2021-05-27

Applications of and Algorithms for Genome Assembly and Genomic Analyses with an Emphasis on Marine Teleosts

Brandon D. Pickett
Brigham Young University

Applications of and Algorithms for Genome Assembly and Genomic Analyses with an Emphasis on Marine Teleosts

Brandon D. Pickett

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Perry G. Ridge, Chair
John S. K. Kauwe, III
Stephen R. Piccolo
Dennis K. Shiozawa
Mark J. Clement

Department of Biology

Brigham Young University

ABSTRACT

Applications of and Algorithms for Genome Assembly
and Genomic Analyses with an Emphasis
on Marine Teleosts

Brandon D. Pickett
Department of Biology, BYU
Doctor of Philosophy

The burgeoning frequency of genome sequencing in recent years is a testament to both the improvements in sequencing technologies and the utility of genomic analyses for biological discovery. The rapid proliferation in technological advancements and availability of complementary data types and techniques has obfuscated the optimal process of genome assembly and raised the barrier to entry to unprecedented levels. In this dissertation, we describe the genome assemblies performed for several marine teleosts and discuss the algorithms and applications required for genome assembly, including some of our specific contributions to the genome assembly and annotation space. In Chapter 1 and Chapter 2, we review the taxonomy, life history, and biogeography of the Roundjaw Bonefish (*Albula glossodonta*) and describe its genome assembly. The genome assemblies with some analyses are described for the Bluefin (*Caranx melampygus*) and Giant (*Caranx ignobilis*) Trevallies in Chapter 3 and Chapter 4, respectively. Chapter 5 and Chapter 6 define and assess algorithms for the annotation of simple sequence repeats in genomic sequences. Publicly available annotations of carbapenem-resistance plasmids were epidemiologically analyzed in Chapter 7. The resiliency of phylogenetic trees to the removal of taxa is explored with a new nodal stability metric and algorithm, TANOS, in Chapter 8. Finally, in Chapter 9, a review of and commentary on vertebrate genome assembly is presented with recommendations for new projects. The aim of this dissertation, and the final chapter in particular, is to explore genome assembly methods and reduce the barrier to entry for new entrants.

Keywords: genome, genome assembly, genomics, assembly, annotation, algorithm, taxonomy, bonefish, giant trevally, bluefin trevally, kingfish, SSR, SA-SSR, Kmer-SSR, kmer, suffix array, ulua, ‘omilu, o‘io

ACKNOWLEDGEMENTS

While my name is on this dissertation, the time, effort, money, and support of others was instrumental in its completion over the last six years. I will list the specific names of groups and individuals, but I must first acknowledge the unseen people who have contributed indirectly to my success by ensuring the university runs smoothly or by anonymously donating to scholarship funds and the university in general.

I was supported primarily by the Department of Biology (https://biology.byu.edu), and I am grateful for both the availability of funds to support me and the people administering the department. I am specifically grateful to the Department Chairs, Dennis K. Shiozawa, Ph.D., John S. K. Kauwe, III, Ph.D., and Richard A. Gill, Ph.D., the Graduate Coordinator, "Uncle" Byron J. Adams, Ph.D., and the Graduate Secretary, Gentri Glaittli. Graduate Studies (https://gradstudies.byu.edu) and the Department of Biology generously supported travel for conferences and awards for competitions. Together, the DNA Sequencing Center (DNASC; https://dnasc.byu.edu) and Illumina (https://www.illumina.com; San Diego, CA, USA) donated short-read sequencing for one of our projects; thank you. The DNASC played a critical role in my research, and I am particularly indebted to Edward R. Wilcox, Ph.D., for his expertise. Similarly, I am grateful to the Office of Research Computing (https://rc.byu.edu) for their support of the computing cluster and my work. I am also grateful to Fly Fishers International (https://flyfishersinternational.org) for funding me with a Conservation Scholarship and to the Society of Freshwater Science for providing funding in the form of a Systematics Award.

I owe my advisor and mentor, Perry G. Ridge, Ph.D., more gratitude than can be effectively expressed. He not only taught my first official bioinformatics course as an undergraduate, but also inspired much of my interest in the field. He convinced me to join his lab when he was a new faculty member, and I am grateful for his instruction during that period of

my academic career – instruction which continued as I returned to BYU for my doctorate. I am grateful for his patience, flexibility, mentorship, support during personal times of strife, friendship, and example as an excellent scientist and human being. From Perry, I have learned the importance of hard work and persistence, as well as the importance of putting first things first.

I am similarly grateful to John "Keoni" S. K. Kauwe, III, Ph.D., for encouraging my initial interest in bioinformatics. I thank him for investing large amounts of his research budget on sequencing fish genomes when I (initially) had no idea what I was doing. I am also grateful for his time and energy spent collecting samples for sequencing. While fishing for incredible sportfish sounds like a treat (because it is!), it also requires a lot of travel time away from family, funding, days in the hot sun, and truly impressive skill. Keoni is also an exceptional example of how to perform good science and how to treat others with respect. Mahalo!

My committee members, Stephen R. Piccolo, Ph.D., Dennis K. Shiozawa, Ph.D., and Mark J. Clement, Ph.D., provided excellent counsel about my research and guidance on our projects. They also taught well-organized, highly effective courses in bioinformatics and computer science, allowed me to be a teaching assistant for them, provided specimens for DNA sequencing and genome assembly, and/or travelled with me to conferences. Thank you for your insights, your kindness, and your one-on-one counsel provided at various times.

I also am grateful to Mark. T. W. Ebbert, Ph.D., for his quick friendship and early mentorship that started when I was a lowly undergrad, and he was a Ph.D. student and postdoc in the lab. Mark has continued to counsel me in my career and life and provide friendship, despite having moved across the country with his family in pursuit of his own career goals. I am grateful to Justin B. Miller, Ph.D., for his close and enduring friendship beginning when we were in

undergraduate classes together. I could not have asked for a better officemate or support through the trials of a graduate program and other life events. Justin started his doctorate with me under Dr. Ridge, and I was inspired by his ability to reason, innovate, and laugh. His family, especially Elisandra Miller, were also exceptionally kind, and I appreciate their friendship and support.

I have also been privileged to work with and, in some cases, befriend others in various departments at BYU and in other institutions. I am grateful to Jessica R. Glass, Ph.D., for her expertise with carangoid fishes and population genomics and her friendship from across the globe. I am indebted to Elizabeth M. Wallace, Ph.D., for her expertise in albulid fishes and guidance navigating a new, complex subject. I thank my coauthors on miscellaneous projects, Sheena Talma, Gareth S. Powell, and Galen E. Card, Ph.D., for the opportunity to collaborate and try something new.

I am grateful to my wife, Alyssa E. Pickett, for her love and support during my program. As a doctorate student herself, she understood my experience in a unique way. I am grateful for her friendship, companionship, and time while she completed her master's program and started a rigorous doctorate program. I am also grateful to my daughter, Mara. They have brought me great joy and purpose, and they have been endlessly encouraging in my pursuit of an education. They also joined me in and often planned much-needed adventures outside the lab. I love you both!

Similarly, my parents, Heather Rae Pickett and John R. Pickett, grandparents, Norman D. Pickett, Roberta M. Pickett, Claudia A. Robertson, and Thomas A. Robertson, and siblings, Bret E. Pickett, Kayla R. Rogler and Evan C. Rogler, and Cami R. Pickett, were supportive and patient with my seemingly endless schooling. I am also grateful for the continued love and encouragement received from many extended family members: Linda S. Mahlum and David J.

Mahlum, Brigett A. Ingram and family, Mary E. Robertson and family, Lisa M. Mahlum, and Jennifer M. Mahlum. Further, I would like to thank my new family through marriage for their love and support: David B. and Loralee L. Evans, Heather E. and Daniel W. Shallenberger family, Brooke E. and Devin R. Clark family, Laurel E. and Randy A. Francis family, Jonathan P. and Mynette K. Evans, and Joseph P. and Joanne Livingston.

During my time at BYU, I was greatly advantaged by the love and support of family and a few close friends living in the area who not only helped me, but also affected me in deeply personal ways: Ashley and Nic Haws family, LeAnn and Steve Gourley and family, Lane and Liz Livingston and family, Lee and Suzanna Livingston and family, Lance and Diane Livingston and family, Karen and Dave Marcum and family, Lindsay and Cam Lee family, Scott N. Gassaway, Tanner and Rachel Crandall and family, and Victoria Violette.

In my experience, the saying that "it takes a village" to raise a child is very true. I have had many villages now, all of which have had unique impacts on my life. Many people had this impact primarily before beginning my doctorate program and have not been included explicitly here, but their impact and importance are, nonetheless, very real and treasured – without them, I could not have made it to this point. To close, I would like to make two final acknowledgements. First, as a religious person graduating from a religious institution, I have the unique opportunity to publicly recognize the influence of my Heavenly Father and my Savior, Jesus Christ. They have transformed my life and my nature. Among many other things, I owe Them my gratitude for the opportunity to pursue knowledge and participate in the advancement of my chosen scientific field. Second, I wish to acknowledge those who supported me in beginning my graduate degree, but who were, unfortunately, unable to witness the completion of it: *in memoriam*, C. D. "Dan" P. Warren (1934-2017) and Hyrum W. Smith (1943-2019).

TABLE OF CONTENTS

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

LIST OF TABLES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

LIST OF FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

CHAPTER 1

Lingering Taxonomic Challenges Hinder Conservation and Management of Global Bonefishes

Brandon D. Pickett[1], Elizabeth M. Wallace[2], Perry G. Ridge[1], John S. K. Kauwe[1]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[2]*Florida Fish and Wildlife Conservation Commission, Fish and Wildlife Research Institute, St. Petersburg, Florida, USA*

A peer-reviewed, production version of this manuscript has been published in
***Fisheries*, 45(7):347-358, DOI: 10.1002/fsh.10438**.

I hereby confirm that the use of this article is compliant with all publishing agreements.

ABSTRACT

Despite expanding research on the popular recreational fishery, bonefish taxonomy remains murky. The genus *Albula*, comprising these iconic circumtropical marine sportfishes, has a complex taxonomic history driven by highly-conserved morphology. Presently, 12 putative species are spread among three species complexes. The cryptic morphology hinders visual identification, requiring genetic species identification in some cases. Unclear nomenclature can have unintended consequences, including exacerbating taxonomic uncertainty and complicating resolution efforts. Further, ignoring this reality in publications may erode management and conservation efforts. In the Indian and Pacific oceans, ranges and areas of overlap are unclear; precluding certainty about which species support the fishery and hindering conservation efforts. Species overlap, at both broad and localized spatial scales, may mask population declines if one is targeted primarily (as demonstrated in the western Atlantic fishery). Additional work is necessary, especially to increase our understanding of spatiotemporal ecology across life history stages and taxa. If combined with increased capacity to discern between cryptic species, population structure may be ascertained, and fisheries stakeholders will be enabled to make informed decisions. To assist in such efforts, we have constructed new range maps for each species and species complex. For bonefishes, conservation genomic approaches may resolve lingering taxonomic uncertainties, supporting effective conservation and management efforts. These methods apply broadly to taxonomic groups with cryptic diversity, aiding species delimitation and taxonomic revisions.

BACKGROUND

Bonefish (Albulidae) *Albula spp.* are tropical, marine, benthivorous fish found principally in sand flats, sea grasses, and mangroves. They are characterized by an inferior mouth with the snout extending beyond the mandible (Hildebrand 1963; Datovo and Vari 2014) (Figure 1). Although bonefish are a source of food in some parts of the world (Breder 1948; Scott and Scott 1988), the principal interests to humans are fishing and tourism as bonefish are prized sportfish since they are elusive and difficult to land. The sportfishing tourism industry for bonefish in the Bahamas was estimated at $141 million USD (Fedler 2010), while the flats fishery (bonefish and other flats species) in the Florida Keys was estimated at $465 million USD (Fedler 2013). Despite a culture, sometimes enforced by law, of catch-and-release fishing (Adams and Cooke 2015; Adams 2016), bonefish catch rates appear to be declining around the globe (Friedlander and Rodgers 2008; Santos et al. 2019a). Preserving bonefish diversity and the flats fisheries depends on increasing our understanding of each species' ecology and life history; however, most research has focused on a single species, *Albula vulpes* (Linnaeus 1758). In part, this is a result of the complicated taxonomy that is currently under revision. Much of the difficulty emanates from several cryptic species – species that are effectively impossible to discern visually due to high morphological similarity. After providing a brief background in bonefish life history and ecology, global depletions of bonefish populations, and cryptic species, we discuss bonefish taxonomic history and the resulting implications for conservation and management.

Ecology and Life History

Bonefish are circumtropical shorefish with an interesting life history. Although the bulk of our knowledge comes from *A. vulpes* and is, in some cases, based on a single site or region,

most characteristics and behaviors may be similar across the genus, except perhaps for the *Albula nemoptera* complex. Additional research for all species, including *A. vulpes*, is still required to fill in the gaps in our understanding of bonefish spatiotemporal ecology.

Like all elopomorphs, bonefishes spend time in development as transparent, ribbon-like larvae called leptocephali (Hollister 1939; Rasquin 1955; Inoue et al. 2004). The leptocephali feed principally on plankton as they grow in length to about 6-9 cm (Hollister 1936; Pfeiler 1984; Vásquez-Yeomans et al. 2009). Exact pelagic larval duration may vary considerably across taxa, however in *A. vulpes* ranges 41-71 days (Mojica et al. 1994; Adams and Cooke 2015). They then undergo a fascinating metamorphosis in which they shrink to about two cm, resulting in individuals reaching the same length three times during development. During the approximately ten day metamorphosis, the leptocephalus transitions to a miniature of the adult form (Hollister 1936; Pfeiler 1984). Pre-metamorphic larvae have some swimming capacity; however, considering ocean currents, they may disperse hundreds of km away from their spawning site (Zeng et al. 2019).

Post-metamorphic larvae move into shallower water to utilize mangroves and estuaries as nurseries for 2-4 years. Evidence from Florida (USA) and Cuba, based on *A. vulpes* and *A. sp. cf. vulpes* (Wallace and Tringali 2010), suggests that juveniles prefer the less saline waters in estuaries compared to the more saline environment of the flats where adults are typically found (Santos et al. 2019b). However, *A. goreensis* (Cuvier and Valenciennes 1847; commonly, Channel Bonefish) appears to also utilize more exposed beach habitat (Haak et al. 2019), and preferred juvenile habitat for other species is unknown. The juvenile diet consists primarily of amphipods and carideans, though diet analyses are limited (Griffin et al. 2019). Despite the

importance of early life history to population stability and resilience (Lefcheck et al. 2019), relatively little is known of juvenile behavior and ecology.

Adults grow to lengths of 100 cm (Scott and Scott 1988) and up to 8 kg in weight (Robins and Ray 1986), though size reports vary among species and locations; a typical adult is probably half as long and heavy (Donovan et al. 2015; Kamikawa et al. 2015). Bonefish lifespans can extend past 20 years, though they average less (Posada et al. 2008). Their diet consists primarily of mollusks and crustaceans, but other benthic fauna is not unusual (Warmke and Erdman 1963; Colton and Alevizon 1983; Liston et al. 2013). Some evidence suggest they forage nomadically, changing location every few days (Ault et al. 2008), though they have high site-fidelity for a general area (Murchie et al. 2013; Boucek et al. 2019; Moxham et al. 2019). In *A. vulpes*, spawning migrations of varied distances (over 80 km documented) occur October through May (Murchie et al. 2015), sometimes near the full or new moons (Adams et al. 2019). Large pre-spawning aggregations with hundreds to thousands of fish form in relatively shallow water, and then move to deep-water drop-offs at dusk to spawn (Danylchuk et al. 2011; Danylchuk et al. 2019). Though other bonefishes may exhibit similar spawning behaviors to *A. vulpes*, timing likely varies across taxa and reproductive ecology has not been evaluated in other species. This information is important for conservation and management globally, as pre-spawning aggregations are vulnerable to harvest and coastal migratory corridors are susceptible to human disturbance.

Relative to *A. vulpes*, the literature on the ecology and life history of other bonefish species is sparse. Differences have been identified between species complexes and some individual species. Of particular importance is research to determine fishery species composition at local scales in areas of known species overlap and further elucidate spawning behaviors and

locations for species supporting fisheries. Without this fundamental information, population declines within a particular fishery (i.e., island or nation) may be masked due to the presence of cryptics and conservation efforts may be confounded due to interspecific variability.

Population Declines

Decreases in bonefish catch rates and instances of shifting baselines have been reported around the globe. However, accurate data from all relevant components of the fishery (recreational catch and release, subsistence harvest, targeted and incidental commercial harvest) are often lacking. Anthropogenic habitat loss is suspected as the primary contributor to population declines in most areas, but exploitation in under-regulated fisheries is also a significant problem (Bunce et al. 2008; Adams et al. 2012g; Filous et al. 2019a). Even in catch and release fisheries, the negative impact to the target species may be larger than previously thought (Dallas et al. 2010; Raby et al. 2014; Brownscombe et al. 2015; Cook et al. 2015), and recent research has focused on understanding and mitigating the effects of catch-and-release practices (Hannan et al. 2015; Adams 2016; Brownscombe et al. 2017). Regardless of the precise cause, The International Union for the Conservation of Nature (IUCN) Red List of Threatened Species™ reports *A. glossodonta* (Forsskål 1775; commonly, Roundjaw Bonefish) as "Vulnerable", *A. vulpes* as "Near Threatened", and *A. esuncula* (Garman 1899; commonly, Eastern Pacific Bonefish) as "Least Concern" (Nielsen et al. 2010; Adams et al. 2014). Five other species are listed as “Data Deficient” and the remaining four have not yet been evaluated (see Table 1). Insufficient data is clearly a bottleneck for ecological work with most bonefish species. Yet, even for *A. vulpes* where information is relatively plentiful, data is still deficient to (a) determine how much population decline is caused by overfishing as opposed to anthropogenic

habitat loss and (b) which species in the *A. vulpes* species complex may be most vulnerable (Adams et al. 2014). Indeed, information is not available for many areas and species, but available data does raise concerns: (a) catch rates are decreasing in the Southwestern Indian Ocean and the Florida Keys (Florida, USA) according to fishers (Bunce et al. 2008; Frezza and Clem 2015; Santos et al. 2019a) (b) demand from recreational tourist fishers is increasing in the Bahamas (Danylchuk et al. 2008), (c) data from the National Oceanic and Atmospheric Administration (NOAA) Marine Recreational Information Program (MRIP) suggest population declines in the Western Atlantic Ocean (National Marine Fisheries Service, Fisheries Statistics Division, pers. comm.), (d) data from Hawai'i's Department of Land and Natural Resources/Division of Aquatic Resources and the United States Fish Commission demonstrate precipitous declines in landings in Hawaiian waters (Friedlander and Rodgers 2008), and (e) unsustainable fishing practices and extirpation of spawning groups have been documented in the South Pacific Ocean (Johannes and Yeeting, 2000; Ram-Bidesi, 2011; Ram-Bidesi and Petaia, 2010). The clear consensus is that population declines are occurring; the uncertainties are to what extent they are occurring, specific causes, and which species are at the highest risk.

Cryptic Species

In bonefishes, the presence of morphologically cryptic species creates challenges to conservation and management (Colborn et al. 2001; Pfeiler et al. 2002; Wallace and Tringali 2016). Correct identification of cryptic species is a prerequisite to examinations of biogeographic and ecological processes as well as conservation applications (Jörger and Schrödl 2013). Cryptic species are relatively widespread, and their recognition is generally considered nontrivial (Bickford et al. 2007; Trontelj and Fišer 2009; Reist et al. 2013). Black basses (*Micropterus*

spp.) and Charrs (*Salvelinus*), iconic sportfishes themselves, are similarly under active taxonomic revision (Reist et al. 2013; Taylor et al. 2019). The conservation and management challenges for any group with cryptic species are inherently similar. In bonefishes, the presence of cryptic species and broadly overlapping ranges make it very difficult to determine the species composition in various fisheries. Occurrences of secondary contact (Pfeiler et al. 2008b) and hybrids (Wallace and Tringali 2016; Rennert et al. 2019) have been documented among bonefish. While the extent and frequency of hybrids are unknown, they further challenge efforts to understand bonefish relationships and ecology. Unsurprisingly, *Albula* is too often described as monotypic and placeholder names are perpetuated after formal descriptions have updated the terms for a given species (Galdino Brandão et al. 2016; Joshi et al. 2016; Abdussamad 2017). Without distinguishing between cryptic species of bonefish in areas of overlap, conservation and management decisions will remain difficult. Increased understanding of spatiotemporal ecology for the various life stages and ability to discriminate between the various cryptic species are necessary to discern population structure and making effective policy decisions.

TAXONOMIC HISTORY

Bonefish were initially described by Linnaeus (1758) as *Albula vulpes*. Twenty-three independent discoveries of bonefish were described under various names, but were eventually synonymized into a circum-global *A. vulpes* by 1940 (Whitehead 1986; Colborn et al. 2001; Bowen et al. 2008) as no significant characters were able to consistently delineate species (Hildebrand 1963). However a second bonefish species, *A. nemoptera* (Fowler 1911; commonly, Threadfin Bonefish), was recognized at this time; it is both rarely encountered by anglers due to its deep-water habitat and easily distinguished by an elongated caudal ray of the dorsal fin

(Fowler 1911; Rivas and Warlen 1967). This new status quo was later broken by Shaklee and Tamaru (1981) when they demonstrated by molecular analysis that two species of bonefish are present in Hawaiian waters, *A. glossodonta* and *A. neoguinaica* (Cuvier and Valenciennes 1847). *A. neoguinaica* was subsequently renamed to *A. forsteri* (Bloch and Schneider 1801) and then *A. argentea* (Forster in Bloch and Schneider 1801) (see Bowen et al. (2008) for further details). Colborn et al. (2001) confirmed and extended the results of Shaklee and Tamaru's study with additional molecular analyses, screening 174 specimens from 26 globally distributed sites for a portion of the mtDNA cytochrome b gene. They concluded that the three species (*A. vulpes*, *A. glossodonta*, and A. *neoguinaica* (now *A. argentea*)) are distinct and that up to five additional species may be present, which they labeled as *A. spp. A-E*. Since then, these and additional species have been described resulting in twelve putative species spread across three species complexes (see Table 2 for a summary of species names and distributions and Figures 3-6 and Supplementary Figures 1-16 for maps of their distributions).

Some morphological traits enable distinction between the complexes, but expertise is usually required. The currently accepted phylogeny, based on portions of the mtDNA cytochrome b gene, is represented in Figure 2. The three complexes form distinct clades, with the *A. vulpes* and *A. argentea* complexes as sisters relative to the *A. nemoptera* complex. Given the currently accepted relationships (Figure 2), we summarize each of the three complexes. Note that we are not reviewing the two deep-water bonefish species in the genus *Pterothrissus*. See Wallace (2014) for a discussion on whether *Pterothrissus* belongs in the order Albuliformes and Hidaka et al. (2017) for more recent taxonomic reclassification.

Albula argentea complex

Bonefish in the *A. argentea* complex are distributed throughout the Indian and Western and Central Pacific Oceans (Pfeiler et al. 2011; Wallace 2014) (Figure 4; Supplementary Figures 1-4). This species complex is well reviewed by Hidaka et al. (2008). In brief, the complex is comprised of three species: *A. argentea*, *A. oligolepis* (Hidaka et al. 2008; commonly, Smallscale Bonefish), and *A. virgata* (Jordan and Jordan 1922; commonly, Longjaw Bonefish). The species in this complex were resurrected from synonymy with *A. vulpes*, beginning with Shaklee and Tamaru's study (1981). The Hawaiian specimens they identified as *A. neoguinaica* are now known as *A. virgata* as a result of Hidaka et al. (2008); their work clarified *A. forsteri* as a junior synonym of *A. argentea*, accounting for the non-endemic specimens that Shaklee and Tamaru (1981) identified as *A. neoguinaica*. *Albula oligolepis* was described as a new species in the same paper (Hidaka et al. 2008). These are distinct from *A. glossodonta* (in the *A. vulpes* complex), whose range overlaps in the Indian and Western Pacific Oceans (Supplementary Figure 10), due to molecular differences and because *A. oligolepis* has a more pointed lower jaw. All species in the *A. argentea* complex share this trait relative to those in the *A. vulpes* complex. *Albula oligolepis* is *A. sp. D* from Colborn et al. (2001).

Albula nemoptera complex

The threadfin bonefish, *A. nemoptera,* was first described by Fowler (1911) in the genus *Dixonina* but later synonymized with *Albula* (Rivas and Warlen 1967). The range for the species in this complex is the Western Atlantic and Eastern Pacific Oceans and they are typically found in deeper water (often in estuaries (Robins and Ray 1986)) than bonefish in the *A. argentea* and *A. vulpes* complexes (Bowen et al. 2008) (Figure 5; Supplementary Figures 5-7). *Albula nemoptera spp.* (*A. sp. E* from Colborn et al. (2001)) are further distinguished by shorter total

length, elongated anal fin and caudal ray of the dorsal fin, mouth reaching a point below the eye, small scales, and a few differences in dentition and meristic characters (Rivas and Warlen 1967; Robins and Ray 1986). The Western Atlantic Ocean form is *A. nemoptera* and the Eastern Pacific Ocean form is designated *A. pacifica* (Beebe 1942; commonly, Pacific Shafted Bonefish) (Pfeiler et al. 2006; Pfeiler 2008). Based on cytochrome b sequence data, they were designated sister species (Pfeiler 2008); additional nuclear gene sequence data supports this (Wallace 2014). We will discuss neither *A. nemoptera* nor *A. pacifica* further in this review as they are easily distinguished morphologically from other bonefish and not the target of a large sportfishing industry.

Albula vulpes complex

Bonefish in the *A. vulpes* complex can be found around the globe (Figure 6; Supplementary Figure 8). Presently, seven species are recognized: *A. vulpes*, *A. glossodonta*, *A. esuncula*, *A. sp. cf. vulpes*, *A. koreana* (Kwun and Kim 2011; commonly, Korean Bonefish), *A. gilberti* (Pfeiler et al. 2011; commonly, Cortez Bonefish), and *A. goreensis*.

A. vulpes (Bonefish)

This is the original bonefish, described by Linnaeus (1758), with which all other species were synonymized by 1940 (Whitehead 1986; Colborn et al. 2001; Bowen et al. 2008). As additional species were later recognized or resurrected, the range of this species has decreased from worldwide to only the Caribbean, Gulf of Mexico, and Western Atlantic Ocean (Wallace 2014) (Supplementary Figure 9).

A. glossodonta (Roundjaw Bonefish)

Albula glossodonta was identified in Hawaiian waters by Shaklee and Tamaru (1981) based on molecular data. It possesses the largest range of any bonefish species, encompassing the Indian Ocean and Western and Central Pacific Ocean (Wallace 2014) (Supplementary Figure 10). Recent studies suggest that *A. glossodonta* individuals are larger, on average, than *A. vulpes* (Donovan et al. 2015). They may also live half as long and spawn between March and September, instead of between October and May as *A. vulpes* does (Filous et al. 2019b).

A. esuncula (Eastern Pacific Bonefish)

Albula esuncula occurs in the Eastern Pacific Ocean; it was previously identified as *A. sp. C* in Colborn et al. (2001) and later clarified in Pfeiler et al. (2008a). Its range stretches south to Panama and reaches north to Sinaloa, Mexico where it occurs sympatrically with *A. gilberti* (Supplementary Figures 11 & 16). *Albula gilberti* (*A. sp. A* in Colborn et al. (2001)) is found northward in the Gulf of California, stretching south to Sinaloa, Mexico. Thus, these two species occur principally in parapatry, except in the southern Gulf of California, where they are found in sympatry. *Albula esuncula* was formally described by Pfeiler et al. (2011) as a necessary step in the description of *A. gilberti*. They are morphological cryptics; however, they may be distinguished genetically (Pfeiler et al. 2008a; Díaz-Viloria et al. 2017).

A. sp. cf. vulpes

Continuing the nomenclature of Colborn et al. (2001), *A. sp. F* was postulated as another species by Valdez-Moreno et al. (2010). Further identification was then provided by Wallace and Tringali (2010) and the species is presently referred to by the placeholder *A. sp. cf. vulpes*. A formal description is forthcoming. This species is a morphological cryptic of *A. vulpes*; its range

is the Western Atlantic Ocean, Gulf of Mexico, and Caribbean (Wallace and Tringali 2010; Wallace 2014) (Supplementary Figure 12).

A. koreana (Korean Bonefish)

This species was described by Kwun and Kim (2011) after morphological and molecular comparison with *A. argentea*; it has a restricted range in the southern Sea of Japan and East China Sea (Supplementary Figure 13). They differ based on vertebrae count and tooth patch distributions on the parasphenoid and mesopterygoid bones. Molecular differences (nuclear and mitochondrial) were also identified (Kwun et al. 2011; Wallace 2014).

A. gilberti (Cortez Bonefish)

Albula gilberti occurs in the Eastern Pacific Ocean (previously *A. sp. A* from Colborn et al. (2001)). Its range extends northward in the Gulf of California, stretching south around Sinaloa, Mexico – where it is sympatric, likely through secondary contact, with *A. esuncula* (Pfeiler et al. 2008b) (Supplementary Figures 14 & 16).

A. goreensis (Channel Bonefish)

Wallace (2014) resurrected *A. goreensis*, a morphological cryptic, from synonymy with *A. vulpes*. *Albula goreensis* is *A. sp. B* from Colborn et al. (2001) and has previously been referred to as *A. garcia* (Bowen et al. 2008; Valdez-Moreno et al. 2010; Galdino Brandão et al. 2016). Its range extends across the tropical Western and Eastern Atlantic Ocean, Gulf of Mexico, and the Caribbean (Whitehead 1990; Bowen et al. 2008; Wallace 2014) (Supplementary Figure 15). Recent work suggests *A. goreensis* adults are smaller than *A. vulpes* and they may occupy a different hydrodynamic niche (Haak et al. 2019; Rennert et al. 2019).

A Note on Distribution Maps

We generated new distribution maps for each of the bonefish species. Much of this information was derived from the IUCN reports, when available. The remaining information resulted from sieving the literature and the personal knowledge of the authors. Deviations from IUCN reported ranges are based on genetically verified collections. While uncertainties exist, these maps represent the best information currently available regarding bonefish species ranges. The full extent of ranges remains unknown for many species – absence on a map indicates no recorded and genetically verified collections. In areas with appropriate habitat, bonefish may occur there – we simply lack data. Alternately, the coastline of a country may be indicated, though appropriate bonefish habitat likely has a patchy distribution. Further, the exact width of highlighted areas is not intended to carry meaning – highlighted areas are simply wide enough to see easily. In some areas, the highlighted width is thinner to avoid overlapping other areas. All maps were generated by hand using Adobe Illustrator CC 2019 (https://www.adobe.com/creativecloud.html); native vector graphics files are available in multiple formats on The Open Science Framework at the following DOI: 10.17605/OSF.IO/J4KSW.

CONSERVATION AND MANAGEMENT IMPLICATIONS

Pursuing the goals of conserving bonefish diversity and ensuring the long-term sustainability of recreational fisheries is a complicated challenge. For the global fishery, a primary impediment is the dearth of necessary biological and ecological information. Bonefish taxonomy remains under active revision, many life history and ecological traits are unknown, and the presence of cryptics creates additional conservation challenges. The focus of this review

has been the current state of the taxonomic revisions, which have been hampered by divergent lineages with highly conserved morphology. The difficulties regarding species identification have also impeded our understanding of basic life history characteristics and behaviors. Recent research suggests differences between (a) cryptic species in the Western Atlantic Ocean and Caribbean Sea (Adams et al. 2008; Haak et al. 2019; Rennert et al. 2019), (b) cryptic species in the Pacific Ocean (Donovan et al. 2015), and (c) species in the Atlantic Ocean and the Indian and Pacific Oceans (Filous et al. 2019b). However, life history traits for many taxa remain unknown.

Research efforts have broached topics such as juvenile habitat (Szekeres 2017; Santos et al. 2019b), energy dynamics (Murchie et al. 2011; Szekeres et al. 2014; Nowell et al. 2015), spawning (Luck et al. 2019; Mejri et al. 2019a; Mejri et al. 2019b), habitat use (Brownscombe et al. 2019) and threats (Steinberg 2015; Cissell and Steinberg 2019; Sweetman et al. 2019), migration (Murchie et al. 2015; Boucek et al. 2019; Perez et al. 2019), anthropogenic exploitation (Filous et al. 2019a), leptocephalus larval dispersion (Zeng et al. 2019), gear restriction (Donovan et al. 2016), light pollution (Szekeres et al. 2017), and local ecological knowledge (Kamikawa et al. 2015; Rehage et al. 2019; Santos et al. 2019a). Research efforts have begun to expand beyond *A. vulpes*, especially into *A. glossodonta*. Nevertheless, additional research is still needed; of principle importance is understanding species composition of fisheries at local scales.

Future Directions

The continuation of research efforts on the aforementioned variety of topics in fisheries around the globe is crucial, as is clarifying the taxonomic status of bonefishes. The designation of species and evolutionarily significant units (ESUs) provides the necessary foundation for

conservation efforts and protections afforded through the Endangered Species Act, IUCN Red List, and Convention on International Trade in Endangered Species of Wild Fauna and Flora (CITES). Taxonomic clarity can further aid prioritization of conservation and management actions given the realities of increasing anthropogenic ecosystem alterations and limited resources for conservation. Since relatively few morphological characters are capable of distinguishing between only some species, bonefish research will continue to require a large genetic component. Identification has routinely been accomplished based on mitochondrial cytochrome b sequence identity (Colborn et al. 2001; Pfeiler et al. 2002; Pfeiler et al. 2006; Pfeiler 2008; Valdez-Moreno et al. 2010; Kwun and Kim 2011; Kwun et al. 2011; Wallace 2014; — 2015; Díaz-Viloria et al. 2017), though some bonefishes may also be identified using microsatellite markers (Seyoum et al. 2008; Wallace 2015; Wallace and Tringali 2016). To resolve interspecific relationships, a robust phylogenetic analysis of the family will require more data as single-gene methods – especially from mtDNA – provide an incomplete picture of evolutionary history (Pamilo and Nei 1988; Nichols 2001; Song et al. 2008). A multi-locus approach, especially at the whole-genome or transcriptome scale, would improve confidence in species delimitation and could provide higher-resolution insights into population structure.

In combination with other biological and ecological studies, genetic / genomic approaches can illuminate a wide range of biodiversity issues necessary for conservation goals at population, species, and higher taxonomic levels. Remaining information needs regarding how bonefish species are distributed, such as ESUs, species delimitation, stock identification, adaptation, bottlenecks, introgressive hybridization, and phylogenetic relationships, can be addressed with advanced genomics techniques. To meet these needs, pooled sequencing of specimens will allow the identification of orders of magnitude more markers and will help assess

variation and perform accurate identification. In addition, at least one assembled and annotated genome from each species complex would be a valuable resource and would facilitate additional research on Elopomorpha. Efforts are currently underway with the goal of improved ability to identify species and further study the life history and ecology of the various bonefish species.

Further, protection of presumed endangered species of bonefish is impossible without a multidisciplinary approach. *Albula glossodonta*, Red List Vulnerable and targeted by consumptive fisheries, may be at greatest risk of regional extirpation and many others in the genus remain data deficient. Larger-scale genetic or genomic analyses may provide key information necessary to make important management decisions. Conservation of bonefishes must include actions at multiple spatial and temporal scales. Effectively managed reserves (such as for spawning sites) play an important role; however, additional consideration must be given to migration corridors, as well as larval settlement and juvenile nursery habitats – all of which will vary among species. These areas extend beyond the scale practical for formal reserve status and will require proactive management largely focused on mitigation of coastal habitat degradation. As we learn more about the distinct larval settlement and juvenile nursery habitat requirements among sympatric bonefishes, it will aid comprehensive and proactive habitat protections and mitigation efforts. Habitat conservation efforts will necessarily include limitations on coastal development. In consumptive fisheries, determination of sustainable harvest levels and enforcement of regulations remain high priorities. Clarification of taxonomic status, species boundaries, and areas of overlap are foundational to all of these directed conservation efforts.

Ultimately, fisheries managers and conservationists remain in a quandary over bonefish preservation until additional data are obtained. Presently, twelve putative species are distributed across three species complexes. The geographic extent, size, and species composition of global

fisheries remains unelucidated. Studies with higher-density genetic variation data from populations around the globe, will greatly aid clarification of relationships among these iconic sportfishes. Such approaches are invaluable conservation tools, especially among sympatric cryptic species. These methods will assist ongoing bonefish conservation efforts, and similar genomic techniques will aid species and population delineation in other groups containing morphological cryptics.

ACKNOWLEDGEMENTS

We thank Derek Olthuis, Dr. Jocelyn Curtis-Quick, and Dr. Christopher Haak for providing photos. Suggestions that improved this manuscript were provided by two anonymous reviewers. We also thank Diane Rome Peebles for providing the illustration. There is no conflict of interest declared in this article.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Taxonomic and conservation statuses of each bonefish species. All species, except A. sp. cf. vulpes, are recognized in Eschmeyer's Catalog of Fishes (Fricke et al. 2019). Near Threatened, Vulnerable, Least Concern, and Data Deficient are formal classifications of the International Union for the Conservancy of Nature (IUCN); the term Unevaluated indicates the IUCN has not yet evaluated the status of that species. Common names all include bonefish (e.g., smallscale bonefish).

Scientific name	Common name	Taxonomic status	Conservation status
Albula argentea complex			
A. argentea (Forster in Bloch and Schneider 1801)	NA	Described species	Data Deficient
A. oligolepis (Hidaka et al. 2008)	Smallscale	Described species	Data Deficient
A. virgata (Jordan and Jordan 1922)	NA	Described species	Data Deficient
Albula nemoptera complex			
A. nemoptera (Fowler 1911)	Threadfin	Described species	Data Deficient
A. pacifica (Beebe 1942)	Pacific Shafted	Described species	Unevaluated
Albula vulpes complex			
A. vulpes (Linnaeus 1758)	Bonefish	Described species	Near Threatened
A. glossodonta (Forsskål 1775)	Roundjaw	Described species	Vulnerable
A. esuncula (Garman 1899)	Eastern Pacific	Described species	Least Concern
A. sp. cf. vulpes (Wallace and Tringali 2010)	NA	Provisional species	Unevaluated
A. koreana (Kwun and Kim 2011)	Korean	Described species	Data Deficient
A. gilberti (Pfeiler et al. 2011)	Cortez	Described species	Unevaluated
A. goreensis (Cuvier and Valenciennes 1847)	Channel	Described species	Unevaluated

Table 2. Summary of other applied names and geographic distribution. See Figures 3-6 and Supplementary Figures 1-16 for maps of the geographic distributions.

Species	Other applied names	Distribution
Albula argentea complex		
Albula argentea (Forster in Bloch and Schneider 1801)	*A. forsteri*, *A. neoguinaica*	Western & Central Pacific
Albula oligolepis (Hidaka et al. 2008)	*A. sp. D*	Indian & Western Pacific
Albula virgata (Jordan and Jordan 1922)	*A. neoguinaica*	Hawai'i, USA
Albula nemoptera complex		
Albula nemoptera (Fowler 1911)	*A. sp. E*, *Dixonina nemoptera*	Western Atlantic & Caribbean
Albula pacifica (Beebe 1942)	*A. nemoptera*	Tropical Eastern Pacific
Albula vulpes complex		
Albula vulpes (Linnaeus 1758)	NA	Western Atlantic & Caribbean
Albula glossodonta (Forsskål 1775)	NA	Indian, Western & Central Pacific
Albula esuncula (Garman 1899)	*A. sp. C*, *A. neoguinaica*	Tropical Eastern Pacific, Southern Gulf of California
Albula sp. cf. vulpes Wallace and Tringali (2010)	*A. sp. F*	Western Atlantic & Caribbean
Albula koreana (Kwun and Kim 2011)	NA	Western Pacific (East China Sea)
Albula gilberti (Pfeiler et al. 2011)	*A. sp. A*	Eastern Pacific, Gulf of California
Albula goreensis (Cuvier and Valenciennes 1847)	*A. sp. B*, *A. garcia*, *A. nova sp.*	Tropical Atlantic & Caribbean

*Amended from Wallace (2014)

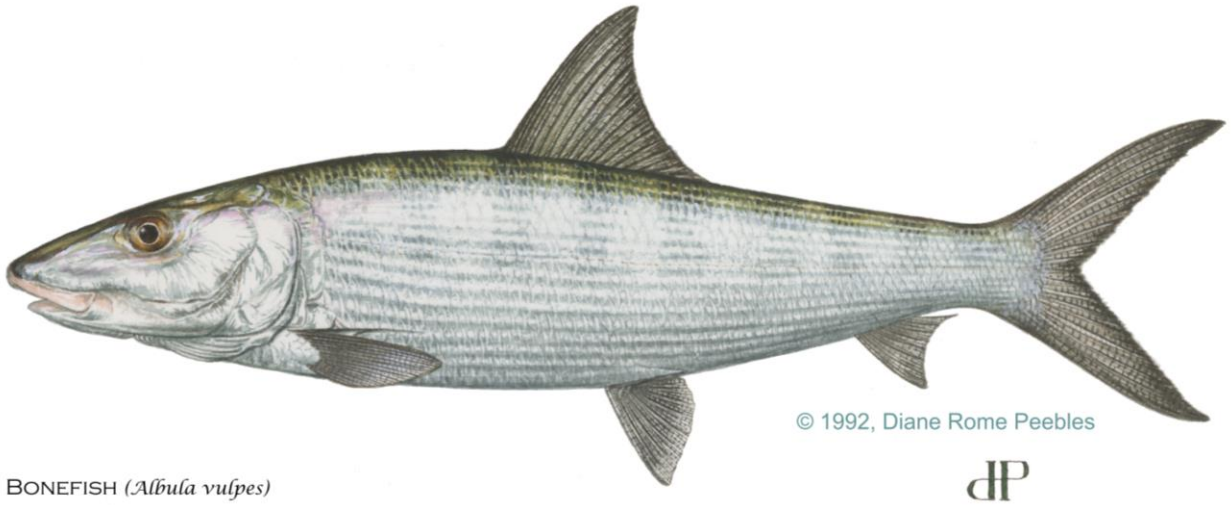


Figure 1. Illustration of *Albula vulpes* – copyright Diane Rome Peebles, used with permission.

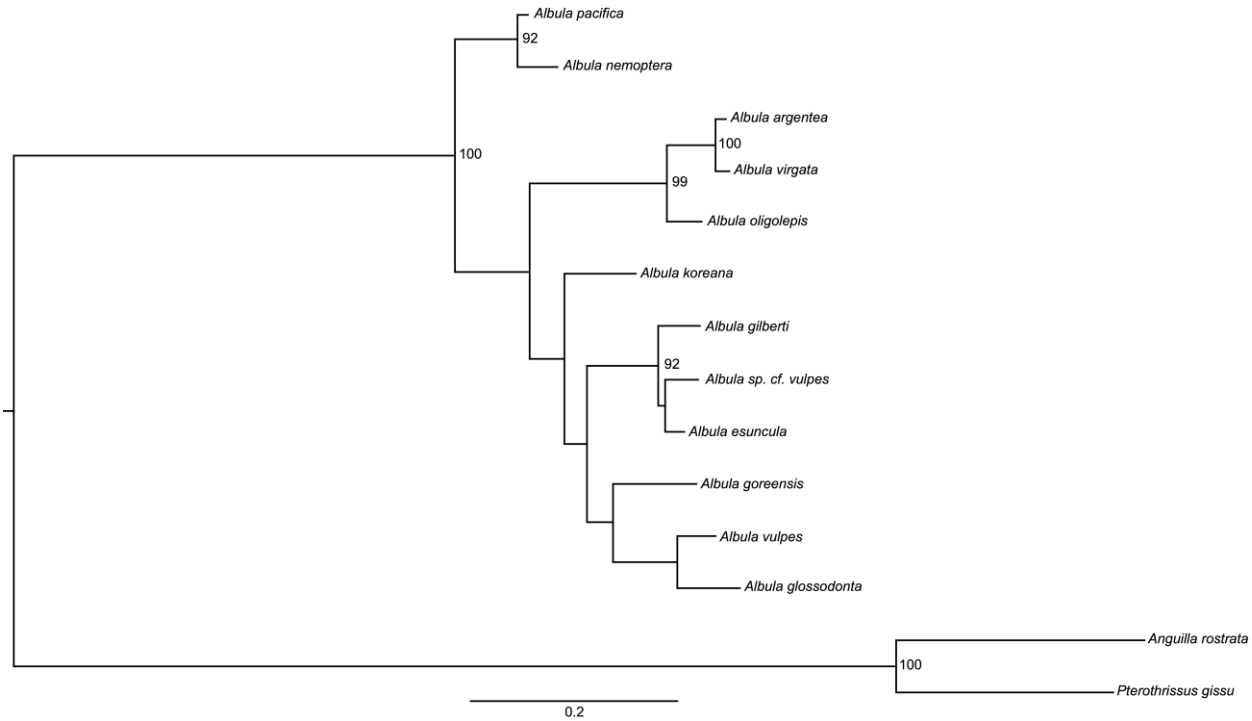


Figure 2. Relationships among all species of *Albula*. Tree topology was inferred using RAxML (Stamatakis 2014) with a portion of the cytochrome b mitochondrial gene. Branch lengths represent sequence divergence between taxa, and bootstrap support values are shown when above 90%. For additional details, see Wallace (2014). A text-based version of the tree can be found in Supplementary File 1 (Appendix 1 herein).

Figure 3. Distribution map of each species complex in *Albula*. Individual maps for each complex can be found in Supplementary Figures 1, 5, and 8. Please see the note on distribution maps.

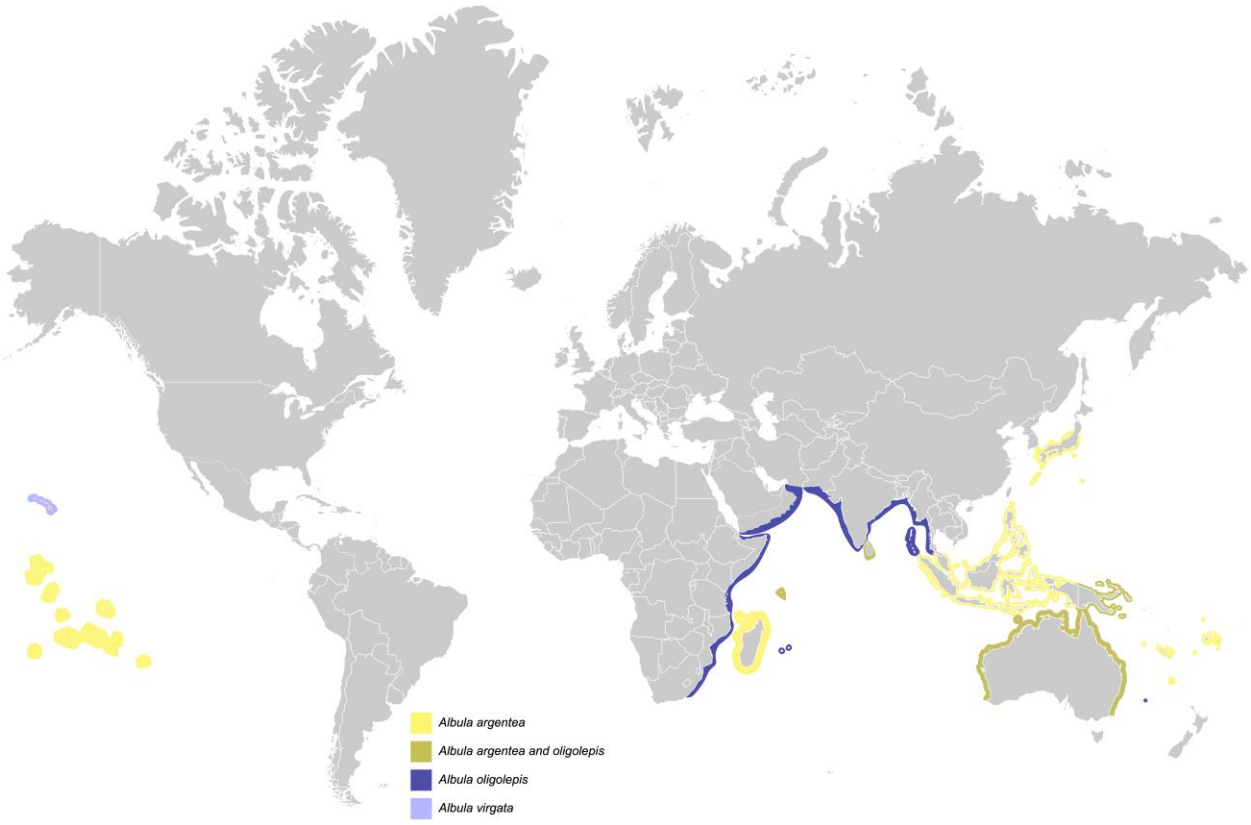


Figure 4. Distribution map of species in the *Albula argentea* species complex. A non-specific map showing this complex can be found in Supplementary Figure 1. Individual maps for each species can be found in Supplementary Figures 2-4. Please see the note on distribution maps.

Figure 5. Distribution map of species in the *Albula nemoptera* species complex. A non-specific map showing this complex can be found in Supplementary Figure 5. Individual maps for each species can be found in Supplementary Figures 6 and 7. Please see the note on distribution maps.

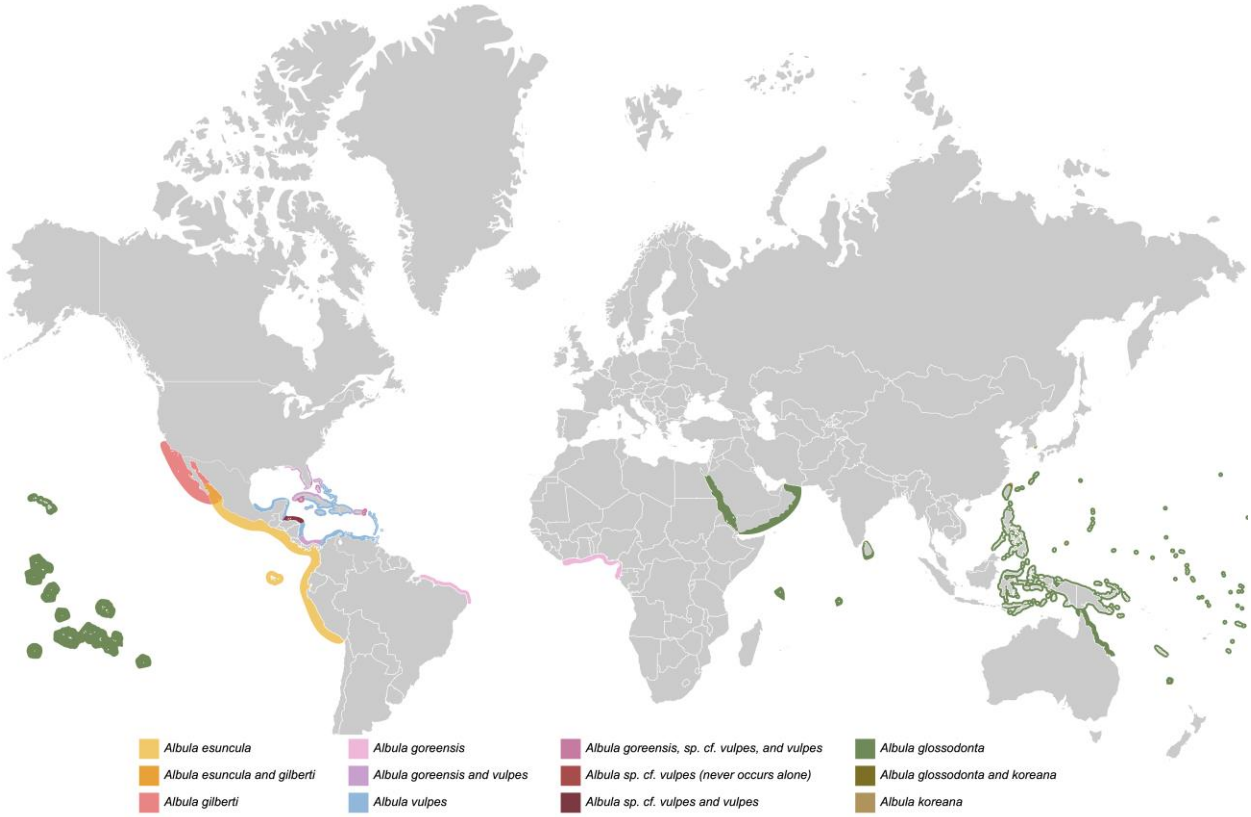
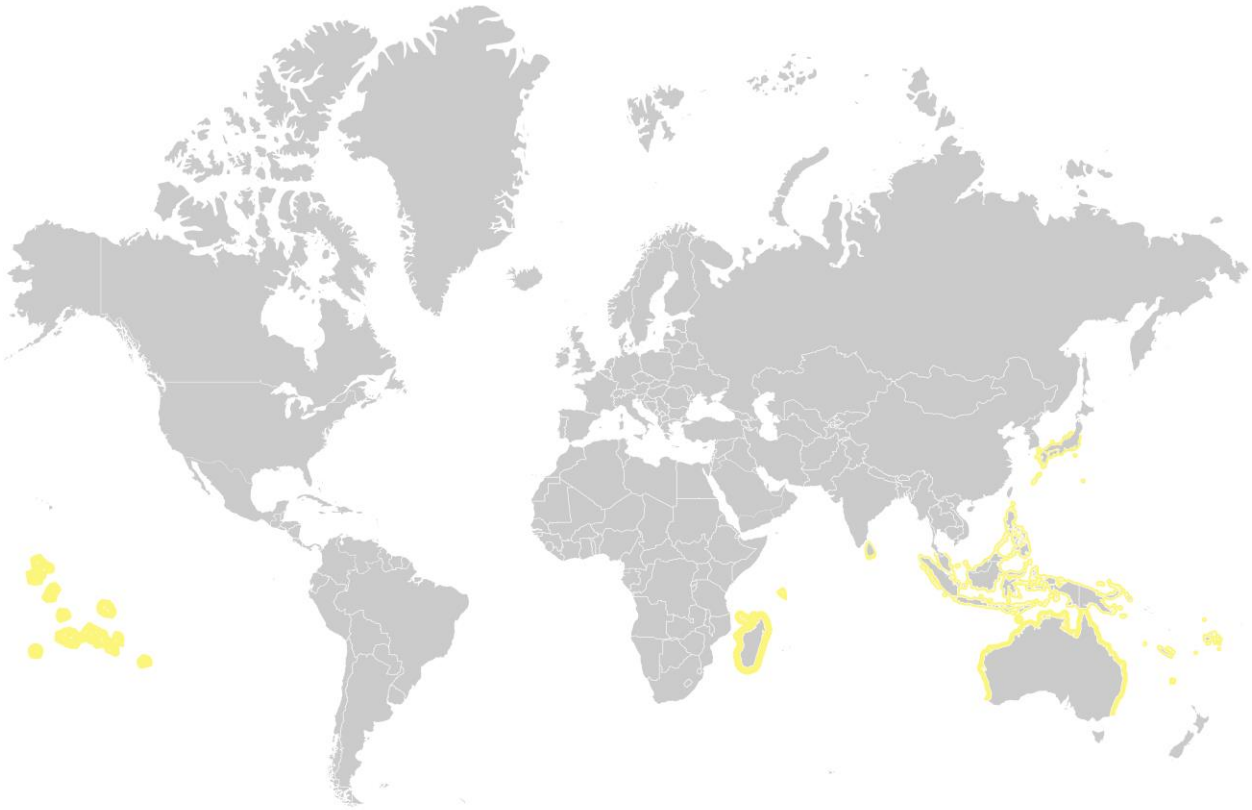
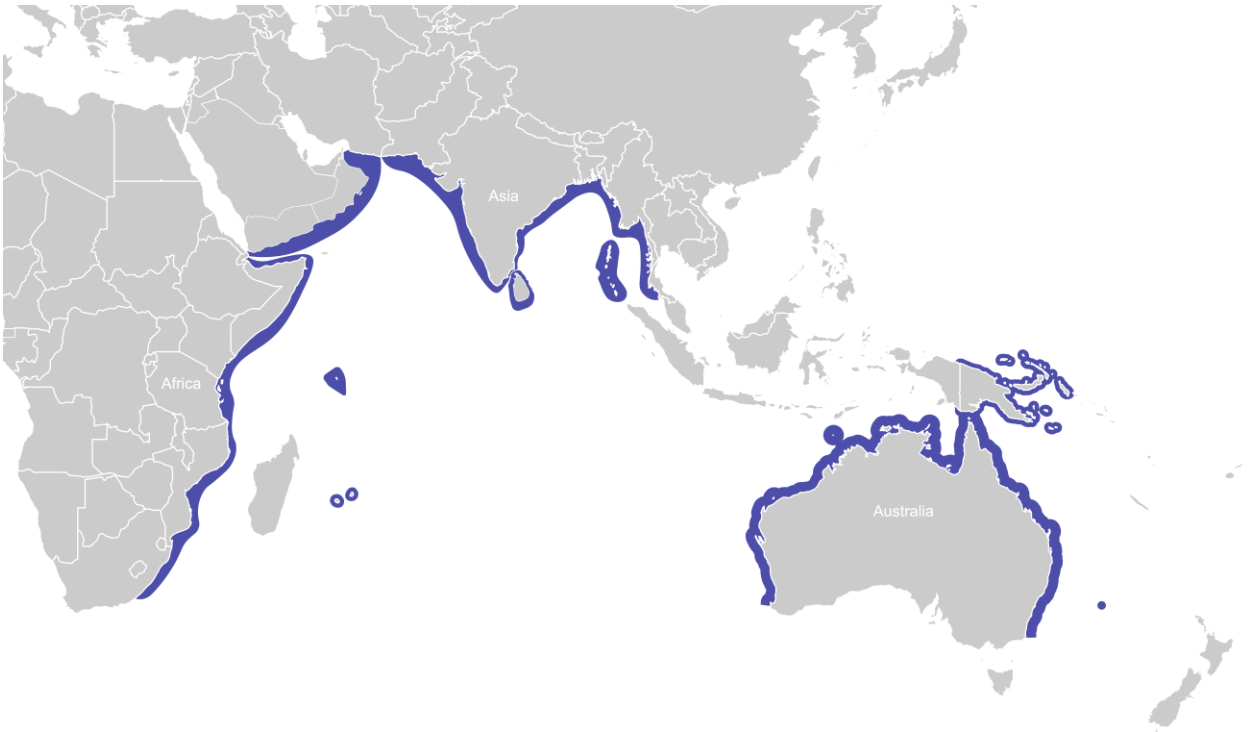


Figure 6. Distribution map of species in the *Albula vulpes* species complex. A non-specific map showing this complex can be found in Supplementary Figure 8. Individual maps for each species can be found in Supplementary Figures 9-15. A subset of this map showing only *Albula esuncula* and *Albula gilberti* may be found in Supplementary Figure 16. Please see the note on distribution maps.

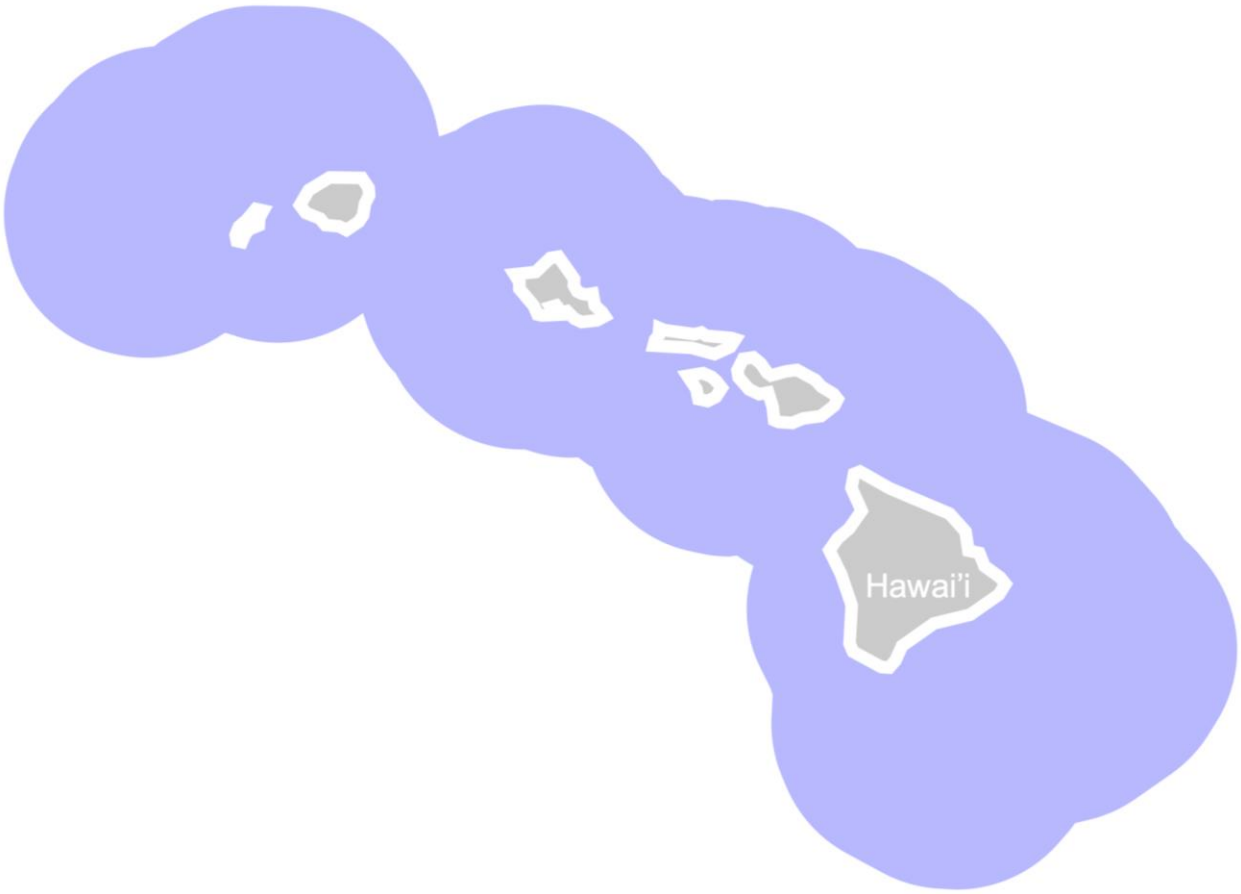
Supplementary Figure 1. Distribution map of the *Albula argentea* species complex. A map showing each of the species in this complex can be found in Figure 4. Individual maps for each species can be found in Supplementary Figures 2-4. To see how the distribution of this complex compares with other complexes, see Figure 3. Please see the note on distribution maps.



Supplementary Figure 2. Distribution map of *Albula argentea*. To see how the distribution of *Albula argentea* compares with other species in the *Albula argentea* species complex, see Figure 4. Please see the note on distribution maps.



Supplementary Figure 3. Distribution map of *Albula oligolepis*. To see how the distribution of *Albula oligolepis* compares with other species in the *Albula argentea* species complex, see Figure 4. Please see the note on distribution maps.



Supplementary Figure 4. Distribution map of *Albula virgata*. To see how the distribution of *Albula virgata* compares with other species in the *Albula argentea* species complex, see Figure 4. Please see the note on distribution maps.

Supplementary Figure 5. Distribution map of the *Albula nemoptera* species complex. A map showing each of the species in this complex can be found in Figure 5. Individual maps for each species can be found in Supplementary Figures 6 and 7. To see how the distribution of this complex compares with other complexes, see Figure 3. Please see the note on distribution maps.

Supplementary Figure 6. Distribution map of *Albula nemoptera*. To see how the distribution of *Albula nemoptera* compares with other species in the *Albula nemoptera* species complex, see Figure 5. Please see the note on distribution maps.

Supplementary Figure 7. Distribution map of *Albula pacifica*. To see how the distribution of *Albula pacifica* compares with other species in the *Albula nemoptera* species complex, see Figure 5. Please see the note on distribution maps.

Supplementary Figure 8. Distribution map of the *Albula vulpes* species complex. A map showing each of the species in this complex can be found in Figure 6. Individual maps for each species can be found in Supplementary Figures 9-15. To see how the distribution of this complex compares with other complexes, see Figure 3. Please see the note on distribution maps.

Supplementary Figure 9. Distribution map of *Albula vulpes*. To see how the distribution of *Albula vulpes* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 10. Distribution map of *Albula glossodonta*. To see how the distribution of *Albula glossodonta* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 11. Distribution map of *Albula esuncula*. View Supplementary Figure 16 to see the areas of sympatry and parapatry with *Albula gilberti*. To see how the distribution of *Albula esuncula* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 12. Distribution map of *Albula sp. cf. vulpes*. To see how the distribution of *Albula sp. cf. vulpes* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 13. Distribution map of *Albula koreana*. To see how the distribution of *Albula koreana* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 14. Distribution map of *Albula gilberti*. View Supplementary Figure 16 to see the areas of sympatry and parapatry with *Albula esuncula*. To see how the distribution of *Albula gilberti* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 15. Distribution map of *Albula goreensis*. To see how the distribution of *Albula goreensis* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

Supplementary Figure 16. Distribution map of *Albula esuncula* and *Albula gilberti*. This map shows the approximate areas of sympatry and parapatry between these two species. View Supplementary Figures 11 and 14 to see individual maps for these species. To see how the distribution of *Albula esuncula* and *Albula gilberti* compares with other species in the *Albula vulpes* species complex, see Figure 6. Please see the note on distribution maps.

REFERENCES

Abdussamad, E. M. 2017. Common Pelagic Finfish Families and their Identification. ICAR - Central Marine Fisheries Research Institute, Kerala, India.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012a. *Albula argentea*. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T194298A2310290.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012b. *Albula glossodonta*, Shortjaw Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T194299A2310398.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012c. *Albula koreana*. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T199659A2608983.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012d. *Albula nemoptera*, Caribbean Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T190357A1949274.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012e. *Albula oligolepis*, Smallscale Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T194301A2310530.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012f. *Albula virgata*, Longjaw Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T194302A2310633.

Adams, A., K. Guindon, A. Horodysky, T. Macdonald, R. Mcbride, J. Shenker, and R. Ward. 2012g. *Albula vulpes*, Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T194303A2310733.

Adams, A. J. 2016. Guidelines for evaluating the suitability of catch and release fisheries: Lessons learned from Caribbean flats fisheries. Fisheries Research. 186:672-680.

Adams, A. J. and S. J. Cooke. 2015. Advancing the science and management of flats fisheries for bonefish, tarpon, and permit. Environmental Biology of Fishes. 98:2123-2131.

Adams, A. J., A. Z. Horodysky, R. S. Mcbride, K. Guindon, J. Shenker, T. C. Macdonald, H. D. Harwell, R. Ward, and K. Carpenter. 2014. Global conservation status and research needs for tarpons (Megalopidae), ladyfishes (Elopidae) and bonefishes (Albulidae). Fish and Fisheries. 15(2):280-311.

Adams, A. J., J. M. Shenker, Z. R. Jud, J. P. Lewis, E. Carey, and A. J. Danylchuk. 2019. Identifying pre-spawning aggregation sites for bonefish (*Albula vulpes*) in the Bahamas

to inform habitat protection and species conservation. Environmental Biology of Fishes. 102(2):159-173.

Adams, A. J., R. K. Wolfe, M. D. Tringali, E. M. Wallace, and G. T. Kellison. 2008. Rethinking the status of Albula spp. biology in the Caribbean and Western Atlantic. Pages 203-214 *in* J. S. Ault, Editor. Biology and management of the world Tarpon and Bonefish fisheries.

Ault, J. S. 2008. Biology and management of the world Tarpon and Bonefish fisheries. CRC Press, Boca Raton, FL, USA.

Ault, J. S., R. Humston, M. F. Larkin, E. Perusquia, N. A. Farmer, J. Luo, N. Zurcher, S. G. Smith, L. R. Barbieri, and J. M. Posada. 2008. Population Dynamics and Resource Ecology of Atlantic Tarpon and Bonefish. Pages 217-258 *in* J. S. Ault, Editor. Biology and management of the world Tarpon and Bonefish fisheries.

Beebe, W. 1942. Eastern Pacific expeditions of the New York Zoological Society, XXX. Atlantic and Pacific fishes of the genus Dixonina. Zoologica. 27:43-48.

Bickford, D., D. J. Lohman, N. S. Sodhi, P. K. L. Ng, R. Meier, K. Winker, K. K. Ingram, and I. Das. 2007. Cryptic species as a window on diversity and conservation. Trends in Ecology & Evolution. 22(3):148-155.

Bloch, M. E. and J. G. Schneider. 1801. Systema Ichthyologiae: Iconibus CX Illustratum. Sumtibus Auctoris Impressum et Bibliopolio Sanderiano Commissum, Berlin, Germany.

Boucek, R. E., J. P. Lewis, B. D. Stewart, Z. R. Jud, E. Carey, and A. J. Adams. 2019. Measuring site fidelity and homesite-to-pre-spawning site connectivity of bonefish (*Albula vulpes*): using mark-recapture to inform habitat conservation. Environmental Biology of Fishes. 102(2):185-195.

Bowen, B. W., S. A. Karl, and E. Pfeiler. 2008. Resolving Evolutionary Lineages and Taxonomy of Bonefishes (Albula spp.). Pages 147-154 *in* J. S. Ault, Editor. Biology and management of the world Tarpon and Bonefish fisheries. CRC Press, Boca Raton, FL, USA.

Breder, C. M. J. 1948. The Bonefishes - Family Albulidae. Pages 60-61 *in* C. M. J. Breder, Editor. Field Book of Marine Fishes of the Atlantic Coast, 10 Ed. New York, NY, USA and London, England.

Breder, C. M. J. and D. E. Rosen. 1966. Order Clupeiformes. Pages 73-75. Modes of Reproduction in Fishes. The American Museum of Natural History, New York, NY, USA.

Brownscombe, J. W., A. J. Danylchuk, J. M. Chapman, L. F. G. Gutowsky, and S. J. Cooke. 2017. Best practices for catch-and-release recreational fisheries – angling tools and tactics. Fisheries Research. 186:693-705.

Brownscombe, J. W., L. P. Griffin, T. Gagne, C. R. Haak, S. J. Cooke, and A. J. Danylchuk. 2015. Physiological stress and reflex impairment of recreationally angled bonefish in Puerto Rico. Environmental Biology of Fishes. 98:2287-2295.

Brownscombe, J. W., L. P. Griffin, T. O. Gagne, C. R. Haak, S. J. Cooke, J. T. Finn, and A. J. Danylchuk. 2019. Environmental drivers of habitat use by a marine fish on a heterogeneous and dynamic reef flat. Marine Biology. 166(2):1-18.

Bunce, M., L. D. Rodwell, R. Gibb, and L. Mee. 2008. Shifting baselines in fishers' perceptions of island reef fishery degradation. Ocean & Coastal Management. 51:285-302.

Chen, J.-N., S. Samadi, and W.-J. Chen. 2015. Elopomorpha (Teleostei) as a New Model Fish Group for Evolutionary Biology and Comparative Genomics. Pages 329-344 *in* P. Pontarotti, Editor. Evolutionary Biology: Biodiversification from Genotype to Phenotype. Springer International Publishing, Switzerland.

Cissell, J. R. and M. K. Steinberg. 2019. Mapping forty years of mangrove cover trends and their implications for flats fisheries in Ciénaga de Zapata, Cuba. Environmental Biology of Fishes. 102(2):417-427.

Colborn, J., R. E. Crabtree, J. B. Shaklee, E. Pfeiler, and B. W. Bowen. 2001. The Evolutionary Enigma of Bonefishes (*Albula spp.*): Cryptic Species and Ancient Separations in a Globally Distributed Shorefish. Evolution. 55:807-820.

Colton, D. E. and W. S. Alevizon. 1983. Feeding Ecology of Bonefish in Bahamian Waters. Transactions of the American Fisheries Society. 112(2A):178-184.

Cook, K. V., R. J. Lennox, S. G. Hinch, and S. J. Cooke. 2015. Fish Out of Water: How Much Air is Too Much? Fisheries. 40:452-461.

Cuvier, G. and A. Valenciennes. 1847. Histoire naturelle des poissons, Volume 19. Levrault, Paris.

Dallas, L. J., A. D. Shultz, A. J. Moody, K. A. Sloman, and A. J. Danylchuk. 2010. Chemical excretions of angled bonefish *Albula vulpes* and their potential use as predation cues by juvenile lemon sharks *Negaprion brevirostris*. Journal of Fish Biology. 77(4):947-962.

Danylchuk, A. J., S. J. Cooke, T. L. Goldberg, C. D. Suski, K. J. Murchie, S. E. Danylchuk, A. D. Shultz, C. R. Haak, E. J. Brooks, A. Oronti, J. B. Koppelman, and D. P. Philipp. 2011. Aggregations and offshore movements as indicators of spawning activity of bonefish (*Albula vulpes*) in The Bahamas. Marine Biology. 158(9):1981-1999.

Danylchuk, A. J., S. E. Danylchuk, S. J. Cooke, T. L. Goldberg, J. B. Koppelman, and D. P. Philipp. 2008. Ecology and management of Bonefish (*Albula spp.*) in the Bahamian Archipelago. Pages 79-92 *in* J. S. Ault, Editor. Biology and management of the world Tarpon and Bonefish fisheries. CRC Press, Boca Raton, FL, USA.

Danylchuk, A. J., J. Lewis, Z. Jud, J. Shenker, and A. Adams. 2019. Behavioral observations of bonefish (*Albula vulpes*) during prespawning aggregations in the Bahamas: clues to identifying spawning sites that can drive broader conservation efforts. Environmental Biology of Fishes. 102(2):175-184.

Datovo, A. and R. P. Vari. 2014. The adductor mandibulae muscle complex in lower teleostean fishes (Osteichthyes: Actinopterygii): comparative anatomy, synonymy, and phylogenetic implications. Zoological Journal of the Linnean Society. 171:554-622.

Díaz-Viloria, N., L. Sánchez-Velasco, R. Perez-Enriquez, A. Zárate-Villafranco, M. J. Miller, and S. P. A. Jiménez-Rosenberg. 2017. Morphological description of genetically identified Cortez bonefish (*Albula gilberti*, Pfeiler and van der Heiden 2011) leptocephali from the southern Gulf of California. Mitochondrial DNA Part A. 28:717-724.

Donovan, M. K., A. M. Friedlander, K. K. Harding, E. M. Schemmel, A. Filous, K. Kamikawa, and N. Torkelson. 2015. Ecology and niche specialization of two bonefish species in Hawai‘i. Environmental Biology of Fishes. 98:2159-2171.

Donovan, M. K., A. M. Friedlander, P. Usseglio, W. Goodell, I. Iglesias, E. M. Schemmel, K. A. Stamoulis, A. Filous, J. Giddens, K. Kamikawa, H. Koike, K. Mccoy, and C. B. Wall. 2016. Effects of Gear Restriction on the Abundance of Juvenile Fishes along Sandy Beaches in Hawai‘i. PLoS ONE. 11:e0155221.

Fedler, T. 2010. The Economic Impact of Flats Fishing in The Bahamas. The Bahamian Flats Fishing Alliance.

Fedler, T. 2013. Economic Impact of the Florida Keys Flats Fishery. Bonefish and Tarpon Trust.

Filous, A., R. J. Lennox, E. E. G. Clua, and A. J. Danylchuk. 2019a. Fisheries selectivity and annual exploitation of the principal species harvested in a data-limited artisanal fishery at a remote atoll in French Polynesia. Ocean & Coastal Management. 178(1 August 2019):1-13.

Filous, A., R. J. Lennox, R. R. Coleman, A. M. Friedlander, E. E. G. Clua, and A. J. Danylchuk. 2019b. Life-history characteristics of an exploited bonefish *Albula glossodonta* population in a remote South Pacific atoll. Journal of Fish Biology. 95(2):562-574.

Forsskål, P. 1775. Descriptiones Animalium: Avium, Amphibiorum, Piscium, Insectorum, Vermium. Hauniæ.

Fowler, H. W. 1911. A new albuloid fish from Santo Domingo. Proceedings of the Academy of Natural Sciences of Philadelphia. 62:651-654.

Frezza, P. E. and S. E. Clem. 2015. Using local fishers' knowledge to characterize historical trends in the Florida Bay bonefish population and fishery. Environmental Biology of Fishes. 98:2187-2202.

Fricke, R., W. N. Eschmeyer, and R. Van Der Laan (Editors). 2019. Eschmeyer's Catalog of Fishes: Genera, Species, References. (http://researcharchive.calacademy.org/research/ichthyology/catalog/fishcatmain.asp). Electronic version accessed 15 May 2019.

Friedlander, A. M. and S. K. Rodgers. 2008. Coral Reef Fishes and Fisheries of South Moloka'i. Pages 59-66 *in* M. E. Field, S. A. Cochran, J. B. Logan, and C. D. Storlazzi, Editors. The Coral Reef of South Moloka'i, Hawai'i – Portrait of a Sediment-Threatened Fringing Reef. U.S. Department of the Interior, Reston, VA, USA.

Galdino Brandão, J. H. S., J. De Araújo Bitencourt, F. B. Santos, L. A. Watanabe, H. Schneider, I. Sampaio, and P. R. a. D. M. Affonso. 2016. DNA barcoding of coastal ichthyofauna from Bahia, northeastern Brazil, South Atlantic: High efficiency for systematics and identification of cryptic diversity. Biochemical Systematics and Ecology. 65:214-224.

Garman, S. 1899. The Fishes. Memoirs of the Museum of Comparative Zoölogy, at Harvard College, Cambridge, Mass. 24:1-431.

Griffin, L. P., C. R. Haak, J. W. Brownscombe, C. R. Griffin, and A. J. Danylchuk. 2019. A comparison of juvenile bonefish diets in Eleuthera, The Bahamas, and Florida, U.S. Environmental Biology of Fishes. 102(2):147-157.

Haak, C. R., M. Power, G. W. Cowles, and A. J. Danylchuk. 2019. Hydrodynamic and isotopic niche differentiation between juveniles of two sympatric cryptic bonefishes, *Albula vulpes* and *Albula goreensis*. Environmental Biology of Fishes. 102(2):129-145.

Hannan, K. D., Z. C. Zuckerman, C. R. Haak, and A. D. Shultz. 2015. Impacts of sun protection on feeding behavior and mucus removal of bonefish, *Albula vulpes*. Environmental Biology of Fishes. 98:2297-2304.

Hidaka, K., Y. Iwatsuki, and J. E. Randall. 2008. A review of the Indo-Pacific bonefishes of the *Albula argentea* complex, with a description of a new species. Ichthyological Research. 55:53-64.

Hidaka, K., Y. Tsukamoto, and Y. Iwatsuki. 2017. *Nemoossis*, a new genus for the eastern Atlantic long-fin bonefish *Pterothrissus belloci* Cadenat 1937 and a redescription of *P. gissu* Hilgendorf 1877 from the northwestern Pacific. Ichthyological Research. 64:45-53.

Hildebrand, S. F. 1963. Family Albulidae. Pages 132-147 *in* H. B. Bigelow, Editor. Fishes of the Western North Atlantic. Sears Foundation for Marine Research, Bingham Oceanographic Laboratory, Yale University, New Haven, Connecticut, USA.

Hollister, G. 1936. A Fish Which Grows by Shrinking. Bulletin - New York Zoological Society. 39:104-109.

Hollister, G. 1939. Young *Megalops cyprinoides* from Batavia, Dutch East India, Including a Study of the Caudal Skeleton and a Comparison with the Atlantic Species, *Tarpon atlanticus*. Zoologica. 24:449-475.

Inoue, J. G., M. Miya, K. Tsukamoto, and M. Nishida. 2004. Mitogenomic evidence for the monophyly of elopomorph fishes (Teleostei) and the evolutionary origin of the leptocephalus larva. Molecular Phylogenetics and Evolution. 32:274-286.

Jordan, D. S. and E. K. Jordan. 1922. A list of the fishes of Hawai‘i, with notes and descriptions of new species. Memoirs of the Carnegie Museum. 10:6-7.

Jörger, K. M. and M. Schrödl. 2013. How to describe a cryptic species? Practical challenges of molecular taxonomy. Frontiers in Zoology. 10(1):59.

Joshi, K. K., M. P. Sreeram, P. U. Zacharia, E. M. Abdussamad, M. Varghese, O. M. M. J. Mohamed Habeeb, K. Jayabalan, K. P. Kanthan, K. Kannan, K. M. Sreekumar, G. George, and M. S. Varsha. 2016. Check list of fishes of the Gulf of Mannar ecosystem, Tamil Nadu, India. Journal of the Marine Biological Association of India. 58:34-54.

Kamikawa, K. T., A. M. Friedlander, K. K. Harding, A. Filous, M. K. Donovan, and E. Schemmel. 2015. Bonefishes in Hawai‘i and the importance of angler-based data to inform fisheries management. Environmental Biology of Fishes. 98:2147-2157.

Kwun, H. J. and J. K. Kim. 2011. A new species of bonefish, *Albula koreana* (Albuliformes: Albulidae) from Korea and Taiwan. Zootaxa. 63:57-63.

Kwun, H. J., J. K. Kim, R. Doiuchi, and T. Nakabo. 2011. Molecular and morphological evidence for the taxonomic status of a newly reported species of *Albula* (Albuliformes: Albulidae) from Korea and Taiwan. Animal Cells and Systems. 15:45-51.

Lefcheck, J. S., B. B. Hughes, A. J. Johnson, B. W. Pfirrmann, D. B. Rasher, A. R. Smyth, B. L. Williams, M. W. Beck, and R. J. Orth. 2019. Are coastal habitats important nurseries? A meta-analysis. Conservation Letters. 12(4):e12645.

Linnaeus, C. 1758. Systema Naturæ, Volume 1, 10 Edition. Stockholm, Sweden.

Liston, S. E., P. E. Frezza, M. Robinson, and J. J. Lorenz. 2013. Assessment of Benthic Fauna Communities on Florida Keys' Shallow Banks as an Indicator of Prey Availability for Bonefish (*Albula vulpes*). Bonefish and Tarpon Trust.

Luck, C., S. Mejri, J. Lewis, P. S. Wills, M. Riche, J. Shenker, A. Adams, and M. J. Ajemian. 2019. Seasonal and spatial changes in sex hormone levels and oocyte development of bonefish (*Albula vulpes*). Environmental Biology of Fishes. 102(2):209-219.

Mejri, S., W. R. Halstead, C. A. Luck, C. Robinson, T. E. Van Leeuwen, A. J. Adams, J. Shenker, M. J. Ajemian, and P. S. Wills. 2019a. A novel attempt at artificial spawning of captive bonefish (*Albula* spp.). Aquaculture Research. 50(9):2718-2723.

Mejri, S., C. Luck, R. Tremblay, M. Riche, A. Adams, M. J. Ajemian, J. Shenker, and P. S. Wills. 2019b. Bonefish (*Albula vulpes*) oocyte lipid class and fatty acid composition related to their development. Environmental Biology of Fishes. 102(2):221-232.

Mojica, R. J., J. M. Shenker, C. W. Harnden, and D. E. Wagner. 1994. Recruitment of bonefish, *Albula vulpes*, around Lee Stocking Island, Bahamas. Fishery Bulletin. 93(4):666-674.

Moxham, E. J., P. D. Cowley, R. H. Bennett, and R. G. Von Brandis. 2019. Movement and predation: a catch-and-release study on the acoustic tracking of bonefish in the Indian Ocean. Environmental Biology of Fishes. 102(2):365-381.

Murchie, K. J., S. J. Cooke, A. J. Danylchuk, S. E. Danylchuk, T. L. Goldberg, C. D. Suski, and D. P. Philipp. 2013. Movement patterns of bonefish (*Albula vulpes*) in tidal creeks and coastal waters of Eleuthera, The Bahamas. Fisheries Research. 147:404-412.

Murchie, K. J., S. J. Cooke, A. J. Danylchuk, and C. D. Suski. 2011. Estimates of field activity and metabolic rates of bonefish (*Albula vulpes*) in coastal marine habitats using acoustic tri-axial accelerometer transmitters and intermittent-flow respirometry. Journal of Experimental Marine Biology and Ecology. 396:147-155.

Murchie, K. J., A. D. Shultz, J. A. Stein, S. J. Cooke, J. Lewis, J. Franklin, G. Vincent, E. J. Brooks, J. E. Claussen, and D. P. Philipp. 2015. Defining adult bonefish (*Albula vulpes*) movement corridors around Grand Bahama in the Bahamian Archipelago. Environmental Biology of Fishes. 98:2203-2212.

National Marine Fisheries Service, Fisheries Statistics Division. Personal Communication on 17 January 2019.

Nelson, J. S., T. C. Grande, and M. V. H. Wilson. 2016. Cohort Elopomorpha. Pages 133-153 *in* Fishes of the World, 5 Ed. John Wiley & Sons, Ltd.

Nichols, R. 2001. Gene trees and species trees are not the same. Trends in Ecology and Evolution. 16:358-364.

Nielsen, J. G., T. Monroe, T. Iwamoto, I. Harrison, W. Eschmeyer, B. Smith-Vaniz, R. Robertson, B. Collette, J. Tyler, A. Dominici-Arosemena, W. Bussing, M. Lopez, H. Molina, E. Salas, L. Sierra, and R. Viquez. 2010. *Albula esuncula*, Eastern Pacific Bonefish. The IUCN Red List of Threatened Species™. The International Union for Conservation of Nature. T178043A7489678.

Nowell, L. B., J. W. Brownscombe, L. F. G. Gutowsky, K. J. Murchie, C. D. Suski, A. J. Danylchuk, A. Shultz, and S. J. Cooke. 2015. Swimming energetics and thermal ecology of adult bonefish (*Albula vulpes*): a combined laboratory and field study in Eleuthera, The Bahamas. Environmental Biology of Fishes. 98:2133-2146.

Pamilo, P. and M. Nei. 1988. Relationships between Gene Trees and Species Trees. Molecular Biology and Evolution. 5(5):568-583.

Perez, A. U., J. J. Schmitter-Soto, A. J. Adams, and W. D. Heyman. 2019. Connectivity mediated by seasonal bonefish (*Albula vulpes*) migration between the Caribbean Sea and a tropical estuary of Belize and Mexico. Environmental Biology of Fishes. 102(2):197-207.

Pfeiler, E. 1984. Inshore migration, seasonal distribution and sizes of larval bonefish, *Albula*, in the Gulf of California. Environmental Biology of Fishes. 10(1/2):117-122.

Pfeiler, E. 2008. Resurrection of the name *Albula pacifica* (Beebe, 1942) for the shafted bonefish (Albuliformes: Albulidae) from the eastern Pacific. Revista de Biología Tropical (International Journal of Tropical Biology). 56:839-844.

Pfeiler, E., B. G. Bitler, and R. Ulloa. 2006. Phylogenetic Relationships of the Shafted Bonefish *Albula Nemoptera* (Albuliformes: Albulidae) from the Eastern Pacific Based on Cytochrome B Sequence Analyses. Copeia. 2006:778-784.

Pfeiler, E., B. G. Bitler, R. Ulloa, A. M. Van Der Heiden, and P. A. Hastings. 2008a. Molecular Identification of the Bonefish *Albula esuncula* (Albuliformes: Albulidae) from the Tropical Eastern Pacific, with Comments on Distribution and Morphology. Copeia. 2008:763-770.

Pfeiler, E., J. Colborn, M. R. Douglas, and M. E. Douglas. 2002. Systematic status of bonefishes (*Albula spp.*) from the eastern Pacific Ocean inferred from analyses of allozymes and mitochondrial DNA. Environmental Biology of Fishes. 63:151-159.

Pfeiler, E., M. A. Mendoza, and F. A. Manrique. 1988. Premetamorphic bonefish (*Albula sp.*) leptocephali from the Gulf of California with comments on life history. Environmental Biology of Fishes. 21(4):241-249.

Pfeiler, E., A. M. Van Der Heiden, R. S. Ruboyianes, and T. Watts. 2011. *Albula gilberti*, a new species of bonefish (Albuliformes: Albulidae) from the eastern Pacific, and a description of adults of the parapatric *A. esuncula*. Zootaxa. 3088(1):1-14.

Pfeiler, E., T. Watts, J. Pugh, and A. M. Van Der Heiden. 2008b. Speciation and demographic history of the Cortez bonefish, *Albula sp. A* (Albuliformes: Albulidae), in the Gulf of California inferred from mitochondrial DNA. Journal of Fish Biology. 73:382-394.

Posada, J. M., D. Debrot, and C. Weinberger. 2008. Aspects of the Biology and Recreational Fishery of Bonefish (*Albula vulpes*) from Los Roques Archipelago National Park, Venezuela. Pages 103-114 *in* J. S. Ault, Editor. Biology and management of the world Tarpon and Bonefish fisheries. CRC Press, Boca Raton, FL, USA.

Raby, G. D., J. R. Packer, A. J. Danylchuk, and S. J. Cooke. 2014. The understudied and underappreciated role of predation in the mortality of fish released from fishing gears. Fish and Fisheries. 15:489-505.

Rasquin, P. 1955. Observations on the metamorphosis of the bonefish, *Albula vulpes* (Linnaeus). Journal of Morphology. 97:77-117.

Rehage, J. S., R. O. Santos, E. K. N. Kroloff, J. T. Heinen, Q. Lai, B. D. Black, R. E. Boucek, and A. J. Adams. 2019. How has the quality of bonefishing changed over the past 40 years? Using local ecological knowledge to quantitatively inform population declines in the South Florida flats fishery. Environmental Biology of Fishes. 102(2):285-298.

Reist, J. D., M. Power, and J. B. Dempson. 2013. Arctic charr (*Salvelinus alpinus*): a case study of the importance of understanding biodiversity and taxonomic issues in northern fishes. Biodiversity. 14(1):45-56.

Rennert, J. J., J. M. Shenker, J. A. Angulo-Valdés, and A. J. Adams. 2019. Age, growth, and age at maturity of bonefish (Albula species) among Cuban habitats. Environmental Biology of Fishes. 102:253-265.

Rivas, L. R. and S. M. Warlen. 1967. Systematics and biology of the bonefish *Albula Nemoptera* (Fowler). Fishery Bulletin U.S. Fish and Wildlife Services. 66:251-258.

Robins, C. R. and G. C. Ray. 1986. Bonefishes: Family Albulidae. Page 48 *in* C. R. Robins and G. C. Ray, Editors. A Field Guide to Atlantic Coast Fishes of North America. Houghton Mifflin Company, Boston, MA, USA.

Santos, R. O., J. S. Rehage, E. K. N. Kroloff, J. E. Heinen, and A. J. Adams. 2019a. Combining data sources to elucidate spatial patterns in recreational catch and effort: fisheries-dependent data and local ecological knowledge applied to the South Florida bonefish fishery. Environmental Biology of Fishes. 102:299-317.

Santos, R. O., R. Schinbeckler, N. Viadero, M. F. Larkin, J. J. Rennert, J. M. Shenker, and J. S. Rehage. 2019b. Linking bonefish (*Albula vulpes*) populations to nearshore estuarine habitats using an otolith microchemistry approach. Environmental Biology of Fishes. 102:267-283.

Scott, W. B. and M. G. Scott. 1988. Family Albulidae / Bonefishes. Page 74. Atlantic Fishes of Canada. University of Toronto Press, Toronto, Canada.

Seyoum, S., E. M. Wallace, and M. D. Tringali. 2008. PERMANENT GENETIC RESOURCES: Twelve polymorphic microsatellite markers for the bonefish, *Albula vulpes* and two congeners. Molecular ecology resources. 8:354-356.

Shaklee, J. B. and C. S. Tamaru. 1981. Biochemical and Morphological Evolution of Hawaiian Bonefishes (*Albula*). Systematic Zoology. 30:125.

Song, H., J. E. Buhay, M. F. Whiting, and K. A. Crandall. 2008. Many species in one: DNA barcoding overestimates the number of species when nuclear mitochondrial pseudogenes are coamplified. Proceedings of the National Academy of Sciences. 105(36):13486-13491.

Stamatakis, A. 2014. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics. 30(9):1312-1313.

Steinberg, M. K. 2015. A nationwide assessment of threats to bonefish, tarpon, and permit stocks and habitat in Belize. Environmental Biology of Fishes. 98:2277-2285.

Sweetman, B. M., J. R. Foley, and M. K. Steinberg. 2019. A baseline analysis of coastal water quality of the port Honduras marine reserve, Belize: a critical habitat for sport fisheries. Environmental Biology of Fishes. 102:429-442.

Szekeres, P. 2017. Exploring the Behaviour and Physiology of Juvenile Bonefish (*Albula vulpes*): Fundamental and Applied Perspectives. Master's thesis. Carleton University, Ottawa, Ontario, Canada.

Szekeres, P., J. W. Brownscombe, F. Cull, A. J. Danylchuk, A. D. Shultz, C. D. Suski, K. J. Murchie, and S. J. Cooke. 2014. Physiological and behavioural consequences of cold shock on bonefish (*Albula vulpes*) in The Bahamas. Journal of Experimental Marine Biology and Ecology. 459:1-7.

Szekeres, P., A. Wilson, C. Haak, A. Danylchuk, J. Brownscombe, C. Elvidge, A. Shultz, K. Birnie-Gauvin, and S. Cooke. 2017. Does coastal light pollution alter the nocturnal behavior and blood physiology of juvenile bonefish (*Albula vulpes*)? Bulletin of Marine Science. 93:491-505.

Taylor, A. T., J. M. Long, M. D. Tringali, and B. L. Barthel. 2019. Conservation of Black Bass Diversity: An Emerging Management Paradigm. Fisheries. 44(1):20-36.

Trontelj, P. and C. Fišer. 2009. Perspectives: Cryptic species diversity should not be trivialised. Systematics and Biodiversity. 7(1):1-3.

Valdez-Moreno, M., L. Vásquez-Yeomans, M. Elías-Gutiérrez, N. V. Ivanova, and P. D. N. Hebert. 2010. Using DNA barcodes to connect adults and early life stages of marine fishes from the Yucatan Peninsula, Mexico: potential in fisheries management. Marine and Freshwater Research. 61:655.

Vásquez-Yeomans, L., E. Sosa-Cordero, M. R. Lara, A. J. Adams, and J. A. Cohuo. 2009. Patterns of distribution and abundance of bonefish larvae *Albula spp.* (Albulidae) in the western Caribbean and adjacent areas. Ichthyological Research. 56(3):266-275.

Wallace, E. M. 2014. Assessing Biodiversity, Evolution, and Biogeography in Bonefishes (Albuliformes): Resolving Relationships and Aiding Management. Doctoral dissertation. University of Minnesota, St. Paul, MN, USA.

Wallace, E. M. 2015. High intraspecific genetic connectivity in the Indo-Pacific bonefishes: implications for conservation and management. Environmental Biology of Fishes. 98:2173-2186.

Wallace, E. M. and M. D. Tringali. 2010. Identification of a novel member in the family Albulidae (bonefishes). Journal of Fish Biology. 76:1972-1983.

Wallace, E. M. and M. D. Tringali. 2016. Fishery composition and evidence of population structure and hybridization in the Atlantic bonefish species complex (*Albula spp.*). Marine Biology. 163:142.

Warmke, G. L. and D. S. Erdman. 1963. Records of marine mollusks eaten by bonefish in Puerto Rican waters. The Nautilus. 76(4):115-120.

Whitehead, P. J. P. 1986. The Synonymy of *Albula vulpes* (Linnaeus, 1758) (Teleostei, Albulidae). Cybium. 10:211-230.

Whitehead, P. J. P. 1990. Albulidae. Pages 122-124 *in* J. C. Quéro, J. C. Hureau, C. Karrer, A. Post, and L. Saldanha, Editors. Check-list of the fishes of the eastern tropical Atlantic. UNESCO, Paris, France.

Zeng, X., A. Adams, M. Roffer, and R. He. 2019. Potential connectivity among spatially distinct management zones for Bonefish (*Albula vulpes*) via larval dispersal. Environmental Biology of Fishes. 102:233-252.

CHAPTER 2

Genome Assembly of the Roundjaw Bonefish (*Albula glossodonta*), a Vulnerable Circumtropical Sportfish

Brandon D. Pickett[1*], Sheena Talma[2*], Jessica R. Glass[3], Daniel Ence[4], Paul D. Cowley[3], Perry G. Ridge[1], John S. K. Kauwe[1,5]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[2]*Department of Ichthyology and Fisheries Science, Rhodes University, Makhanda, South Africa*

[3]*South African Institute for Aquatic Biodiversity, Makhanda, South Africa*

[4]*School of Forest Resources and Conservation, University of Florida, Gainesville, Florida, USA*

[5]*Brigham Young University - Hawai'i, Laie, Hawai'i, USA*

[*]*These authors contributed equally to this work*

ABSTRACT

Background: Bonefishes are cryptic species indiscriminately targeted by subsistence and recreational fisheries worldwide. The roundjaw bonefish, *Albula glossodonta* is the most widespread bonefish species in the Indo-Pacific and is listed as vulnerable to extinction by the IUCN's Red List due to anthropogenic activities. Whole-genome datasets allow for improved population and species delimitation, which – prior to this study – were lacking for *Albula* species.

Results: We generated a high-quality genome assembly of an *A. glossodonta* individual from Hawai'i, USA. The assembled contigs had an NG50 of 4.75 Mbp and a maximum length of 28.2 Mbp. Scaffolding yielded an NG50 of 14.49 Mbp, with the longest scaffold reaching 42.29 Mbp. Half the genome was contained in 20 scaffolds. The genome was annotated with 28.3 K protein-coding genes. We then analyzed 66 *A. glossodonta* individuals and 38,355 SNP loci to evaluate population genetic connectivity between six atolls in Seychelles and Mauritius in the Western Indian Ocean. We observed genetic homogeneity between atolls in Seychelles and evidence of reduced gene flow between Seychelles and Mauritius. The South Equatorial Current could be one mechanism limiting gene flow of *A. glossodonta* populations between Seychelles and Mauritius.

Conclusions: Quantifying the spatial population structure of widespread fishery species such as bonefishes is necessary for effective transboundary management and conservation. This population genomic dataset mapped to a high-quality genome assembly allowed us to discern shallow population structure in a widespread species in the Western Indian Ocean. The genome assembly will be useful for addressing the taxonomic uncertainties of bonefishes globally.

INTRODUCTION

Bonefishes (*Albula* spp.) are popular and economically important sportfishes found in the tropics around the globe. In the Florida Keys (Florida, USA) alone, $465 million of the annual economy is attributed to sportfishing tourism for bonefish and other fishery species inhabiting coastal flats [1]. Considering only bonefish, the sportfishing industry generates $169 million annually in the Bahamas [2, 3]. Unfortunately, population declines of bonefish have been observed around the globe, raising questions about how best to conserve bonefish and manage the associated fisheries [4]. *Albula* contains many morphological cryptic species, which, when combined with baseline data gaps, creates a significant hurdle to effective management [5-7].

All bonefish species were historically synonymized to a single species, *Albula vulpes* (Linnaeus 1758) [8], by 1940 [9-11], except for the threadfin bonefish, *A. nemoptera* (Fowler 1911) [12], which is morphologically distinct [12, 13]. Molecular testing in the last several decades has enabled specific distinctions that were not previously possible [6, 9, 14-16]. Presently, three species complexes (*A. argentea*, *A. nemoptera*, and *A. vulpes* complexes) contain the twelve putative albulid species, although identification remains difficult in most cases [4]. The roundjaw bonefish (Fig. 1), *A. glossodonta* (Forsskål 1775) [17], is one of seven species in the *A. vulpes* complex.

Most of the species in the *A. vulpes* complex can be found in the Caribbean Sea and Atlantic Ocean. By contrast, *A. glossodonta* can be found throughout the Indian and Pacific Oceans; this range overlaps slightly with *A. koreana* (Kwun and Kim 2011) [18] from the *A. vulpes* complex and drastically with each species in the *A. argentea* complex [4]. *Albula glossodonta* may be distinguished genetically from other species, but morphological identification based on its more-rounded jaw and larger average size is difficult for non-experts

[4, 19]. This difficulty, alongside underregulated fisheries and anthropogenic habitat loss, poses significant threats to the future of this species. In point of fact, *A. glossodonta* has been evaluated as "Vulnerable" on the International Union for the Conservation of Nature's (IUCN) Red List of Threatened Species™ [7], and several incidents of overexploitation, including regional extirpation, have been reported [20-24].

The threat to *A. glossodonta* and other bonefish species will persist unless identification is made easier and population genomics techniques are employed to understand and identify evolutionarily significant units, areas of overlap between species, presence and extent of hybridization, and life-history traits, especially migration and spawning [4]. Genetic identification has hitherto been accomplished using only a portion of the mitochondrial cytochrome b gene and some microsatellite markers [6, 9, 15, 18, 25-32], which likely provide an insufficient taxonomic history [4, 33-35]. To contribute to a more robust capacity for identification and enable more complex genomics-based analyses, we present a high-quality genome assembly of an *A. glossodonta* individual. A transcriptome assembly was also created and was used alongside computational annotation methods to create structural and functional annotations for the genome assembly. Additionally, we present results from a population genomic analysis of *A. glossodonta* populations in Seychelles and Mauritius, two island nations that support lucrative bonefish fly fishing industries. The raw data, assembly, and annotations are available on the National Center for Biotechnology Information (NCBI) website under BioProject Accessions PRJNA668352 and PRJNA702254.

METHODS

An overview of the methods used in this study is provided here. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided in the Supplementary Bioinformatics Methods [see Additional File 1].

Tissue Collection and Preservation

Blood, gill, heart, and liver tissues from one *A. glossodonta* individual were collected off the coast of Molokaʻi (near Kaunakakai, Hawaiʻi, USA) in February 2016. Heart tissue from a second individual was also collected at the same location in September 2017. Tissue samples were flash-frozen in liquid nitrogen, and blood samples were preserved in EDTA. All samples were packaged in dry ice for transportation to Brigham Young University (BYU; Provo, Utah, USA) and stored at -80°C until sequencing. The blood sample from the first individual was used for short-read DNA sequencing. The gill, heart, and liver samples from the same individual were used for short-read RNA sequencing. The heart sample from the second individual was used for long-read sequencing and Hi-C sequencing.

For population genomic analyses, tissues (dorsal muscle samples or fin clips) were collected by fly fishing charter operators from 96 individuals of *A. glossodonta* from six coral atolls in the Southwest Indian Ocean (SWIO; Fig 1; Table S1 [Additional File 2]). All tissues were preserved in 95% EtOH at -20℃ until sequencing, and thereafter cataloged and preserved in -80℃ in the tissue biobank of South African Institute for Aquatic Biodiversity (Makhanda, South Africa) [36].

Sequencing

DNA Sequencing

DNA was prepared for long-read sequencing with Pacific Biosciences (PacBio; Menlo Park, California, USA) [37] SMRTbell Library kits, following the protocol "Procedure & Checklist – Preparing >30 kb SMRTbell Libraries Using Megaruptor Shearing and BluePippin Size-Selection for PacBio RS II and Sequel Systems". Continuous long-read (CLR) sequencing was performed on thirteen SMRT cells for a 10-hour movie on the PacBio Sequel at the BYU DNA Sequencing Center (DNASC) [38], a PacBio Certified Service Provider. Short-read sequencing was performed in Rapid Run mode for 250 cycles in one lane on the Illumina (San Diego, California, USA) [39] Hi-Seq 2500 at the DNASC after sonication with Covaris (Woburn, Massachusetts, USA) [40] Adaptive Focus Acoustics technology and preparation with New England Biolabs (Ipswich, Massachusetts, USA) [41] NEBNext Ultra II End Repair and Ligation kits with adapters from Integrated DNA Technologies (Coralville, Iowa, USA) [42].

mRNA Sequencing

RNA was prepared with Roche (Basel, Switzerland) [43] KAPA Stranded RNA-Seq kit, following manufacturer recommendations. Paired-end sequencing was performed in High Output mode for 125 cycles on the three samples together in one lane on the Illumina Hi-Seq 2500 at the DNASC.

Hi-C Sequencing

DNA was prepared with Phase Genomics (Seattle, Washington, USA) [44] Proximo Hi-C Kit (Animal) using the Sau3AI restriction enzyme (cut site: GATC) following recommended protocols. Paired-end sequencing was performed in Rapid Run mode for 250 cycles in one lane on the Illumina Hi-Seq 2500 at the DNASC.

ddRAD Library Preparation and Sequencing

We employed double digest restricted site-associated (ddRAD) sequencing to measure intraspecific genetic variation across six sampling localities in the SWIO. We extracted total DNA using Qiagen DNeasy Tissue kits per the manufacturer's protocol (Qiagen, Inc., Valencia, California, USA) [45]. We examined the quality of DNA extractions visually using gel electrophoresis and by quantifying isolated DNA using a Qubit fluorometer (Life Technologies, Carlsbad, California, USA) [46].

We modified a protocol developed by Peterson et al. [47] to prepare samples for ddRAD sequencing. We used the rare cutter *PstI* (5′-CTGCAG-3′ recognition site) and common cutter *MspI* (5′-CCGG-3′ recognition site). We carried out double digests of 150 – 200 ng total DNA per sample using the two enzymes in the manufacturer's supplied buffer (New England Biolabs) for 8 hours at 37℃. We randomly distributed samples from different localities across the sequencing plate to minimize bias during library preparation. We visually examined samples using gel electrophoresis to determine digestion success and then ligated barcoded Illumina adapters to DNA fragments [47]. After ligation, we pooled samples into 12 libraries and performed a clean-up using the QIAquick PCR Purification Kit. We then performed PCR using Phusion *Taq* (New England Biolabs) and Illumina indexed primers [47]. Library DNA concentration was checked using a Qubit fluorometer, followed by normalization, a second round of pooling into four libraries, and an additional QIAquick cleanup step. We then re-measured DNA concentration using a Qubit and combined equal amounts from each of the four pools into one. We analyzed this final pool using a BioAnalyzer (Agilent, Santa Clara, California, USA) [48] and performed size-selection using a Pippin Prep (Sage Science, Beverly, Massachusetts, USA) [49], selecting for fragments between 300 – 500 bp. This was followed by a final measure of concentration using a BioAnalyzer. We sent the library to the University of

Oregon Genomics and Cell Characterization Core Facility [50] where concentrations were verified via qPCR before 100 bp single-end sequencing on an Illumina Hi-Seq 4000.

Read Error Correction

Illumina DNA

Illumina whole-genome sequencing (WGS) reads were corrected using Quake v0.3.5 [51], which depended upon old versions of R (v3.4.0) [52] and the R package VGAM (v0.7-8) [53, 54]. Quake attempts to automatically choose a k-mer cutoff, traditionally based on k-mer counts provided by Jellyfish [55]. To generate q-mer counts instead of k-mer counts, BFCounter v0.2 [56] was used. Quake suggested a q-mer cutoff of 2.33, which was subsequently used by the correction phase of Quake. Unlike the WGS reads, the Illumina DNA reads created with the Hi-C library preparation were not corrected.

Illumina RNA

Illumina RNA-seq reads underwent a correction procedure using Rcorrector v1.0.2 [57]. Rcorrector automatically chooses a k-mer cutoff based on k-mer counts provided by Jellyfish [55], which Rcorrector runs automatically for the user. Alternately, Jellyfish can be run externally or bypassed with an alternate k-mer counting program, and counts can subsequently be provided to Rcorrector, which may be started at what it calls “stage 3”. We bypassed Jellyfish by using BFCounter v0.2 [56] to count k-mers. Note that Rcorrector made no changes to the reads.

PacBio CLRs

Several methods were attempted for the correction of the PacBio CLRs. The corrected reads from each method that did not fail were assembled, and the assembly results were used to

choose the correction strategy. Ultimately, a hybrid correction strategy was employed. First, the reads were self-corrected using Canu v1.6 [58]. Second, the self-corrected reads were further corrected using Illumina short-reads (previously corrected with Quake) using CoLoRMap downloaded April 2018 [59].

Genome Size Estimation

Genome size was estimated using a k-mer analysis on the corrected Illumina WGS reads. First, the k-mer coverage was estimated using ntCard v1.0.1 [60]. The k-mer coverage histogram was computationally processed to calculate the area under the curve and identify the peak to determine genome size according to the following equation: $a / p = s$, where a is the area under the curve, p is the number of times the k-mers occur (the x-value) at the peak, and s is the genome size.

Genome Assembly, Polishing, and Scaffolding

Multiple assemblies were generated from various correction strategies. The final assembly was based on a hybrid correction strategy as described in the previous section "PacBio CLRs". The assembly was created using Canu v1.6 [58]. The assembly underwent two rounds of polishing with the corrected Illumina WGS reads using RaCon v1.3.1 [61]. The polished contigs were scaffolded in a stepwise fashion using two types of long-range information: Hi-C and RNA-seq reads. Both scaffolding steps required read mapping to the contigs before determining how to order and orient contigs. The Hi-C data alignments were performed following the Arima Genomics (San Diego, California, USA) [62] Mapping Pipeline [63], which relied on bwa v0.7.17-r1998 [64], Picard v2.19.2 [65], and SAMtools v1.6 [66]. BEDTools v2.28.0 [67] was

used to prepare the Hi-C alignments for scaffolding. The RNA-seq data were aligned using HiSat v0.1.6-beta [68]. Scaffolding was performed for the Hi-C and RNA-seq data, respectively, with SALSA, downloaded 29 May 2019 [69, 70], and Rascaf, downloaded June 2018 [71]. Assembly continuity statistics, e.g., N50 and auN [72], were calculated with caln50 downloaded 10 April 2020 [73] and a custom Python [74] script. Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 [75] and OrthoDB v10 [76] (Table S2 [Additional File 2]).

Transcriptome Assembly

The transcriptome was assembled from Illumina RNA-seq reads from all three tissues (i.e., gill, heart, and liver). The raw reads were used because Rcorrector did not modify any bases, thus making the raw reads and the "corrected" reads identical. The transcripts were assembled using Trinity v2.6.6 [77]. Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 [75] and OrthoDB v10 [76] (Table S2 [Additional File 2]).

ddRAD Sequence Assembly and Filtering

We assembled all ddRAD data using the program *ipyrad* v0.9.31 [78]. The input parameters for *ipyrad* are included in the supplementary materials (Table S3 [Additional File 2]). All *A. glossodonta* reads were mapped to the genome assembly. In step one of the *ipyrad* workflow, we demultiplexed sequences by identifying individual sample barcode sequences and restriction overhangs. During step two, we trimmed barcodes and adapters from reads, which were then hard-masked using a *q*-score threshold of 20 and filtered for a maximum number of undetermined bases per read. In step three we clustered reads with a minimum depth of coverage of six to retain clusters in the ddRAD assembly. During step four, we jointly estimated sequencing error rate and heterozygosity from site patterns across the clustered reads assuming a

maximum of two consensus alleles per individual. In step five, we determined consensus base calls for each allele using the parameters from step four and filtered for a maximum number of undetermined sites per locus. During step six, we clustered consensus sequences and aligned reads for each sample. During step seven, we filtered the data by the maximum number of alleles per locus, the maximum number of shared heterozygous sites per locus, and other criteria [78] and formatted output files for downstream analyses. We included all loci shared by at least 10 individuals.

We performed additional filtering steps after running *ipyrad* to account for missing data and rare alleles. Using VCFtools v0.1.16 [79] and BCFtools v1.6 [66], we removed individuals missing more than 98% of genotype calls. We retained only biallelic single nucleotide polymorphisms (SNPs) and removed (i) indels, (ii) loci with minor allele frequencies < 0.05 to exclude singletons and false polymorphic loci due to potential sequencing errors, (iii) alleles with a minimum count < 2, and (iv) loci with high mean depth values (> 100). We then implemented an iterative series of filtering steps based on missing data and genotype call rates to maximize genomic coverage per individual (Table S4 [Additional File 2]) [80]. Thereafter, we removed loci out of Hardy-Weinberg Equilibrium to filter for excess heterozygosity. We then used PLINK v1.9 [81] to perform linkage disequilibrium pruning by calculating the squared coefficient of correlation (r^2) on all SNPs within a 1 kb window [82]. We removed all SNPs with an r^2 value greater than 0.6.

Computational Annotation of Assembled Genome

The MAKER v3.01.02-beta [83] pipeline was used to annotate the assembly. With minor modifications (see Supplementary Bioinformatics Methods, Additional File 1), annotation proceeded according to the process described in the most recent Maker Wiki tutorial [84]. A

custom repeat library was created using RepeatModeler v1.0.11 [85]. The transcriptome assembly, genome assembly, and proteins from UniProtKB Swiss-prot [86, 87] were used as input to MAKER to create initial annotations. Gene models based on these annotations were used to train the following *ab initio* gene predictors: AUGUSTUS v3.3.2 [88, 89] and SNAP downloaded 3 June 2019 [90]. AUGUSTUS was trained using BUSCO [75] as a wrapper; SNAP was trained without a wrapper. Genemark-ES v4.38 [91-93] was also trained on the assembled genome. These models were all provided to MAKER for a second round of structural annotation. The gene models based on those annotations were filtered with gFACs v1.1.1 [94] and again provided to AUGUSTUS and SNAP. As Genemark-ES does not accept initial gene models, it did not need to be run again. The gene models from the *ab initio* gene predictors were again provided to MAKER for a third and final round of annotation. Functional annotations were added using MAKER accessory scripts, the BLAST+ Suite v2.9.0 [95, 96], and InterProScan v5.45-80.0 [97, 98]. The annotations in GFF3 format were validated with GenomeTools v1.6.1 [99] and manually curated to adhere to GenBank submission guidelines.

Statistical Analysis of Population Genomic Data

Detection of Loci under Selection

Before conducting population genomic analyses, we performed outlier tests to identify loci putatively under selection, which are generally identified by a significant difference in allele frequencies between populations [100]. Specifically, we implemented two outlier detection methods that accommodate missing data: *pcadapt* v4.1.0 [100] and BayeScan v2.1 [101]. The assumption behind *pcadapt* is that loci associated with population structure, ascertained via principal component analysis (PCA), are under selection [100]. *pcadapt* is advantageously fast and able to handle large numbers of loci. The number of principal components (K) was chosen

based on visualization of a scree plot of the eigenvalues of a covariance matrix. Once the *K*-value was chosen, the Mahalanobis distance (*D* test statistic) was calculated using multiple linear regression of the number of SNPs versus *K* [100, 102]. To account for false discovery rates, the *p*-values generated using the Mahalanobis distance *D* were transformed to q-values using the R v3.6.3 [52] *q-value* package v2.15.0 [103] with the cut-off point (α) set to 10% (0.1).

BayeScan measures allele frequencies between different populations and identifies loci that are perceived to be undergoing natural selection based on their F_{ST} values [104, 105]. The method applies linear regression to generate population- and locus-specific F_{ST} estimates and calculates subpopulation F_{ST} coefficients by taking the difference in allele frequency between each population and the common gene pool. BayeScan incorporates uncertainties in allele frequencies due to small sample sizes, as well as imbalances in the number of samples between populations [101]. We assigned each of the six sampling localities as a population. Our analyses were based on 1:50 prior odds and included 100,000 iterations and a false discovery rate of 10%. We used the default values for the remaining parameters and visualized results in R v3.6.3 following the developer's manual [106]. After running both *pcadapt* and BayeScan, we used R to assess the number of outliers identified by both programs and subsequently removed outlier loci to generate a neutral dataset for downstream analyses.

Population Structure and Genetic Differentiation

To examine population structure, we used a model-based clustering method to reconstruct the genetic ancestry of individuals using sparse nonnegative matrix factorization (sNMF) and least-squares optimization. Model-based analyses were performed using the package *LEA* v2.6.0 [107] in R. The sNMF function in *LEA* estimates the number of ancestral populations and the probability of the number of gene pools from which each individual originated by calculating an

ancestry coefficient and investigating the model's fit through cross-entropy criterion [108]. We calculated and visualized cross-entropy scores of *K* population clusters ranging from 1–10 with 10 replicates. To complement sNMF, we also used principal component analysis (PCA), a distance-based approach based on variation in allele distributions, implemented in VCFtools v0.1.16 [79]. For sNMF and PCA analyses, no populations were assigned *a priori*. We assigned each of the six sampling localities as populations for subsequent visualization, grouped into four "island groups" based on the proximity of some of the atolls that comprised our sampling localities (Fig. 2). The five Seychelles atolls we sampled were spread amongst three separate clusters of islands that are commonly referred to as the "outer island groups" due to the geographic locations of these outlying coralline islands relative to the densely-populated, granitic "inner islands" of the Seychelles Archipelago. The island groups consisted of (i) Amirantes (St. Joseph's Atoll), (ii) Farquhar (Farquhar and Providence Atolls), (iii) Aldabra (Aldabra and Cosmoledo Atolls), as well as (iv) Mauritius (St. Brandon's Atoll; Table S1 [Additional File 2]). We computed summary statistics in R v3.6.3, including pairwise F_{ST} estimates (StAMPP v1.6.1 [109]), isolation by distance via the Mantel Rand test (adegenet v2.1.3 [110]), and expected and observed heterozygosity (hierfstat v0.5-7 [111]) to compare genetic diversity and differentiation between the four island groups.

RESULTS

Sequencing

DNA Sequencing

Paired-end, short-read sequencing (Illumina) yielded 109.5M pairs of reads comprised of 53.86Gbp. The mean and N50 read lengths were 245.981 and 250, respectively. Continuous

long-read sequencing (PacBio) generated 9.5M reads with a total of 69.85Gbp. The mean and N50 read lengths were 7,352.726 and 13,831, respectively. The longest read was 103,889bp. The read length distribution is plotted in Figure 2. Result summaries for both sequencing runs are available in Table 1.

mRNA Sequencing

RNA-seq from the three tissues (i.e., gill, heart, and liver) generated 270.7M pairs of reads totaling 67.2Gbp. The gill tissue yielded 107.7M pairs of reads, with a total of 26.7Gbp. The heart tissue generated 19.6Gbp across 78.8M pairs of reads. The 84.2M pairs of reads from the liver tissue were comprised of 20.9Gbp. Across all three tissues, the mean and N50 read lengths were 124.122 and 125, respectively. The combined results from all three tissues are summarized in Table 1.

Hi-C Sequencing

Sequencing yielded 88.7M pairs of reads comprised of 44.28Gbp. The mean and N50 read lengths were 249.493 and 250, respectively. A summary of these results is presented in Table 1.

ddRAD sequencing

After data processing using *ipyrad*, we recovered a mean of 114,324 reads per individual for *A. glossodonta* and an average of 107,105 loci per individual. Following filtering for missing data, minor allele frequency, and linkage disequilibrium, the dataset contained 66 individuals and 38,355 SNPs. BayeScan, being a more conservative outlier detection method than *pcadapt*, did not identify any outliers; we thus used only outlier detection results from *pcadapt*. Subsequent

removal of *pcadapt* outliers (N = 155) resulted in a neutral dataset containing 38,200 SNPs with 9% missing data.

Read Error Correction

Illumina DNA

When Quake corrects paired-end reads, three outcomes are possible for each pair of reads: (i) both reads are either already correct or correctable, (ii) one read is either correct or correctable and the other is low-quality, or (iii) both reads are low-quality. Of the original 218.96M reads (109.5M pairs of reads), Quake corrected 62.7M of them and removed 51.6M of them. 5.97M pairs of reads were discarded because both reads were rated as erroneous. 39.6M pairs of reads had one read that was correct or correctable and one read that was low-quality; these were also discarded. The remaining 63.88M pairs of reads were either correct or correctable and were kept in the final read set containing 29.11Gbp of sequence.

Illumina RNA

No corrections were made to the RNA-seq reads by the error correction software.

PacBio CLRs

The dual-correction strategy (self-correction followed by hybrid-correction) reduced the number of reads from 9.5M to 2.79M and the total number of bases from 69.85Gbp to 36.79Gbp. The mean and N50 read lengths were changed from 7,354 and 13,831 to 13,193 and 15,483, respectively. The longest read was 63,271 bases. The distribution of read lengths can be viewed in Fig. 3.

Genome Size Estimation

The genome size was estimated to be approximately 0.933Gbp as a result of the k-mer analysis, which was consistent with the authors' expectations based on two closely related elopomorph species [112, 113].

Genome Assembly, Polishing, and Scaffolding

The initial assembly from Canu was comprised of 3.8K contigs with a total assembly size of 1.05Gbp. The mean contig length, N50, NG50, and maximum contig length were 276.2Kbp, 3.6Mbp, 4.7Mbp, and 28.2Mbp, respectively. The L50 was 57, and the LG50 was 43. The auN was 8.17M. After two rounds of polishing these contigs with the corrected Illumina WGS reads using RaCon, the assembly statistics changed only marginally. The number of contigs, L50, and LG50 were unchanged. The assembly size decreased by 318.7Kbp (0.03%). The mean contig length, N50, NG50, and maximum contig length were reduced by 83.8bp (0.03%), 1.3Kbp (0.04%), 1.5Kbp (0.03%), and 3.8Kbp (0.01%), respectively. The auN decreased by 2Kbp (0.02%).

The scaffolding with the Hi-C data joined some polished contigs together, reducing the sequence count to 3.6K (-4.69%). The number of bases, excluding unknown bases (Ns), was unchanged; however, it is important to note that when SALSA creates gaps while ordering and orienting contigs, it always uses a gap size of 500bp. The result, in this case, was adding 116Kbp of Ns, which means 232 gaps were created. These gaps were spread across 113 scaffolds. No scaffold had more than six gaps (seven contigs ordered and oriented together). The mean scaffold length, scaffold N50, scaffold NG50, and maximum scaffold length increased by 13.6Kbp (4.92%), 3.8Mbp (106.25%), 5.79Mbp (121.90%), and 14.1Mbp (49.85%), respectively. Coupled with these increases were decreases of 29 (50.88%) and 22 (51.16%) in the L50 and LG50, respectively. The auN increased to 14.1M (+72.81%). The quality of the

Hi-C scaffolding can be visualized (Fig. 4) via a contact matrix generated by PretextMap [114] and PretextView [115].

The genome assembly was further improved by scaffolding with RNA-seq data. As expected, the magnitude of the changes between sets of scaffolds was smaller than what was observed between contigs and scaffolds. The total number of sequences was reduced by 176 to 3.4K (-4.69%). The number of known bases was again unchanged; however, it is important to note that when Rascaf orders and orients contigs (or other scaffolds) it always inserts a gap of 17bp to represent gaps of unknown size. Rascaf added 179 new gaps (3,043 unknown bases) across 148 sequences. Three gaps (1,500 unknown bases) from SALSA were removed, but the rest remained unchanged. The most gaps added to a single sequence by Rascaf was five. The sequence with the most total gaps (from either source) had seven gaps (six from Hi-C), thus eight contigs were joined together.

This resulting set of scaffolds (which also includes all the contigs that were not joined to another contig in some way) had a mean length of 304.5Kbp (+5.11% from the Hi-C only value) and a maximum length of 42.29Mbp (+0.08%). The N50 and NG50 increased to 7.97Mbp (+7.04%) and 14.49Mbp (+37.58%), respectively. Decreases to 26 (-7.14%) and 20 (-4.76%) were observed for L50 and LG50, respectively. The auN increased to 14.7M (+4.37%). Table 2 summarizes the assembly continuity statistics, and the area under the N-curve (auN) is visualized in Fig. 5.

The assembly correctness, as assessed with single-copy orthologs, was also evaluated at each stage (Table S2 [Additional File 2]). The results suggest that the modifications made to the primary Canu-based assembly from polishing and scaffolding did not significantly impact the correct assembly of single-copy orthologs. The final set of scaffolds had 3,481 complete single-

copy orthologs (95.6% of 3,640 from the ODB10 Actinopterygii set). Of these 88.4% (3,076) were present in the assembly only once, and 11.6% (405) were present more than once. Twenty-five (0.7%) and 135 (3.7%) single-copy orthologs were fragmented in and missing from the assembly, respectively.

Transcriptome Assembly

The transcriptome assembly generated by Trinity was comprised of 455K sequences with a mean sequence length of 1,177bp. The N50 and L50 were 2.6Kbp and 56K, respectively. The N90 and L90 were, respectively, 410bp and 270K. Of the 3,640 single-copy orthologs in the ODB10 Actinopterygii set, 86.4% (3,144) were complete; 39.5% (1,241) of which were present only once in the transcript set. 128 (3.5%) single-copy orthologs were fragmented in the transcript set, 368 (10.1%) were missing. (See Table S2 [Additional File 2])

Computational Annotation

Computational structural and functional annotation yielded 28.3K protein-coding genes. Of these, 17.2K and 15.6K have annotated 5′ and 3′ UTRs, respectively. 1.8K tRNA genes were also identified. The annotations are available with the assembly on GenBank.

Population Genomic Analysis

Cross-entropy scores generated by the model-based population differentiation analysis, sNMF, provided support for a single population of *A. glossodonta* across all localities. However, individual ancestry plots generated by sNMF showed evidence of genetic differentiation in individuals from Mauritius (St. Brandon's Atoll), compared to the Seychelles sites (Fig. 6A). This differentiation was corroborated by PCA visualization of the first two principal

components, where St. Brandon's Atoll individuals clustered separately from the four Seychelles island groups (Fig. 6B). Together, both population differentiation analyses indicated weak geographic population structure across all sampling localities, with reduced gene flow between St. Brandon's Atoll and the Seychelles sites.

Pairwise F_{ST} results also indicated greater genetic differentiation between St. Brandon's Atoll and all other island groups (Table 3). Estimates of observed and expected heterozygosity were similar across island groups (Table S5 [Additional File 2]), suggesting no differences in genetic diversity between sampling localities and providing no evidence for distinguishing metapopulation processes such as inbreeding. A test of isolation by distance between sampling sites was not significant ($p = 0.1501$).

DISCUSSION

Albula glossodonta is an important fishery species in the Indo-Pacific for both subsistence and recreational purposes [20, 30, 116, 117]. Given this species' current "Vulnerable" IUCN status [7, 118] amidst recent taxonomic uncertainties [4], understanding patterns of gene flow and population structure in *A. glossodonta* is important for fisheries management [30, 119].

We observed a genetically homogenous population of *A. glossodonta* across five island atolls in the Seychelles Archipelago, with limited gene flow between Seychelles and Mauritius. Unlike highly migratory species such as eels (Anguillidae), which are close relatives of bonefishes, adult bonefishes are known for high site fidelity with relatively short migrations (~10-100 km) [117, 120, 121]. We hypothesized that adult bonefishes would not migrate between the Seychelles islands, or between the Seychelles and St. Brandon's Atoll in Mauritius,

since these distances span 400–2,000 km. Consequently, the observed trend of genetic homogeneity across the Seychelles is likely not a result of adult long-distance migrations, but rather pelagic larval dispersal, the primary dispersal mechanism for bonefishes [32, 121-123]. Bonefish larvae, also referred to as leptocephali, have a long pelagic larval duration ranging from 41–72 days, which enables them to drift long distances with ocean currents [21, 124]. The estimated pelagic larval duration for *A. glossodonta* is 57 days, based on observations of individuals from French Polynesia in the South Pacific [21]. The Seychelles islands are located in the South Equatorial Current, which flows westwards from the Indian Ocean towards the eastern coast of continental Africa, enabling larvae to be transported across the Seychelles islands, even across depths exceeding 4000 m (Fig. 2) [125, 126].

Genetic homogeneity is not always an outcome of long pelagic larval duration, as demonstrated by *Anguilla marmorata*, for which 2–5 stocks were identified in the Indo-Pacific [127, 128], and *A. glossodonta*, where putative stocks between the Indian and Pacific Oceans were suggested [119]. Indeed, we found evidence of restricted gene flow between the Seychelles sampling sites and St. Brandon's Atoll, Mauritius, which is ~1500–2000 km from the Seychelles Islands (Fig. 2). This genetic structuring was unexpected, given the long pelagic larval duration of *A. glossodonta*. However, there is evidence of limited gene flow between Seychelles and Mauritius in other marine fish species with pelagic larvae, such as *Lutjanid kasmira* [129], *Lethrinus nebulosus* [130], and *Pristipomoides filamentous* [131].

We attribute the observed genetic structure between Seychelles and St. Brandon's Atoll to the ocean currents in the southwestern Indian Ocean and their role in larval transport [132, 133]. St. Brandon's Atoll is in the direct path of one of the bifurcated arms of the South Equatorial Current as it passes through the Mascarene Plateau [125, 134]. The South Equatorial

Current pushes water westward, which may create a barrier to gene flow to islands south of Seychelles such as Mauritius and Réunion [130, 131, 134]. Although there are currently no bonefish – or even elopomorph – larval dispersal models for the Indian Ocean, pelagic larval dispersal simulation models of coral species in the southwestern Indian Ocean corroborate the biogeographic break between Seychelles and Mauritius, suggesting connectivity is limited even when the pelagic larval duration is between 50–60 days [125, 134]. However, these models considered coral larvae, which are completely reliant on currents for their dispersal [122, 134, 135]. Whilst the dispersal behavior of *A. glossodonta* larvae is unknown, we speculate that, similar to eels (Anguillidae; which also have long pelagic larval durations), bonefishes could disperse greater distances than passive corals by having the ability to swim (e.g., *Anguilla japonica* [136]) or may even take part in vertical migrations (e.g., *Anguilla japonica* [137, 138]). While officially undescribed, swimming ability in bonefish leptocephali has been observed [139], and vertical migrations have previously been theorized [122, 140].

Genome-wide datasets have enabled researchers to better-delineate population connectivity across seascapes for marine species where conventional markers (e.g., mtDNA, microsatellites) have not provided sufficient genomic resolution [127, 141, 142]. Such advances in genomic sequencing have altered our view of population connectivity in other marine fishes such as yellowfin tuna (*Thunnus albacores* [143]) and the American eel (*Anguilla rostrata* [144]). These studies, including ours, highlight the power of large genomic datasets for investigating connectivity in open-ocean environments containing few, if any, natural barriers that were traditionally thought to drive population structure. Although there has been a rapid increase in the number of studies using next-generation sequencing datasets for marine fishes,

few studies to date have employed the use of genomic datasets on elopomorphs, and none on bonefish [144-146].

Conclusions

This is the first genome assembly and annotation for an albulid species, as well as the first use of a genome-wide single-nucleotide polymorphism dataset to investigate population structure for *Albula glossodonta* or any bonefish species in the Indian Ocean. Individuals of *A. glossodonta* were genetically homogenous across four coralline island groups in the Seychelles Archipelago, but they showed evidence of genetic differentiation between the Seychelles and Mauritius (St. Brandon's Atoll). These patterns of connectivity are likely facilitated by pelagic larval dispersal, which is presumed to be strongly shaped by currents in the southwestern Indian Ocean. Only with high-resolution genomic data were we able to discern this pattern of population structure between Seychelles and Mauritius. Our dataset serves as a valuable resource for future genomic studies of bonefishes to facilitate their management and conservation.

DATA AVAILABILITY

The raw reads, genome assembly, and annotations are available under BioProject PRJNA668352 and BioSamples SAMN16516506-SAMN16516510 and SAMN17284271. The ddRAD reads are available under BioProject PRJNA702254, BioSamples SAMN18012541-SAMN18012606.

AUTHOR CONTRIBUTIONS

PDC: Conceptualization; Funding Acquisition; Investigation; Supervision; Resources; Writing - Review & Editing. **DE:** Methodology; Validation; Writing - Original Draft Preparation; Writing - Review & Editing. **JRG:** Conceptualization; Formal Analysis; Investigation; Supervision; Methodology; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **JSKK:** Conceptualization; Funding Acquisition; Investigation; Supervision; Resources; Writing - Review & Editing. **BDP:** Conceptualization; Data Curation; Formal Analysis; Investigation; Methodology; Software; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **PGR:** Funding Acquisition; Supervision; Resources; Writing - Review & Editing. **ST:** Investigation; Resources; Writing - Original Draft Preparation; Writing - Review & Editing.

ORCIDs

Paul D. Cowley, Ph.D.: 0000-0003-1246-4390

Daniel Ence, Ph.D.: 0000-0001-6099-9985

Jessica R. Glass, Ph.D.: 0000-0002-9843-1786

John S. K. Kauwe III, Ph.D.: 0000-0001-8641-2468

Brandon D. Pickett: 0000-0001-8235-4440

Perry G. Ridge, Ph.D.: 0000-0001-6944-2753

Sheena Talma: 0000-0003-2971-6523

ACKNOWLEDGEMENTS

We thank the artist, Tim Johnson [147], for creating the beautiful illustration (Fig. 1). We thank the Brigham Young University DNA Sequencing Center [38] and Office of Research Computing [148] for their continued support of our research. We thank Elizabeth M. Wallace, Clayton Ching, Josiah Ching, Derek Olthuis, Zachary Emig, Weston Gleave, and the fly fishing guides from FlyCastaway [149] and Alphonse Fishing Company [150], especially Daniel Hoenings and Matthieu Cosson, for the collection of samples in Hawai‘i and the western Indian Ocean. We are grateful to Taryn Bodill and Martinus Scheepers of the South African Institute for Aquatic Biodiversity [36] for laboratory assistance and Thomas Near of Yale University [151] for the use of laboratory space, funding, and equipment. We also thank the Seychelles Fishing Authority [152], the Island Conservation Society [153], the Islands Development Company Ltd. [154], the Seychelles Islands Foundation [155], the Ministry of Agriculture, Climate Change and Environment [156], and Shane and Hafiza Talma for their logistical support.

FUNDING

BDP was supported by a Conservation Scholarship [157] from Fly Fishers International [158]. ST was supported by the South African Institute for Aquatic Biodiversity [36], the Mandela Rhodes Foundation [159], the Marine Research Grant [160] from the Western Indian Ocean Marine Science Association [161], and the Yale University Department of Ecology and Evolutionary Biology [162].

CONFLICT OF INTEREST

None declared.

ADDITIONAL FILES

Additional File 1: Supplementary Bioinformatics Methods. Herein as Appendix 2.

Additional File 2: Supplementary Tables. Herein as Appendix 3.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Sequencing Information. The results from each type of DNA and RNA sequencing from *Albula glossodonta*. PE= Paired-end reads. SMRT=Single-Molecule, Real-Time sequencing. CLR=Continuous Long-reads.

Company	Illumina	Illumina	Illumina	PacBio
Instrument	Hi-Seq 2500	Hi-Seq 2500	Hi-Seq 2500	Sequel I
Mode	Rapid Run	High Output	Rapid Run	NA
Sequencing Type	PE	PE	Hi-C, PE	SMRT, CLR
Duration	250 cycles	125 cycles	250 cycles	30 hours
Specimen	1	1	2	2
Tissues	Blood	Gill, Heart, Liver	Heart	Heart
Molecule	DNA	RNA	DNA	DNA
Millions of Read(Pair)s	109.5	270.7	88.7	9.5
Mean Read Length	246	124	245	7,353
Read N50	250	125	250	13,831
Nucleotides (Gb)	53.86	67.2	44.28	69.85

Table 2. Continuity Statistics. Continuity statistics for the *Albula glossodonta* genome assembly at various stages. Note that the auN value is the area under the NG curve, not the N curve. Also note that when submitted to GenBank, the gaps were all converted to a length of 100 bp.

	Contigs	**Polished Contigs**	**Scaffolds (Hi-C)**	**Scaffolds (Hi-C + RNA-seq)**
Sequences	3,799	3,799	3,621	3,445
Known Bases	1.04935 Gb	1.04903 Gb	1.04903 Gb	1.04903 Gb
Mean Length	276,217.073	276,133.196	289,707.267	304,507.986
Max. Length	28,203,290	28,199,443	42,256,846	42,290,388
NG50	4,747,926	4,746,442	10,532,420	14,490,288
NG90	503,090	503,135	739,806	827,489
LG50	43	43	21	20
LG90	289	289	181	162
auN	8,165,188	8,163,173	14,106,761	14,723,001
Sequences with Gaps	-	-	133	236
Gaps	-	-	232	408
Unknown Bases	-	-	116,000	117,543
Mean Gap Length	-	-	500.000	288.096

Table 3. Pairwise FST Comparisons by Island Group.

	Amirantes	Farquhar	Aldabra
Farquhar	0.0014*		
Aldabra	0.0005	0.0020*	
Mauritius	0.0034*	0.0043*	0.0040*

Figure 1. Roundjaw Bonefish (*Albula glossodonta*) adult. Quantitative morphological data for this illustration of A. glossodonta were obtained primarily from two articles: Hidaka et al. 2008 [163] and Shaklee and Tamaru 1981 [14]. These were then evaluated by the artist, with assistance and input from the authors, to select specific values for details such as the number of pored lateral line scales (76) and the number of rays in the pectoral (18), dorsal (16), pelvic (10), and anal fins (9). Each of these was portrayed in the illustration to be at or near the middle of the ranges reported in the aforementioned articles. While some limited information was found in the literature describing coloration and general shape, the artist found particular benefit in some excellent, detailed photographs by Derek Olthuis of samples that were both personally caught in Hawai‘i and later genetically identified as A. glossodonta by Dr. J. S. K. Kauwe. Illustration Copyright: Tim Johnson, used with permission.

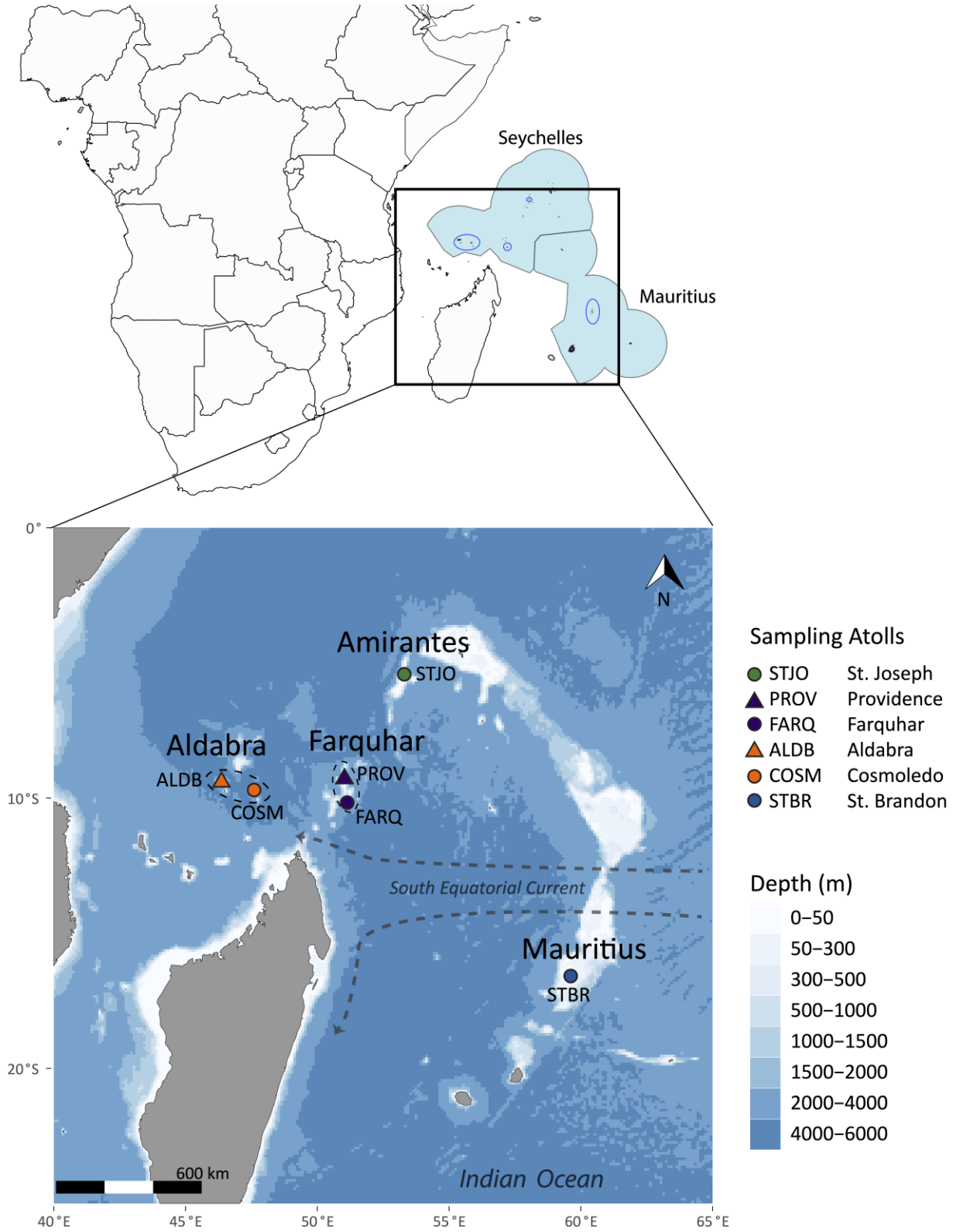


Figure 2. Sampling localities for *A. glossodonta* population genomic analysis. The upper panel shows the marine boundaries for the Seychelles and Mauritius in light blue. Locations of sampling sites are indicated by dark blue ovals. The lower panel shows the atolls comprising the four island groups: Amirantes, Farquhar, Aldabra, and Mauritius.

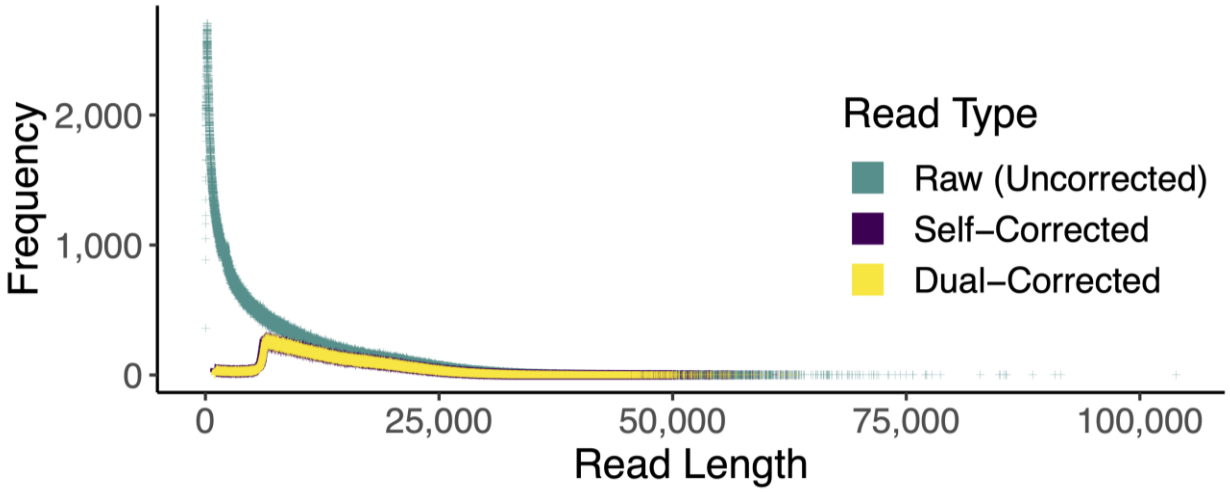


Figure 3. Frequency of Pacific Biosciences Read Lengths. The change in read length distribution is demonstrated as reads are corrected. The dramatic shift from raw to corrected reads is evident. The self-corrected (purple) data points are slightly larger than the dual-corrected (yellow) data points to make the purple distribution visible, the size has no meaning.

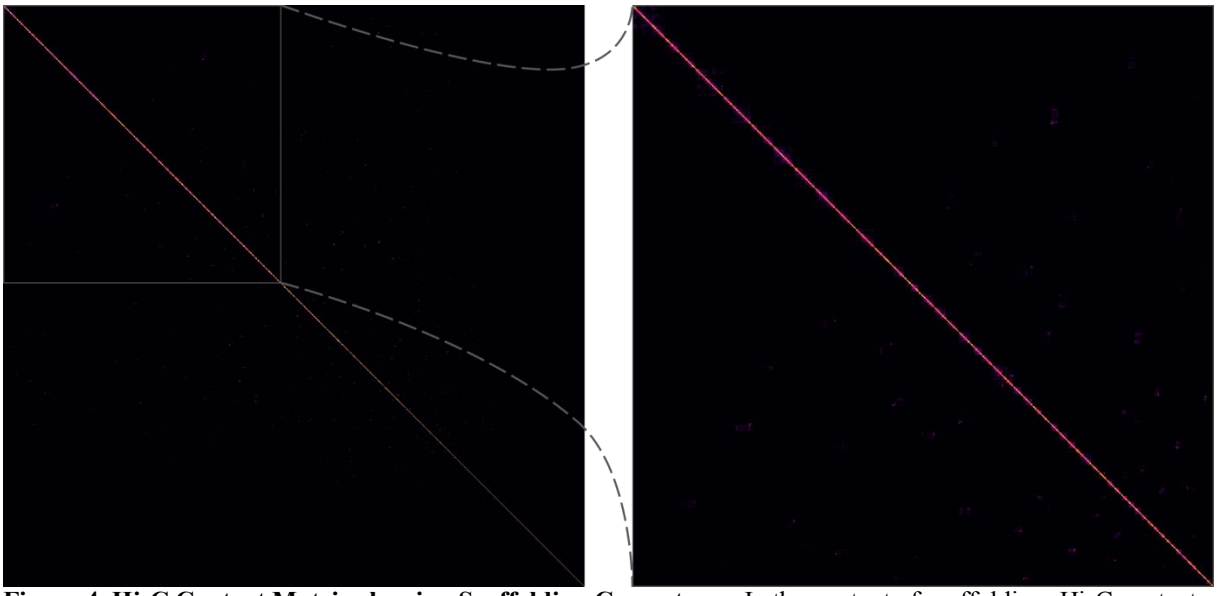


Figure 4. Hi-C Contact Matrix showing Scaffolding Correctness. In the context of scaffolding, Hi-C contact matrices show how correct the scaffolds are. Off-diagonal marks, especially those that are bright and large, are evidence of mis-assembly and/or incorrect scaffolding. The interpretations of the lighter and smaller off-diagonal marks in this image are ambiguous because the assembly is unphased with some relatively short contigs/scaffolds. Additional detail may be viewed by zooming in on the high-resolution image.

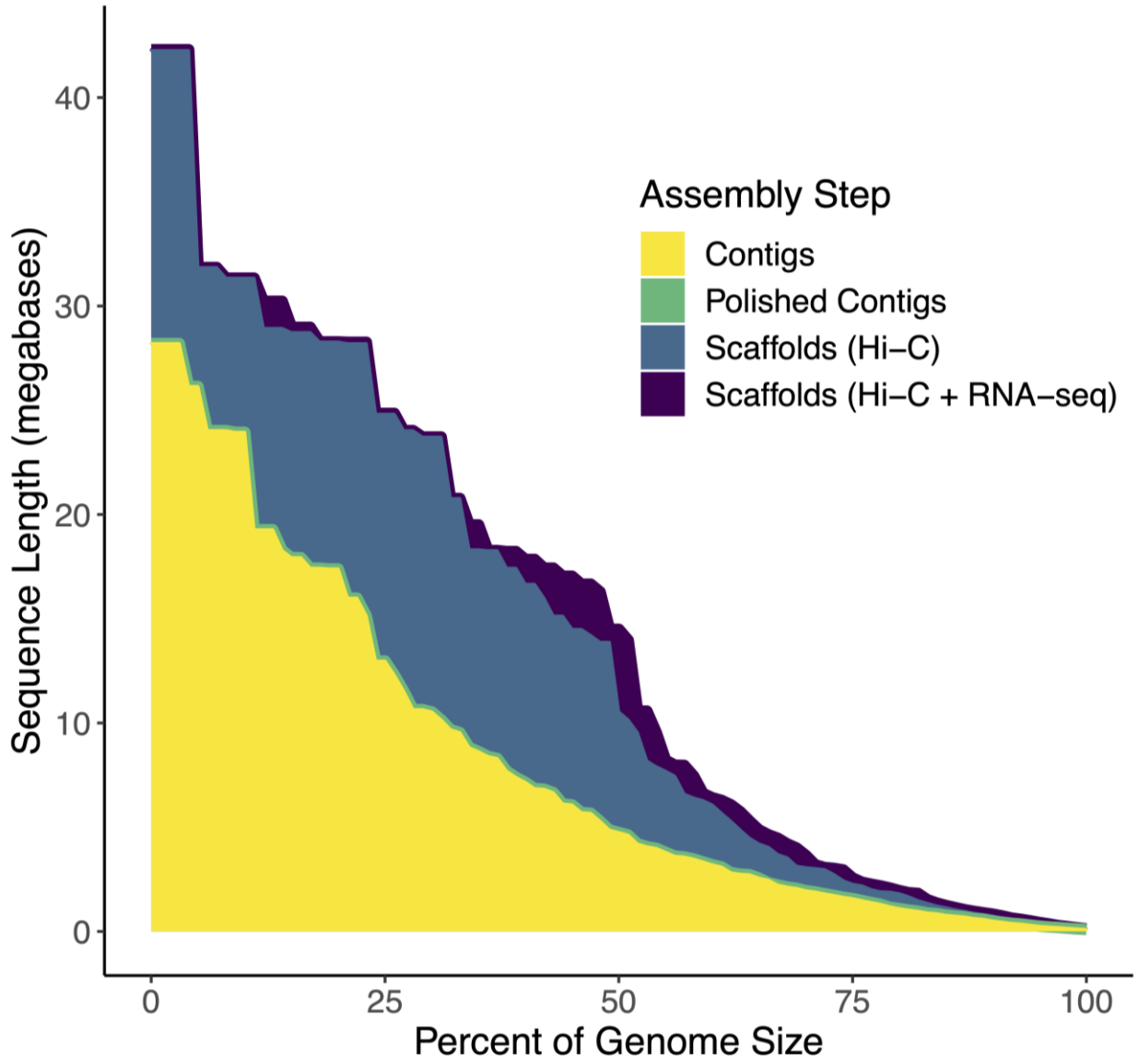


Figure 5. Area Under the N-curve (auN) for each Assembly Step. The N-curve and the area under it are plotted for each major step of the assembly: contigs, polished contigs, scaffolds from only Hi-C data, and scaffolds from both Hi-C and RNA-seq data. The auN for the polished contigs (green) is very similar to the contigs (yellow). Most of the curve was completely blocked by the contigs (yellow) curve. To show that the polished contigs (green) share nearly the same curve, the line was plotted more thickly so it can just barely be seen. Similarly, the Hi-C + RNA-seq scaffolds (purple) curve is very similar to the Hi-C only scaffolds (blue) curve. In this case, differences are more apparent. In certain places, e.g., at the highest peak, the Hi-C + RNA-seq scaffolds (purple) are plotted more thickly so it can be seen behind the Hi-C only scaffolds (blue).

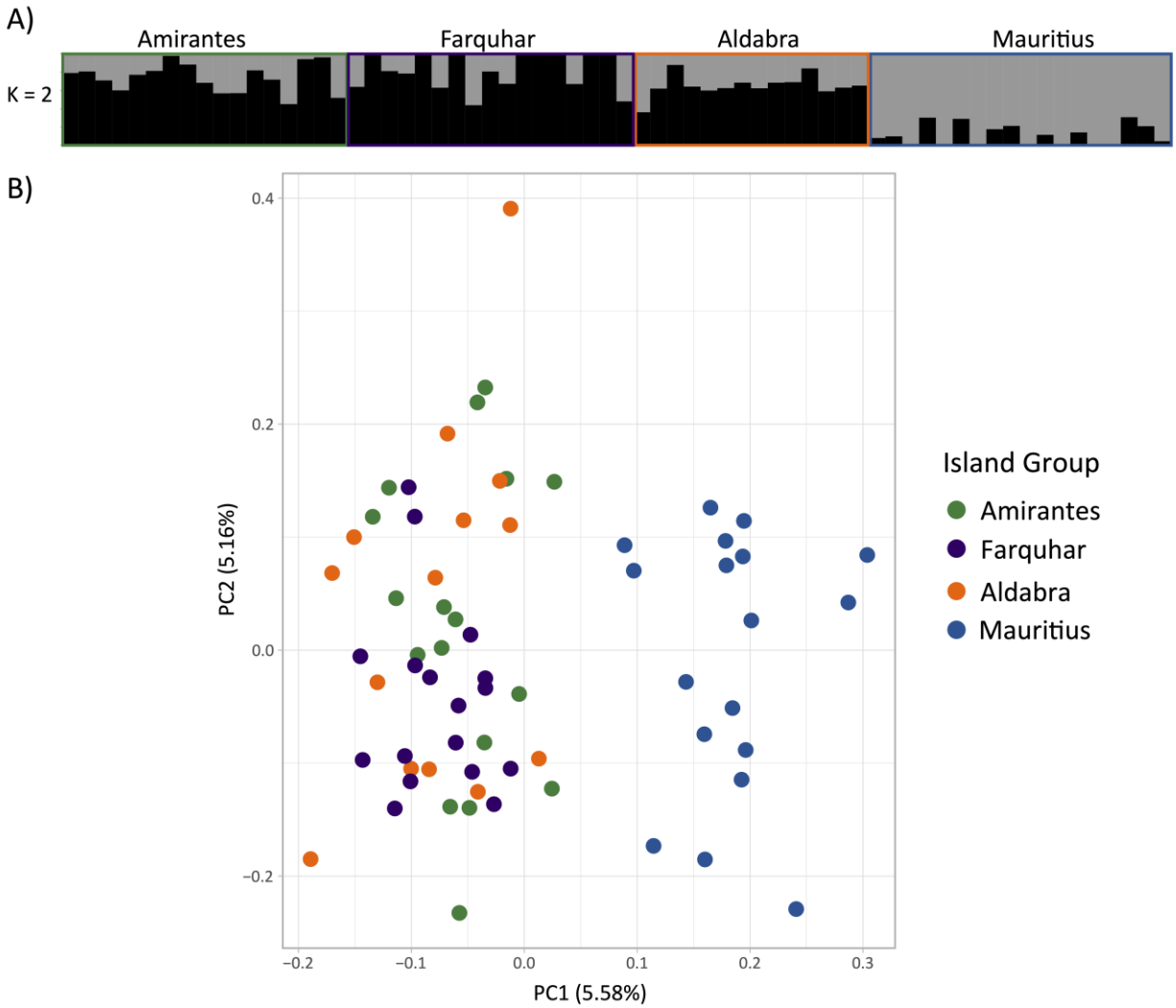


Figure 6. Population Differentiation Analyses. Weak geographic population structure is present across all sampling localities, with reduced gene flow between St. Brandon's Atoll and the Seychelles sites. Island groups are colored as in Fig. 2. (A) Individual ancestry plots generated using sNMF, indicating K = 2 putative populations. (B) Principal component analysis biplot showing the first two principal components.

REFERENCES

1. Fedler T. *Economic Impact of the Florida Keys Flats Fishery*. 2013. Vero Beach, FL, USA: The Bonefish and Tarpon Trust.

2. Fedler T. *The Economic Impact of Flats Fishing in The Bahamas*. 2010. The Bahamian Flats Fishing Alliance.

3. Fedler T. *The 2018 Economic Impact of Flats Fishing in The Bahamas*. 2019. Miami, FL, USA: The Bonefish and Tarpon Trust.

4. Pickett BD, Wallace EM, Ridge PG and Kauwe JSK. Lingering Taxonomic Challenges Hinder Conservation and Management of Global Bonefishes. Fisheries. 2020; 45 7:347-58. doi:10.1002/fsh.10438.

5. Jörger KM and Schrödl M. How to describe a cryptic species? Practical challenges of molecular taxonomy. Frontiers in Zoology. 2013; 10 1:59. doi:10.1186/1742-9994-10-59.

6. Wallace EM. *Assessing Biodiversity, Evolution, and Biogeography in Bonefishes (Albuliformes): Resolving Relationships and Aiding Management*. Doctoral dissertation, University of Minnesota, St. Paul, MN, USA, 2014.

7. Adams AJ, Horodysky AZ, McBride RS, Guindon K, Shenker J, MacDonald TC, et al. Global conservation status and research needs for tarpons (Megalopidae), ladyfishes (Elopidae) and bonefishes (Albulidae). Fish and Fisheries. 2014; 15 2:280-311. doi:10.1111/faf.12017.

8. Linnaeus C. Systema Naturæ. 10 ed. Stockholm, Sweden 1758.

9. Colborn J, Crabtree RE, Shaklee JB, Pfeiler E and Bowen BW. The Evolutionary Enigma of Bonefishes (*Albula spp.*): Cryptic Species and Ancient Separations in a Globally Distributed Shorefish. Evolution. 2001; 55:807-20. doi:10.1111/j.0014-3820.2001.tb00816.x.

10. Bowen BW, Karl SA and Pfeiler E. Resolving Evolutionary Lineages and Taxonomy of Bonefishes (Albula spp.). In: Ault JS, editor. Biology and management of the world Tarpon and Bonefish fisheries. Boca Raton, FL, USA: CRC Press; 2008. p. 147-54.

11. Whitehead PJP. The Synonymy of *Albula vulpes* (Linnaeus, 1758) (Teleostei, Albulidae). Cybium. 1986; 10:211-30.

12. Fowler HW. A new albuloid fish from Santo Domingo. Proc Acad Nat Sci Philadelphia. 1911; 62:651-4.

13. Rivas LR and Warlen SM. Systematics and biology of the bonefish *Albula Nemoptera* (Fowler). Fishery Bulletin US Fish and Wildlife Services. 1967; 66 2:251-8.

14. Shaklee JB and Tamaru CS. Biochemical and Morphological Evolution of Hawaiian Bonefishes (*Albula*). Syst Zool. 1981; 30:125. doi:10.2307/2992412.

15. Seyoum S, Wallace EM and Tringali MD. PERMANENT GENETIC RESOURCES: Twelve polymorphic microsatellite markers for the bonefish, *Albula vulpes* and two congeners. Mol Eco Res. 2008; 8:354-6. doi:10.1111/j.1471-8286.2007.01954.x.

16. Wallace EM and Tringali MD. Identification of a novel member in the family Albulidae (bonefishes). J Fish Biol. 2010; 76:1972-83. doi:10.1111/j.1095-8649.2010.02639.x.

17. Forsskål P. Descriptiones Animalium: Avium, Amphibiorum, Piscium, Insectorum, Vermium. Hauniæ 1775.

18. Kwun HJ and Kim JK. A new species of bonefish, *Albula koreana* (Albuliformes: Albulidae) from Korea and Taiwan. Zootaxa. 2011; 63:57-63.

19. Donovan MK, Friedlander AM, Harding KK, Schemmel EM, Filous A, Kamikawa K, et al. Ecology and niche specialization of two bonefish species in Hawai‘i. Environmental Biology of Fishes. 2015; 98:2159-71. doi:10.1007/s10641-015-0427-z.

20. Filous A, Lennox RJ, Clua EEG and Danylchuk AJ. Fisheries selectivity and annual exploitation of the principal species harvested in a data-limited artisanal fishery at a remote atoll in French Polynesia. Ocean & Coastal Management. 2019; 178 1 August 2019:1-13. doi:10.1016/j.ocecoaman.2019.104818.

21. Filous A, Lennox RJ, Coleman RR, Friedlander AM, Clua EEG and Danylchuk AJ. Life-history characteristics of an exploited bonefish *Albula glossodonta* population in a remote South Pacific atoll. J Fish Biol. 2019; 95 2:562-74. doi:10.1111/jfb.14057.

22. Johannes RE and Yeeting B. I-Kiribati knowledge and management of Tarawa's Lagoon resources. Atoll Research Bulletin. 2000; 489:1-24. doi:10.5479/si.00775630.489.1.

23. Ram-Bidesi V. An economic assessment of destructive fishing methods in Kiribati: A case study of *te ororo* fishing in Tarawa. SPC Fisheries Newsletter. 2011; 135 May/August:21-7.

24. Ram-Bidesi V and Petaia S. *Socio-economic assessment of fishing practices by North and South Tarawa fishers in Kiribati*. 2010.

25. Pfeiler E, Colborn J, Douglas MR and Douglas ME. Systematic status of bonefishes (*Albula spp.*) from the eastern Pacific Ocean inferred from analyses of allozymes and mitochondrial DNA. Environmental Biology of Fishes. 2002; 63:151-9. doi:10.1023/A:1014263528547.

26. Pfeiler E. Resurrection of the name *Albula pacifica* (Beebe, 1942) for the shafted bonefish (Albuliformes: Albulidae) from the eastern Pacific. Rev Biol Trop. 2008; 56:839-44.

27. Pfeiler E, Bitler BG and Ulloa R. Phylogenetic Relationships of the Shafted Bonefish *Albula Nemoptera* (Albuliformes: Albulidae) from the Eastern Pacific Based on Cytochrome B Sequence Analyses. Copeia. 2006; 2006:778-84. doi:10.1643/0045-8511(2006)6[778:PROTSB]2.0.CO;2.

28. Kwun HJ, Kim JK, Doiuchi R and Nakabo T. Molecular and morphological evidence for the taxonomic status of a newly reported species of *Albula* (Albuliformes: Albulidae) from Korea and Taiwan. Animal Cells and Systems. 2011; 15:45-51. doi:10.1080/19768354.2011.555151.

29. Valdez-Moreno M, Vásquez-Yeomans L, Elías-Gutiérrez M, Ivanova NV and Hebert PDN. Using DNA barcodes to connect adults and early life stages of marine fishes from the Yucatan Peninsula, Mexico: potential in fisheries management. Marine and Freshwater Research. 2010; 61:655. doi:10.1071/MF09222.

30. Wallace EM. High intraspecific genetic connectivity in the Indo-Pacific bonefishes: implications for conservation and management. Environmental Biology of Fishes. 2015; 98:2173-86. doi:10.1007/s10641-015-0416-2.

31. Díaz-Viloria N, Sánchez-Velasco L, Perez-Enriquez R, Zárate-Villafranco A, Miller MJ and Jiménez-Rosenberg SPA. Morphological description of genetically identified Cortez bonefish (*Albula gilberti*, Pfeiler and van der Heiden 2011) leptocephali from the southern Gulf of California. Mitochondrial DNA Part A. 2017; 28:717-24. doi:10.3109/24701394.2016.1174226.

32. Wallace EM and Tringali MD. Fishery composition and evidence of population structure and hybridization in the Atlantic bonefish species complex (*Albula spp.*). Mar Biol. 2016; 163:142. doi:10.1007/s00227-016-2915-x.

33. Pamilo P and Nei M. Relationships between Gene Trees and Species Trees. Mol Biol Evol. 1988; 5 5:568-83. doi:0.1093/oxfordjournals.molbev.a040517.

34. Nichols R. Gene trees and species trees are not the same. Trends Eco Evol. 2001; 16:358-64. doi:10.1016/S0169-5347(01)02203-0.

35. Song H, Buhay JE, Whiting MF and Crandall KA. Many species in one: DNA barcoding overestimates the number of species when nuclear mitochondrial pseudogenes are coamplified. Proceedings of the National Academy of Sciences. 2008; 105 36:13486-91. doi:10.1073/pnas.0803076105.

36. The South African Insitute for Aquatic Biodiversity (SAIAB). https://www.saiab.ac.za. Accessed 1 February 2021.

37. Pacific Bioscienes. https://www.pacb.com. Accessed 1 February 2021.

38. Brigham Young University DNA Sequencing Center. https://dnasc.byu.edu. Accessed 1 February 2021.

39. Illumina. https://www.illumina.com. Accessed 1 February 2021.

40. Covaris. https://www.covaris.com. Accessed 1 February 2021.

41. New England Biolabs. https://www.neb.com. Accessed 1 February 2021.

42. Integrated DNA Technologies. https://www.idtdna.com. Accessed 1 February 2021.

43. Roche. https://sequencing.roche.com. Accessed 1 February 2021.

44. Phase Genomics. https://phasegenomics.com. Accessed 1 February 2021.

45. Qiagen. https://www.qiagen.com/. Accessed 1 February 2021.

46. Life Technologies. https://www.thermofisher.com. Accessed 1 February 2021.

47. Peterson BK, Weber JN, Kay EH, Fisher HS and Hoekstra HE. Double Digest RADseq: An Inexpensive Method for De Novo SNP Discovery and Genotyping in Model and Non-Model Species. PLoS ONE. 2012; 7 5:e37135. doi:10.1371/journal.pone.0037135.

48. Agilent. https://www.agilent.com. Accessed 1 February 2021.

49. Sage Science. https://sagescience.com. Accessed 1 February 2021.

50. University of Oregon Genomics and Cell Characterization Core Facility. https://gc3f.uoregon.edu. Accessed 1 February 2021.

51. Kelley DR, Schatz MC and Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. Genome Biol. 2010; 11:R116. doi:10.1186/gb-2010-11-11-r116.

52. R Core Team. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing, 2021. https://www.r-project.org.

53. Yee TW and Wild CJ. Vector Generalized Additive Models. Journal of Royal Statistical Society, Series B. 1996; 58 3:481-93. doi:10.1111/j.2517-6161.1996.tb02095.x.

54. Yee TWM, Cleve. VGAM: Vector Generalized Additive Models. The Comprehensive R Archive Network. 2009; v0.7-8.

55. Marcais G and Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 2011; 27 6:764-70. doi:10.1093/bioinformatics/btr011.

56. Melsted P and Pritchard JK. Efficient counting of k-mers in DNA sequences using a bloom filter. BMC Bioinform. 2011; 12 333 doi:10.1186/1471-2105-12-333.

57. Song L and Florea L. Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads. GigaScience. 2015; 4 48 doi:10.1186/s13742-015-0089-y.

58. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH and Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017; 27 5:722-36. doi:10.1101/gr.215087.116.

59. Haghshenas E, Hach F, Sahinalp SC and Chauve C. CoLoRMap: Correcting Long Reads by Mapping short reads. Bioinformatics. 2016; 32:i545-i51. doi:10.1093/bioinformatics/btw463.

60. Hamid M, Khan H and Birol I. ntCard: a streaming algorithm for the cardinality estimation of genomics data. Bioinformatics. 2017; 33 9:1324-30. doi:10.1093/bioinformatics/btw832.

61. Vaser R, Sović I, Nagarajan N and Šikić M. Fast and accurate de novo genome assembly from long uncorrected reads. Genome Res. 2017; 27 5:737-46. doi:10.1101/gr.214270.116.

62. Arima Genomics. https://arimagenomics.com. Accessed 1 February 2021.

63. Arima Genomics Mapping Pipeline. https://github.com/ArimaGenomics/mapping_pipeline. Accessed 1 February 2021.

64. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv. 2013; 1303.3997.

65. Broad Institute. Picard Toolkit. Broad Institute, GitHub repository: Broad Institute, 2019. http://broadinstitute.github.io/picard.

66. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009; 25 16:2078-9. doi:10.1093/bioinformatics/btp352.

67. Quinlan AR and Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010; 26 6:841-2. doi:10.1093/bioinformatics/btq033.

68. Kim D, Langmead B and Salzberg SL. HISAT: a fast spliced aligner with low memory requirements. Nat Methods. 2015; 12 4:357-60. doi:10.1038/nmeth.3317.

69. Ghurye J, Pop M, Koren S, Bickhart D and Chin C-S. Scaffolding of long read assemblies using long range contact information. BMC Genomics. 2017; 18 1:1-11. doi:10.1186/s12864-017-3879-z.

70. Ghurye J, Rhie A, Walenz BP, Schmitt A, Selvaraj S, Pop M, et al. Integrating Hi-C links with assembly graphs for chromosome-scale assembly. PLoS Comput Biol. 2019; 15 8:e1007273. doi:0.1371/journal.pcbi.1007273.

71. Song L, Shankar DS and Florea L. Rascaf: Improving Genome Assembly with RNA Sequencing Data. Plant Genome. 2016; 9 3:1-12. doi:10.3835/plantgenome2016.03.0027.

72. Li H. auN: a new metric to measure assembly contiguity. *Heng Li's Blog*. 2020. http://lh3.github.io/2020/04/08/a-new-metric-on-assembly-contiguity.

73. Li H: calN50 GitHub Repository. https://github.com/lh3/calN50. Accessed 10 April 2020.

74. Python Programming Language. https://www.python.org. Accessed 1 February 2021.

75. Simão FA, Waterhouse RM, Ioannidis P, Kriventseva EV and Zdobnov EM. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics. 2015; 31 19:3210-2. doi:10.1093/bioinformatics/btv351.

76. Kriventseva EV, Kuznetsov D, Tegenfeldt F, Manni M, Dias R, Simão FA, et al. OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. Nucleic Acids Res. 2019; 47 D1:D807-D11. doi:10.1093/nar/gky1053.

77. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol. 2011; 29 7:644-52. doi:10.1038/nbt.1883.

78. Eaton DAR. PyRAD: assembly of de novo RADseq loci for phylogenetic analyses. Bioinformatics. 2014; 30 13:1844-9. doi:10.1093/bioinformatics/btu121.

79. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, Depristo MA, et al. The variant call format and VCFtools. Bioinformatics. 2011; 27 15:2156-8. doi:10.1093/bioinformatics/btr330.

80. O'Leary SJ, Puritz JB, Willis SC, Hollenbeck CM and Portnoy DS. These aren't the loci you'e looking for: Principles of effective SNP filtering for molecular ecologists. Mol Ecol. 2018; 27 16:3193-206. doi:10.1111/mec.14792.

81. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, et al. PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. The American Journal of Human Genetics. 2007; 81 3:559-75. doi:10.1086/519795.

82. Hill WG and Robertson A. Linkage disequilibrium in finite populations. Theoretical and Applied Genetics. 1968; 38 6:226-31. doi:10.1007/BF01245622.

83. Holt C and Yandell M. MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. BMC Bioinform. 2011; 12:491. doi:10.1186/1471-2105-12-491.

84. Holt C and Yandell M: MAKER Tutorial for WGS Assembly and Annotation Winter School 2018. http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER_Tutorial_for_WGS_Assembly_and_Annotation_Winter_School_2018 (2018). Accessed 1 March 2018.

85. Smit AFA and Hubley R. RepeatModeler Open-1.0. 2008.

86. The Uniprot Consortium. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019; 47 D1:D506-D15. doi:10.1093/nar/gky1049.

87. Boutet E, Lieberherr D, Tognolli M, Schneider M and Bairoch A. UniProtKB/Swiss-Prot: The Manually Annotated Section of the UniProt KnowledgeBase. In: Edwards D, editor. Plant Bioinformatics: Methods and Protocols. Totowa, NJ: Humana Press; 2007. p. 89-112.

88. Stanke M, Schöffmann O, Morgenstern B and Waack S. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. BMC Bioinform. 2006; 7:62. doi:10.1186/1471-2105-7-62.

89. Stanke M and Waack S. Ggene prediction with a hidden Markov model and a new intron submodel. Bioinformatics. 2003; 19 Suppl. 2:ii215-ii25. doi:10.1093/bioinformatics/btg1080.

90. Korf I. Gene finding in novel genomes. BMC Bioinform. 2004; 5:59.

91. Lomsadze A, Ter-Hovhannisyan V, Chernoff YO and Borodovsky M. Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Res. 2005; 33 20:6964-506. doi:10.1093/nar/gki937.

92. Brůna T, Lomsadze A and Borodovsky M. GeneMark-EP+: eukaryotic gene prediction with self-training in the space of genes and proteins. NAR Genom Bioinform. 2020; 2 2:lqaa026. doi:10.1093/nargab/lqaa026.

93. Lomsadze A, Burns PD and Borodovsky M. Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. Nucleic Acids Res. 2014; 42 15:e119. doi:10.1093/nar/gku557.

94. Caballero M and Wegrzyn J. gFACs: Gene Filtering, Analysis, and Conversion to Unify Genome Annotations Across Alignment and Gene Prediction Frameworks. Genomics Proteomics Bioinformatics. 2019; 17 3:305-10. doi:10.1016/j.gpb.2019.04.002.

95. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC Bioinform. 2009; 10:421. doi:Artn 421\nDoi 10.1186/1471-2105-10-421.

96. Altschul SF, Gish W, Miller W, Myers EW and Lipman DJ. Basic Local Alignment Search Tool. J Mol Biol. 1990; 215:403-10. doi:10.1016/S0022-2836(05)80360-2.

97. Jones P, Binns D, Chang H-Y, Fraser M, Li W, McAnulla C, et al. InterProScan 5: genome-scale protein function classification. Bioinformatics. 2014; 30 9:1236-40. doi:10.1093/bioinformatics/btu031.

98. Mitchell AL, Attwood TK, Babbitt PC, Blum M, Bork P, Bridge A, et al. InterPro in 2019: improving coverage, classification and access to protein sequence annotations. Nucleic Acids Res. 2019; 47 D1:D351-D60. doi:10.1093/nar/gky1100.

99. Gremme G, Steinbiss S and Kurtz S. GenomeTools: A Comprehensive Software Library for Efficient Processing of Structured Genome Annotations. IEEE/ACM Trans Comput Biol Bioinform. 2013; 10 3:645-56. doi:10.1109/TCBB.2013.68.

100. Luu K, Bazin E and Blum MGB. pcadapt: an R package to perform genome scans for selection based on principal component analysis. Mol Eco Res. 2017; 17 1:67-77. doi:10.1111/1755-0998.12592.

101. Foll M and Gaggiotti O. A Genome-Scan Method to Identify Selected Loci Appropriate for Both Dominant and Codominant Markers: A Bayesian Perspective. Genetics. 2008; 180 2:977-93. doi:10.1534/genetics.108.092221.

102. Martins H, Caye K, Luu K, Blum MGB and François O. Identifying outlier loci in admixed and in continuous populations using ancestral population differentiation statistics. Mol Ecol. 2016; 25 20:5029-42. doi:10.1111/mec.13822.

103. Storey JD, Bass AJ, Dabney A and Robinson D. qvalue: Q-value estimation for false discovery rate control. The Comprehensive R Archive Network. 2017; v2.15.0.

104. Beaumont MA and Balding DJ. Identifying adaptive genetic divergence among populations from genome scans. Mol Ecol. 2004; 13 4:969-80. doi:10.1111/j.1365-294x.2004.02125.x.

105. Vitalis R, Dawson K and Boursot P. Interpretation of Variation Across Marker Loci as Evidence of Selection. Genetics. 2001; 158 4:1811-23. doi:10.1093/genetics/158.4.1811.

106. Foll M: BayeScan v2.1User Manual. http://cmpg.unibe.ch/software/BayeScan/files/BayeScan2.1_manual.pdf (2012). Accessed 1 February 2021.

107. Frichot E and François O. LEA: An R package for landscape and ecological association studies. Methods in Ecology and Evolution. 2015; 6 8:925-9. doi:10.1111/2041-210x.12382.

108. Shryock DF, Havrilla CA, Defalco LA, Esque TC, Custer NA and Wood TE. Landscape genetic approaches to guide native plant restoration in the Mojave Desert. Ecological Applications. 2017; 27 2:429-45. doi:10.1002/eap.1447.

109. Pembleton LW, Cogan NOI and Forster JW. StAMPP: an R package for calculation of genetic differentiation and structure of mixed-ploidy level populations. Mol Eco Res. 2013; 13 5:946-52. doi:10.1111/1755-0998.12129.

110. Jombart T and Ahmed I. adegenet 1.3-1: new tools for the analysis of genome-wide SNP data. Bioinformatics. 2011; 27 21:3070-1. doi:10.1093/bioinformatics/btr521.

111. Goudet J. hierfstat, a package for r to compute and test hierarchical F-statistics. Molecular Ecology Notes. 2005; 5 1:184-6. doi:10.1111/j.1471-8286.2004.00828.x.

112. Hardie DC and Hebert PDN. Genome-size evolution in fishes. Canadian Journal of Fisheries and Aquatic Sciences. 2004; 61 9:1636-46. doi:10.1139/F04-106.

113. Hinegardner RR, Donn Eric. Cellular DNA Content and the Evolution of Teleostean Fishes. The American Naturalist. 1972; 106 951:621-44.

114. High Performance Assembly Group - Wellcome Sanger Institute. PretextMap. 2020; 0.1.4.

115. High Performance Assembly Group - Wellcome Sanger Institute. PretextView. 2019; 0.0.1.

116. Kamikawa KT, Friedlander AM, Harding KK, Filous A, Donovan MK and Schemmel E. Bonefishes in Hawai‘i and the importance of angler-based data to inform fisheries management. Environmental Biology of Fishes. 2015; 98:2147-57. doi:10.1007/s10641-015-0421-5.

117. Moxham EJ, Cowley PD, Bennett RH and von Brandis RG. Movement and predation: a catch-and-release study on the acoustic tracking of bonefish in the Indian Ocean. Environmental Biology of Fishes. 2019; 102 2:365-81. doi:10.1007/s10641-019-00850-1.

118. Adams A, Guindon K, Horodysky A, MacDonald T, McBride R, Shenker J, et al. *Albula glossodonta, Shortjaw Bonefish*. Report no. T194299A2310398, 2012. The International Union for Conservation of Nature.

119. Williams CT, Mcivor AJ, Wallace EM, Lin YJ and Berumen ML. Genetic diversity and life-history traits of bonefish *Albula* spp. from the Red Sea. J Fish Biol. 2020:1-10. doi:10.1111/jfb.14638.

120. Larkin MF. *Assessment of South Florida's Bonefish Stock*. Dissertation, University of Miami, Coral Gables, Florida, USA, 2011.

121. Perez AU, Schmitter-Soto JJ, Adams AJ and Heyman WD. Connectivity mediated by seasonal bonefish (*Albula vulpes*) migration between the Caribbean Sea and a tropical estuary of Belize and Mexico. Environmental Biology of Fishes. 2019; 102 2:197-207. doi:10.1007/s10641-018-0834-z.

122. Zeng X, Adams A, Roffer M and He R. Potential connectivity among spatially distinct management zones for Bonefish (*Albula vulpes*) via larval dispersal. Environmental Biology of Fishes. 2019; 102:233-52. doi:10.1007/s10641-018-0826-z.

123. Danylchuk AJ, Cooke SJ, Goldberg TL, Suski CD, Murchie KJ, Danylchuk SE, et al. Aggregations and offshore movements as indicators of spawning activity of bonefish (*Albula vulpes*) in The Bahamas. Mar Biol. 2011; 158 9:1981-99. doi:10.1007/s00227-011-1707-6.

124. Friedlander A, Caselle JE, Beets J, Lowe CG, Bowen BW, Ogawa TK, et al. Biology and Ecology of the Recreational Bonefish Fishery at Palmyra Atoll National Wildlife Refuge with Comparisons to Other Pacific Islands. In: Ault JS, editor. Biology and management of the world Tarpon and Bonefish fisheries. Boca Raton, FL, USA: CRC Press; 2008. p. 27-56.

125. Crochelet E, Roberts J, Lagabrielle E, Obura D, Petit M and Chabanet P. A model-based assessment of reef larvae dispersal in the Western Indian Ocean reveals regional connectivity patterns — Potential implications for conservation policies. Regional Studies in Marine Science. 2016; 7 September:159-67. doi:10.1016/j.rsma.2016.06.007.

126. Badal MR, Rughooputh S, Rydberg L, Robinson IS and Pattiaratchi C. Eddy formation around South West Mascarene Plateau (Indian Ocean) as evidenced by satellite 'global ocean colour' data. Western Indian Ocean Journal of Marine Science. 2010; 8 2:139-45. doi:10.4314/wiojms.v8i2.56969.

127. Gagnaire P-A, Minegishi Y, Zenboudji S, Valade P, Aoyama J and Berrebi P. Within-population structure highlighted by differential introgression across semipermeable barriers to gene flow in *Anguilla marmorata*. Evolution. 2011; 65 12:3413-27. doi:10.1111/j.1558-5646.2011.01404.x.

128. Donovan S, Pezold F, Chen Y and Lynch B. Phylogeography of *Anguilla marmorata* (Teleostei: Anguilliformes) from the eastern Caroline Islands. Ichthyological Research. 2012; 59 1:70-6. doi:10.1007/s10228-011-0245-z.

129. Muths D, Gouws G, Mwale M, Tessier E and Bourjea J. Genetic connectivity of the reef fish *Lutjanus kasmira* at the scale of the western Indian Ocean. Canadian Journal of Fisheries and Aquatic Sciences. 2012; 69 5:842-53. doi:10.1139/f2012-012.

130. Healey AJE, Gouws G, Fennessy ST, Kuguru B, Sauer WHH, Shaw PW, et al. Genetic analysis reveals harvested Lethrinus nebulosus in the Southwest Indian Ocean comprise two cryptic species. ICES Journal of Marine Science. 2018; 75 4:1465-72. doi:10.1093/icesjms/fsx245.

131. Mzingirwa FA, Mkare TK, Nyingi DW and Njiru J. Genetic diversity and spatial population structure of a deepwater snapper, *Pristipomoides filamentosus* in the south-west Indian Ocean. Mol Biol Rep. 2019; 46 5:5079-88. doi:10.1007/s11033-019-04962-w.

132. Muths D, Grewe P, Jean C and Bourjea J. Genetic population structure of the Swordfish (*Xiphias gladius*) in the southwest Indian Ocean: Sex-biased differentiation, congruency between markers and its incidence in a way of stock assessment. Fish Res. 2009; 97 3:263-9. doi:10.1016/j.fishres.2009.03.004.

133. Obura D. The Diversity and Biogeography of Western Indian Ocean Reef-Building Corals. PLoS ONE. 2012; 7 9:e45013. doi:10.1371/journal.pone.0045013.

134. Gamoyo M, Obura D and Reason CJC. Estimating Connectivity Through Larval Dispersal in the Western Indian Ocean. Journal of Geophysical Research: Biogeosciences. 2019; 124 8:2446-59. doi:10.1029/2019JG005128.

135. Otwoma LM, Reuter H, Timm J and Meyer A. Genetic connectivity in a herbivorous coral reef fish (*Acanthurus leucosternon* Bennet, 1833) in the Eastern African region. Hydrobiologia. 2018; 806 1:237-50. doi:10.1007/s10750-017-3363-4.

136. Chang Y-LK, Miller MJ, Tsukamoto K and Miyazawa Y. Effect of larval swimming in the western North Pacific subtropical gyre on the recruitment success of the Japanese eel. PLoS ONE. 2018; 13 12:e0208704. doi:10.1371/journal.pone.0208704.

137. Kudo K. Larval vertical-migration strategy of Japanese eel. In: *MTS/IEEE Oceans 2001 An Ocean Odyssey* Honolulu, HI, USA, 5-8 November 2001 2001, pp.870-5. Escondido, CA, USA: Marine Technology Society.

138. Shinoda A, Aoyama J, Miller MJ, Otake T, Mochioka N, Watanabe S, et al. Evaluation of the larval distribution and migration of the Japanese eel in the western North Pacific. Reviews in Fish Biology and Fisheries. 2011; 21 3:591-611. doi:10.1007/s11160-010-9195-1.

139. Pfeiler E. Inshore migration, seasonal distribution and sizes of larval bonefish, *Albula*, in the Gulf of California. Environmental Biology of Fishes. 1984; 10 1/2:117-22. doi:10.1007/BF00001668.

140. Mojica RJ, Shenker JM, Harnden CW and Wagner DE. Recruitment of bonefish, *Albula vulpes*, around Lee Stocking Island, Bahamas. Fish Bull. 1994; 93 4:666-74.

141. Lemopoulos A, Prokkola JM, Uusi-Heikkilä S, Vasemägi A, Huusko A, Hyvärinen P, et al. Comparing RADseq and microsatellites for estimating genetic diversity and relatedness — Implications for brown trout conservation. Ecol Evol. 2019; 9 4:2106-20. doi:10.1002/ece3.4905.

142. Willette DA, Allendorf FW, Barber PH, Barshis DJ, Carpenter KE, Crandall ED, et al. So, you want to use next-generation sequencing in marine systems? Insight from the Pan-Pacific Advanced Studies Institute. Bull Mar Sci. 2014; 90 1:79-122. doi:10.5343/bms.2013.1008.

143. Mullins RB, McKeown NJ, Sauer WHH and Shaw PW. Genomic analysis reveals multiple mismatches between biological and management units in yellowfin tuna (*Thunnus albacares*). ICES Journal of Marine Science. 2018; 75 6:2145-52. doi:10.1093/icesjms/fsy102.

144. Babin C, Gagnaire P-A, Pavey SA and Bernatchez L. RAD-Seq Reveals Patterns of Additive Polygenic Variation Caused by Spatially-Varying Selection in the American Eel (*Anguilla rostrata*). Genome Biology and Evolution. 2017; 9 11:2974-86. doi:10.1093/gbe/evx226.

145. Benestan L, Quinn BK, Maaroufi H, Laporte M, Clark FK, Greenwood SJ, et al. Seascape genomics provides evidence for thermal adaptation and current-mediated population structure in American lobster (*Homarus americanus*). Mol Ecol. 2016; 25 20:5073-92. doi:10.1111/mec.13811.

146. Valenzuela-Quiñonez F. How fisheries management can benefit from genomics? Briefings in Functional Genomics. 2016; 15 5:352-7. doi:10.1093/bfgp/elw006.

147. Johnson T: Tim Johnson Gallery. https://timjohnsongallery.com. Accessed 8 March 2021.

148. Brigham Young University Office of Research Computing. https://rc.byu.edu. Accessed 1 February 2021.

149. FlyCastaway. https://www.flycastaway.com. Accessed 1 February 2021.

150. Alphonse Fishing Company. https://www.alphonsefishingco.com. Accessed 1 February 2021.

151. Yale University. https://www.yale.edu. Accessed 1 February 2021.

152. Seychelles Fishing Authority. http://www.sfa.sc. Accessed 1 February 2021.

153. Island Conservation Society. http://www.islandconservationseychelles.com. Accessed 1 February 2021.

154. Islands Development Company Ltd. https://www.idcseychelles.com. Accessed 1 February 2021.

155. Seychelles Islands Foundation. https://www.sif.sc. Accessed 1 February 2021.

156. Ministry of Agriculture, Climate Change and Environment. https://pcusey.sc/about-meecc. Accessed 1 February 2021.

157. Fly Fishers International Conservation Scholarship. https://flyfishersinternational.org/Conservation/Projects-Programs/Scholarship-Program. Accessed 1 February 2021.

158. Fly Fishers International. https://flyfishersinternational.org. Accessed 1 February 2021.

159. The Mandela Rhodes Foundation. https://www.mandelarhodes.org. Accessed 1 February 2021.

160. The Western Indian Ocean Marine Science Association Marine Research Grant. https://www.wiomsa.org/research-support/marg. Accessed 1 February 2021.

161. The Western Indian Ocean Marine Science Association. https://www.wiomsa.org. Accessed 1 February 2021.

162. Yale University Department of Ecology and Evolutionary Biology. https://eeb.yale.edu. Accessed 1 February 2021.

163. Hidaka K, Iwatsuki Y and Randall JE. A review of the Indo-Pacific bonefishes of the *Albula argentea* complex, with a description of a new species. Ichthyological Research. 2008; 55:53-64. doi:10.1007/s10228-007-0010-5.

CHAPTER 3

De novo genome assembly of the marine teleost, Bluefin Trevally (*Caranx melampygus*)

Brandon D. Pickett[1], Jessica R. Glass[2], Perry G. Ridge[1], John S. K. Kauwe[1,3]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[2]*South African Institute for Aquatic Biodiversity, Makhanda, South Africa*

[3]*Brigham Young University - Hawai'i, Laie, Hawai'i, USA*

ABSTRACT

The bluefin trevally, *Caranx melampygus*, also known as the bluefin kingfish or bluefin jack, is known for its remarkable, bright-blue fins. This marine teleost is a widely-prized sportfish, but few resources have been devoted to the genomics and conservation of this species because it is not targeted by large-scale commercial fisheries. Population declines from recreational and artisanal overfishing have been observed in Hawai‘i, USA, resulting in both an interest in aquaculture and concerns about the long-term conservation of this species. Most research to-date has been performed in Hawai‘i, raising questions about the status of bluefin trevally populations across its Indo-Pacific range. Genomic resources allow for expanded research on stock status, genetic diversity, and population demography. We present a high-quality nuclear genome assembly of a Hawaiian bluefin trevally from noisy long-reads with a contig NG50 of 1.2Mbp. Some of the contigs were arranged into scaffolds using RNA-seq data from eight tissues from the same individual. This is the first whole-genome assembly for the carangoid clade Carangini. Using this assembled genome, a multiple sequentially Markovian coalescent model was implemented to assess population demography. Estimates of effective population size suggest population expansion has occurred since the Late Pleistocene. This genome will be a valuable resource for comparative phylogenomic studies of carangoid fishes and will help elucidate demographic history and delineate stock structure for bluefin trevally populations throughout the Indo-Pacific.

INTRODUCTION

The bluefin trevally (*Caranx melampygus*; Cuvier 1833) is a marine teleost fish (Carangiformes: Carangoidei) inhabiting coastal environments throughout the tropical and subtropical Indo-Pacific (Fig. 1). *C. melampygus* is a top predator on coral and rocky reef ecosystems, reaching up to 117 cm in length and feeding predominantly on shallow-water fishes and invertebrates (Sudekum et al. 1991; Meyer et al. 2001). In the Northwestern Hawaiian Islands, for example, bluefin trevallies consume an estimated 11,000 metric tons of prey per year, confirming their role as important predators in this region (Sudekum et al. 1991). *Caranx melampygus* is also targeted by small-scale and recreational fisheries in Hawai‘i, where it is known by its Native Hawaiian name, *‘omilu* (Meyer et al. 2001). In recent decades, the *C. melampygus* population in Hawai‘i has been impacted by overharvesting and habitat destruction (Friedlander and Dalzell 2004). For this reason, there has been significant interest in Hawai‘i in captive breeding for aquaculture (Moriwake et al. 2001; Zhao and Lu 2006). Because the bulk of research on the bluefin trevally has been conducted in Hawai‘i, observations of population declines raise concerns for populations in other parts of its range, where abundance and biomass estimates remain unknown.

Recent genomic evidence suggests *C. melampygus* comprises a unique population in Hawai‘i compared to several localities sampled across the Indo-Pacific (Glass et al. In Press), and an analysis of complete mitochondrial genomes suggests individuals from Guam are also genetically distinct (Genomic Resources Development Consortium et al. 2014). Given population declines and evidence of unique stock structure in Hawai‘i, whole genome data for *C. melampygus* would provide unprecedented value for inferring demographic history, estimating effective population size, and testing for selection and local adaptation. Juvenile and adult

individuals frequently utilize estuarine habitats, for example, and have a strong tolerance for freshwater in coastal locations where estuaries are present (Blaber and Cyrus 1983). Studying the evolution and physiology of *C. melampygus* in a genomic context is valuable to the broader scientific and reef fish community, especially given interest in the genomic mechanisms of adaptation of marine and anadromous fishes to freshwater (Kültz 2015). Furthermore, whole genome data provide baseline biological information for delineating wild stocks, a critical component of transboundary fisheries management, while also serving as an important reference for the aquaculture industry to examine genomic signatures of growth in captivity and susceptibility to disease (Zhao and Lu 2006). At present, published whole genome data are available for only seven out of approximately 150 carangoid species: *Echeneis naucrates* (Linnaeus 1758) (Koepfli et al. 2015) , *Trachinotus ovatus* (Linnaeus 1758) (Zhang et al. 2019), *Selene dorsalis* (Gill 1863) (Malmstrøm et al. 2017) , and four *Seriola* sp. (Purcell et al. 2015; Araki et al. 2018; Ozaki and Araki 2017; Yasuike et al. 2018), all of which diverged from *C. melampygus* approximately 48–50 Mya (Harrington et al. 2016). Here, we present an annotated *de novo* genome assembly of *C. melampygus* to facilitate future research for aquaculture development and expand the genomic resources of carangoid fishes for comparative phylogenomic analysis.

MATERIALS AND METHODS

An overview of the methods used in this study is provided here. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided in the Supplementary Bioinformatics Methods (Supplementary File 1; Appendix 4 herein).

Sample Acquisition & Sequencing

One *C. melampygus* individual was captured in 3-9 m of water <1 km off the coast of O‘ahu (near Kaneohe, Hawai‘i, USA: 21°26'45.3"N 157°48'07.5"W) in April 2018. The specimen was caught using a Shimano (Sakai, Osaka, Japan) ocean rod outfitted with a Daiwa (Cypress, California, USA) Saltiga 6500 reel and a white feather jig. Brain, eye, fin, gill, heart, kidney, liver, and muscle tissue samples were collected immediately upon capture, flash-frozen in liquid nitrogen, and packaged in dry ice for transportation to Brigham Young University (BYU; Provo, Utah, USA) for storage at -80° until sequencing. All tissue samples were used for short-read RNA sequencing. The heart tissue was also used for long-read DNA sequencing.

DNA was prepared for long-read sequencing with a Pacific Biosciences (PacBio; Menlo Park, California, USA; https://www.pacb.com) SMRTbell Library kit, adhering to the following protocol: “Procedure & Checklist – Preparing >30 kb SMRTbell Libraries Using Megaruptor Shearing and BluePippin Size-Selection for PacBio RS II and Sequel Systems”. Continuous long-read (CLR) sequencing was performed on ten SMRT cells for a 10-hour movie on the PacBio Sequel at the BYU DNA Sequencing Center (DNASC; https://dnasc.byu.edu), a PacBio Certified Service Provider. RNA was prepared with Roche (Basel, Switzerland; https://sequencing.roche.com) KAPA Stranded RNA-Seq kit, following recommended protocols. Paired-end sequencing was performed in Rapid Run mode for 250 cycles with the eight samples across two lanes on the Illumina (San Diego, California, USA; https://www.illumina.com) Hi-Seq 2500 at the DNASC.

Sequence Assembly and Scaffolding

The PacBio CLR reads were self-corrected and assembled with Canu v1.6 (Koren et al. 2017). The contigs were scaffolded using RNA-seq reads. The scaffolding step required read mapping to the contigs before determining how to order and orient contigs. The RNA-seq reads were aligned using HiSat v0.1.6-beta (Kim et al. 2015). Scaffolding was performed with RNA-seq data using the latest (June 2018) commit of Rascaf (Song et al. 2016). Assembly continuity statistics, e.g., N50 and auNG (Li 2020), were calculated with caln50 downloaded April 2020 (https://github.com/lh3/calN50) and a custom Python (https://www.python.org) script. The genome size provided to Canu and used for assembly statistics was based on values recorded in the Animal Genome Size Database (Gregory 2018). A C-value was not listed in the database for *C. melampygus*; we used 0.8 (782.4 Mbp) as an upper limit based on recorded genome size values for other *Caranx* species.

The transcriptome was assembled from Illumina RNA-seq reads from all eight tissues (i.e., brain, eye, fin, gill, heart, kidney, liver, and muscle). The transcripts were assembled using Trinity v2.6.6 (Grabherr et al. 2011). Both the genome and transcriptome assemblies were assessed for correctness using single-copy orthologs with BUSCO v4.0.6 (Simão et al. 2015) and the Actinopterygii subset of OrthoDB v10 (Kriventseva et al. 2019).

Computational Annotation

The MAKER v3.01.02-beta (Holt and Yandell 2011) pipeline was used to annotate the genome assembly. Generally speaking, annotation proceeded according to the process described in the most recent Maker Wiki tutorial (Holt and Yandell 2018). A custom repeat library was created using RepeatModeler v1.0.11 (Smit and Hubley 2008). The transcriptome assembly, genome assembly, and proteins from UniProtKB Swiss-prot (The Uniprot Consortium 2019;

Boutet et al. 2007) were used as input to MAKER to create initial annotations. Gene models based on these annotations were used to train the following *ab initio* gene predictors: AUGUSTUS v3.3.2 (Stanke et al. 2006; Stanke and Waack 2003) and SNAP downloaded 3 June 2019 (Korf 2004). AUGUSTUS was trained using BUSCO (Simão et al. 2015) as a wrapper; SNAP was trained without a wrapper. Genemark-ES v4.38 (Lomsadze et al. 2005; Brůna et al. 2020; Lomsadze et al. 2014) was also trained, though necessarily without the initial models from MAKER. These models were all provided to MAKER for a second round of structural annotation. The gene models based on those annotations were filtered with gFACs v1.1.1 (Caballero and Wegrzyn 2019) and again provided to AUGUSTUS and SNAP. As Genemark-ES does not accept initial gene models, it had no need to be run again. The gene models from the *ab initio* gene predictors were again provided to MAKER for a third and final round of annotation. Functional annotations were added using MAKER accessory scripts, the BLAST+ Suite v2.9.0 (Camacho et al. 2009; Altschul et al. 1990), and InterProScan v5.45-80.0 (Jones et al. 2014; Mitchell et al. 2019).

Demographic History

We inferred the historical demography of *C. melampygus* and its close relative, the giant trevally (*Caranx ignobilis*), by implementing the multiple sequentially Markovian coalescent (MSMC) model (Schiffels and Durbin 2014) to generate estimates of effective population size (N_e) over time. MSMC estimates the rate of coalescent events between two alleles at each locus along an unphased, single diploid genome. We used the self-corrected PacBio reads, filtered for scaffolds > 500Kbp, and applied additional cutoffs to ensure sufficient sequencing depth and quality using MSMC-tools downloaded 8 October 2020 (https://github.com/stschiff/msmc-tools;

Schiffels and Wang 2020; Mather et al. 2020). We used a draft *de novo* genome for *C. ignobilis* (Pickett et al. 2021). We ran MSMC v1.1.0 using the following time patterning parameters to estimate 20-time intervals and one free coalescent rate parameter: "1*2+16*1+1*2". We then generated 1,000 bootstrap estimates using a simulated dataset that randomly pulled, with replacement, 500Kbp long segments and arranged them into 52 segments per "chromosome." We generated 30 simulated "chromosomes" to construct artificial 780Mbp long genomes, reflecting the estimated size of the *C. melampygus* genome, to determine confidence intervals around N_e estimates. We used the same MSMC parameters for *C. ignobilis*, except that we generated 30 simulated "chromosomes" to construct 630Mbp long genomes to reflect the estimated size of the *C. ignobilis* genome (Pickett et al. 2021). After running MSMC, we converted population sizes and times into number of individuals and years, respectively, using a per site per generation mutation rate ($\mu = 3.7\ e^{-8}$) from another marine teleost species (Liu et al. 2016). For *C. melampygus*, we used a generation time of four, based on the average age of sexual maturity of *C. melampygus* (two) multiplied by two (Mather et al. 2020; Nadachowska-Brzyska et al. 2016). For *C. ignobilis*, we used a generation time of six, given an average age of three for sexual maturity in this species. The scripts to perform this analysis are available on GitHub (https://github.com/pickettbd/msmc-slurmPipeline) with supporting documentation.

Data Availability

Raw reads have been deposited in the National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA) under BioProject PRJNA670455. The genome assembly and annotations are associated with the same BioProject and can be found in GenBank under accession JAFELL010000000.

RESULTS AND DISCUSSION

Sequencing

Continuous long-read sequencing (PacBio) generated 4.45M reads with a total of 52.67Gbp, which is approximately 67x physical coverage of the genome. The mean and N50 read lengths were 11,834.678 and 19,264, respectively. The longest read was 116,429bp. The read length distribution is plotted in Figure 2. A summary of the results for the sequencing run is available in Table 1. This genome represents the first for the *Caranx* genus and ranks among the highest quality genomes available for Carangoidea in terms of N50 (Zhang et al. 2019).

RNA-seq from the eight tissues (i.e., brain, eye, fin, gill, heart, kidney, liver, and muscle) generated 257.47M pairs of reads totaling 114.61Gbp. Across all eight tissues, the mean and N50 read lengths were 222.6 and 249, respectively. The combined results from all eight tissues are represented in Table 1, while the results from each tissue are made available in Table 2.

PacBio CLR Error Correction

The self-correction strategy reduced the number of reads from 4.45M to 1.77M and the total number of bases from 52.67Gbp to 29.6Gbp for an approximate physical coverage of 37.8x. The mean and N50 read lengths were changed from 11,835 and 19,264 to 16,769 and 19,027, respectively. The longest read was 78,163 bases. The distribution of read lengths can be viewed in Figure 2.

Genome Assembly and Scaffolding

The initial assembly from Canu was comprised of 3.6K contigs with a total assembly size of 711Mbp. The mean contig length, N50, NG50, and maximum contig length were 198.8Kbp, 1.5Mbp, 1.2Mbp, and 8.9Mbp, respectively. The L50 was 120, and the LG50 was 147. The auNG was 1.93M. The scaffolding with the RNA-seq data joined some contigs together, reducing the sequence count to 3.3K (-8.08%). The number of bases, excluding unknown bases (Ns), was unchanged; however, it is important to note that when Rascaf creates gaps while ordering and orienting contigs, it always uses a gap size of 17bp to represent gaps of unknown size. The result in this case was adding 4.9Kbp of Ns, which means 289 gaps were created. These gaps were spread across 254 scaffolds. No scaffold had more than three gaps (four contigs ordered and oriented together). The mean scaffold length, scaffold N50, and scaffold NG50 increased by 17.5Kbp (8.79%), 213.8Kbp (12.70%), and 156.7Kbp (11.75%), respectively. Coupled with these increases were decreases of 13 (10.33%) and 17 (11.56%) in the L50 and LG50, respectively. The maximum scaffold length was unchanged from the maximum contig length. The auNG increased to 2.14M (+11.05%). Table 3 summarizes the assembly continuity statistics, and the area under the NG-curve (auNG) is visualized in Figure 3.

The assembly correctness, as assessed with single-copy orthologs, was also evaluated at the contig and scaffold level. The results suggest that the modifications made to the primary Canu-based assembly from scaffolding did not significantly impact the correct assembly of single-copy orthologs. The final set of scaffolds had 3,474 complete single-copy orthologs (95.5% of 3,640 from the ODB10 Actinopterygii set). Of these 89.8% (3,267) were present in the assembly only once, and 6.4% (207) were present more than once. Twenty-two (0.6%) and 144 (3.9%) single-copy orthologs were fragmented in and missing from the assembly, respectively.

Transcriptome Assembly & Computational Annotation

The transcriptome assembly generated by Trinity was comprised of 680K sequences with a mean sequence length of 1,171bp. The N50 and L50 were 2.4Kbp and 89K, respectively. The N90 and L90 were, respectively, 434bp and 419K. Of the 3,640 single-copy orthologs in the ODB10 Actinopterygii set, 93.3% (3,399) were complete; 33.8% (1,148) of which were present only once in the transcript set. 112 (3.1%) single-copy orthologs were fragmented in the transcript set, 129 (3.6%) were missing. Computational structural and functional annotation using the transcriptome assembly and the MAKER pipeline yielded 32.9K protein-coding genes. Of these, 21.8K and 20.7K have annotated 5′ and 3′ UTRs, respectively. 2.3K tRNA genes were also identified. The annotations are available in GFF3 format alongside the assembly.

Population Demography

Results of MSMC modeling indicated a gradual increase in effective population size (N_e) of both *C. melampygus* and *C. ignobilis* beginning around 150 kya, with strong fluctuations in *C. melampygus* population sizes between ~30-75 kya (Fig. 4). The increase in N_e was greater for *C. melampygus* than *C. ignobilis*. Our observed corroborate a previous demographic analysis of both species from Hawai‘i using mitochondrial loci, which also recovered evidence of population expansion compared to *C. ignobilis* (Santos et al. 2011). Other demographic components of wild populations (e.g., population structure, nonrandom mating, selection) are also known to affect estimates of coalescent rates (Mazet et al. 2016). For example, decreases in sea level have been linked to the isolation of marine populations (Cacciapaglia et al. 2021; Norris and Hull 2012), which would lead to demographic changes such as population structure and nonrandom mating. Sea levels decreased globally from the beginning of the Upper Pleistocene

(~129 kya) until the last glacial maximum (~19–26 kya), with several fluctuations in-between caused by glacial-interglacial cycles (Grant et al. 2014). Moreover, ocean circulation patterns were weaker during glacial periods (Rahmstorf 2002), which would limit connectivity between populations of marine fishes such as *C. melampygus* and *C. ignobilis* that disperse primarily via pelagic larval drifting.

Recent evidence suggests *C. melampygus* and *C. ignobilis* individuals are a genetically unique population in Hawai‘i (Glass et al. 2021). During the last glacial maximum, exposed limestone bridges linked the Hawaiian Islands of Maui, Lāna‘i, and Moloka‘i and supported reef habitats which became drowned after sea levels began rising (Grigg et al. 2002). These limestone reef features may have created increased habitat availability in Hawai‘i during periods of glaciation and supported population expansion. Notably, these species are large-bodied and associated with coastal habitats, including rock and coral reefs, but are not reef-obligate. Overall, some reef fishes exhibit evidence of dramatic declines in population size during glaciation periods (Gaither et al. 2010), whereas others exhibit evidence of population expansion similar to what is reported here for *C. melampygus* (Delrieu-Trottin et al. 2017). An analysis of demographic history for *C. melampygus* individuals from the widespread, Indo-West Pacific population, and individuals of *C. ignobilis* from other identified populations (Glass et al. 2021) would allow us to compare population expansion and contractions over time and assess how sea level changes may have affected *C. melampygus* and *C. ignobilis* differently across the Indo-Pacific.

Conclusion

The assembled genome of *Caranx melampygus* represents the first whole-genome assembly and annotation for the genus *Caranx* and second in the clade Carangini, the most speciose subclade of Carangoidea. The high quality of this reference genome builds on previous carangoid whole genome datasets and is important for delineating stock structure and demographic history of *C. melampygus*, especially given evidence of a unique genetic lineage in Hawai‘i. The bluefin trevally genome is also a valuable resource for comparative phylogenomic studies of carangoid fishes.

AUTHOR CONTRIBUTIONS

JRG: Data Curation; Formal Analysis; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **JSKK:** Conceptualization; Funding Acquisition; Investigation; Supervision; Resources; Writing - Review & Editing. **BDP:** Conceptualization; Data Curation; Formal Analysis; Investigation; Methodology; Software; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **PGR:** Funding Acquisition; Supervision; Resources; Writing - Review & Editing.

ACKNOWLEDGEMENTS

We thank the Brigham Young University DNA Sequencing Center (https://dnasc.byu.edu) and Office of Research Computing (https://rc.byu.edu) for their continued support of our research. For creating the beautiful illustration (Fig. 1), we thank the artist, Tim Johnson (https://timjohnsongallery.com).

FUNDING

None Declared.

CONFLICT OF INTEREST

None Declared.

ORCIDS

Brandon D. Pickett: 0000-0001-8235-4440

Jessica R. Glass, Ph.D.: 0000-0002-9843-1786

Perry G. Ridge, Ph.D.: 0000-0001-6944-2753

John S. K. Kauwe, Ph.D.: 0000-0001-8641-2468

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Sequencing Information. The results from each type of DNA and RNA sequencing from *Caranx melampygus*. PE= Paired-end reads. SMRT=Single-Molecule, Real-Time sequencing. CLR=Continuous Long-reads.

Company	Illumina	PacBio
Instrument	Hi-Seq 2500	Sequel I
Mode	Rapid Run	NA
Sequencing Type	PE	SMRT, CLR
Duration	250 cycles	30 hours
Specimen	1	1
Tissues	Brain, Eye, Fin, Gill, Heart, Kidney, Liver, Muscle	Heart
Molecule	RNA	DNA
Millions of Read(Pair)s	257.47	4.45
Mean Read Length (bp)	222.6	11,834.7
Read N50 (bp)	249	19,264
Nucleotides (Gbp)	114.61	52.67

Table 2. RNA Sequencing Details per Tissue. The results of RNA sequencing for each tissue from one *Caranx melampygus* individual. The eight tissues were spread across two lanes and run on an Illumina Hi-Seq 2500 in Rapid Run mode for 250 cycles to generate paired-end reads. Unless otherwise specified, lengths of nucleotide sequences are measured in base pairs (bp).

	Millions of Read Pairs	**Mean Read Length**	**Read N50**	**Nucleotides (Gbp)**
Brain	31.3	219.8	249	13.76
Eye	37.96	219.9	249	16.7
Fin	33.02	219.9	249	14.52
Gill	28.97	225.4	249	13.06
Heart	32.98	228.9	249	15.09
Kidney	32.51	222.5	249	14.47
Liver	30.10	224.6	249	13.52
Muscle	30.63	220.3	249	13.49
All	**257.47**	**222.6**	**249**	**114.61**

Table 3. Continuity Statistics. Continuity statistics for the *Caranx melampygus* genome assembly at the contig and scaffold level. Note that the auNG value is the area under the NG-curve and is unitless. Unless otherwise specified, all nucleotide sequences and gaps are measured in base pairs (bp).

	Contigs	Scaffolds
Sequences	3,577	3,288
Known Bases	710.963 Mbp	710.963 Mbp
Mean Length	198,759.666	216,229.722
Max. Length	8,932,605	8,932,605
NG50	1,176,926	1,333,605
NG90	24,428	24,595
LG50	147	130
LG90	3,179	2,892
auNG	1,927,338	2,140,376
Sequences with Gaps	-	254
Gaps	-	289
Unknown Bases	-	4,913
Mean Gap Length	-	17

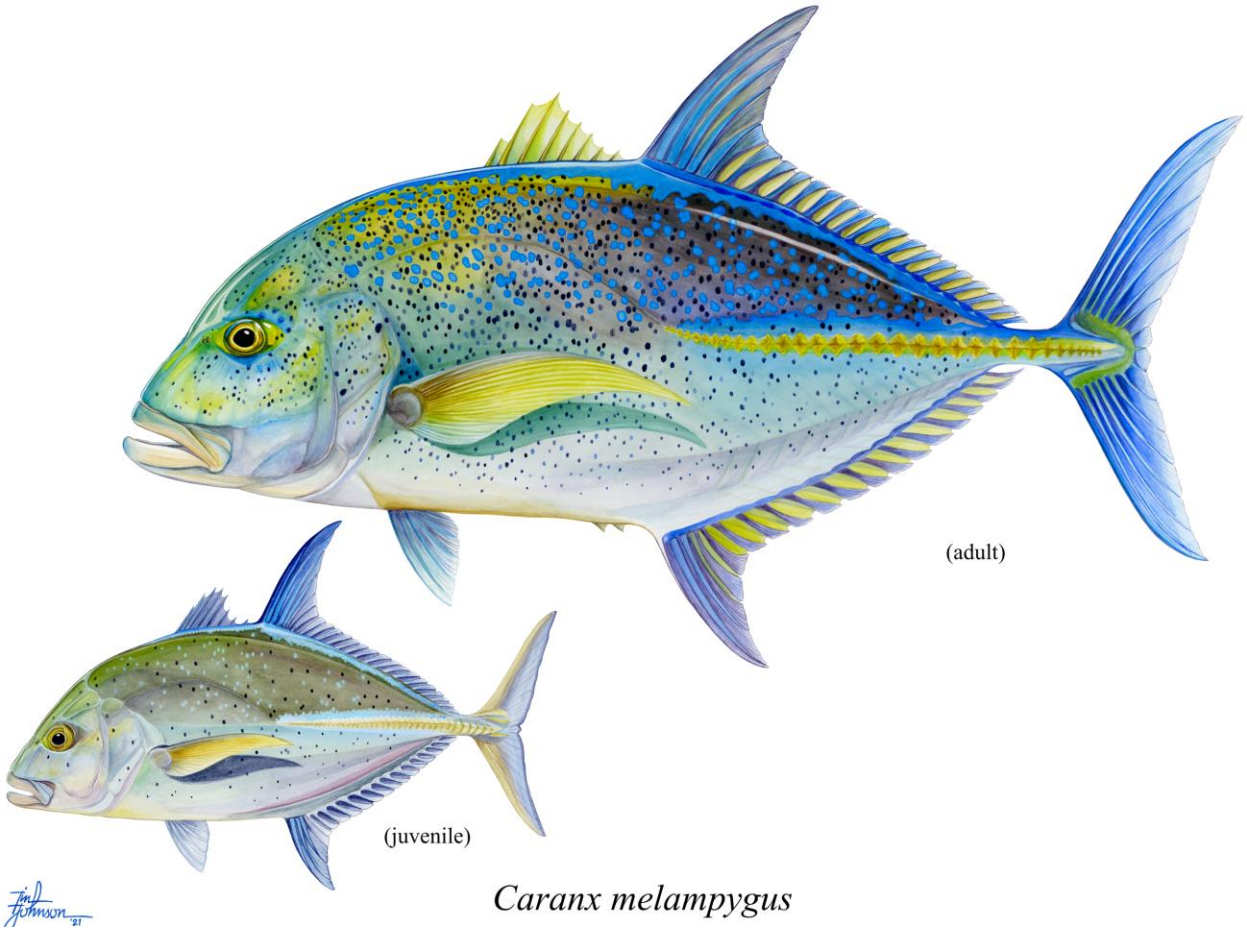


Figure 1. Bluefin trevally (*Caranx melampygus*) adult and juvenile. Quantitative morphological data for this illustration of *C. melampygus* were obtained primarily from (Heemstra et al. 2021). These were then evaluated by the artist who selected specific values for details such as number of lateral line scutes (32), number of rays (23) and spines (8) in the dorsal fin, and number of rays (19) and spines (2) in the anal fin. Each of these was portrayed in the illustration to be near the middle of the ranges reported. Illustration copyright: Tim Johnson, used with permission.

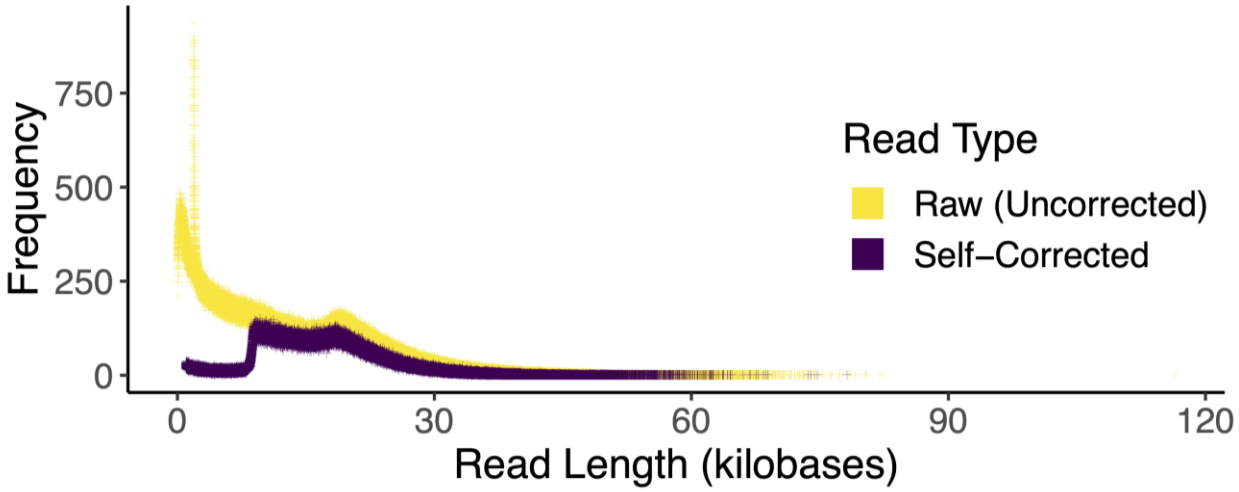


Figure 2. Frequency of Pacific Biosciences Read Lengths. The change in read length distribution is demonstrated as reads are corrected. The dramatic shift from raw to corrected reads is evident.

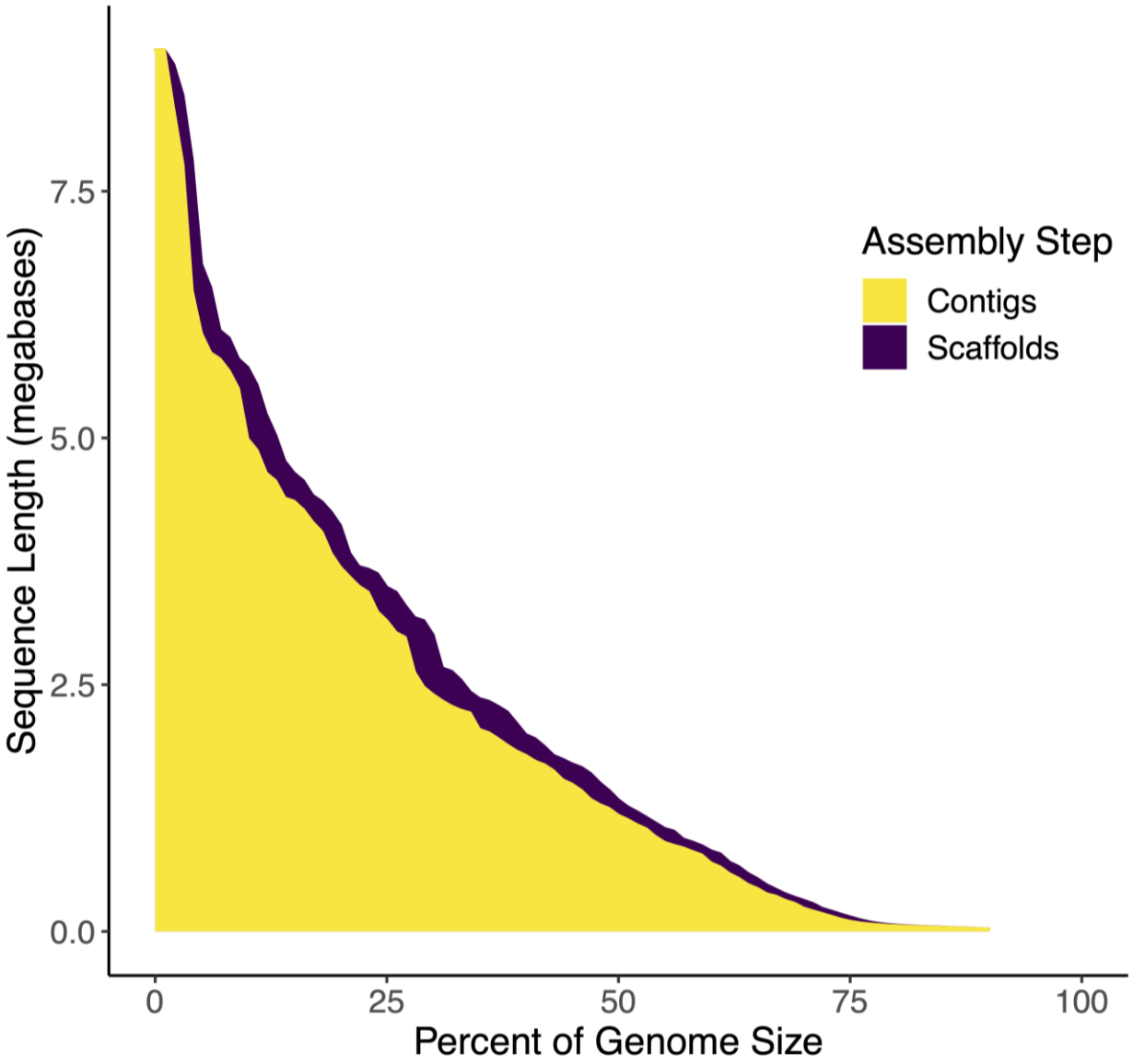


Figure 3. Area Under the NG-curve (auNG) for each Assembly Step. The NG-curve and the area under it are plotted for the contigs and scaffolds. This visually demonstrates that the scaffold NGx is equal or larger at any value of x (i.e., percent of the genome size). As these scaffolds were generated with only RNA-seq data, the difference is not as dramatic as it might be with another data type (e.g., Hi-C).

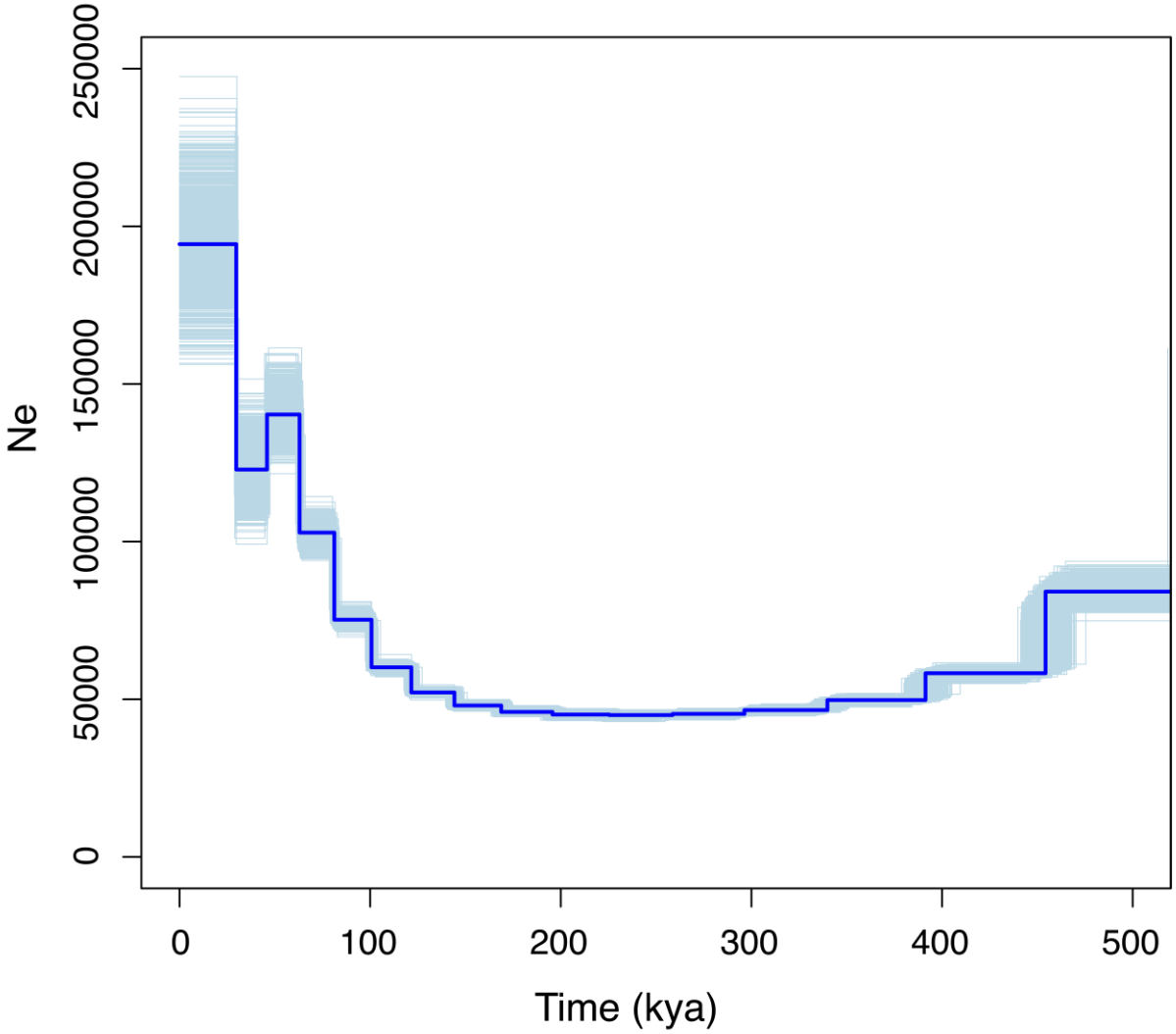


Figure 4. MSMC Analysis of Demographic History. Inferred demographic history of *C. melampygus* over time using MSMC. The dark blue line represents median effective population size (N_e) estimates. The light blue lines indicate 1,000 individual bootstrap replicates.

LITERATURE CITED

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, 1990 Basic Local Alignment Search Tool. *J. Mol. Biol.* 215:403-410.

Araki, K., J.-y. Aokic, J. Kawase, K. Hamada, A. Ozaki *et al.*, 2018 Whole Genome Sequencing of Greater Amberjack (*Seriola dumerili*) for SNP Identification on Aligned Scaffolds and Genome Structural Variation Analysis Using Parallel Resequencing. *Int. J. Genomics* 2018:7984292.

Blaber, S. J. M., and D. P. Cyrus, 1983 The biology of Carangidae (Teleostei) in Natal estuaries. *J. Fish Biol.* 22 (2):173-188.

Boutet, E., D. Lieberherr, M. Tognolli, M. Schneider, and A. Bairoch, 2007 UniProtKB/Swiss-Prot: The Manually Annotated Section of the UniProt KnowledgeBase, pp. 89-112 in *Plant Bioinformatics: Methods and Protocols*, edited by D. Edwards. Humana Press, Totowa, NJ.

Brůna, T., A. Lomsadze, and M. Borodovsky, 2020 GeneMark-EP+: eukaryotic gene prediction with self-training in the space of genes and proteins. *NAR Genom. Bioinform.* 2 (2):lqaa026.

Caballero, M., and J. Wegrzyn, 2019 gFACs: Gene Filtering, Analysis, and Conversion to Unify Genome Annotations Across Alignment and Gene Prediction Frameworks. *Genomics Proteomics Bioinformatics* 17 (3):305-310.

Cacciapaglia, C. W., M. B. Bush, ., and R. van Woesik, 2021 Legacies of an ice-age world may explain the contemporary biogeographical provinces of corals. *Frontiers of Biogeography* In press.

Camacho, C., G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos *et al.*, 2009 BLAST+: architecture and applications. *BMC Bioinform*. 10:421.

Cuvier, G., 1833 *Histoire naturelle des poissons* vol. 9. Levrault, Paris.

Delrieu-Trottin, E., S. Mona, J. Maynard, V. Neglia, M. Veuille *et al.*, 2017 Population expansions dominate demographic histories of endemic and widespread Pacific reef fishes. *Sci. Reps.* 7 (1):40519.

Friedlander, A. M., and P. Dalzell, 2004 A review of the biology and fisheries of two large jacks, ulua (*Caranx ignobilis*) and omilu (*Caranx melampygus*), in the Hawaiian Archipeligo, pp. 171-185 in *Status of Hawai'i's coastal fisheries in the new millennium: Proceedings of the 2001 Fisheries Symposium*, edited by A. M. Friedlander. American Fisheries Society, Hawai'i Chapter, Honolulu, HI, USA.

Gaither, M. R., R. J. Toonen, D. R. Robertson, S. Planes, and B. W. Bowen, 2010 Genetic evaluation of marine biogeographical barriers: perspectives from two widespread Indo-Pacific snappers (*Lutjanus kasmira* and *Lutjanus fulvus*). *J. Biogeogr.* 37 (1):133-147.

Genomic Resources Development Consortium, S. R. Keller, D. M. Nelson, C. Pylant, S. R. Santos *et al.*, 2014 Genomic Resources Notes accepted 1 October 2013 – 30 November 2013. *Mol. Eco. Res*. 14 (2):435-436.

Gill, T. N., 1863 Synopsis of the carangoids of the eastern coast of North America. *Proc. Acad. Nat. Sci.* Philadelphia 14 (9 [2nd]):430-443.

Glass, J. R., S. R. Santos, J. S. K. Kauwe, B. D. Pickett, and T. J. Near, 2021 Phylogeography of two coastal marine predators (*Caranx ignobilis* and *Caranx melampygus*) across the Indo-Pacific. *Bull. Mar. Sci.* 97 (2):257-280.

Grabherr, M. G., B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson *et al.*, 2011 Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat. Biotechnol.* 29 (7):644-652.

Grant, K. M., E. J. Rohling, C. B. Ramsey, H. Cheng, R. L. Edwards *et al.*, 2014 Sea-level variability over five glacial cycles. *Nat. Commun*. 5 (1):5076.

Gregory, T. R., 2018 Animal Genome Size Database. http://www.genomesize.com.

Grigg, R., E. Grossman, S. Earle, S. Gittings, D. Lott *et al.*, 2002 Drowned reefs and antecedent karst topography, Au'au Channel, S.E. Hawaiian Islands. *Coral Reefs* 21 (1):73-82.

Harrington, R. C., B. C. Faircloth, R. I. Eytan, W. L. Smith, T. J. Near *et al.*, 2016 Phylogenomic analysis of carangimorph fishes reveals flatfish asymmetry arose in a blink of the evolutionary eye. *BMC Evol. Biol.* 16:224.

Heemstra, P. C., E. Heemstra, D. A. Ebert, W. Holleman, and J. R. Randal, 2021 *Coastal Fishes of the Western Indian Ocean*. Makhanda, South Africa: National Research Foundation - South African Institute for Aquatic Biodiversity (NRF-SAIAB).

Holt, C., and M. Yandell, 2011 MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinform.* 12:491.

Holt, C., and M. Yandell, 2018 MAKER Tutorial for WGS Assembly and Annotation Winter School 2018. http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER_Tutorial_for_WGS_Assembly_and_Annotation_Winter_School_2018.

Jones, P., D. Binns, H.-Y. Chang, M. Fraser, W. Li *et al.*, 2014 InterProScan 5: genome-scale protein function classification. *Bioinformatics* 30 (9):1236-1240.

Kim, D., B. Langmead, and S. L. Salzberg, 2015 HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12 (4):357-360.

Koepfli, K.-P., B. Paten, and S. J. O'Brien, 2015 The Genome 10K Project: A Way Forward. *Ann. Rev. Ani. Biosci.* 3:57-111.

Koren, S., B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman *et al.*, 2017 Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* 27 (5):722-736.

Korf, I., 2004 Gene finding in novel genomes. BMC Bioinform. 5:59.

Kriventseva, E. V., D. Kuznetsov, F. Tegenfeldt, M. Manni, R. Dias *et al.*, 2019 OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Res*. 47 (D1):D807-D811.

Kültz, D., 2015 Physiological mechanisms used by fish to cope with salinity stress. *J. Exp. Bio.* 218 (12):1907-1914.

Li, H., 2020 auN: a new metric to measure assembly contiguity in *Heng Li's Blog*.

Linnaeus, C., 1758 *Systema Naturæ* vol. 1. Stockholm, Sweden.

Liu, S., M. M. Hansen, and M. W. Jacobsen, 2016 Region-wide and ecotype-specific differences in demographic histories of threespine stickleback populations, estimated from whole genome sequences. *Mol. Ecol.* 25 (20):5187-5202.

Lomsadze, A., P. D. Burns, and M. Borodovsky, 2014 Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. *Nucleic Acids Res.* 42 (15):e119.

Lomsadze, A., V. Ter-Hovhannisyan, Y. O. Chernoff, and M. Borodovsky, 2005 Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Res. 33 (20):6964-6506.

Malmstrøm, M., M. Matschiner, O. K. Tørresen, K. S. Jakobsen, and S. Jentoft, 2017 Whole genome sequencing data and de novo draft assemblies for 66 teleost species. *Sci. Data* 4:160132.

Mather, N., S. M. Traves, and S. Y. W. Ho, 2020 A practical introduction to sequentially Markovian coalescent methods for estimating demographic history from genomic data. *Ecol. Evol.* 10 (1):579-589.

Mazet, O., W. Rodríguez, S. Grusea, S. Boitard, and L. Chikhi, 2016 On the importance of being structured: instantaneous coalescence rates and human evolution—lessons for ancestral population size inference? *Heredity* 116 (4):362-371.

Meyer, C. G., K. N. Holland, B. M. Wetherbee, and C. G. Lowe, 2001 Diet, resource partitioning and gear vulnerability of Hawaiian jacks captured in fishing tournaments. *Fish. Res.* 53 (2):105-113.

Mitchell, A. L., T. K. Attwood, P. C. Babbitt, M. Blum, P. Bork *et al.*, 2019 InterPro in 2019: improving coverage, classification and access to protein sequence annotations. *Nucleic Acids Res.* 47 (D1):D351-D360.

Moriwake, A. M., V. N. Moriwake, A. C. Ostrowski, and C.-S. Lee, 2001 Natural spawning of the bluefin trevally *Caranx melampygus* in captivity. *Aquaculture* 203 (1-2):159-164.

Nadachowska-Brzyska, K., R. Burri, L. Smeds, and H. Ellegren, 2016 PSMC analysis of effective population sizes in molecular ecology and its application to black-and-white Ficedula flycatchers. *Mol. Ecol.* 25 (5):1058-1072.

Norris, R. D., and P. M. Hull, 2012 The temporal dimension of marine speciation. *Evol. Ecol*. 26 (2):393-415.

Ozaki, A., and K. Araki, 2017 *Seriola quiqueradiata* isolate Squ1, whole genome shotgun sequencing project. *GenBank*. BDMU00000000.

Pickett, B. D., J. R. Glass, P. G. Ridge, J. S. K. Kauwe, 2021 Genome assembly of the marine apex predator, Giant Trevally (*Caranx ignobilis*). Chapter 4 herein.

Purcell, C. M., C. L. Chabot, M. T. Craig, N. Martinez-Takeshita, L. G. Allen *et al.*, 2015 Developing a genetic baseline for the yellowtail amberjack species complex, *Seriola lalandi* sensu lato, to assess and preserve variation in wild populations of these globally important aquaculture species. *Cons. Genetics* 16:1475-1488.

Rahmstorf, S., 2002 Ocean circulation and climate during the past 120,000 years. *Nature* 419 (6903):207–214.

Santos, S. R., Y. Xiang, and A. W. Tagawa, 2011 Population Structure and Comparative Phylogeography of Jack Species (*Caranx ignobilis* and *C. melampygus*) in the High Hawaiian Islands. *J. Hered.* 102 (1):47-54.

Schiffels, S., and R. Durbin, 2014 Inferring human population size and separation history from multiple genome sequences. *Nat. Genet.* 46 (8):919-925.

Schiffels, S., and K. Wang, 2020 MSMC and MSMC2: The Multiple Sequentially Markovian Coalescent, pp. 147-166 in *Statistical Population Genomics*, edited by J. Y. Dutheil. Springer US, New York, NY.

Simão, F. A., R. M. Waterhouse, P. Ioannidis, E. V. Kriventseva, and E. M. Zdobnov, 2015 BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* 31 (19):3210-3212.

Smit, A. F. A., and R. Hubley, 2008 RepeatModeler Open-1.0, http://www.repeatmasker.org.

Song, L., D. S. Shankar, and L. Florea, 2016 Rascaf: Improving Genome Assembly with RNA Sequencing Data. *Plant Genome* 9 (3):1-12.

Stanke, M., O. Schöffmann, B. Morgenstern, and S. Waack, 2006 Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinform*. 7:62.

Stanke, M., and S. Waack, 2003 Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics* 19 (Suppl. 2):ii215-ii225.

Sudekum, A. E., J. D. Parrish, R. L. Radtke, and S. Ralston, 1991 Life History and Ecology of Large Jacks in Undisturbed, Shallow, Oceanic Communities. *Fish. Bull*. 89 (3):493-513.

The Uniprot Consortium, 2019 UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.* 47 (D1):D506-D515.

Yasuike, M., Y. Iwasaki, I. Nishiki, Y. Nakamura, A. Matsuura *et al.*, 2018 The yellowtail (*Seriola quinqueradiata*) genome and transcriptome atlas of the digestive tract. *DNA Res.* 25 (5):547-560.

Zhang, D.-C., L. Guo, H.-Y. Guo, K.-C. Zhu, S.-Q. Li *et al.*, 2019 Chromosome-level genome assembly of golden pompano (*Trachinotus ovatus*) in the family Carangidae. *Sci. Data* 6:216.

Zhao, Z., and Y. Lu, 2006 Establishment and characterization of two cell lines from bluefin trevally *Caranx melampygus*. *Dis. Aquat. Organ.* 68 (2):91-100.

CHAPTER 4

Genome assembly of marine apex predator, Giant Trevally (*Caranx ignobilis*)

Brandon D. Pickett[1], Jessica R. Glass[2], Perry G. Ridge[1], John S. K. Kauwe[1,3]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[2]*South African Institute for Aquatic Biodiversity, Makhanda, South Africa*

[3]*Brigham Young University - Hawai'i, Laie, Hawai'i, USA*

ABSTRACT

Caranx ignobilis, commonly known as the kingfish or giant trevally, is a large, reef-associated apex predator. It is a prized sportfish, targeted heavily throughout its tropical and subtropical range in the Indian and Pacific Oceans, and it has drawn significant interest in aquaculture due to an unusual tolerance for freshwater. In this study, we present a high-quality nuclear genome assembly of a *C. ignobilis* individual from Hawaiian waters, which have recently been shown to host a genetically distinct population. The assembly has a contig NG50 of 7.3Mbp and scaffold NG50 of 46.3Mbp. Twenty-five of the 203 scaffolds contain 90% of the genome. We also present the raw Pacific Biosciences continuous long-reads from which the assembly was created. A Hi-C dataset (Dovetail Genomics Omni-C) and Illumina-based RNA-seq from eight tissues are also presented; the latter of which can be particularly useful for annotation and studies of freshwater tolerance. Overall, this genome assembly and supporting data is a valuable tool for ecological and comparative genomics studies of kingfish and other carangoid fishes.

BACKGROUND & SUMMARY

The "genomic revolution" continues to rapidly advance our understanding of human evolution, as well as the evolution of non-model organisms [1]. Comparative genomic approaches using whole genome datasets allow for new discoveries at every scale: from genome to chromosome to organism to entire clades of organisms. Genomic datasets for non-model marine teleost fishes, the most diverse clade of vertebrates, are invaluable for investigating evolutionary questions relating to adaptation, selection, genome duplication, and phylogenetic conservatism in vertebrates.

We present a high-quality genome assembly of the marine teleost, giant trevally (*Caranx ignobilis*; Carangiformes: Carangoidei; Fig. 1). This assembly serves as a valuable resource for the field of evolutionary biology, ecology, and phylogenetics. *Caranx ignobilis* is a member of the Carangini clade, the most specious subclade within Carangoidei. Carangoid fishes are known for their extreme diversity in morphology and ecology [2,3]. The giant trevally, specifically, is known to be highly tolerant of freshwater environments, leading to a surge of interest in this species for aquaculture [4-6] and making it an ideal candidate species to investigate linkages between genotype and phenotype in the context of freshwater adaptation by marine fishes [7,8]. *Caranx ignobilis* is an apex predator in tropical and subtropical reefs and coastal environments in the Indian and Pacific Oceans [9] and is heavily targeted by small-scale and recreational fisheries throughout its range. Understanding its evolutionary and ecological role in ecosystem structure and function is important for fisheries management and the protection of reef and coral ecosystems. Importantly, new putative populations of *C. ignobilis* in the Indian and Pacific Oceans have recently been described using genomic datasets [10]. A high-quality genome thus allows for the inference of demographic history, genomic signals of selection and adaption, and

comparative genomic studies with other Carangoid fishes, such as hybridization with the closely related bluefin trevally, *Caranx melampygus* [11].

For this *C. ignobilis* assembly, we present results using 58.25 Gb of Pacific Biosciences (PacBio) Single-molecule, Real-time (SMRT) sequencing data. Illumina paired-end sequencing data was also generated with libraries for both RNA-seq and Hi-C, totaling 347.6 Gb. Both were used for scaffolding purposes and are valuable datasets individually. The estimated genome size was 625.92 Mb [12,13], of which 96.7% is covered by known bases in the primary haploid assembly. In addition to being highly-contiguous, the genome assembly contained complete, unduplicated copies of >95% of expected single-copy orthologs, suggesting the assembly is reasonably accurate and complete. The assembly and supporting sequencing datasets are sufficiently high-quality to serve as a valuable resource for a variety of prospective comparative and population genomics studies.

METHODS

An overview of the methods used in this study is provided here. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided in the Supplementary Bioinformatics Methods (Supplementary File 1; Appendix 5 herein).

Sample Acquisition & Sequencing

Blood, brain, eye, fin, gill, heart, kidney, liver, and muscle tissues from one *C. ignobilis* individual were collected off the coast of O‘ahu (near Kaneohe, Hawai‘i, USA) in April 2019. Blood was preserved in EDTA, and other tissue samples were flash-frozen in liquid nitrogen. All

samples were packaged in dry ice for transportation to Brigham Young University (BYU; Provo, Utah, USA) for storage at -80°C until sequencing. The blood sample was used to create the Omni-C dataset. All non-blood tissue samples were used for short-read RNA sequencing; the heart tissue was also used for long-read DNA sequencing.

DNA was prepared for long-read sequencing with a Pacific Biosciences (PacBio; Menlo Park, California, USA; https://www.pacb.com) SMRTbell Library kit, adhering to the following protocol: "Procedure & Checklist - Preparing gDNA Libraries Using the SMRTbell Express Template Preparation Kit 2.0". Continuous long-read (CLR) sequencing was performed on seven SMRT cells for a 10-hour movie on the PacBio Sequel at the BYU DNA Sequencing Center (DNASC; https://dnasc.byu.edu), a PacBio Certified Service Provider. RNA was prepared with Roche (Basel, Switzerland; https://sequencing.roche.com) KAPA Stranded RNA-seq kit, following recommended protocols. Paired-end sequencing was performed in High Output mode for 125 cycles with the eight samples across two lanes on the Illumina (San Diego, California, USA; https://www.illumina.com) Hi-Seq 2500 at the DNASC. Finally, the "Omni-C Proximity Ligation Assay Protocol" version 1.0 was followed using a Dovetail Genomics Omni-C kit to prepare for Illumina Paired-end sequencing. Adapters were provided by Integrated DNA Technologies, and sequencing proceeded in Rapid Run mode for 250 cycles in one lane on an Illumina Hi-Seq 2500.

Sequence Assembly, Duplicate Purging, and Scaffolding

The PacBio CLR reads were self-corrected and assembled with Canu v1.8 [14]. To get a haploid representation of the genome, duplicates were purged with purge_dups v1.2.5 [15]. The primary set of 329 contigs was selected for scaffolding with Omni-C data, which required reads

to be mapped to the assembly before determining how to order and orient the contigs. The Omni-C reads were aligned following the Arima Genomics (San Diego, California, USA; https://arimagenomics.com) Mapping Pipeline commit #2e74ea4 (https://github.com/ArimaGenomics/mapping_pipeline), which relied on BWA-MEM2 v2.1 [16,17], Picard v2.19.2 [18], and SAMtools v1.9 [19]. BEDTools v2.28.0 [20] was used to prepare the Omni-C alignments for scaffolding with SALSA commit #974589f [21]. Before the scaffolding step was performed, SALSA cleaned the assembly by breaking mis-assemblies as determined by Omni-C read mappings. This set of contigs was then used simultaneously for both the remainder of the SALSA pipeline and for scaffolding with Rascaf v1.0.2 commit #690f618 [22] using the RNA-seq data from all tissues aligned using HiSat v0.1.6-beta [23]. The two sets of scaffolds were combined using custom Python (https://www.python.org) scripts, which used the Omni-C scaffolds as a starting point and added compatible joins from the RNA-seq evidence. Contamination was removed from the final set of scaffolds as identified during the NCBI submission process; all gaps were also adjusted to a fixed size (100 Ns).

Genome Assembly Validation

At each phase of the assembly, continuity statistics, e.g., N50 and auN, were calculated with caln50 commit #3e1b2be (https://github.com/lh3/calN50) and a custom Python script (Table 3). The genome size (625.92 Mb) provided to Canu and used for assembly statistics was based on the C-value of 0.64 from Hardie and Hebert [12] as recorded in the Animal Genome Size Database [13]. Assembly correctness was also assessed at each phase using single-copy orthologs from the Actinopterygii set of OrthoDB v10 [24] as identified by BUSCO v4.0.6 [25] (Table 4). The scaffolds were visually inspected using a Hi-C contact matrix created with PretextView v0.1.4

(https://github.com/wtsi-hpag/PretextView) and PretextMap v0.1.4 (https://github.com/wtsi-hpag/PretextMap) with SAMtools v1.10 [19].

Visual comparisons with other carangoid genomes were created for cursory validation and observation of general synteny. Dot plots were generated using Mashmap v2.0 commit #ffeef48 [26] (-f 'one-to-one' --pi 95 -s 10000) and a comparison of single-copy orthologs was created using ChrOrthLink commit #d29b10b after assessment with BUSCO v3.0.6 [25] using the Vertebrata set from OrthoDB v9 [27]. The genome assemblies obtained from NCBI for these analyses were the following (alphabetical order): *Caranx melampygus* (bluefin trevally) [11], *Echeneis naucrates* (live suckershark) [28,29], *Seriola dumerili* (greater amberjack) [28,29], *Seriola quinqueradiata* (yellowtail) [30,31], *Seriola rivoliana* (longfin yellowtail) [32], *Trachinotus ovatus* (golden pompano) [33,34], and *Trachurus trachurus* (Atlantic horse mackerel) [35,36].

TECHNICAL VALIDATION

Sequencing

Continuous long-read sequencing (PacBio) generated 3.74M reads with a total of 58.25 Gbp, which is approximately 93x physical coverage of the genome. The mean and N50 read lengths were 15,591.278 and 27,441, respectively. The longest read was 129,643bp. The read length distribution is plotted in Figure 2. A summary of the results for the sequencing run is available in Table 1. This genome represents the second for the *Caranx* genus and ranks highly in terms of N50 among available carangoid genomes [34,36].

RNA-seq from the eight tissues (i.e., brain, eye, fin, gill, heart, kidney, liver, and muscle) generated 435.99M pairs of reads totaling 108.30Gbp. Across all eight tissues, the mean and N50

read lengths were 124.21 and 125, respectively. The combined results from all eight tissues are represented in Table 1, while the results from each tissue are made available in Table 2. Omni-C sequencing generated 80.92 Gb of data across 169.1M read pairs. The N50 and mean read length were respectively 250 and 239.3. The Omni-C results are also represented with in Table 1 with the PacBio and RNA-seq data.

PacBio CLR Error Correction

The correction process reduced the number of reads from 3.74M to 656K and the total number of bases from 58.3Gbp to 23.9Gbp for an approximate physical coverage of 38.3x. The mean and N50 read lengths were changed from 15,591 and 27,441 to 36,475 and 40,065, respectively. The longest read was 126,321 bases. The distribution of corrected read lengths can be viewed relative to the raw read lengths in Figure 2.

Genome Assembly, Duplicate Purging, and Scaffolding

The initial assembly from Canu was comprised of 1.8K contigs with a total assembly size of 758Mbp. This was a diploid assembly in the sense that both haplotypes were present and intermixed, separated whenever a bubble in the assembly graph prevented a single reasonable contig. Duplicate purging to get a haploid representation of the genome (albeit with inevitable haplotype switching) and fixing mis-assemblies with evidence from Hi-C data yielded 343 contigs with a total assembly size of 605Mbp. The mean contig length, N50, NG50, and maximum contig length were to 1.8Mbp, 7.7Mbp, 7.3Mbp, and 19.6Mbp, respectively. The L50 was 23, and the LG50 was 25. The auN was to 8.55M. These values represent modest reductions

from the original Canu assembly (as expected), and they can be visualized in the area under the N-curve as presented in Figure 3. (Also see Table 3)

Paired-end Illumina reads, such as those produced from Hi-C or RNA-seq libraries can provide information to order and orient contigs into scaffolds, but they contain insufficient information to utilize for gap-filling procedures. Accordingly, the result on assembly statistics should increase length, decrease number of sequences, and leave the number of known bases unchanged. This pattern is evident in the assembly statistics from our iterative scaffolding procedure (Table 3). It is important to note that SALSA and Rascaf introduce gaps of unknown size, and they respectively use fixed runs of Ns of lengths 500 and 17 to represent such gaps. For submission to NCBI, these gaps were converted to a fixed length of 100 Ns, and the evidence for whether joins were supported by Hi-C data or RNA-seq data was submitted in an accompanying file in AGP format (https://www.ncbi.nlm.nih.gov/assembly/agp/AGP_Specification). The NCBI submission process also identified minor contaminants in some sequences, which were manually removed. The final set of scaffolds had an NG50 of 46.3Mbp and an auN of 42.6M (Fig. 3; Table 3). All joins are represented in a contact matrix (Fig. 4), which shows the Hi-C evidence for the assembly. Some joins are poorly supported by the Hi-C evidence, which is not surprising as some joins were made by RNA-seq evidence instead. Without manual curation, it is difficult to ascertain whether any individual such join is spurious.

The assembly correctness, as assessed with single-copy orthologs, was also evaluated at the contig and scaffold level (Table 4). The results suggest that the modifications made to the primary contig assembly from scaffolding did not significantly impact the correct assembly of single-copy orthologs. The final set of scaffolds had 3,546 complete single-copy orthologs (97.4% of 3,640 from the OrthoDB10 Actinopterygii set). Of these 85.7% (3,120) were present

in the assembly only once, and 11.7% (426) were present more than once. Twelve (0.3%) and 82 (2.3%) single-copy orthologs were fragmented in and missing from the assembly, respectively.

Comparison of Giant Trevally with Other Carangoid Genomes

We compared the *C. ignobilis* genome to published genomes of other carangoids spanning the carangoid phylogeny, including the live sharksucker (*Echeneis naucrates*) [28,29], golden pompano (*Trachinotus ovatus*) [33,34], yellowtail (*Seriola quinqueradiata*) [30,31], longfin yellowtail (*Seriola rivoliana*) [32], greater amberjack (*Seriola dumerili*) [37,38], and the more closely-related species: Atlantic horse mackerel (*Trachurus trachurus*) [35,36] and bluefin trevally (*Caranx melampygus*) [11]. We generated dot plots to visualize genome alignments and look for general synteny between the genomes (Fig. 5). Some structural variation can be seen, but additional analysis would be required to explore each of such further. We similarly compared the same assemblies by visualizing the grouping of single-copy orthologs plotted along the assemblies (Fig. 6). Large groupings of orthologs consistently appear together between genomes, though specific patterns become difficult to inspect at the genome scale when the contigs/scaffolds get small. The longest scaffolds in the *C. ignobilis* assembly have single-copy orthologs from more than on chromosome from other assemblies with chromosome number assigned, and this is evident with the nearby *E. naucrates*. If the relative sizes of the chromosomes from the *E. naucrates* assembly are taken as baseline truth, this calls into question whether some of the *C. ignobilis* RNA-seq scaffolding joins are valid. Karyotype analysis, additional sequencing data (e.g., Ultra-long Nanopore (Oxford, England, UK)), and/or more in-depth, one-on-one comparisons would help elucidate the structure. Ultimately, our results indicate the utility of this

genomic dataset for future comparative studies on genome structure and evolution within Carangiformes and marine teleosts more broadly.

DATA RECORDS

Raw reads have been deposited in the National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA) [39-48] under BioProject PRJNA670456 [49], BioSamples SAMN16516519-SAMN16516526 and SAMN16629462 [50-58]. The genome assembly is associated with the same BioProject under the "container" BioSample SAMN18021194 [59] and can be found in GenBank under accession JAFHLA000000000. See Table 5 for a complete list of datasets and their mapping to BioSamples.

CODE AVAILABILITY

No significant computer programs were generated in this work. Custom scripts referenced in the text are described in the Supplementary Bioinformatics Methods and/or are available on GitHub at https://github.com/pickettbd/caranx-ignobilis_assembly-paper_misc-scripts.

AUTHOR CONTRIBUTIONS

JRG: Funding Acquisition; Writing - Original Draft Preparation; Writing - Review & Editing. **JSKK:** Conceptualization; Funding Acquisition; Investigation; Supervision; Resources; Writing - Review & Editing. **BDP:** Conceptualization; Data Curation; Formal Analysis; Funding Acquisition; Investigation; Methodology; Software; Visualization; Writing - Original Draft

Preparation; Writing - Review & Editing. **PGR:** Funding Acquisition; Supervision; Resources; Writing - Review & Editing.

ACKNOWLEDGEMENTS

We thank the Brigham Young University DNA Sequencing Center (https://dnasc.byu.edu) and Office of Research Computing (https://rc.byu.edu) for their continued support of our research. We thank the artist, Elaine Heemstra, and the South African Institute for Aquatic Biodiversity (https://www.saiab.ac.za) for the use of the illustration (Fig. 1). We are grateful to the Vertebrate Genomes Project (https://vgp.github.io), specifically Erich Jarvis of The Rockefeller University (New York City, New York, USA) and Richard Durbin of the Wellcome Sanger Institute and University of Cambridge (Cambridge, England, UK), for providing access to the *Trachurus trachurus* genome assembly before an official publication was released. We thank Chul Lee of Seoul National University (Seoul, Republic of South Korea) and Ann McCartney and Arang Rhie of the National Institutes of Health – National Human Genome Research Institute (Bethesda, Maryland, USA) for helpful discussion about assembly validation and ChrOrthLink.

FUNDING

Illumina and the Brigham Young University DNA Sequencing Center granted an Illumina Pilot Award to BDP, JRG, and JSKK, resulting in complimentary sequencing for the RNA.

COMPETING INTERESTS

The authors declare no competing interests.

ORCIDS

Brandon D. Pickett: 0000-0001-8235-4440

Jessica R. Glass, Ph.D.: 0000-0002-9843-1786

Perry G. Ridge, Ph.D.: 0000-0001-6944-2753

John S. K. Kauwe, Ph.D.: 0000-0001-8641-2468

ADDITIONAL INFORMATION

Supplementary Information

Supplementary Bioinformatics Methods (Supplementary File 1) can be found in Appendix 5 herein.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Sequencing Information. The results from each type of DNA and RNA sequencing from *Caranx ignobilis*. PE= Paired-end reads. SMRT=Single-Molecule, Real-Time sequencing. CLR=Continuous Long-reads.

Company	Illumina	Illumina	PacBio
Instrument	Hi-Seq 2500	Hi-Seq 2500	Sequel I
Mode	High Output	Rapid Run	NA
Sequencing Type	PE	Omni-C, PE	SMRT, CLR
Duration	125 cycles	250 cycles	10 hours
Specimen	1	1	1
Tissues	Brain, Eye, Fin, Gill, Heart, Kidney, Liver, Muscle	Blood	Heart
Molecule	RNA	DNA	DNA
Millions of Read(Pair)s	435.99	169.11	3.74
Mean Read Length	124.2	239.3	15,591.3
Read N50	125	250	27,441
Nucleotides (Gb)	108.30	80.92	58.25

Table 2. RNA Sequencing Details per Tissue. The results of RNA sequencing for each tissue from one *Caranx ignobilis* individual. The eight tissues were spread across two lanes and run on an Illumina Hi-Seq 2500 in Rapid Run mode for 250 cycles to generate paired-end reads. Unless otherwise specified, lengths of nucleotide sequences are measured in base pairs (bp).

	Millions of Read Pairs	**Mean Read Length**	**Read N50**	**Nucleotides (Gb)**
Brain	45.59	124.17	125	11.32
Eye	52.02	124.26	125	12.93
Fin	50.13	124.16	125	12.45
Gill	55.56	124.22	125	13.80
Heart	57.87	124.29	125	14.39
Kidney	58.73	124.16	125	14.58
Liver	58.25	124.23	125	14.47
Muscle	57.84	124.16	125	14.36
All	**435.99**	**124.21**	**125**	**108.30**

Table 3. Continuity Statistics. Continuity statistics for the *Caranx ignobilis* genome assembly at the contig and scaffold level. The final set of scaffolds (far right column) is the same as "Scaffolds (SALSA + Rascaf" except that the contaminants were manually removed from the assembly and gaps were unified to 100 Ns. Note that the auNG value is the area under the NG-curve and is unitless. Unless otherwise specified, all nucleotide sequences and gaps are measured in base pairs (bp).

	Contigs	Contigs (purge_dups)	Contigs (purge_dups + SALSA)	Scaffolds (SALSA)	Scaffolds (SALSA + Rascaf)	Scaffolds
Sequences	1,804	329	343	240	209	203
Known Bases	757.523 Mb	605.140 Mb	605.140 Mb	605.140 Mb	605.140 Mb	605.115
Mean Length	0.420 Mb	1.839 Mb	1.764 Mb	2.521 Mb	2.895 Mb	2.981 Mb
Max. Length	23.990 Mb	23.990 Mb	19.607 Mb	32.157 Mb	89.251 Mb	89.251 Mb
NG50	7.412 Mb	7.412 Mb	7.261 Mb	23.385 Mb	46.318 Mb	46.303 Mb
NG90	1.097 Mb	0.950 Mb	0.700 Mb	1.386 Mb	1.410 Mb	1.410 Mb
LG50	24	24	25	12	5	5
LG90	103	105	114	39	25	25
auNG	9.090 M	9.051 M	8.549 M	19.716 M	42.606 M	42.600 M
Sequences with Gaps	-	-	-	40	35	35
Gaps	-	-	-	103	134	133
Unknown Bases	-	-	-	51,500	52,027	13,300
Mean Gap Length	-	-	-	500	388.261	100

Table 4. Summary BUSCO Results. Summary BUSCO results for the *Caranx ignobilis* genome assembly at the various contig and scaffold stages. Each value is the percentage of single-copy orthologs (n=3,640) in the Actinopterygii lineage dataset from OrthoDB v10.

	Contigs	Contigs (purge_dups)	Contigs (purge_dups + SALSA)	Scaffolds (SALSA)	Scaffolds (SALSA + Rascaf)	Scaffolds
Complete	97.6	97.5	97.6	97.2	97.3	97.4
Single Copy	85.9	95.9	96.0	95.7	95.7	95.8
Duplicated	11.7	1.6	1.6	1.5	1.6	1.6
Fragmented	0.3	0.6	0.5	0.5	0.5	0.5
Missing	2.1	1.9	1.9	2.3	2.2	2.1

Table 5. Database Information for Raw Sequences. All samples were collected from the same *Caranx ignobilis* specimen in April 2019 off the coast of O'ahu (near Kaneohe, Hawai'i, USA). They are combined under the BioProject PRJNA670456. The genome assembly is deposited in GenBank under accession JAFHLA000000000 with the "container" BioSample SAMN18021194.

Specimen	Tissue	BioSample Number	Sequencing Type	SRA Accession
1	Blood	SAMN16629462	Dovetail Omni-C	SRR13036356
1	Brain	SAMN16516519	Illumina RNA-seq	SRR13036363
1	Eye	SAMN16516520	Illumina RNA-seq	SRR13036362
1	Fin	SAMN16516521	Illumina RNA-seq	SRR13036361
1	Gill	SAMN16516522	Illumina RNA-seq	SRR13036360
1	Heart	SAMN16516523	Illumina RNA-seq	SRR13036359
1	Heart	SAMN16516523	PacBio CLR WGA	SRR13036357
1	Kidney	SAMN16516524	Illumina RNA-seq	SRR13036355
1	Liver	SAMN16516525	Illumina RNA-seq	SRR13036354
1	Muscle	SAMN16516526	Illumina RNA-seq	SRR13036353

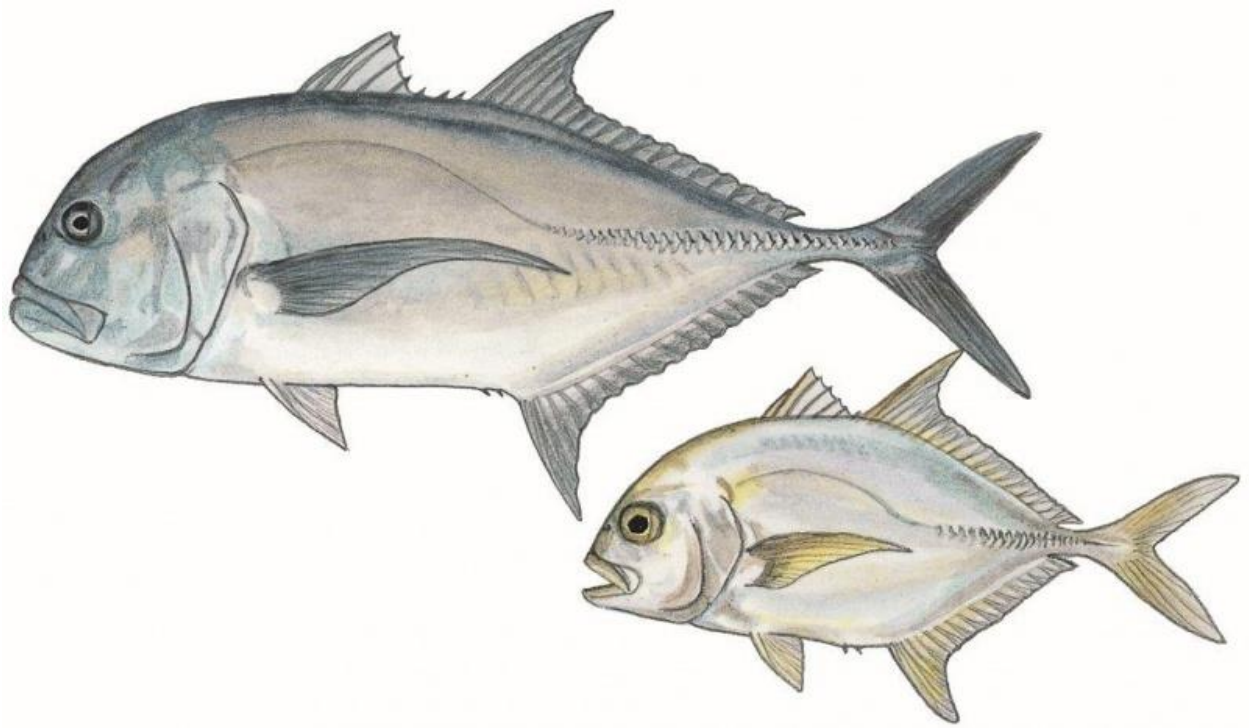


Figure 1. Giant trevally (*Caranx ignobilis*) adult and juvenile. Illustration by Elaine Heemstra, courtesy of the South African Institute for Aquatic Biodiversity.

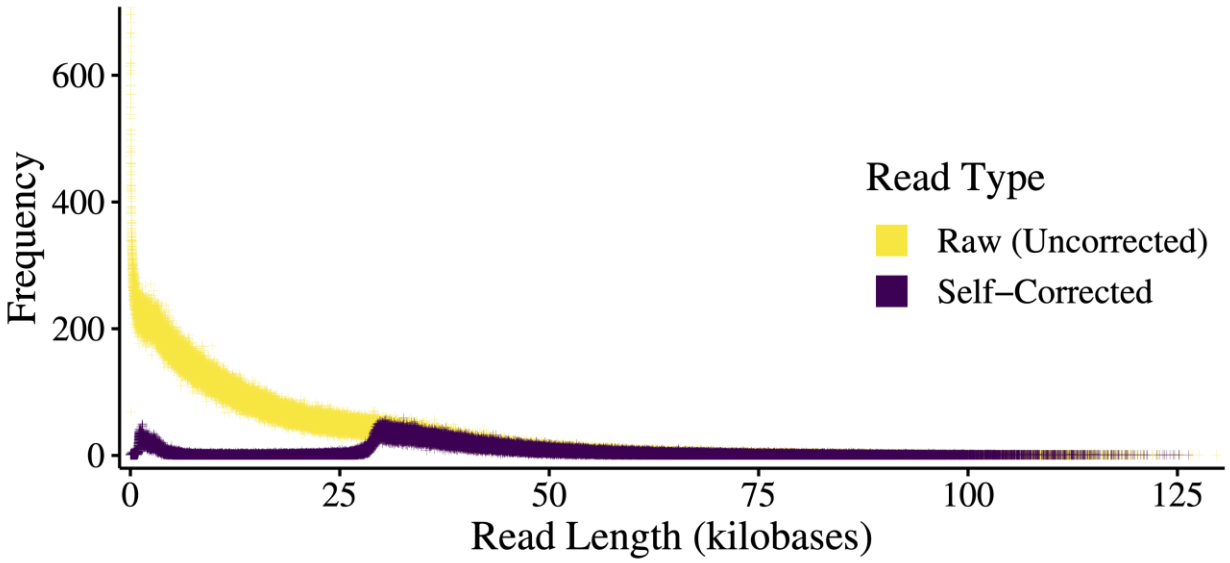


Figure 2. Frequency of Pacific Biosciences Read Lengths. The change in read length distribution is demonstrated as reads are corrected. The dramatic shift from raw to corrected reads is evident. Reads were corrected by consensus using the correction phase of Canu v1.8.

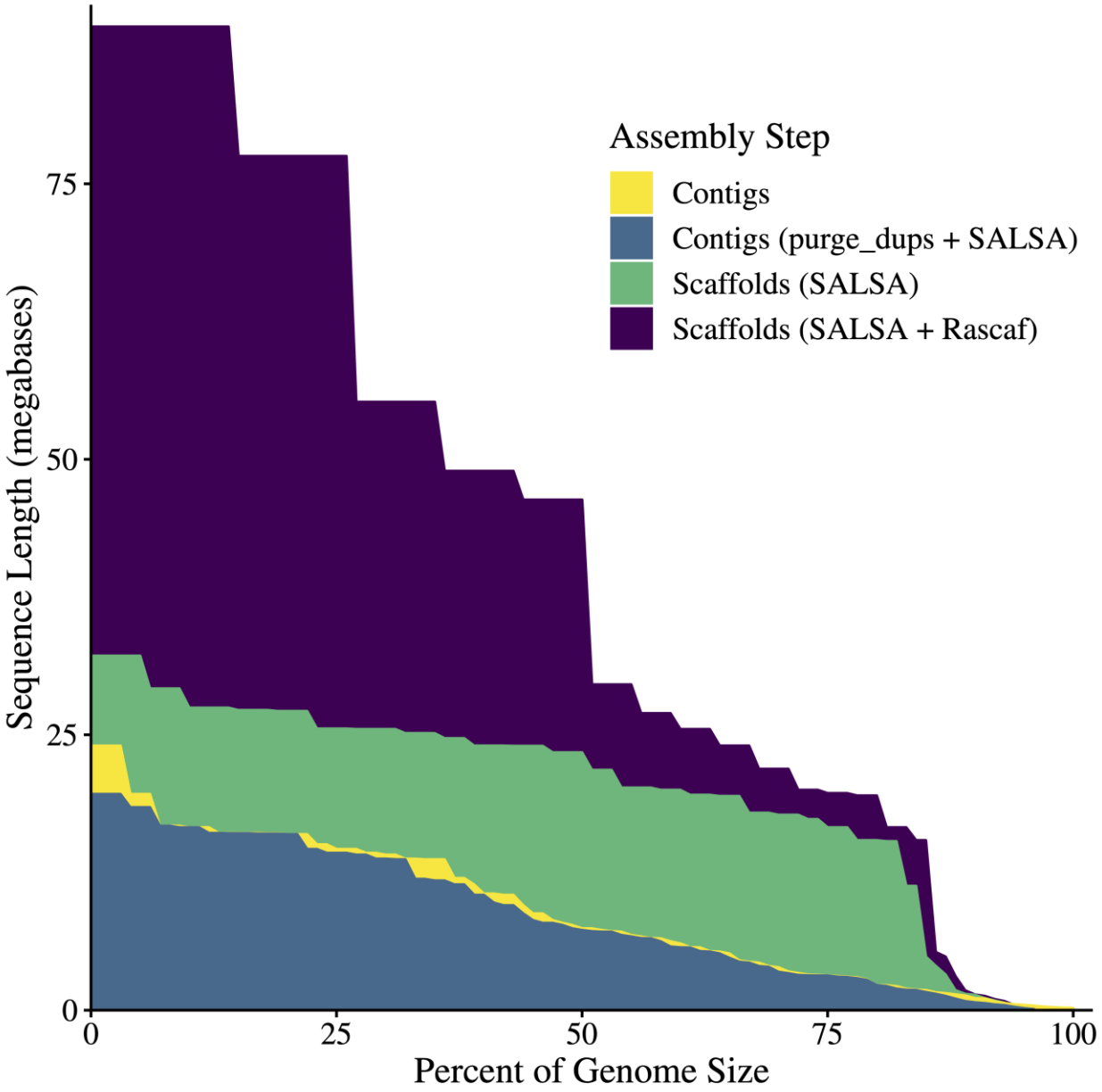


Figure 3. Area Under the NG-curve (auNG) for each Assembly Step. The NG-curve and the area under it are plotted for the contigs and scaffolds. This visually demonstrates increase in continuity from contigs to scaffolds. Scaffolding with RNA-seq data – which has minimal effect on its own (data not shown) – further increases the scaffold-level continuity. This plot also shows that duplicate purging and fixing mis-assemblies slightly reduced contig-level continuity, which is expected.

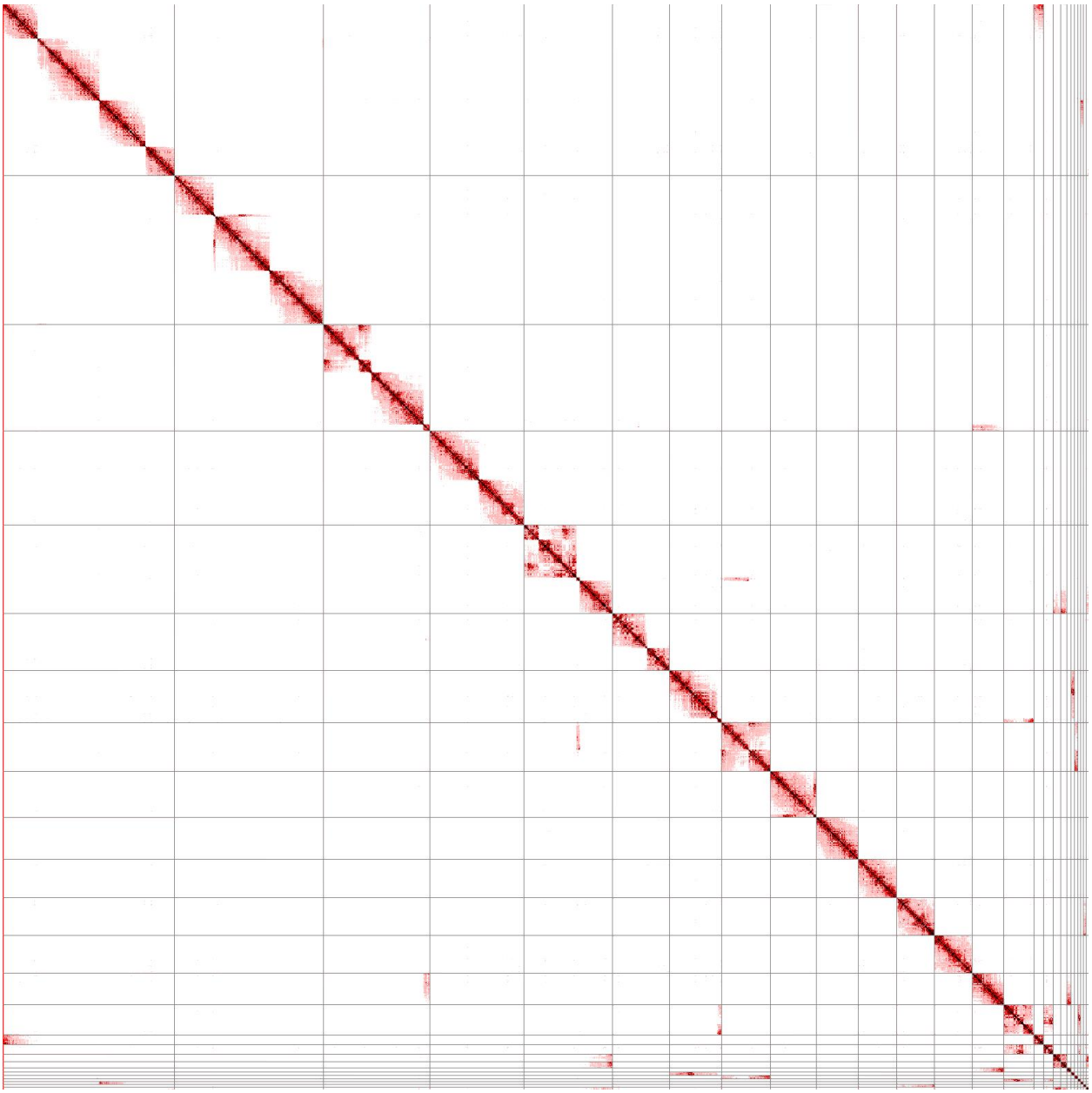


Figure 4. Hi-C Contact Matrix. In the context of scaffolding, Hi-C contact matrices show how correct the scaffolds are based on Hi-C alignment evidence. The longest 26 scaffolds are shown, ordered by descending length from top-left to bottom-right; grey lines show scaffold boundaries. Off-diagonal marks, especially those that are dark and large, are possible evidence of mis-assembly and/or incorrect scaffolding. Regions with sharp edges similar to where the grey lines appear, but without the grey lines (e.g., three such locations occur in the top-left square), are joins between contigs in that scaffold that lack Hi-C evidence. The lack of Hi-C alignment evidence could suggest that these joins are invalid, but evidence for these joins does exist from RNA-seq alignments. Detection of any spurious joins would, at a minimum, require manual curation. Such curation would enable additional adjustments that would fix minor issues evident from the contact matrix.

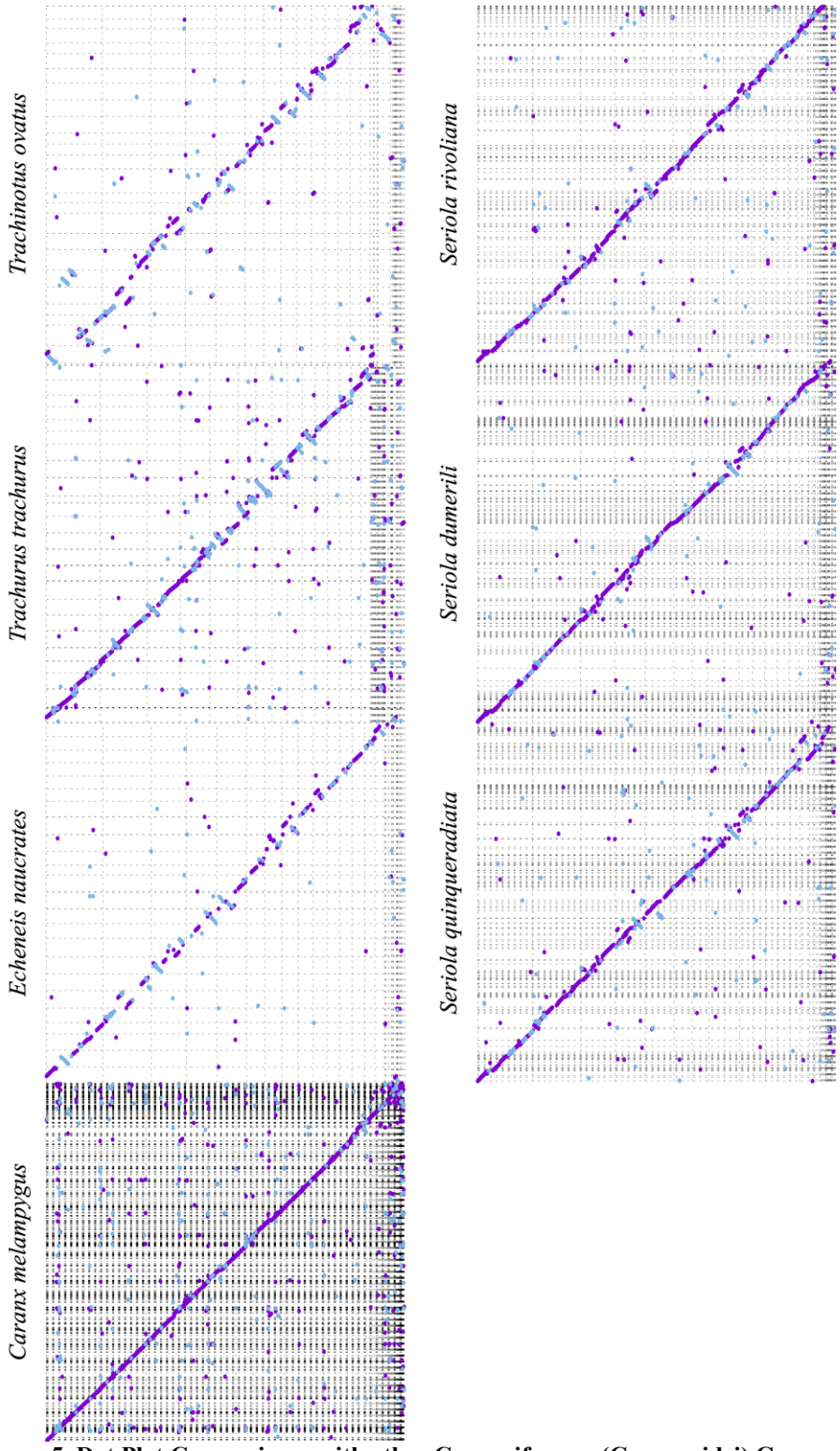


Figure 5. Dot Plot Comparisons with other Carangiformes (Carangoidei) Genomes. Dot plots show the relative continuity of the various segments of two genomes. The purple dots show segments that align in the positive orientation, blue in the negative. The x-axis is the *Caranx ignobilis* genome. The y-axes for each plot are other carangoid genomes. Dots off the diagonal indicate structural variation between the genome assemblies. For assemblies that did not have duplicates purged to reduce the assembly to pseudohaplotypes (*Caranx melampygus* and *Seriola* spp.), the extra dots are presumably the alignment to the secondary copy.

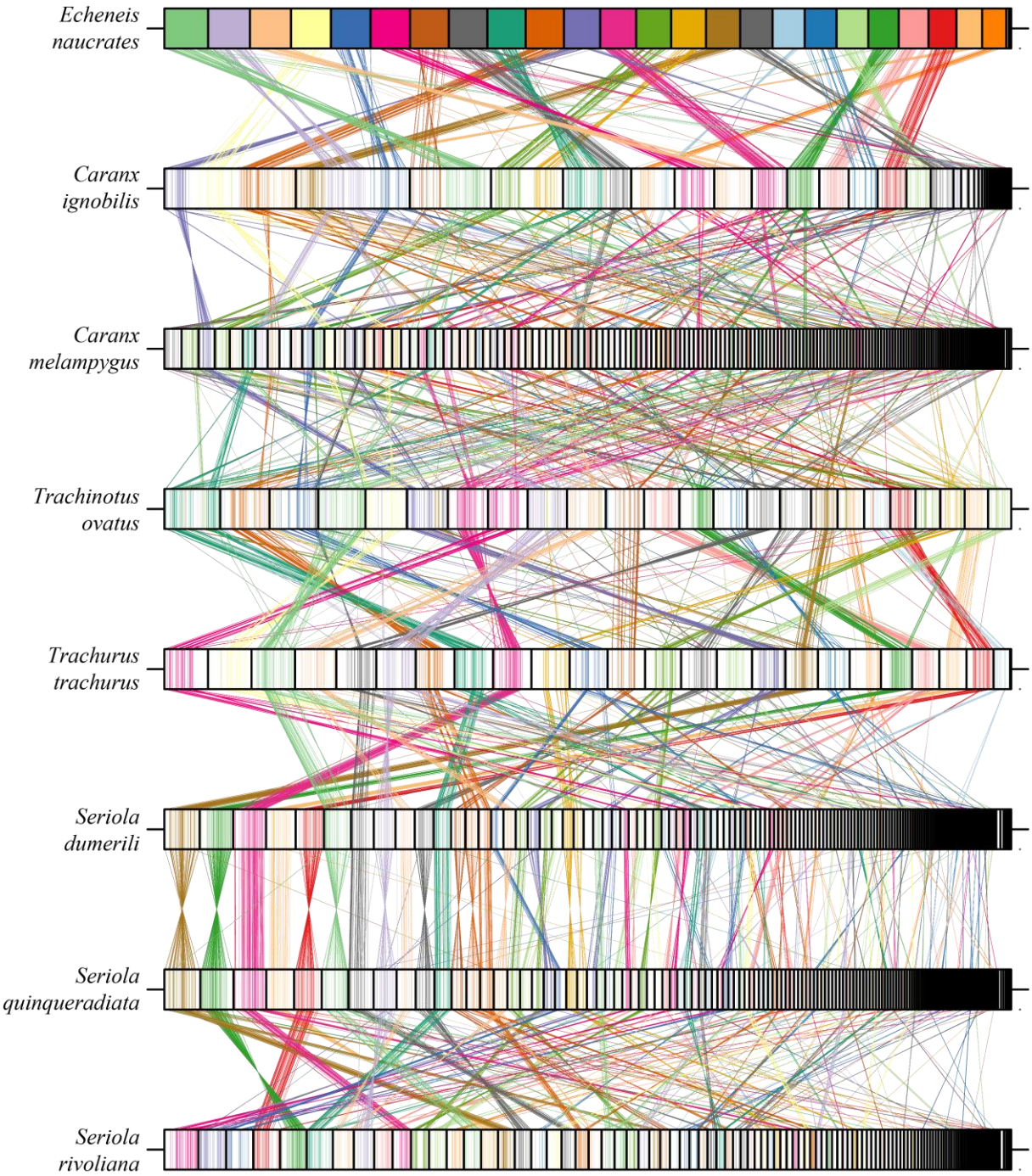


Figure 6. Single-copy Ortholog Comparisons with other Carangiformes (Carangoidei) Fishes. Single-copy orthologs from the Actinopterygii set of OrthoDB v9 were identified with BUSCO v3.0.6 and visualized using ChrOrthLink. "Chromosomes" (usually contigs or scaffolds) are ordered based on length. Comparisons are difficult to assess when "chromosome" sizes vary greatly, especially at the genome scale. Additional information could be gleaned when comparing genomes one-by-one with chromosomes ordered based on similarity. At this scale, however, it is clear that groupings of single-copy orthologs cluster together across genomes, suggesting orthology not just between these genes, but with general genomic structure within larger regions. The longest scaffolds in the *Caranx ignobilis* assembly have single-copy orthologs from more than one chromosome from other assemblies with chromosome number assigned, and this is evident with the *Echeneis naucrates* assembly. If the relative sizes of the chromosomes from the *E. naucrates* assembly are taken as baseline truth, this calls into question whether some of the *C. ignobilis* RNA-seq scaffolding joins are valid.

REFERENCES

1 Koonin, E. V., Aravind, L. & Kondrashov, A. S. The Impact of Comparative Genomics on Our Understanding of Evolution. *Cell* **101**, 573-576, doi:10.1016/s0092-8674(00)80867-3 (2000).

2 Price, S. A., Claverie, T., Near, T. J. & Wainwright, P. C. Phylogenetic insights into the history and diversification of fishes on reefs. *Coral Reefs* **34**, 997-1009, doi:10.1007/s00338-015-1326-7 (2015).

3 Frédérich, B., Marramà, G., Carnevale, G. & Santini, F. Non-reef environments impact the diversification of extant jacks, remoras and allies (Carangoidei, Percomorpha). *Proceedings of the Royal Society B: Biological Sciences* **283**, 20161556, doi:10.1098/rspb.2016.1556 (2016).

4 Abdussamad, E. M., Kassim, H. M. & Balasubramanian, T. S. Distribution, biology and behaviour of the giant trevally, *Caranx ignobilis* - a candidate species for mariculture. *Bengladesh Journal of Fisheries Research* **12**, 89-94 (2008).

5 Kappen, D. C., Kaippilly, D. & D., D. N. Pioneer Attempt on Cage Culture of GiantTrevally, *Caranx Ignobilis* through Farmer Participatory Approach in Thiruthipuram Backwaters, Kochi, Kerala, India. *Ambient Science* **5**, 6-8, doi:10.21276/ambi.2018.05.2.ta02 (2018).

6 Mutia, M. T. M., Muyot, F. B., Magistrado, M. L., Muyot, M. C. & Baral, J. L. Induced Spawning of Giant Trevally, *Caranx ignobilis* (Forsskål, 1775) using Human Chorionic Gonadotropin (hCG) and Luteinising Hormone-releasing Hormone Analogue (LHRHa). *Asian Fisheries Science* **33**, 118-127, doi:10.33997/j.afs.2020.33.2.004 (2020).

7 Cossins, A. R. & Crawford, D. L. Fish as models for environmental genomics. *Nature Reviews Genetics* **6**, 324-333, doi:10.1038/nrg1590 (2005).

8 Kültz, D. Physiological mechanisms used by fish to cope with salinity stress. *J. Exp. Bio.* **218**, 1907-1914, doi:10.1242/jeb.118695 (2015).

9 Glass, J. R., Daly, R., Cowley, P. D. & Post, D. M. Spatial trophic variability of a coastal apex predator, the giant trevally *Caranx ignobilis*, in the western Indian Ocean. *Marine Ecology Progress Series* **641**, 195-208 (2020).

10 Glass, J. R., Santos, S. R., Kauwe, J. S. K., Pickett, B. D. & Near, T. J. Phylogeography of two coastal marine predators (*Caranx ignobilis* and *Caranx melampygus*) across the Indo-Pacific. *Bull. Mar. Sci.* **97**, 257-280, doi:10.5343/bms.2019.0114 (2021).

11 Pickett, B. D., Glass, J. R., Ridge, P. G. & Kauwe, J. S. K. *De novo* genome assembly of the marine teleost, Bluefin Trevally (*Caranx melampygus*). Chapter 3 herein.

12 Hardie, D. C. & Hebert, P. D. N. Genome-size evolution in fishes. *Canadian Journal of Fisheries and Aquatic Sciences* **61**, 1636-1646, doi:10.1139/F04-106 (2004).

13 Gregory, T. R. *Animal Genome Size Database*, <http://www.genomesize.com> (2018).

14 Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* **27**, 722-736, doi:10.1101/gr.215087.116 (2017).

15 Guan, D. *et al.* Identifying and removing haplotypic duplication in primary genome assemblies. *Bioinformatics* **36**, 2896-2898, doi:10.1093/bioinformatics/btaa025 (2020).

16 Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv* **1303.3997** (2013).

17 Vasimuddin, M., Misra, S., Li, H. & Aluru, S. in *2019 IEEE IPDPS*. 314-324 (Institute of Electrical and Electronics Engineers (IEEE), 2019).

18 Broad Institute. Picard Toolkit. *GitHub* http://broadinstitute.github.io/picard (2019).

19 Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078-2079, doi:10.1093/bioinformatics/btp352 (2009).

20 Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841-842, doi:10.1093/bioinformatics/btq033 (2010).

21 Ghurye, J. *et al.* Integrating Hi-C links with assembly graphs for chromosome-scale assembly. *PLoS Comput. Biol.* **15**, e1007273, doi:0.1371/journal.pcbi.1007273 (2019).

22 Song, L., Shankar, D. S. & Florea, L. Rascaf: Improving Genome Assembly with RNA Sequencing Data. *Plant Genome* **9**, 1-12, doi:10.3835/plantgenome2016.03.0027 (2016).

23 Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* **12**, 357-360, doi:10.1038/nmeth.3317 (2015).

24 Kriventseva, E. V. *et al.* OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Res.* **47**, D807-D811, doi:10.1093/nar/gky1053 (2019).

25 Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V. & Zdobnov, E. M. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* **31**, 3210-3212, doi:10.1093/bioinformatics/btv351 (2015).

26 Jain, C., Koren, S., Dilthey, A., Phillippy, A. M. & Aluru, S. A fast adaptive algorithm for computing whole-genome homology maps. *Bioinformatics* **34**, i748-i756, doi:10.1093/bioinformatics/bty597 (2018).

27 Kriventseva, E. V. *et al.* OrthoDB v8: update of the hierarchical catalog of orthologs and the underlying free software. *Nucleic Acids Res.* **43**, D250-D256, doi:10.1093/nar/gku1220 (2015).

28 *NCBI RefSeq* https://identifiers.org/insdc.gca:GCF_900963305.1 (2019).

29 Vertebrate Genomes Project. *Echeneis naucrates, Live Sharksucker*, <https://vgp.github.io/genomeark/Echeneis_naucrates> (2019).

30 *NCBI GenBank* https://identifiers.org/insdc.gca:GCA_002217815.1 (2017).

31 Yasuike, M. *et al.* The yellowtail (*Seriola quinqueradiata*) genome and transcriptome atlas of the digestive tract. *DNA Res.* **25**, 547-560, doi:10.1093/dnares/dsy024 (2018).

32 *NCBI GenBank* https://identifiers.org/insdc.gca:GCA_002994505.1 (2018).

33 *NCBI GenBank* https://identifiers.org/insdc.gca:GCA_900607315.1 (2018).

34 Zhang, D.-C. *et al.* Chromosome-level genome assembly of golden pompano (*Trachinotus ovatus*) in the family Carangidae. *Sci. Data* **6**, 216, doi:10.1038/s41597-019-0238-8 (2019).

35 *NCBI GenBank* https://identifiers.org/insdc.gca:GCA_905171665.1 (2021).

36 Vertebrate Genomes Project. *Trachurus trachurus, Atlantic Horse Mackerel*, <https://vgp.github.io/genomeark/Trachurus_trachurus> (2020).

37 Araki, K. *et al.* Whole Genome Sequencing of Greater Amberjack (*Seriola dumerili*) for SNP Identification on Aligned Scaffolds and Genome Structural Variation Analysis Using Parallel Resequencing. *Int. J. Genomics* **2018**, 7984292, doi:10.1155/2018/7984292 (2018).

38 *NCBI RefSeq* https://identifiers.org/insdc.gca:GCF_002260705.1 (2017).

39 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036353 (2021).

40 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036354 (2021).

41 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036355 (2021).

42 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036356 (2021).

43 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036357 (2021).

44 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036359 (2021).

45 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036360 (2021).

46 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036361 (2021).

47 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036362 (2021).

48 *NCBI Sequence Read Archive* https://identifiers.org/ncbi/insdc.sra:SRR13036363 (2021).

49 *NCBI BioProject* https://identifiers.org/bioproject:PRJNA670456 (2021).

50 *NCBI BioSample* https://identifiers.org/biosample:SAMN16629462 (2021).

51 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516519 (2021).

52 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516520 (2021).

53 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516521 (2021).

54 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516522 (2021).

55 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516523 (2021).

56 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516524 (2021).

57 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516525 (2021).

58 *NCBI BioSample* https://identifiers.org/biosample:SAMN16516526 (2021).

59 *NCBI BioSample* https://identifiers.org/biosample:SAMN18021194 (2021).

CHAPTER 5

SA-SSR: a suffix array-based algorithm for exhaustive and efficient SSR discovery in large genetic sequences

Brandon D. Pickett[1], Sarah M. Karlinsey[1], Corinne E. Penrod[1], Michael J. Cormier[1], Mark T. W. Ebbert[1], Dennis K. Shiozawa[1], Clint J. Whipple[1], Perry G. Ridge[1]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

A peer-reviewed, production version of this manuscript has been published in
***Bioinformatics*, 32(17):2707-2709, DOI: 10.1093/bioinformatics/btw298**.

I hereby confirm that the use of this article is compliant with all publishing agreements.

ABSTRACT

Summary: Simple Sequence Repeats (SSRs) are used to address a variety of research questions in a variety of fields (e.g., population genetics, phylogenetics, forensics, etc.), due to their high mutability within and between species. Here, we present an innovative algorithm, SA-SSR, based on suffix and longest common prefix arrays for efficiently detecting SSRs in large sets of sequences. Existing SSR detection applications are hampered by one or more limitations (i.e., speed, accuracy, ease-of-use, etc.). Our algorithm addresses these challenges while being the most comprehensive and correct SSR detection software available. SA-SSR is 100% accurate and detected >1000 more SSRs than the second-best algorithm, while offering greater control to the user than any existing software.

Availability and implementation: SA-SSR is freely available at https://github.com/ridgelab/SA-SSR

Supplementary information: Supplementary data are available at Bioinformatics online and in Appendix 6 herein.

1. INTRODUCTION

Simple Sequence Repeats (SSRs), microsatellites, or short tandem re-peats (STRs), are tandem repeats of short (often 2–5 bp) nucleotide strings (Madesiset al., 2013). There are generally 10–100 such re-peats at each SSR locus resulting in a DNA segment that is amenable to rapid molecular characterization. Given their repetitive nature, the lengths of SSR loci tend to increase or decrease due to polymerase slippage during DNA replication (Schlotterer and Tautz, 1992). As a consequence, SSR loci have high mutation rates and frequently generate multiple polymorphic alleles. SSR loci are common in both nuclear and organellar genomes, and when flanked by unique sequence, PCR primers can be readily designed to amplify simple sequence length polymorphisms. SSRs have proven highly useful for a variety of molecular genetics, population genetics, and phylogenetics applications because it is simple to genotype them using PCR, and because they are highly polymorphic.

While SSRs have been extensively characterized in many model species, the expense and effort traditionally required to develop SSRs has limited their use in non-model species. Fortunately, next-generation sequencing has enabled researchers to quickly produce large quantities of genomic and/or transcriptomic data for nearly any species. While a high-quality genome is still difficult to assemble, there is usually adequate sequence information to identify thousands of unique SSR loci with minimal sequencing. Thus, researchers working in non-model systems need user friendly and customizable bioinformatics algorithms to identify SSR loci.

A complete, accurate, characterization of SSRs in non-model systems increases the likelihood researchers are able to identify SSRs where flanking genotyping primers can be designed. SSR differences can be used to differentiate between related species or provide insights into specific phenotypes/adaptations. Finally, since the majority of researchers do not

have formal computational training, a straightforward, intuitive application is likely to enable traditional bench/field scientists to use SSRs in their research.

Many tools exist to find SSRs with varying degrees of utility, but few tools have both a useful command line interface for scripting and meaningful, parseable output. Identifying SSRs in a sequence is challenging because the search is prohibitive in time and memory requirements. Most existing tools use either an exhaustive, combinatorial search approach or a heuristic approach (Limet al., 2013). Exhaustive searches have time complexity that grows exponentially, while heuristic approaches trade comprehensiveness for run time. We developed an algorithm that is both efficient and complete.

Conceptually, finding SSRs in a nucleotide sequence is relatively straightforward, but the size of current datasets makes it a substantial challenge. SSR detection in sequence data is a substring operation—a large class of problems common in computer science. Many algorithms and data structures have been developed to reduce the time and space requirements for string operations. The suffix tree boasts linear time and space requirements for generating its representation of the string and can be used to perform many important substring operations in O(*nlogn*) time. After Weiner discovered suffix trees (Weiner, 1973), McCreight (McCreight, 1976) and Ukkonen (Ukkonen, 1995) each simplified it, paving the way for the development of the suffix array (Abouelhodaet al., 2004;Kurtz,1999;Manber and Myers, 1993). Suffix arrays have the same properties as suffix trees, but they are as many as five times more memory efficient (Kurtz, 1999;Manber and Myers, 1993).

2. ALGORITHM

A suffix array is an array of character positions representing a list of all possible suffixes of a string, ordered lexicographically, and longest common prefix arrays are arrays of the lengths of the longest common prefix of each adjacent suffix in the suffix array. Using suffix and longest common prefix arrays, we designed and implemented a novel algorithm for finding SSRs in a nucleotide sequence in linear (O(n)) time and space. The algorithm makes no distinction between microsatellites or minisatellites—it can find tandem repeats of any length or period size.

SSRs are identified by calculating three different parameters, k, r, and p from the suffix and longest common prefix arrays, where k equals the length of an SSR repeating unit or period size, r equals the number of times it repeats after the original occurrence, and p equals the position of the first nucleotide of the first period of the SSR (see Supplementary Texts 1 and 2, and Supplementary Figure S1 for a more detailed explanation). SSRs are identified by calculating k , p and r from the suffix and longest common prefix arrays (Supplementary Fig. S1 C). Let i equal the index of any entry in the suffix array (except the first position), where SA and LCPA are the suffix and longest common prefix arrays, respectively:

$$k = |SA_i - SA_{i-1}| \quad (1)$$

$$r = \left\lfloor \frac{LCPA_i}{k_i} \right\rfloor \quad (2)$$

$$p = MIN(SA_{i-1}, SA_i) \quad (3)$$

If $r > 0$, an SSR of length $k * (r + 1)$ exists at position p in the original sequence, otherwise if $r = 0$ there is no SSR at position p . The base unit (e.g., AG in the SSR AGAGAG) of the SSR starts at position p and ends at position $p + (k - 1)$. Thus, by comparing each adjacent element in the suffix array we can find SSRs in a sequence.

3. RESULTS

Our algorithm requires at most 9*n* bytes of memory, where n is the length of the entire query sequence. For each nucleotide in the sequence, we generously assume one byte in the original sequence (using 8-bit characters), four bytes in the suffix array (using 32-bit integers) and four bytes in the longest common prefix array (using 32-bit integers). The time complexity for building a suffix array and its longest common prefix array is O(*n*). Our algorithm then requires 3 * ($n - 1$) constant time computations to find SSRs, thus keeping the total time and space complexities at O(*n*).

We evaluated the performance of our algorithm compared to seven existing applications (see Supplementary Table S1 for a list of algorithms) on the *Arabidopsis thaliana* (chromosome 4), *Caenorhabditis elegans*, *Drosophila melanogaster*, *Escherichia coli* and *Zaire ebolavirus* genomes (GenBank Accessions: NC_003075.7, GCA_001483305.1, GCA_001014345.1, GCA_001432175.2 and NC_002549.1, respectively), comprised of 13,121 sequences totaling 248,846,830 nucleotides. Sequences ranged in length from 516 to 18,590,000 nucleotides with a median size of 4,662 (Supplementary Figures S2–S6 show a distribution of sequence lengths). Dozens of applications exist for SSR detection. We selected algorithms for comparison that: (i) were capable of processing the *Arabidopsis thaliana* chromosome (the longest of the sequences), (ii) had a non-interactive, Linux, command-line interface, (iii) were freely available for immediate download, and (iv) had 10 or more citations per year or were published in the last three years. Several additional algorithms met our requirements, but used antiquated shared libraries, or had compile/run-time errors. All comparisons were run on a 6-core Intel Haswell Westmere (2.67 GHz) processor with 24 GB of memory (1066 MHz DDR3).

SA-SSR, like other algorithms, calls any detected sequence repeat an SSR. Reported numbers and accuracy reflect the assumption that all sequence repeats are SSRs. SA-SSR

maximized the number of SSRs identified, while maintaining low memory requirements and runtime, and providing higher flexibility to the user to control desired output (results summarized in Table 1 with more detailed results in Supplementary Table S2). We counted the total number of SSRs identified by SA-SSR and each of the algorithms with period sizes one to seven and minimum total length of 16 nucleotides (period sizes and lengths likely to be of most interest in common applications). Next, we determined the accuracy of each of the tested algorithms, including SA-SSR, by writing a script to scan the entire sequence to verify whether or not a reported SSR was present. Most of the tested algorithms, including SA-SSR, were 100% accurate. However, compared to other algorithms, SA-SSR, found the highest number of correct (38,088 SSRs) and unique SSRs (on average >18,000 SSRs more than the other algorithms). MREPS, SSR-Pipeline, and TRF only missed 1,340, 3,047, and 7,423 correct SSRs detected by SA-SSR, respectively. However, TRF was only 23% accurate. Results of algorithm comparisons and software features are summarized in Supplementary Tables S2–S31.

Finally, we designed SA-SSR with intuitive features and formatting requirements. Like other SSR detection applications, SA-SSR takes FASTA files as input. However, some of the other applications, including some of those with high performance, are difficult to use. For example, MREPS displays an error message if any characters are not A, C, G, T or N, or if too many N's are present. Even if a user has the skills to remove all the characters that are not A, C, G or T, this makes the output positions of SSRs incorrect because those characters are not accounted for. Additionally, MREPS output is in a relatively un-structured text document that is not trivial to parse. As another example, SSR-Pipeline can only look for one period size at a time, requiring the user to manually re-run the software repeatedly for each period size of interest. Finally, SA-SSR provides greater flexibility to the user. For example, the user can

choose whether to perform an exhaustive or faster (still nearly complete) search, change output filters to report (or not) overlapping SSRs, or report only user-specified SSRs.

SA-SSR is freely available at: http://github.com/ridgelab/SA-SSR.

ACKNOWLEDGEMENTS

We thank the Fulton Supercomputing Laboratory (https://marylou.byu.edu) at Brigham Young University for their consistent efforts to support our research.

FUNDING

This work was supported by start-up funds from Brigham Young University to PGR and a mentoring environment grant from Brigham Young University to CJW.

CONFLICT OF INTEREST

None declared.

SUPPLEMENTAL MATERIALS

The supplemental materials are available online or in Appendix 6 herein.

TABLES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Summary of results from comparisons of SA-SSR with other SSR detection algorithms. This is a combination of results across each of the genomes included in the comparison. For more detailed results see Supplementary Tables S2, S4–S31. [a] MREPS timing includes the pre- and post-processing time for each genome necessary to adjust positions to account for removing 'incorrect symbols' and Ns. The additional times are an average of multiple approaches. [b] We only considered SSRs with period sizes 1–7 (inclusive) and lengths of at least 16 nucleotides (nt). The difference between the number of SSRs in range and reported is due exclusively to SSR length (less than 16 nt) and period size (greater than 7). [c] Whenever possible, we salvaged correct SSRs that were inside incorrect SSRs reported by other software packages. For example, in *Drosophila melanogaster*, we recovered three for PRoGeRF and 8,408 for TRF. To illustrate, in sequence JXOZ01000043.1, TRF reports a CT repeated 36 times at position 2,171. While TRF does correctly identify a low-complexity region with many CT repeats, there are not 36 perfect repeats in a row. In this case, we salvaged two perfect CT regions, each repeating 8 times. [d] Detailed pairwise comparisons can be found in Supplementary Tables S4–S31.

							Comparison with SA-SSR		
	CPU Time[a] (mm:ss)	**Real Time[a] (mm:ss)**	**SSRs Reported**	**SSRs In Range[b]**	**Number Correct[c]**	**Percent Correct**	**SSRs Unique to Software[d]**	**SSRs Unique to SA-SSR**	**Shared SSRs**
GMATo	329:18	329:18	72,713,858	15,284	6,617	43.29	20	34,237	3,851
MREPS	393:02	393:02	75,552	37,076	37,076	100	71	1,340	36,748
PRoGeRF	3,194:18	3,194:18	5,457,129	2,278	2,268	99.56	2	35,864	2,224
QDD	24:17	24:17	53,248	17,418	17,418	100	10	20,759	17,329
SA-SSR	28,820:12	2,416:32	38,088	38,088	38,088	100	NA	NA	NA
SSR-Pipeline	1,411:21	1,411:21	60,344,067	36,398	36,398	100	68	3,047	35,041
SSRIT	2:12	2:12	13,217	13,217	13,217	100	5	24,951	13,137
TRF	12:14	12:14	2,035,715	147,284	33,876	23.00	12	7,423	30,665

REFERENCES

Abouelhoda, M. I. et al. (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, **2**, 53–86.

Kurtz, S. (1999) Reducing the space requirement of suffix trees. *Softw. Pract. Exp.*, **29**, 1149–1171.

Lim, K. G. et al. (2013) Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance. *Brief. Bioinf.*, **14**, 67–81.

Madesis, P. et al. (2013) Microsatellites: Evolution and contribution. In: Microsatellites. Springer, pp. 1–13.

Manber, U. and Myers, G. (1993) Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, **22**, 935–948.

McCreight, E. M. (1976) A space-economical suffix tree construction algorithm. *J. ACM (JACM),* **23**, 262–272.

Schlotterer, C. and Tautz, D. (1992) Slippage synthesis of simple sequence DNA. *Nucleic Acids Res.*, **20**, 211–215.

Ukkonen, E. (1995) On-line construction of suffix trees. *Algorithmica*, **14**, 249–260.

Weiner, P. (1973) Linear pattern matching algorithms. Switching and Automata Theory, 1973. SWAT'08. In: IEEE Conference Record of 14th Annual Symposium on IEEE, pp. 1–11.

CHAPTER 6

Kmer-SSR: A Fast and Exhaustive SSR Search Algorithm

Brandon D. Pickett[1], Justin B. Miller[1], Perry G. Ridge[1]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

A peer-reviewed, production version of this manuscript has been published in
***Bioinformatics*, 33(24):3922-3928, DOI: 10.1093/bioinformatics/btx538**.

I hereby confirm that the use of this article is compliant with all publishing agreements.

ABSTRACT

Motivation: One of the main challenges with bioinformatics software is that the size and complexity of datasets necessitate trading speed for accuracy, or completeness. To combat this problem of computational complexity, a plethora of heuristic algorithms have arisen that report a 'good enough' solution to biological questions. However, in instances such as Simple Sequence Repeats (SSRs), a 'good enough' solution may not accurately portray results in population genetics, phylogenetics and forensics, which require accurate SSRs to calculate intra- and inter-species interactions.

Results: We present Kmer-SSR, which finds all SSRs faster than most heuristic SSR identification algorithms in a parallelized, easy-to-use manner. The exhaustive Kmer-SSR option has 100% precision and 100% recall and accurately identifies every SSR of any specified length. To identify more biologically pertinent SSRs, we also developed several filters that allow users to easily view a subset of SSRs based on user input. Kmer-SSR, coupled with the filter options, accurately and intuitively identifies SSRs quickly and in a more user-friendly manner than any other SSR identification algorithm.

Availability and implementation: SA-SSR is freely available at https://github.com/ridgelab/Kmer-SSR

1. INTRODUCTION

Simple sequence repeats (SSRs) are short repetitive regions of DNA where at least one base is tandemly repeated many times due to slipped-strand mispairing and errors occurring in DNA replication, repair, or recombination (Levinson and Gutman, 1987). For decades, SSRs have been studied to determine phenotypic differences caused by increased copy numbers of short repetitive sequences (Kashi and King, 2006). Moreover, SSRs account for quantitative genetic variation and phenotypic differences without lowering species fitness (Kashi et al., 1997). SSR concentration varies not only between different species, but also between different chromosomes within the same species, and cannot be explained by assessing the nucleotide composition of sequences (Katti et al., 2001). Because SSRs reveal characteristic functions of DNA replication, recombination and repair, they are important in studying biological systems interactions, as well as studying repeat expansion-based diseases with next-generation sequencing data (Kashi and King, 2006).

Many different approaches have been used to identify SSRs. Here, we propose the use of k-mers. The term k-mer refers to a subsequence of length 'k' derived from a given sequence, while k-mer decomposition refers to all possible substrings of length 'k' that can be made from a sequence. Uses for k-mer decomposition have previously been outlined in instances such as genome assembly and machine learning (Chikhi and Medvedev, 2014; Ghandi et al., 2014). Although k-mers have been used to identify similar subsequences as in (Han et al., 2007), to our knowledge SSR identification has never been attempted through k-mer decomposition.

2. MATERIALS AND METHODS

2.1 Overview

Kmer-SSR utilizes k-mer decomposition to provide an exhaustive or filtered approach to finding all SSRs in a given sequence (Figs 1 and 2). Our version of k-mer decomposition works by identifying all subsequences of length 'k' while tracking the start position of each k-mer. K-mer lengths are defined by the user as the SSR period length. Kmer-SSR minimizes the usage of random access memory (RAM) by performing k-mer decomposition and only storing k-mers that are the same as the preceding k-mer (SSR period length). If a k-mer is not identical to a k-mer found k bases previously, the previously identified k-mers will be discarded and k-mer decomposition will occur for the rest of the sequence.

2.2 Memory Requirements

We used the following techniques to limit memory requirements:

1. Identify SSRs from left to right: Kmer-SSR checks each position starting at the leftmost position of the sequence for each SSR period size (i.e., k-mer length) given by the user. This method allowed us to store only a single potential SSR and immediately either discard it if it was not repeated or write it to a file if it was a valid SSR.
2. Identify SSRs with the largest period size first: Since Kmer-SSR does not store previously identified SSRs in memory, it is necessary to search for SSRs in a specific order, or else risk reporting SSRs fully enclosed within larger SSRs. To avoid this issue, we take the period sizes given by the user and search for SSRs from the longest period size to the smallest (e.g., if the user wants to search for 2-mers and 7-mers, we search for all 7-mer SSRs in the sequences before we search for 2-mer SSRs). When

an SSR is discovered, an atomicity check is conducted to determine if the k-mer can be broken down to a smaller subsequence. An SSR is considered atomic if no smaller SSRs exist inside the first period. For example, ATATATAT would be identified as a 4-mer (ATAT) repeated twice, but ATAT is not atomic because AT (repeated twice) occurs within the first period. Thus, it is ignored because it is an invalid 4-mer and, if the user requested searching for 2-mers, it would be discovered again as a 2-mer (AT) repeated four times. If the atomicity check fails, the SSR is not reported. When an atomic (i.e. valid) SSR is discovered, the iterator moves just past the SSR, minus the current period size being searched, to ensure that overlapping SSRs are identified. For example, ACAACAACACACACAC has ACA repeated three times starting at position 0. Additionally, AC repeats five times starting at position 6. After finding the ACA repeat, we would miss the full AC repeat if we skipped to the end of the ACA repeat and resumed searching from there. Only by backtracking as described above (9–3 = 6), do we find the full AC repeat. Note that each of the nucleotides between positions 0 and 5 need not be searched for SSRs because Kmer-SSR has already found SSRs with larger period sizes than the current period size. In other words, since Kmer-SSR has already found SSRs with larger period sizes, the maximum possible overlap with the current SSR (ACA) and an adjacent following SSR is k (which is three in this example), removing the need to search for SSRs from the start of a valid SSR to k bases from the end of that SSR.

3. Create a Boolean filter array: To ensure that SSRs are unique and do not end in the same positions, we created a Boolean filter array of the same length as the sequence being analyzed, which is initiated to false. In C ++, the implementation of this array

only requires one bit per position, so the memory requirement is nominal. When an SSR is discovered, we first ensure that at least one position in the first or last SSR period size on either end of the SSR is false in the Boolean array. If one position is false, we assign all values within the array that correspond to all positions in the SSR to true. The filter allows us to ignore completely overlapping SSRs because overlapping SSRs will be set to 'true' at the positions at the ends of the SSR.

By utilizing the above-mentioned methods, we were able to limit the amount of RAM needed to O(n), where n is the sequence length, and the constant value is slightly more than one byte (one byte to store each sequence base and one bit allocated in the Boolean filter for each base).

2.3 SSR filters

Next, we implemented a comprehensive filter that allows users to control the output of Kmer-SSR based on atomicity, cyclic duplicates, enclosed SSRs, minimum SSR length and specific SSR period sizes. Pseudocode for Kmer-SSR is in Figure 2. The following are different filters that are optionally applied to the output of Kmer-SSR:

1. Atomicity check: The atomicity check ensures that the smallest period size for each SSR is reported. For instance, if an ATAT repeats four times, it would be reported as an AT repeated eight times because AT is the smallest period size within ATAT.
2. Cyclic duplicates: Many SSRs create equally viable SSRs with slightly different positions reported. For instance, in the sequence ATATATATATATATATA, it is arguably equally valid to report the AT repeated eight times starting at position zero as it would be to report TA repeating eight times starting at position one. To avoid

duplicate reporting of cyclic duplicates and ensure the longest SSR is always reported, we choose and report only the leftmost SSR. So, in this instance, only the AT repeated eight times would be reported.

3. Enclosed SSRs: Occasionally, SSRs might be completely enclosed within other SSRs. For example, in the sequence TAAAATTAAAATTAAAAT, the SSR TAAAAT is repeated three times, but within each TAAAAT there is an A that repeats four times. In this case, we only report the longest SSR, TAAAAT, repeated three times.
4. SSR length: We allow the user to input minimum and maximum SSR lengths via command line options. By default, SSRs are only reported if they are at least 16 nucleotides long.
5. Set specific period sizes: We allow the user to input specific period sizes to be checked (e.g., 1, 3, 5 would look for SSRs with period sizes of one, three and five), or ranges of period sizes (e.g. 1–7 would look for SSRs with period sizes one through seven). By default, Kmer-SSR reports SSRs of period sizes one through seven. SSRs outside of the user specified range are not reported.
6. Number of repeats: We allow the user to input minimum and maximum numbers of repeats via command line options. By default, SSRs must repeat at least twice to be reported.
7. Enumerated SSRs: If the user is interested in a very limited set of SSRs, they may specify those via a command line option and no other SSRs will be reported.
8. Sequence length: The user may specify minimum and maximum bounds on the length of an input sequence, outside of which the program will not search or report SSRs. By

default, if a sequence is less than 100 bases or more than 500 megabases, it will be ignored.

3. RESULTS

We conducted pairwise comparisons of Kmer-SSR against the following SSR identification algorithms: GMATo (Wang et al., 2013), MREPS (Kolpakov et al., 2003), PRoGeRF (Lopes et al., 2015), QDD (Meglécz et al., 2014), SA-SSR (Pickett et al., 2016), SSR-Pipeline (Miller et al., 2013), SSRIT (Temnykh et al., 2001) and TRF (Benson, 1999). These comparisons were performed on DNA sequences from six different species (whole genome assembly unless otherwise noted): *Anolis carolinensis* chromosome 6 (CM000942.1), *Chlamydomonas reinhardtii* (assembly v5.5) (Merchant et al., 2007), *Danio rerio* chromosome 25 (CM002909.1), *Dictyostelium doscoideum* (GCA_0000044695.1), *Physcomitrella patens* chromosome 1 (assembly v3.3), and *Saccharomyces cerevisiae* (GCA_001634645.1). Table 1 displays the computational time of each algorithm and the number of SSRs correctly identified for each dataset (CPU Time and Real Time columns).

Because Kmer-SSR is multithreaded and robust to fasta files with unknown nucleotides, the real time for SSR identification using Kmer-SSR is faster than any other algorithm. Although MREPS reports a faster real time identification of SSRs, the program does not usually run with sequences containing unknown characters. With the addition of the time necessary to make the input fasta files usable for MREPS, it underperformed Kmer-SSR in all six datasets (Table 1, Real Time column). We found that with the exception of TRF, all algorithms tested were 100% accurate in identifying SSRs; however, only Kmer-SSR, MREPS and SSRIT reported all possible filtered SSRs within the range specified for each dataset (Table 1, SSRs In Range

column). Although SSRIT has a faster CPU time than Kmer-SSR, it does not have the multithreading capabilities of Kmer-SSR, nor does it allow for querying of SSRs other than period sizes 2–4 without directly editing the algorithm's source code.

4. DISCUSSION

SSR identification is important in many biological comparisons. It is important to have 100% accuracy in SSR identification because primers often depend on the exact SSR sequence with conserved flanking sequences (Robinson et al., 2004), and phenotypic variations associated with SSRs require an accurate portrayal of a genome. Furthermore, determining the exact SSR copy number is important in species identification and aids in the identification of discrete families and individuals. Kmer-SSR fills a usability gap in SSR identification. While many SSR identification algorithms exist, it is often difficult to install, use and read the output from the algorithms available. Two of the main strengths of Kmer-SSR are its usability and the SSR filters that are easily accessible to help answer biological questions. Installing Kmer-SSR is at least as easy to install as other algorithms. Kmer-SSR was implemented in C ++. It does not require any editing of the source code to find SSRs of different lengths or filter overlapping SSRs, and it provides a robust documentation for its command line options. Step-by-step instructions for installation and implementation of Kmer-SSR are available with the algorithm's source code at http://github.com/ridgelab/Kmer-SSR.

The filters available in Kmer-SSR help answer primary biological questions. Instead of inundating a researcher with duplicate SSRs, Kmer-SSR eliminates overlapping SSRs by only reporting the left-most SSR in each sequence when multiple SSRs are equally valid. Furthermore, longer SSRs are typically more biologically interesting, so completely enclosed

SSRs are not included in the output. Importantly, these filters still allow for overlapping SSRs where at least one period size is completely outside of the previously reported SSR. These filters set Kmer-SSR apart from all other SSR identification algorithms because of its ease of use as well as its utility.

As we compared other algorithms, a few difficulties arose that made it challenging to directly compare the output from each program. We learned that QDD does not allow the sequence header line to contain the vertical bar [|] (and possibly other characters that have special meaning in a regular expression). Also, analysis of 1-mers in longer sequences, such as the lizard genome, exceeded 21 days in SSR-pipeline. MREPS also required pre-splitting of the input sequence files because the algorithm does not accept any characters besides A, T, C and G in the sequence lines (it will accept a very limited number of well-distributed Ns). SSRIT requires directly editing the source code to query period sizes other than lengths two through four. Similarly, QDD requires directly editing its source code to retrieve different period lengths and different SSR lengths. QDD defaults to 1-mers that must be 1 million bases long and 2-mers through 6-mers that must repeat at least 5 times. Furthermore, unlike some other algorithms, the output format for Kmer-SSR is easily parsable, and it can be exported directly to an Excel spreadsheet or another tab delimited parser. GMATO, ProGeRF, SSRIT and SA-SSR have similar output formats (although, ProGeRF and SSRIT do not provide column headers). MREPS and TRF are text-based reports with embedded tables. QDD provides a semicolon-separated value report with a few fixed columns followed by a variable number of columns thereafter depending on the number of SSRs found in a given sequence. SSR-Pipeline provides FASTA formatted output where the SSRs are encoded in the header (see Table 2). MREPS, PRoGeRF and TRF attempt to identify SSRs through heuristics. Heuristics are a common approach to

achieve an adequate solution to a problem that is either too computationally intensive to check all possible solutions or does not have a good approach to calculate the exact solution (Clancey, 1985). Table 2 displays features of each software package per each software package's documentation (Benson, 1999; Kolpakov et al., 2003; Lopes et al., 2015; Meglécz et al., 2014; Miller et al., 2013; Pickett et al., 2016; Temnykh et al., 2001; Wang et al., 2013).

While Kmer-SSR provides a substantially better user experience with more filters and options than all other algorithms, Kmer-SSR has several weaknesses. First, since Kmer-SSR is an exact algorithm, it is not as fast as the heuristic approach of MREPS when there are only canonical nucleotides in a sequence. Second, due to the kmer decomposition approach used in Kmer-SSR, it is unable to identify fuzzy repeat regions where only one or two nucleotides differ from an exact repeat. Although not necessary for many applications, fuzzy repeats would provide Kmer- SSR with increased functionality that is not currently possible with the algorithm's implementation. Third, Kmer-SSR has no web interface.

Unlike all other algorithms, Kmer-SSR offers the convenience of a completely exhaustive search in linear time (though with a larger constant factor than normal). This truly exhaustive search is entirely filter- free. As an example, that means it would report an ACG repeated seven times at position 1, six times at position 4, five times at position 7, etc. This is likely not necessary for most applications. However, with the exhaustive option, we set an upper limit for all SSR identifications. Furthermore, since genome complexity is important in primer design and predicting recombination events (Murray et al., 1999), the exhaustive option could be used as an easy approach to determine the proportion of a sequence that repeats.

ACKNOWLEDGEMENTS

We appreciate Brigham Young University and the Fulton Supercomputing Laboratory (https://marylou.byu.edu) for their continued support of our research. We thank the US Department of Energy Joint Genome Institute for granting access to Chromosome 1 of *Physcomitrella patens*.

FUNDING

This work has been supported by funds provided by Brigham Young University and the Department of Biology.

CONFLICT OF INTEREST

None declared.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Comparisons of all nine SSR-identification algorithms across six genomes with period sizes of 1-7 and a minimum SSR length of 16 bases. We ran all comparisons on a 2.3 Ghz Intel Haswell processor. Although each algorithm was given the same amount of memory and CPUs, due to hardware variability of the CPU, runtimes could vary by up to 20%. Also, MREPS required pre-processing of the fasta files, which typically added anywhere from a few seconds to several minutes to the runtime (not depicted in the table), depending on the pre-processing approach used. Similarly, we did not include the time required to edit SSRIT and QDD's source code in order for their programs to function over the period sizes in these tests. SSR-Pipeline could not finish searching for 1-mers in chromosome 6 of the *Anolis carolinensis* in 21 days of run time. Accordingly, the chromosome was split into 24 approximately equal sized chunks (i.e., approximately 3.3 Mb each) and each chunk was searched for 1-mers separately by SSR-Pipeline. The required time for each chunk was summed (approximately 5 hours) and used in place of 504 hours (21 days).
The SSRs After Adjustments column reflects the number of SSRs that we did not remove or alter for purposes of making the comparison simpler. SSRs that were exact duplicates, duplicates with only the repeat number varying, duplicates that varied only by cycle (e.g., ACG versus CGA with the same number of repeats right next to each other), entirely surrounded by another SSR, or not atomic (e.g., ATAT repeated 2 times instead of AT repeated 8 times) were removed. SSRs that shared the same base and overlapped were combined into one SSR (e.g., AT repeated 8 times at position 1 and AT repeated 6 times at position 11 would be combined to AT repeated 11 times at position 1).
The SSRs In Range column is the number of SSRs from the previous column that were 16 nt or longer and had a period size of 1-7 (inclusive).
The Number Correct column is the number of SSRs In Range that were actually present in the sequence.
The Number Correct and Fixed is the Number Correct plus a few incorrect SSRs that we are able to fix (e.g., a program might report an AT repeated 30 times, but it only repeated 20 times in the sequence).
The Percent Correct and Fixed is the percent of SSRs in Range that were correct or fixed.

										Comparison with Kmer-SSR		
		CPU Time (mm:ss)	Real Time (mm:ss)	SSRs Reported	SSRs After Adjustments	SSRs In Range	Number Correct	Number Correct & Fixed	Percent Correct & Fixed	SSRs Unique to Software	SSRs Unique to Kmer-SSR	SSRs Shared
Anolis carolinensis (chr 6)	GMATo	2:38	2:38	20,623,008	16,369,297	16,871	16,871	16,870	100	0	8,194	10,090
	Kmer-SSR	2:24	0:24	18,284	18,284	18,284	18,284	18,284	100	NA	NA	NA
	MREPS	0:09	0:09	25,639	25,639	18,284	18,284	18,284	100	0	0	18,284
	PRoGeRF	18:07	18:07	16,841,656	16,840,821	17,763	17,762	17,763	100	0	610	17,674
	QDD	19:11	19:11	60,994	60,994	18,009	18,009	18,009	100	0	732	17,552
	SA-SSR	338:47	33:55	18,166	18,166	18,166	18,166	18,166	100	0	442	17,842
	SSR-Pipeline	611:55	611:55	19,173,282	17,301,120	18,044	18,044	18,044	100	0	913	17,371
	SSRIT	1:29	1:29	87,073	74,121	18,284	18,284	18,284	100	0	0	18,284
	TRF	2:09	2:09	422,851	411,644	42,157	13,872	17,307	41.05	0	1,560	16,724
Chlamydomonas reihardtii	GMATo	3:30	3:30	26,512,280	21,624,294	50,401	50,401	50,139	99	0	23,086	34,416
	Kmer-SSR	3:26	0:19	57,502	57,502	57,502	57,502	57,502	100	NA	NA	NA
	MREPS	0:14	0:14	94,875	94,875	57,502	57,502	57,502	100	0	0	57,502
	PRoGeRF	37:55	37:55	8,071,102	8,020,213	32,043	31,989	32,004	100	0	25,588	31,914
	QDD	8:51	8:51	216,943	216,943	55,470	55,470	55,470	100	0	3,002	54,500
	SA-SSR	1,324:33	167:48	56,833	56,833	56,833	56,833	56,833	100	0	1,214	56,288
	SSR-Pipeline	632:10	632:10	26,973,434	23,032,838	56,729	56,729	56,729	100	0	1,793	55,709
	SSRIT	2:00	2:00	310,109	252,223	57,502	57,502	57,502	100	0	0	57,502
	TRF	8:52	8:52	1,022,145	990,316	181,973	25,451	45,773	25.15	0	14,546	42,956
Danio rerio (chr 25)	GMATo	1:12	1:12	9,501,860	7,535,749	22,546	22,546	22,362	99	0	8,463	13,636
	Kmer-SSR	1:10	0:13	22,099	22,099	22,099	22,099	22,099	100	NA	NA	NA
	MREPS	0:05	0:05	26,862	26,862	22,099	22,099	22,099	100	0	0	22,099
	PRoGeRF	8:14	8:14	7,696,269	7,695,012	21,729	21,668	21,684	100	0	494	21,605
	QDD	7:43	7:43	49,016	49,016	21,805	21,805	21,805	100	0	908	21,191
	SA-SSR	2,075:03	648:00	21,862	21,862	21,862	21,862	21,862	100	0	690	21,409
	SSR-Pipeline	1,958:54	1,958:54	8,948,450	7,954,899	21,857	21,857	21,857	100	0	987	21,112
	SSRIT	0:43	0:43	69,645	58,065	22,099	22,099	22,099	100	0	0	22,099
	TRF	5:03	5:03	293,378	283,764	40,343	11,255	16,911	41.92	0	6,144	15,955

		CPU Time (mm:ss)	Real Time (mm:ss)	SSRs Reported	SSRs After Adjustments	SSRs In Range	Number Correct	Number Correct & Fixed	Percent Correct & Fixed	Comparison with Kmer-SSR: SSRs Unique to Software	SSRs Unique to Kmer-SSR	SSRs Shared
Dictyostelium doscoideum	GMATo	1:02	1:02	8,810,607	7,126,425	82,643	82,643	82,526	100	0	28,714	62,967
	Kmer-SSR	1:12	0:08	91,681	91,681	91,681	91,681	91,681	100	NA	NA	NA
	MREPS	0:05	0:05	121,835	121,835	91,681	91,681	91,681	100	0	0	91,681
	PRoGeRF	11:42	11:42	4,629,786	4,604,499	60,176	60,174	60,174	100	0	31,707	59,974
	QDD	3:44	3:44	171,686	171,686	88,017	88,017	88,017	100	0	5,295	86,386
	SA-SSR	723:31	236:01	90,700	90,700	90,700	90,700	90,700	100	0	1,635	90,046
	SSR-Pipeline	246:35	246:35	9,292,900	7,397,561	90,810	90,810	90,810	100	0	1,759	89,922
	SSRIT	0:42	0:42	265,894	202,531	91,681	91,681	91,681	100	0	0	91,681
	TRF	17:30	17:30	642,904	602,301	178,902	40,772	75,742	42.34	0	18,962	72,719
Physcomitrella patens (chr 1)	GMATo	0:59	0:59	7,981,869	6,500,395	7,739	7,739	7,736	100	0	3,259	5,528
	Kmer-SSR	0:58	0:10	8,787	8,787	8,787	8,787	8,787	100	NA	NA	NA
	MREPS	0:04	0:04	12,885	12,885	8,787	8,787	8,787	100	0	0	8,787
	PRoGeRF	7:32	7:32	6,639,989	6,639,933	8,669	8,668	8,668	100	0	131	8,656
	QDD	4:29	4:29	27,774	27,774	8,319	8,319	8,319	100	0	621	8,166
	SA-SSR	642:36	91:59	8,719	8,719	8,719	8,719	8,719	100	0	152	8,635
	SSR-Pipeline	1,498:06	1,498:06	7,763,141	6,874,175	8,720	8,720	8,720	100	0	253	8,534
	SSRIT	0:35	0:35	39,472	35,941	8,787	8,787	8,787	100	0	0	8,787
	TRF	1:53	1:53	223,938	215,818	22,730	6,132	8,192	36.04	0	891	7,896
Saccharyomyces cerevisiae	GMATo	0:23	0:23	3,281,592	2,674,303	1,101	1,101	1,101	100	0	588	887
	Kmer-SSR	0:23	0:04	1,475	1,475	1,475	1,475	1,475	100	NA	NA	NA
	MREPS	0:02	0:02	2,293	2,293	1,475	1,475	1,475	100	0	0	1,475
	PRoGeRF	3:43	3:43	1,065,515	1,065,510	492	492	492	100	0	988	487
	QDD	0:47	0:47	8,672	8,672	1,368	1,368	1,368	100	0	139	1,336
	SA-SSR	338:50	60:55	1,430	1,430	1,430	1,430	1,430	100	0	57	1,418
	SSR-Pipeline	9:32	9:32	3,124,288	2,820,560	1,427	1,427	1,427	100	0	73	1,402
	SSRIT	0:14	0:14	12,276	10,386	1,475	1,475	1,475	100	0	0	1,475
	TRF	0:26	0:26	62,616	61,038	4,634	755	1,242	26.80	0	290	1,185
Combined	GMATo	9:44	9:44	76,711,216	61,830,463	181,301	181,301	180,734	100	0	72,304	127,524
	Kmer-SSR	9:33	1:18	199,828	199,828	199,828	199,828	199,828	100	NA	NA	NA
	MREPS	0:39	0:39	284,389	284,389	199,828	199,828	199,828	100	0	0	199,828
	PRoGeRF	87:13	87:13	44,944,317	44,865,988	140,872	140,753	140,785	100	0	59,518	140,310
	QDD	44:45	44:45	535,085	535,085	192,988	192,988	192,988	100	0	10,697	189,131
	SA-SSR	5,443:20	1,238:38	197,710	197,710	197,710	197,710	197,710	100	0	4,190	195,638
	SSR-Pipeline	4,957:12	4,957:12	75,275,495	65,381,153	197,587	197,587	197,587	100	0	5,778	194,050
	SSRIT	5:43	5:43	784,469	633,267	199,828	199,828	199,828	100	0	0	199,828
	TRF	35:53	35:53	2,667,832	2,564,881	470,739	98,237	165,167	35.09	0	42,393	157,435

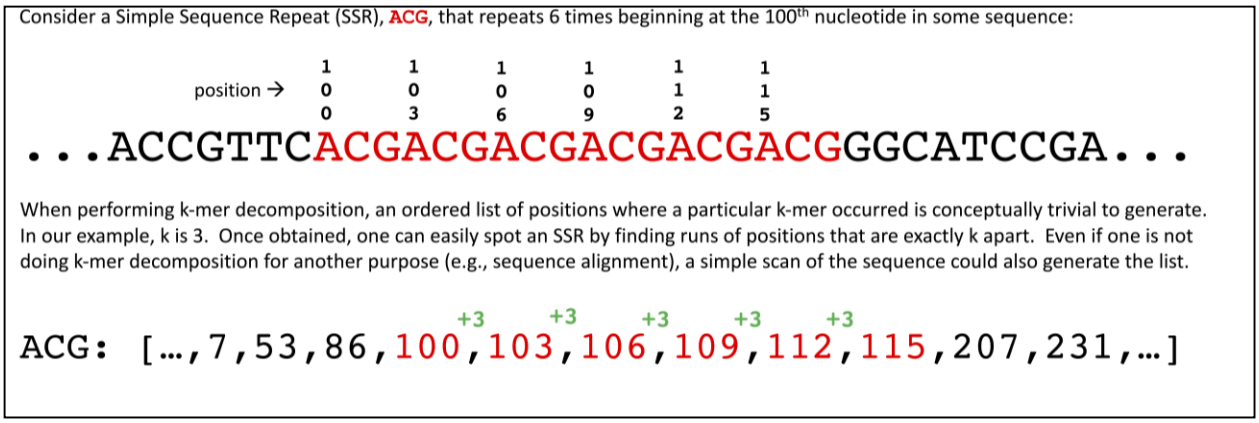


Figure 1. Conceptual Representation of Kmer-SSR. Although we implement some filters and tricks to speed up Kmer-SSR runtime, each SSR is identified through kmer decomposition, which allows the identification of instances when the same SSR period occurs k bases from the previously identified SSR period.

```
Input: P, s              // the list of desired period sizes, a DNA sequence
P ← sort(P)              // sort largest to smallest
F                        // Boolean array of length(s); all values instantiated as False
Function searchForSSR(period, seq, index)
Begin
       base = getSubSequence(pos, period, seq)      // grab the first sequence
       next = getSubSequence(pos, period, seq)
       repeats = 0                                  // the number of times the ssr repeats
       pos = index                                  // starting position of the ssr
              while base == next and pos < (length(seq) - period - 1) do  // while adjacent
                                                                          // periods match
              repeats += 1                  // we found another copy, increment the count
              pos += period
              next = getSubSequence(pos, period, seq)      // grab the next period
       end while
return SSR(base, repeats, index)
End Function

Function passesBooleanFilter (F, ssrStartPos, ssrStopPos)
Begin
   for i ← ssrStartPos to ssrStopPos do      //Positions in first period size
      if F_i == False then                   //If SSR has never been found at the position
         return True                         //SSR is valid
      end if
   end for
return False                                 //SSR is not valid
End Function

Main Program
for i ← 1 to length(P) do                    //For each period size in list
   for j ← 1 to length(s) do                 //For each nucleotide in sequence
      ssr = searchForSSR(P_i, s, j)          //Search for next SSR that repeats
      u = getSSRStartPos(s, ssr)             //Get start position of SSR
      v = getSSRStopPos(s, ssr)              //Get last position of SSR
      if passesUserFilers(ssr) and passesBooleanFilter(F, u, v) then
         print(ssr)                          //Print SSR to output file
         for x ← u to v do                   //For each position in SSR
            F_x ← True                       //Sets Boolean filter to True
         end for
         j += (length(ssr) - P_i - 1)        //Update position in sequence
      end if
   end for
end for
End
```

Figure 2. Pseudocode for the Kmer-SSR algorithm. The function passesBooleanFilter ensures SSRs are not duplicates of previously reported SSRs. The function passesUserFilters (function not shown) completes other user-specified options, which may include minimum SSR length, minimum and maximum number of periods, finding specific SSRs, and sequence length bounds.

REFERENCES

Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.*, **27**, 573.

Chikhi, R. and Medvedev, P. (2014) Informed and automated k-mer size selection for genome assembly. *Bioinformatics*, **30**, 31–37.

Clancey, W. J. (1985) Heuristic classification. *Artif. Intell.*, **27**, 289–350.

Ghandi, M. et al. (2014) Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput. Biol.*, 10**,** e1003711.

Han, W.-S. et al. (2007) Ranked subsequence matching in time-series data-bases. In: *Proceedings of the 33rd international conference on Very large databases*. VLDB Endowment. p. 423-434.

Kashi, Y. et al. (1997) Simple sequence repeats as a source of quantitative gen-etic variation. *Trends Genet.*, **13**, 74–78.

Kashi, Y. and King, D. G. (2006) Simple sequence repeats as advantageous mutators in evolution. *Trends Genet*., **22**, 253–259.

Katti, M. V. et al. (2001) Differential distribution of simple sequence repeats in eukaryotic genome sequences. *Mol. Biol. Evol*., **18**, 1161–1167.

Kolpakov, R. et al. (2003) mreps: efficient and flexible detection of tandem re-peats in DNA. *Nucleic Acids Res*., **31**, 3672–3678.

Levinson, G. and Gutman, G. A. (1987) Slipped-strand mispairing: a major mechanism for DNA sequence evolution. *Mol. Biol. Evol*., **4**, 203–221.

Lopes, RdS. et al. (2015) ProGeRF: Proteome and Genome Repeat Finder Utilizing a Fast Parallel Hash Function. *BioMed. Res. Int.*, **2015**, 1.

Meglécz, E. et al. (2014) QDD version 3.1: a user-friendly computer program for microsatellite selection and primer design revisited: experimental validation of variables determining genotyping success rate. *Mol. Ecol. Resources*, **14**, 1302–1313.

Merchant, S. S.et al. (2007) The Chlamydomonas genome reveals the evolution of key animal and plant functions. *Science*, **318**, 245–250.

Miller, M. P. et al. (2013) SSR_pipeline: A bioinformatic infrastructure for identifying microsatellites from paired-end Illumina high-throughput DNA sequencing data. *J. Hered*., est056.

Murray, J. et al. (1999) Comparative sequence analysis of human minisatellites showing meiotic repeat instability. *Genome Res*., **9**, 130–136.

Pickett, B. D. et al. (2016) SA-SSR: a suffix array-based algorithm for exhaustive and efficient SSR discovery in large genetic sequences. *Bioinformatics*, **32**, 2707–2709.

Robinson, A. J. et al. (2004) Simple sequence repeat marker loci discovery using SSR primer. *Bioinformatics*, **20**, 1475–1476.

Temnykh, S. et al. (2001) Computational and experimental analysis of micro-satellites in rice (Oryza sativa L.): frequency, length variation, transposon associations, and genetic marker potential. *Genome Res*., **11**, 1441–1452.

Wang, X. et al. (2013) GMATo: A novel tool for the identification and analysis of microsatellites in large genomes. *Bioinformation*, **9**, 541–544.

CHAPTER 7

Molecular epidemiology of carbapenem-resistance plasmids using publicly available sequences

Galen E. Card[1*], Brandon D. Pickett[2*], Perry G. Ridge[2], Richard A. Robison[1]

[1]*Department of Microbiology and Molecular Biology, Brigham Young University, Provo, Utah, USA*

[2]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[*]*These authors contributed equally to this work*

A peer-reviewed, production version of this manuscript has been published in
***Genome*, 62(12):785-792, DOI: 10.1139/gen-2019-0100**.

I hereby confirm that the use of this article is compliant with all publishing agreements.

ABSTRACT

Carbapenem-resistant bacteria have quickly become a worldwide concern in nosocomial infections. Of the seven known carbapenemases, four have been shown to be particularly problematic: KPC, NDM, IMP, and VIM. To date, many local and species- or carbapenemase-specific epidemiological studies have been performed, which often focus on the organism itself. This report attempts to perform an inclusive (encompass both species and carbapenemase) epidemiologic study using publicly available plasmid sequences from NCBI. In this report, the gene content of these various plasmids has been characterized, replicon types of the plasmids identified, and the global spread and species promiscuity of the plasmids analyzed. Additionally, support to several groups targeting plasmid maintenance and transfer mechanisms to slow the spread of resistance plasmids is given.

INTRODUCTION

Nosocomial infections have quickly become a significant cause of mortality. In 2002, the US Centers for Dis- ease Control and Prevention estimated that the national mortality rate due to hospital-acquired infections was 5.8% (Klevens et al. 2007). In 2011, that rate increased to 10.4% (Magill et al. 2014). While these same reports show that the chance of acquiring an infection at the hospital has decreased, the infections are becoming strikingly more lethal.

One significant reason for this increase in mortality is the acquisition of antibiotic resistance in bacterial populations (Read and Woods 2014). To mitigate the havoc wrought by β-lactamases on the efficacy of antimicrobials, multiple β-lactam derivatives have been pressed into service. One of these derivate classes, the carbapenems, is used as a last resort for treating extended spectrum β-lactamase infections. Recently, resistance to this class has occurred as well.

Antibacterial resistance is often conferred to these organisms through extra-chromosomal segments of DNA called plasmids (Read and Woods 2014). Plasmids often carry the molecular machinery to replicate themselves and allow for the transfer of the plasmid between different bacterial strains, and even between many gram-negative bacteria (Logan and Weinstein 2017). Additionally, many carbapenemase-carrying plasmids are large; therefore, they often carry a toxin/anti- toxin plasmid addiction system (Tsang 2017) or plasmid partitioning system to prevent the bacterium from losing the plasmid. Furthermore, evidence has been shown for local and global transmission of carbapenemase genes among several bacterial species (Logan and Weinstein 2017; Stoesser et al. 2017), leading to a global crisis in the declination of antibiotic therapy efficacy.

Carbapenemases

Currently there are about nine diverse types of carbapenemases falling into Ambler classes A, B, and D (Yong et al. 2009; Overturf 2010). Each of those nine types have several allele variations. We will focus on four clinically relevant types found in *Enterobacteriaceae*, the class A serine-mediated *Klebsiella pneumoniae* carbapenemase (bla_{KPC}) and three class B metallo-β-lactamases (bla_{MBL}): the New Delhi metallo-β-lactamase (MBL) (bla_{NDM}), the Verona integron-encoded MBL (bla_{VIM}), and the Imipenem- resistant MBL (bla_{IMP}), and highlight their pertinent characteristics.

Klebsiella pneumoniae carbapenemase

First identified in 2001 (Yigit et al. 2001), bla_{KPC} was not the first carbapenemase, as several MBLs that could hydrolyze carbapenem had already been identified in Japan in 1994 (Paterson and Bonomo 2005). This initial variant (now referred to as KPC-2) provided resistance to numerous penicillins, all the cephalosporins, and aztreonam, and was also resistant to the β-lactamase inhibitors clavulanic acid and tazobactam (Yigit et al. 2001). A recent review indicates that there are currently 12 reported variants of the KPC enzyme (Sotgiu et al. 2018). As of 27 February 2018, the Centers for Disease Control and Prevention report that bla_{KPC}-positive infections have been reported from all 50 states and the District of Columbia (Centers for Disease Control and Prevention n.d.; https://www.cdc.gov/hai/organisms/cre/trackingcre.html). KPC enzymes have also been reported from many other nations and in numerous gram-negative species, including *Acinetobacter baumanii*, *Pseudomonas aeruginosa*, and nearly all the *Enterobacteriaceae* (Arnold et al. 2011; Perez and Van Duin 2013; Codjoe and Donkor 2018).

The ease of *bla*$_{KPC}$ gene transfer has been augmented by the *Tn*4401 transposon that flanks the KPC-1 gene (Arnold et al. 2011).

New Delhi metallo-β-lactamase

Originally isolated from India in 2008, there are currently more than 10 reported variants of *bla*$_{NDM}$ (Bedenic´ et al. 2014). They are present in 34 states (Centers for Disease Control and Prevention n.d.; https://www.cdc.gov/hai/organisms/cre/trackingcre.html) and multiple countries including the United Kingdom, Pakistan, In- dia, Sweden, and others (Perez and Van Duin 2013). This type of carbapenemase has shown greater enzymatic activity than the *bla*$_{VIM}$ and *bla* types for the penicillins, cephalosporins, and a few of the carbapenems (Yong et al. 2009). *bla*$_{NDM}$ has shown a greater potential for spread than bla$_{KPC}$, as it has rapidly appeared across the world in the last 10 years.

Verona integron-encoded metallo-β-lactamase

bla$_{VIM}$ has 14 reported variants with amino acid content varying up to 10% (Bedenić et al. 2014). *bla*$_{VIM}$ originated from *Pseudomonas aeruginosa* in the Mediterranean in 1997, but quickly spread into *Enterobacteriaceae* and proceeded to spread globally. Reports indicate that *bla*$_{VIM}$ can hydrolyze all β-lactams except monobactams and remains susceptible to inhibitors (Marsik and Nambiar 2011). Like the other carbapenemases, plasmids are the primary mechanism for horizontal gene transfer of this carbapenemase.

Imipenem-resistant metallo-β-lactamase

bla$_{IMP}$ shares many of the same characteristics as *bla*$_{VIM}$, but the amino acid content between the two diverges by 70% (Bedenić et al. 2014). *bla*$_{IMP}$ also represents the most diverse type of carbapenemase with 18 variants reported (Bedenić et al. 2014). Isolated in 1991 in Japan, it is the earliest carbapenemase of the four, and is resistant to the inhibitor clavulanic acid (Watanabe et al. 1991).

While there are other reports that characterize carbapenemase plasmids, these generally describe a single carbapenemase within a species (see Johnson and Woodford 2013; Sheppard et al. 2016; Stoesser et al. 2017; Piazza et al. 2019; Wang et al. 2018; Chen et al. 2019; Mansour et al. 2019; Mukherjee et al. 2019). This report is the first large-scale attempt to characterize the diversity and promiscuity of plasmids carrying one of four carbapenemase families across multiple bacterial species. However, the impact of this study is limited due to the regional bias introduced by national surveillance and sequencing programs. Additionally, *bla*$_{OXA-48}$ carbapenemase was excluded due to its sequence similarities to other OXA-type β-lactamases and because of its reported decreased efficiency in hydrolytic activity towards carbapenems (Poirel et al. 2012). We identified these carbapenemase-carrying plasmids from seven clinically-relevant gram-negative bacteria (*Enterobacter aerogenes* (also *Klebsiella aerogenes* (Tindall et al. 2017)), *Enterobacter cloacae*, *Escherichia coli*, *Klebsiella pneumoniae*, *Pseudomonas aeruginosa*, *Providencia stuartii*, and *Serratia marcescens*).

MATERIALS AND METHODS

A detailed description and the full dataset can be found in the supplementary data, File S1[1].

Sequence acquisition

In total, 532 complete plasmid sequences were obtained from NCBI nucleotide database by a discontiguous megablast nucleotide search (Altschul et al. 1990) of four representative carbapenemase genes (*bla*$_{\text{IMP-4}}$, *bla*$_{\text{KPC-2}}$, *bla*$_{\text{NDM-1}}$, *bla*$_{\text{VIM-1}}$) to allow for variations within the carbapenemase family. We employed the same Entrez strategy used by Orlek et al. (2017) in the BLAST search to filter for complete plasmids:

> "biomol_genomic[PROP] AND plasmid[filter] NOT complete cds[Title] NOT gene[Title] NOT genes[Title] NOT contig[Title] NOT scaffold[Title] NOT whole genome map[Title] NOT partial sequence[Title] NOT par- tial plasmid[Title] NOT locus[Title] NOT region[Title] NOT fragment[Title] NOT integron[Title] NOT transposon[Title] NOT insertion sequence[Title] NOT insertion element[Title] NOT phage[Title] NOT operon[Title]"

This BLAST search was done separately for the seven organisms of interest: *E. aerogenes*, *E. cloacae*, *E. coli*, *K. pneumoniae*, *P. aeruginosa*, *P. stuartii*, and *S. marcescens*. GenBank files were downloaded for each BLAST alignment that scored >80% identity and query coverage. These sequences were retrieved on 5 March 2018.

Plasmid gene composition

A list of key terms was derived by a random survey of 10% of the acquired GenBank files, with cross reference to QuickGO, the European Bioinformatics Institute's Gene Ontology reference database to classify gene products into one of the following categories: antimicrobial resistance, with β-lactamases as a subset; plasmid trans- fer genes; toxin/antitoxin systems; DNA

[1] Supplementary data are available with the article through the journal Web site at https://nrcresearchpress.com/doi/suppl/10.1139/gen-2019-0100 and in Appendix 7 herein.

maintenance, modifying, and repair proteins; mobile genetic elements; hypothetical genes; and other.

Incompatibility group/replicon typing and plasmid characterization

Plasmid incompatibility groups were determined by nucleotide BLAST (Altschul et al. 1990; Camacho et al. 2009) against a local download of the PlasmidFinder v1.3 *Enterobacteriaceae* database containing the origin sequences for numerous replicon types (Carattoli et al. 2014) downloaded on 1 March 2018. The incompatibility groups were assigned when matches met the following criteria: ≥80% identity, ≥60% subject coverage, and within 1% of the percent identity of the highest match. Accordingly, more than one incompatibility group could be reported for any given plasmid. Further characterization was accomplished as follows: extracting the CDS regions for each plasmid, searching these CDS regions for key terms using regular expressions, and combining the results for plasmid groups of interest (e.g., those that belong to *Enterobacteriaceae*, or those that carry bla_{KPC}). Additionally, associated metadata were extracted for plasmids that identified a country of origin to elucidate the global prevalence of these plasmids.

Nondiscrete plasmid groups

Ultimately, the plasmid sequences were BLASTed against each other to identify any duplicate entries, and the following metadata was identified for any match exceeding 98% coverage and identity match: the organism from which the plasmid was extracted, the country of origin, and the collection date of the plasmid. The tree was constructed using a custom distance metric and Python (https://python.org) code from the CAM package (Miller et al. 2019). The custom distance metric is described in detail in the supplementary data (File S1). Briefly, it is the

sum of the bases from the query and subject included in the alignment divided by the sum of the length of the query and subject sequences. The image of the tree was generated using FigTree v1.4.4 (https://github.com/rambaut/figtree).

This characterization of each plasmid and of groups of plasmids was accomplished using custom scripts, made freely available at https://github.com/ridgelab/plasmidCharacterization and in the supplementary data (File S1).

Statistical analyses

Since plasmid length distributions are not normal (left-skewed, Fig. S1), all statistical analyses were performed with the Mann–Whitney U-test or the Kruskal–Wallis ranked ANOVA where appropriate, for nonparametric distributions. To be conservative due to our large sample size, statistical significance was determined when $p < 0.0001$.

RESULTS

Plasmid gene composition

Due to the inherent inconsistencies of GenBank record annotations, our search method required discarding 86/532 accessions, leaving a total of 446 accessions in this analysis. The criteria for keeping an accession in the analysis was if at least one, and no more than six, carbapenemase genes were identified on the plasmid (full dataset available in Table S1). To account for poor assembly and annotation due to short-read sequencing technologies, we identified from the metadata which technologies were used. Of the 86 GenBank files discarded, 27 used short-read technologies and 46 used long-read technologies. Of the GenBank files retained, 271 GenBank files noted the sequencing technology used, of which 48 used more than

one with 40 of these using a short-read/ long-read strategy. Overall, there was an even distribution of short- and long-read sequencing technologies (198 short- and 121 long-read technologies). Of those 446 plasmids, 198 carry bla_{KPC}, 168 carry bla_{NDM}, 49 carry bla_{IMP}, and 31 carry bla_{VIM}. When identifying species of origin, 7 plasmids belong to *E. aerogenes*, 33 to *E. cloacae*, 142 to *E. coli*, 235 to *K. pneumoniae*, 18 to *P. aeruginosa*, 3 to *P. stuartii*, and 8 to *S. marcescens*. The mean size of all carbapenemase-carrying plasmids was 104,222 bp, with a median length of 87,663 bp. The largest plasmid was 500,840 bp and the smallest 1,635 bp. The average percent gene content of all plasmids was as follows: antimicrobial-resistance genes, 8.0%; plasmid transfer genes, 15.8%; DNA modification genes, 14.7%; mobile genetic elements, 9.3%; hypothetical genes, 33.2%; other/ metabolic genes, 18.9%. The plasmids carried, on average, two β-lactamases, with 22.6% of the plasmids carrying three or more, and the most β-lactamases on a single plasmid was six. The carbapenemase copy number of these plasmids ranged from one to three, with 97.98% of the plasmids harboring only one copy. Of those that harbored more than one carbapenemase gene, they all belonged to the same type.

Plasmid incompatibility group/replicon typing

No incompatibility group presented itself as the most abundant; however, the following six groups constitute 70.4% of the plasmids: IncA/C2 (45/446, 10.1%), IncFIB (39/ 446, 8.7%), IncFII (58/446, 13%), IncN (56/446, 12.6%), IncX3 (54/446, 12.1%), and multi-replicon plasmids (62/446, 13.9%) (see Table S2). Notably, 7.62% (34/446) of the plasmids could not be accurately typed using this method. Sixty-two plasmids carried more than one replicon, and these were significantly larger than those that carried a single replicon (Mann–Whitney U-test $P < 0.0001$, Fig. S2). Previous work has shown the propensity for bla_{NDM} to be located on IncX3

plasmids, and our work supports this claim with 28% of bla_{NDM}-carrying plasmids on an IncX3 plasmid. We also identify IncFII as a common replicon for bla_{NDM} plasmids (25%) (Fig. 1; Table S3) (Wang et al. 2018). Additionally, we have identified multi-replicon plasmids, IncFIB, and IncN to be the common carriers for bla_{KPC}, and IncA/C2 and IncN replicons as the common carriers for bla_{IMP} (Table S3). Notably, most of the replicon types for bla_{VIM}-resistance plasmids (38%) could not be identified using the PlasmidFinder database.

While the carbapenemases do not seem to be found more often on plasmids of a specific incompatibility group over another, there is a species preference, as would be expected (Fig. 2). With species that have more than five plasmids represented, *E. cloacae*, *S. marcescens*, and *E. aerogenes* more commonly contain IncFII plasmids (30%, 50%, and 43%, respectively); *E. coli* commonly contains IncX3 plasmids (27%); and *K. pneumoniae* are predominately carrying IncFIB, IncN, and multi-replicon plasmids (15%, 12%, and 18%, respectively). Most plasmids from *P. aeruginosa* could not be typed from the PlasmidFinder database since the database is designed for the family *Enterobacteriaceae*.

Of the incompatibility groups from multi-replicon plasmids, the most commonly found was IncFII, present in 48.4% of the plasmids. The other two most common incompatibility groups in multi-replicon plasmids are IncR and IncFIB (35.5% and 29.0%, respectively).

Geographic spread and species promiscuity of plasmids

Among all 446 plasmids, only 32 countries are represented, with the United States of America and China being the predominant countries (54 and 86, respectively). One hundred and ninety-four submissions did not list a country of origin for the plasmid. Additionally, of the 446 plasmids, our intra-BLAST analysis identified 42 indiscrete groups containing 114 plasmids

(Fig. 3). The smallest groups contain 2 plasmids (23 groups) and the largest 48. In total, there were 332 discrete plasmids. Of the seven species of interest in this study, the greatest promiscuity has been seen between *E. coli* and *K. pneumoniae*, with the occasional coincident plasmid in *E. cloacae* and one incidence of an indiscrete plasmid shared between *K. pneumoniae* and *S. marcescens*. Twelve plasmids were of environmental or livestock origin, 139 were from clinical isolates, and the remaining 295 did not provide an isolation source.

Additionally, according to this public data, China is the only country where all four carbapenemase types have been observed. In the following countries, three of the four carbapenemases were observed (not observed): Australia (bla_{VIM}), Canada (bla_{IMP}), Switzerland (bla_{IMP}), Taiwan (bla_{VIM}), and the United States of America (bla_{IMP}). bla_{NDM} was the most widespread carbapenemase, present in 25/32 countries. Interestingly, bla_{IMP} was only reported from Asian and Oceanic countries (Australia, China, Japan, Taiwan, and Thailand).

Additionally, in countries that had at least 10 plasmids, we identified the predominant incompatibility group in that country (Table 1).

DISCUSSION

In general, data on carbapenemase-producing plasmids from less common but still clinically important organisms such as *P. aeruginosa* and relevant carbapenemases such as bla_{VIM} is severely lacking. Additionally, global epidemiologic studies of carbapenemase-carrying plasmids are further complicated by the lack of GenBank metadata found. Differences between infection-reporting requirements and research efforts among different countries, and the fact that these plasmids are not routinely sequenced, further complicates these analyses.

The cladogram showing the nondiscrete plasmid groups (Fig. 3) is quite illuminating, but it is also the most biased due to large sequencing projects of local outbreaks. This may be the case for the over-representation of plasmids from China, especially the IncX3 group. However, the intercontinental nature of these nondiscrete plasmids, particularly the IncFIB group present on four separate continents, indicates either that these plasmids are very stable or that they can spread at a speed at which they do not accumulate significant mutations. Conversely, the fact that common incompatibility groups such as IncFII do not cluster with similar nondiscrete plasmids could be explained by them simply being more diverse or that they have not been identified during a sequencing project of a hospital CRE outbreak.

Furthermore, to effectively track and monitor the spread of carbapenem-resistance plasmids in local outbreaks, rapid identification is critical. Current clinical practices (blood culture, followed by isolation and PCR) have a 48–72 h delay before carbapenemase resistance is determined. For the more rapid, nonPCR-based methods using whole blood (such as Knob et al. 2018), it is important to realize that the plasmids of interest are quite large. With their median length over 80 kb, plasmid isolation becomes difficult when necessary for the application, and many of the replicon types identified are from low copy number plasmids.

Also, this report supports rational methods of several groups using targeted approaches to slow the spread of carbapenemase plasmids. First, the antitoxin of the plasmid addiction system is currently a target (Tsang 2017). Targeting this system could prevent its binding with the toxin, resulting in the death of the host harboring the plasmid. However, this would not be a universal target since only 52.9% of the plasmids contain toxin/antitoxin systems (Table S1). And secondly, 90.4% (403/446) of the plasmids carry transfer genes to pass the plasmid be- tween bacteria (Table S1), which is also supported by the evidence shown here of nondiscrete plasmids

appearing in multiple species. Preventing pilus formation could dramatically reduce the spread of these plasmids. This direction is currently being pursued by several groups employing strategies such as bacteriophage, colloidal clays, and antibody therapy (Getino and de la Cruz 2018). Targeting both mechanisms simultaneously may dramatically reduce the spread and persistence of these plasmids in the hospital.

Ultimately, this analysis was very difficult due to the nonstandardization of GenBank metadata and the under-reporting and publication of carbapenemase-carrying plasmids from different countries. This is a severe limitation in the complete comprehension of the carbapenem-resistance epidemic, and more effort needs to be focused on these under-reported carbapenemases and species (VIM and IMP, *P. aeruginosa*). However, we were able to support work done by other groups, by showing the prevalence of diverse targets (toxin/ antitoxin and conjugal transfer) among these plasmids. These efforts may ultimately help stem the tide of increasing global carbapenem resistance.

CONFLICT OF INTEREST STATEMENT

The authors have no conflicts of interest to declare.

ACKNOWLEDGEMENTS

G.E.C. conceptualized this analysis, determined the functional groups of interest, generated the key terms, and analyzed the output from the scripts. **B.D.P.** wrote the scripts for the analysis and assisted in writing a portion of the manuscript. **P.G.R.** and **R.A.R.** advised this work and reviewed the manuscript. We thank the Fulton Supercomputing Laboratory (https://marylou.byu.edu) at Brigham Young University for their consistent efforts to support our

research. We would also like to thank the curators of the PlasmidFinder database (Henrik Hasman and Alessandra Carattoli) for keeping that information up to date and accessible. This work was supported by the U.S. National Institutes of Health (R01 AI116989).

TABLES & FIGURES

Supplementary Tables (File S2) and Supplementary Figures (File S3) are available online or herein as Appendix 8 and Appendix 9, respectively.

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Predominant incompatibility group and carbapenemase prevalence in countries with more than 10 representative plasmids.

Country	*Incompatibility Group*	*Percent of plasmids (no./Total)*	*Percent carbapenemase in predominant Incompatibility group*
Australia	IncFIB	45.5% (5/11)	KPC 80.0% (4/5); IMP 20.0% (1/5)
Brazil	IncN	50.0% (8/16)	KPC 100.0% (8/8)
Canada	IncFII	33.3% (5/15)	KPC 40.0% (2/5); NDM 60.0% (3/5)
China	IncFII	26.7% (23/86)	KPC 82.6% (19/23); NDM 17.4% (4/23)
United States of America	IncFIB	24.1% (13/54)	KPC 61.5% (8/13); NDM 38.5% (5/13)

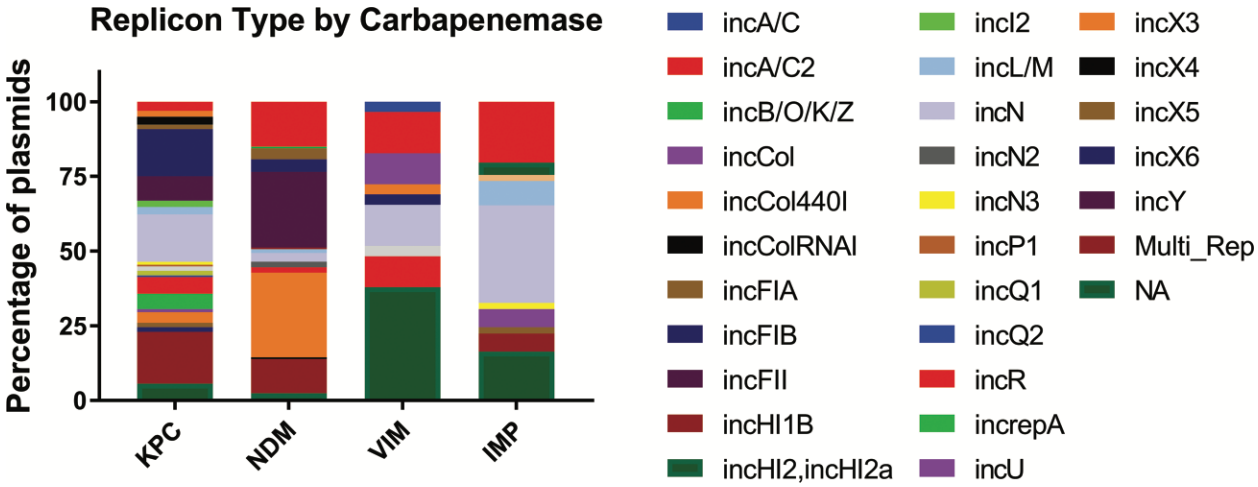


Figure 1. Relative abundance of incompatibility groups among plasmids. Predominant incompatibility groups from each carbapenemase family: KPC, IncFIB (15.8%), IncN (15.8%), and multi-replicon (17.3%); NDM, IncA/C2 (15.1%), IncFII (25.3%), IncX3 (28.3%), and multi-replicon (11.4%); IMP, IncA/C2 (22.4%), IncN (32.7%), and NA (8/49 16.3%); VIM, IncA/C2 (16.1%), IncN (13.8%), IncR (10.3%), and NA (37.9%).

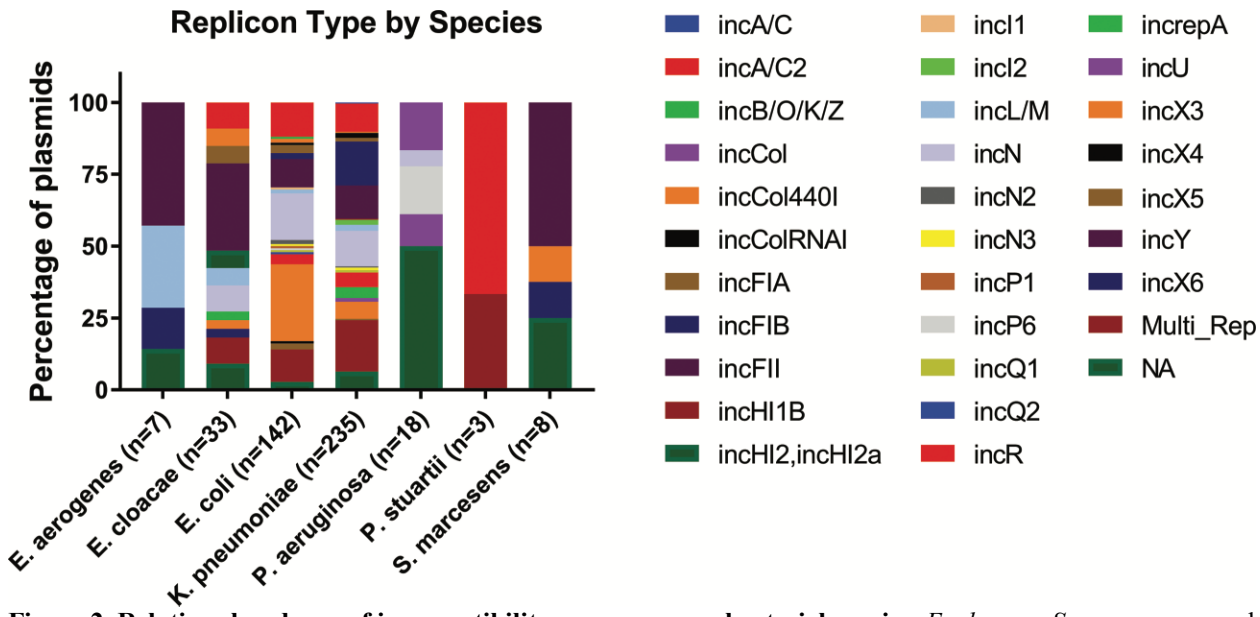


Figure 2. Relative abundance of incompatibility groups among bacterial species. *E. cloacae*, *S. marcescens*, and *E. aerogenes* prefer FII plasmids (30.3%, 50%, and 42.9% respectively), *E. coli* prefer X3 plasmids (26.8%) and FIB and multi-replicon plasmids predominate in *K. pneumoniae* (15.3 and 17.9 respectively). The majority of plasmids from *P. aeruginosa* could not be typed from the PlasmidFinder database (50%).

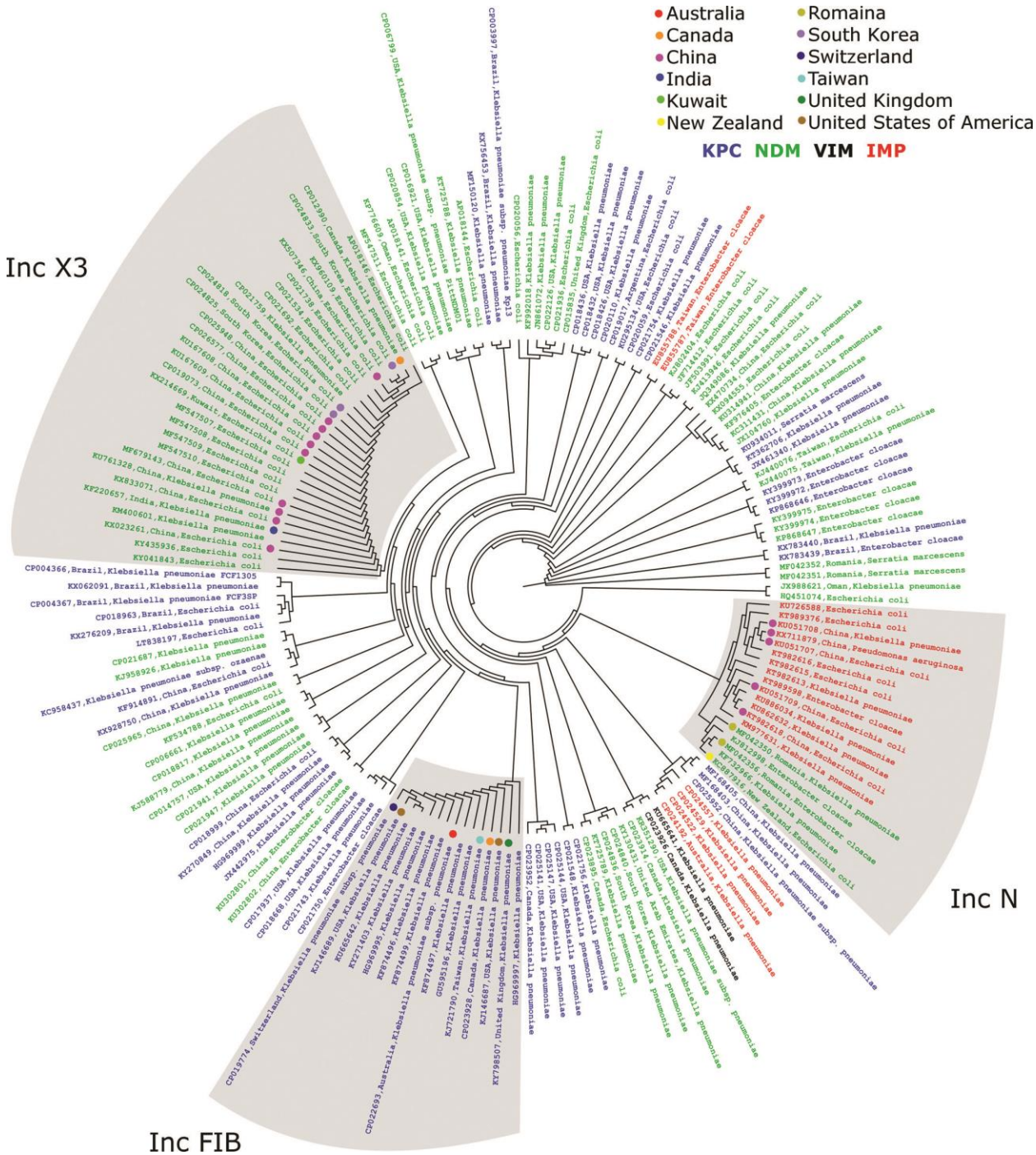


Figure 3. Indiscrete plasmid groups. Cladogram showing the nucleotide relationships between plasmids that have >98% query coverage and identity. The geographic distribution of these plasmids in the three largest groups has been identified by colored dots. Blue text = KPC carrying plasmid, green = NDM, red = IMP, and black = VIM.

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. J. Mol. Biol. **215**: 403-410. doi:10.1016/S0022-2836(05)80360-2. PMID:2231712.

Arnold, R.S., Thom, K.A., Sharma, S., Phillips, M., Johnson, J.K., and Morgan, D.J. 2011. Emergence of *Klebsiella pneumoniae* Carbapenemase (KPC)-producing bacteria. South. Med. J. **104**(1): 40-45. doi:10.1097/SMJ.0b013e3181fd7d5a. PMID:21119555.

Bedenić, B., Plečko, V., Sardelić, S., Uzunović, S., and Godič Torkar, K. 2014. Carbapenemases in gram-negative bacteria: laboratory detection and clinical significance. BioMed Res. Int. **2014**: 841951. doi:10.1155/2014/841951. PMID: 25025071.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T.L. 2009. BLAST+: architecture and applications. BMC Bioinform. **10**: 421. doi:10.1186/1471-2105-10- 421.

Carattoli, A., Zankari, E., Garcia-Fernandez, A., Voldby Larsen, M., Lund, O., Villa, L., et al. 2014. In silico detection and typing of plasmids using PlasmidFinder and plasmid multilocus sequence typing. Antimicrob. Agents Chemother. **58**(7): 3895-3903. doi:10.1128/AAC.02412-14. PMID:24777092.

Centers for Disease Control and Prevention. n.d. Tracking CRE [online]. Available from https://www.cdc.gov/hai/organisms/cre/trackingcre.html [accessed 8 June 2018].

Chen, F.J., Huang, W.C., Liao, Y.C., Wang, H.Y., Lai, J.F., Kuo, S.C., et al. 2019. Molecular epidemiology of emerging carbapenem resistance in *Acinetobacter nosocomialis* and *Acinetobacter pittii* in Taiwan, 2010–2014. Antimicrob. Agents Chemother. doi:10.1128/ aac.02007-18.

Codjoe, F.S., and Donkor, E.S. 2018. Carbapenem resistance: a review. Med. Sci. **6**(1). doi:10.3390/medsci6010001. PMID: 29267233.

Getino, M., and de la Cruz, F. 2018. Natural and artificial strategies to control the conjugative transmission of plasmids. Microbiol. Spectrum, **6**(1). doi:10.1128/microbiolspec.MTBP-0015-2016. PMID:29327679.

Johnson, A.P., and Woodford, N. 2013. Global spread of antibiotic resistance: the example of New Delhi metallo-beta- lactamase (NDM)-mediated carbapenem resistance. J. Med. Microbiol. **62**: 499-513. doi:10.1099/jmm.0.052555-0. PMID: 23329317.

Klevens, R.M., Edwards, J.R., Richards, C.L., Horan, T.C., Gaynes, R.P., Pollock, D.A., and Cardo, D.M. 2007. Estimating health care-associated infections and deaths in U.S. hospitals, 2002. Public Health Reports, **122**(2): 160-166. PMID: 17357358.

Knob, R., Hanson, R.L., Tateoka, O.B., Wood, R.L., Guerrero-Arguero, I., Robison, R.A., et al. 2018. Sequence-specific sepsis-related DNA capture and fluorescent labeling in monoliths prepared by single-step photopolymerization in microfluidic devices. Journal of Chromatography A, **1562**: 12–18. doi:10.1016/j.chroma.2018.05.042. PMID:29859687.

Logan, L.K., and Weinstein, R.A. 2017. The epidemiology of carbapenem-resistant Enterobacteriaceae: the impact and evolution of a global menace. The Journal of Infectious Dis- eases, **215**(suppl_1): S28–S36. doi:10.1093/infdis/jiw282. PMID: 28375512.

Magill, S.S., Edwards, J.R., Bamberg, W., Beldavs, Z.G., Dumyati, G., Kainer, M.A., et al. 2014. Multistate Point-prevalence survey of health care-associated infections. New England Journal of Medicine, **370**(13): 1198-1208. doi:10.1056/NEJMoa1306801. PMID:24670166.

Mansour, W., Grami, R., Jaidane, N., Messaoudi, A., Charfi, K., Ben Romdhane, L., et al. 2019. Epidemiology and whole-genome analysis of NDM-1-producing *Klebsiella pneumoniae* KP3771 from Tunisia. Microb. Drug Resist. **25**(5). doi:10.1089/mdr.2018.0204.

Marsik, F.J., and Nambiar, S. 2011. Review of carbapenemases and AmpC-beta-lactamases. Pediatr. Infect. Dis. J. **30**(12): 1094-1095. doi:10.1097/INF.0b013e31823c0e47. PMID:22105420.

Miller, J.B., McKinnon, L.M., Whiting, M.F., and Ridge, P.G. 2019. CAM: an alignment-free method to recover phylogenies using codon aversion motifs. PeerJ. **7**: e6984. doi:10.7717/peerj.6984. PMID:31198636.

Mukherjee, S., Bhattacharjee, A., Naha, S., Majumdar, T., Debbarma, S.K., Kaur, H., et al. 2019. Molecular characterization of NDM-1-producing *Klebsiella pneumoniae* ST29, ST347, ST1224, and ST2558 causing sepsis in neonates in a tertiary care hospital of North-East India. Infect. Genet. Evol. **69**: 166-175. doi:10.1016/j.meegid.2019.01.024. PMID:30677535.

Orlek, A., Phan, H., Sheppard, A.E., Doumith, M., Ellington, M., Peto, T., et al. 2017. Ordering the mob: insights into replicon and MOB typing schemes from analysis of a curated dataset of publicly available plasmids. Plasmid, **91**: 42-52. doi:10.1016/ j.plasmid.2017.03.002. PMID:28286183.

Overturf, G.D. 2010. Carbapenemases: a brief review for pediatric infectious disease specialists. Pediatric Infect. Disease J. **29**(1): 68-70. PMID:20035208.

Paterson, D.L., and Bonomo, R.A. 2005. Extended-spectrum β-lactamases: a clinical update. Clin. Microbiol. Rev. **18**(4): 657-686. doi:10.1128/CMR.18.4.657-686.2005. PMID:16223952.

Perez, F., and Van Duin, D. 2013. Carbapenem-resistant Entero- bacteriaceae: a menace to our most vulnerable patients. Cleveland Clin. J. Med. **80**(4): 225-233. doi:10.3949/ccjm.80a.12182. PMID:23547093.

Piazza, A., Comandatore, F., Romeri, F., Brilli, M., Dichirico, B., Ridolfo, A., et al. 2019. Identification of *bla*VIM-1 gene in ST307 and ST661 *Klebsiella pneumoniae* clones in Italy: old acquain- tances for new combinations. Microb. Drug Resist. **25**(5): 787-790. doi:10.1089/mdr.2018.0327. PMID:30589602.

Poirel, L., Potron, A., and Nordmann, P. 2012. OXA-48-like carbapenemases: the phantom menace. J. Antimicrob. Chem. **67**(7): 1597-1606. doi:10.1093/jac/dks121. PMID:22499996.

Read, A.F., and Woods, R.J. 2014. Antibiotic resistance management. Evol. Med. Publ. Health, **2014**(1): 147. doi:10.1093/emph/eou024. PMID:25355275.

Sheppard, A.E., Stoesser, N., Wilson, D.J., Sebra, R., Kasarskis, A., Anson, L.W., et al. 2016. Nested Russian doll-like genetic mobility drives rapid dissemination of the carbapenem resistance gene bla_{KPC}. Antimicrob. Agents Chemother. **60**(6): 3767-3778. doi:10.1128/AAC.00464-16. PMID:27067320.

Sotgiu, G., Are, B.M., Pesapane, L., Palmieri, A., Muresu, N., Cossu, A., et al. 2018. Nosocomial transmission of carbapenem-resistant *Klebsiella pneumoniae* in an Italian university hospital: a molecular epidemiological study. J. Hosp. Infect. **99**(4): 413-418. doi:10.1016/j.jhin.2018.03.033. PMID:29621600.

Stoesser, N., Sheppard, A.E., Peirano, G., Anson, L.W., Pankhurst, L., Sebra, R., et al. 2017. Genomic epidemiology of global *Klebsiella pneumoniae* carbapenemase (KPC)-producing *Escherichia coli*. Sci. Rep. **7**(1): 5917. doi:10.1038/s41598-017-06256-2. PMID:28725045.

Tindall, B.J., Sutton, G., and Garrity, G.M. 2017. *Enterobacter aerogenes* Hormaeche and Edwards 1960 (Approved Lists 1980) and *Klebsiella mobilis* Bascomb et al. 1971 (Approved Lists 1980) share the same nomenclatural type (ATCC 13048) on the Approved Lists and are homotypic synonyms, with consequences for the name *Klebsiella mobilis* Bascomb et al. 1971 (Approved Lists 1980). Int. J. Syst. Evol. Microbiol. **67**(2): 502-504. doi:10.1099/ijsem.0.001572. PMID:27902205.

Tsang, J. 2017. Bacterial plasmid addiction systems and their implications for antibiotic drug development. Postdoc J. **5**(5): 3-9. PMID:28781980.

Wang, Y., Tong, M.-K., Chow, K.-H., Cheng, V.C.-C., Tse, C.W.-S., Wu, A.K.-L., et al. 2018. Occurrence of highly conjugative IncX3 epidemic plasmid carrying bla_{NDM} in *Enterobacteriaceae* isolates in geographically widespread areas. Front. Microbiol. **9**: 2272. doi:10.3389/fmicb.2018.02272. PMID:30294321.

Watanabe, M., Iyobe, S., Inoue, M., and Mitsuhashi, S. 1991. Transferable imipenem resistance in *Pseudomonas aeruginosa*. Antimicrob. Agents Chemother. **35**(1): 147-151. doi:10.1128/AAC.35.1.147. PMID:1901695.

Yigit, H., Queenan, A.M., Anderson, G.J., Domenech-Sanchez, A., Biddle, J.W., Steward, C.D., et al. 2001. Novel carbapenem-hydrolyzing β-lactamase, KPC-1, from a carbapenem-resistant strain of *Klebsiella pneumoniae*. Antimicrobial Agents and Chemotherapy, **45**(4): 1151-1161. doi:10.1128/AAC.45. 4.1151-1161.2001. PMID:11257029.

Yong, D., Toleman, M.A., Giske, C.G., Cho, H.S., Sundman, K., Lee, K., and Walsh, T.R. 2009. Characterization of a new metallo-β-lactamase gene, *bla*$_{\text{NDM-1}}$, and a novel erythromycin esterase gene carried on a unique genetic structure in *Klebsiella pneumoniae* sequence type 14 from India. Antimicrob. Agents Chemother. **53**(12): 5046-5054. doi:10.1128/AAC.00774-09. PMID:19770275.

CHAPTER 8

TANOS: TAxon jackknife for NOdal Stability with genomic data

Gareth S. Powell[1*], Brandon D. Pickett[1*], Gavin J. Martin[1], Michael F. Whiting[1], Perry G. Ridge[1], Seth M. Bybee[1]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

[*]*These authors contributed equally to this work*

ABSTRACT

Motivation: As phylogenetic data sets increase in size due to high-throughput sequencing, standard nodal support values (e.g., bootstrap values) quickly reach full support and thus provide minimal value in assessing tree stability within or across topologies. With this increase in loci coverage, some approaching full genomic scales, the main limitation in current and future phylogenetics has shifted to taxon sampling. However, few strategies remain to assess the strength of a given taxon sampling scheme or identifying troublesome and potentially undersampled regions of a topology. How stable is a given node to the utilized taxon sampling?

Results: We present TANOS (TAxon jackknife for NOdal Stability), which uses traditional resampling without replacement for taxa in genomics-scale datasets to compute nodal stability scores for the phylogenetic tree of interest. Resampled trees are compared, and all internal nodes are recorded. After tabulating the presence of each internal node in all jackknifed trees, a measure of nodal stability is generated and reported. Reported values provide insight into the stability of a given node to the included taxon sampling.

Availability and implementation: The source code is freely available on GitHub at https://github.com/pickettbd/TANOS.

1. INTRODUCTION

Resampling methods are those techniques that create many subsamples of data from an original dataset. In phylogenetics these approaches have been applied to both sequence and morphological data matrices as a means to measure nodal "support", via assessing data agreement across a topology (Efron 1979, Lanyon 1985). Bootstrapping, simply explained, is subsampling with replacement and in phylogenetics is commonly applied but is limited to characters (whether nucleotides or morphological features), jackknifing is subsampling without replacement and therefore can be applied to characters or taxa. The failings of both bootstrapping and jackknife approaches to nodal support with traditional phylogenetic datasets (i.e., small Sanger-based datasets) are documented within the literature (Felsenstein 1985). However, jackknifing has the clear philosophical advantage over bootstrap in that it does not skew the observed data; specifically, applying additional weight to a given character in the dataset by resampling it multiple times. Additionally, jackknifing as an approach to taxon stability has not been fully explored, especially in the current day of genomic scale data and phylogenetics. Herein we produce a robust approach to assess taxon sampling schemes while also identifying troublesome and potentially undersampled regions of a topology that are particularly useful with modern and large phylogenetic datasets.

Tukey (1958) coined the term "jackknife" and specifically used the method to explore how a given outcome was affected by subsets of the original observations of the total dataset. In this context jackknifing methods are methods of random subsampling without replacement. From a statistical standpoint, jackknife methodologies and the theory predicating its usage is reviewed by Miller (1974) and subsequently summarized by Efron (1979). Miller argues jackknifing a dataset reduces overall bias in that dataset and attempts to prove this formulaically.

1.1 Character Jackknife in Phylogeny

Lanyon (1985) proposed "a technique for investigating variance within a dataset" and coined the "jackknife approach" within phylogenetics. He provides both a biological and statistical argument for jackknifing approaches being beneficial when reconstructing trees. The biological justification is based around the value of pseudoreplicates in cases of ideal datasets containing redundancy. He argues that any given dataset only contains a small subset of data from the evolutionary history that has actually taken place between a given taxon and its sister species. Apart from this subset of data, the remainder of the data are informative at ancestral nodes and represents the evolutionary history of more than just that terminal taxon. Lanyon refers to these data as redundant across multiple included taxa and pointed out that conflicting data will result in internal inconsistencies across the topology.

Both Lanyon (1985) and Felsenstein (1985) discuss what jackknife techniques add to phylogenetics. Through the use of strict consensus trees, Lanyon (1985) focuses on the utility of identifying where all subtrees agree and identifying disagreement. He also argues those internal inconsistencies are not a reflection of complex speciation events resulting in multiple new taxa, but merely an unresolved region of the tree. It must be pointed out that Lanyon proposed this method within a distance-based framework and used it to specifically find inconsistencies in distance data being used for phylogenetic estimation.

Lanyon (1987) further outlines these techniques as they apply to phylogenetics. He states "*The use of jackknifing and bootstrapping should enable investigators to learn more about their data than was previously possible because of the information on the dispersion of sample statistics. I hasten to add that this situation does not imply that investigators will be able to*

conclude more from their data". He goes on to point out that these methods, as with all statistical procedures, do come with limitations and assumptions that need to be taken into account before using such tools. Lanyon (1987) argued that the real value in jackknifing is the amount of data exploration that these tools allow.

Simmons and Freudenstein (2011) investigated the seeming inflation of support values and what they called "Spurious 99%" bootstrap or jackknife support values. Using both contrived and empirical real world examples the authors demonstrated these erroneous examples of high support at the nodes. The authors end that article with a list of recommendations based on situations with high amounts of missing data, or low overlap in loci across terminals, or supermatrix approaches. Recommendation number two is largely ignored but simply stated "JK (jackknife) resampling be used rather than BS (bootstrap) resampling." These authors were clearly focused on character jackknifing and not a taxon approach; however, it is clear that jackknifing is an underused method in modern phylogenetics.

1.2 Taxon Jackknifing and the Taxon Influence Index

Wrobel (2008) reviewed methods for identifying uncertainty in phylogeny, specifically with those estimated based on molecular data. In this review he contemplated both the character jackknife and the taxon jackknife. He posited that the character jackknife may not be as popular as the bootstrap despite its theoretical similarity due to the often overall lower values it resulted in for nodal support. When giving an overview of taxon jackknife he notes that many argued "species" are not independent and so the statistical implications of removing a subset of taxa are even less understood than in character jackknifing. While certainly true, it underestimates the

power of a taxon jackknife to identify unstable portions of the topology, it also ignores the obvious lack of independence in the vast majority of molecular and morphological data as well.

An alternate use of jackknifing methods has been developed for maximum likelihood analyses. Mariaassou et al. (2012) explored the use of taxon resampling in molecular alignments and execution in maximum likelihood tree reconstruction. These methods are based on the sequential removal of individual terminals and comparison of resulting topologies. They argue that based on the changes in topology when a taxon is removed, a metric for evolutionary importance can be generated, the Taxon Influence Index (TII). This tool has been used to help identify key taxa that have a larger impact on the phylogeny than surrounding species (Denton et al. 2017); however, it is not used as a metric of internal nodal stability nor as a way to identify potential weaknesses in an overall taxon sampling. Instead of using taxon jackknifing to assess the overall stability of a given node to sampling, TII identifies which terminals have the most effect on parent nodes. Conceptually, TII is a tool that can be used with a phylogenetic analysis to identify influential taxa, these taxa could be considered influential due to being relics, representing large diverse clades, or actually reflect flaws in the original sampling. Both of these examples provide the theoretical foundation for TANOS and demonstrate utility even on a smaller scale.

1.3 Needs in a genomics era

In this era, the traditional limitations of phylogenetics due to the lack of character coverage can be argued has largely gone away. What remains is the influence of taxon sampling and the problem of reconstructing relationships between often extremely diverse clades with relatively few representatives. Genomic-scale data are more readily available and its usage in

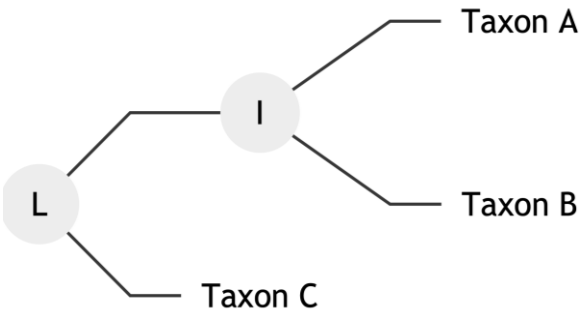
phylogenetics is constantly growing, an improved platform to investigate taxon stability is needed. Here, we present a new program TANOS, capable of evaluating taxon jackknifes of very large molecular datasets based on modern tree reconstruction methods, answering the core question, how stable is a given topology to the removal of taxa?

2. MATERIALS AND METHODS

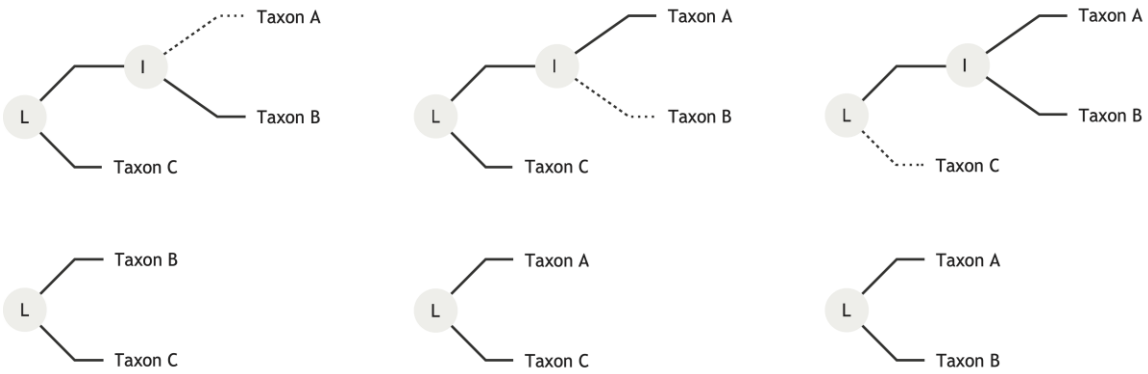
To calculate how stable each node in a given topology is, additional trees must be constructed with taxa removed. These new trees can be compared with the original provided tree, and a score can be assigned to each node. The process can be summarized by the following steps: (a) subset alignments, (b) generate new trees, and (c) compute stability scores. The first two steps are routine, if potentially computationally expensive; they can be completed with basic scripting and existing software packages. The third step is unique and required the conception and implementation of a new algorithm. The process is most easily understood conceptually and visually before getting to the implementation details.

2.1 Conceptual Examples

The core question is how stable the current topology is to the removal of taxa? Accordingly, the same question extends to each internal node of the tree. The answer may vary throughout the tree. Regardless of which node is currently being evaluated, the overall stability to the removal of taxa is a combination of the stability to the removal of each individual taxon. Consider a simple example tree with taxa A-C (internal nodes, L and I, are also labeled for convenient reference):



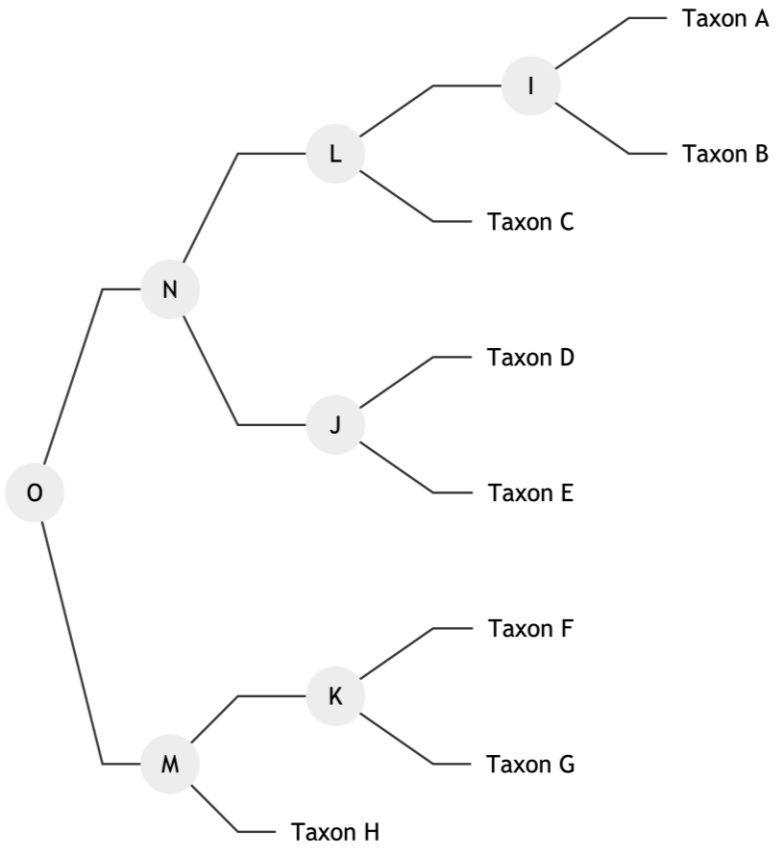
To determine the overall stability of node L, the results for the stability of node L to the individual removal of each taxon must be combined. Once a taxon is removed from the tree, the remaining taxa can be considered as a set. Trees built without that taxon can then be queried to see how many trees also contain a node with the same taxa set. For example, if A is removed from the tree, the node L effectively has only two taxa, forming the set {B,C}. If trees built without A are queried, the percentage of those trees containing this same set as a clade can be identified. Likewise, the trees built without B and C can be queried for presence of sets {A,C} and {A,B}, respectively.



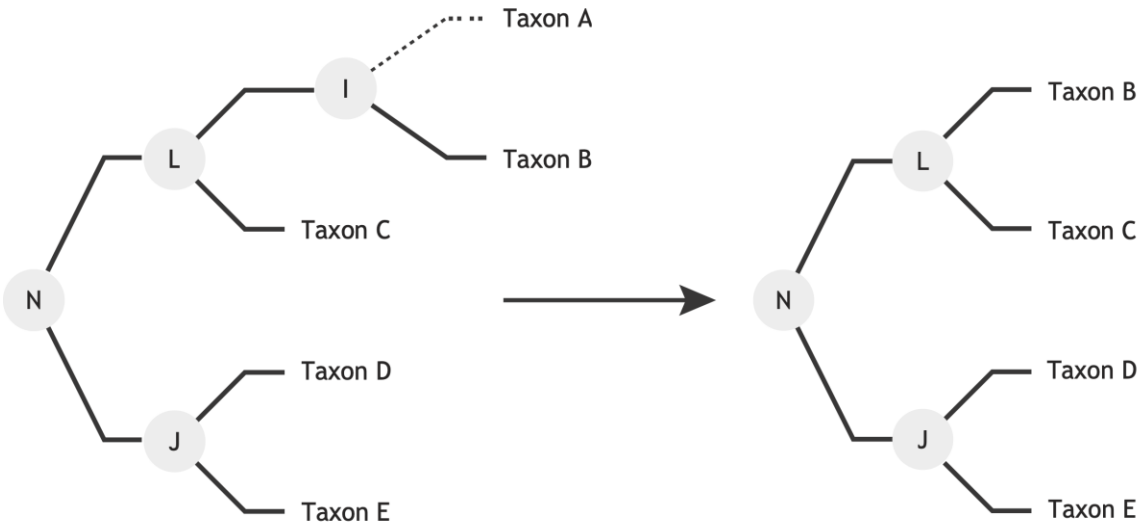
As this entire tree has only three total taxa, each tree built without the removed taxon is guaranteed to have the set being searched for; thus, stability for that given node to the removal of A would be 1 (100%), likewise for the removal of B and C. If the stability is averaged for all taxa, the final score is 1 ((1+1+1)/3). One could follow the same procedure to evaluate node I, but the effort would be similarly wasted as a set containing a single taxon (the result of removing

one from a set of two taxa) will necessarily be found in any subsequent trees made with that taxon. This simple example tree demonstrates that the score at the root and parents of terminal nodes is, by definition, always 1.

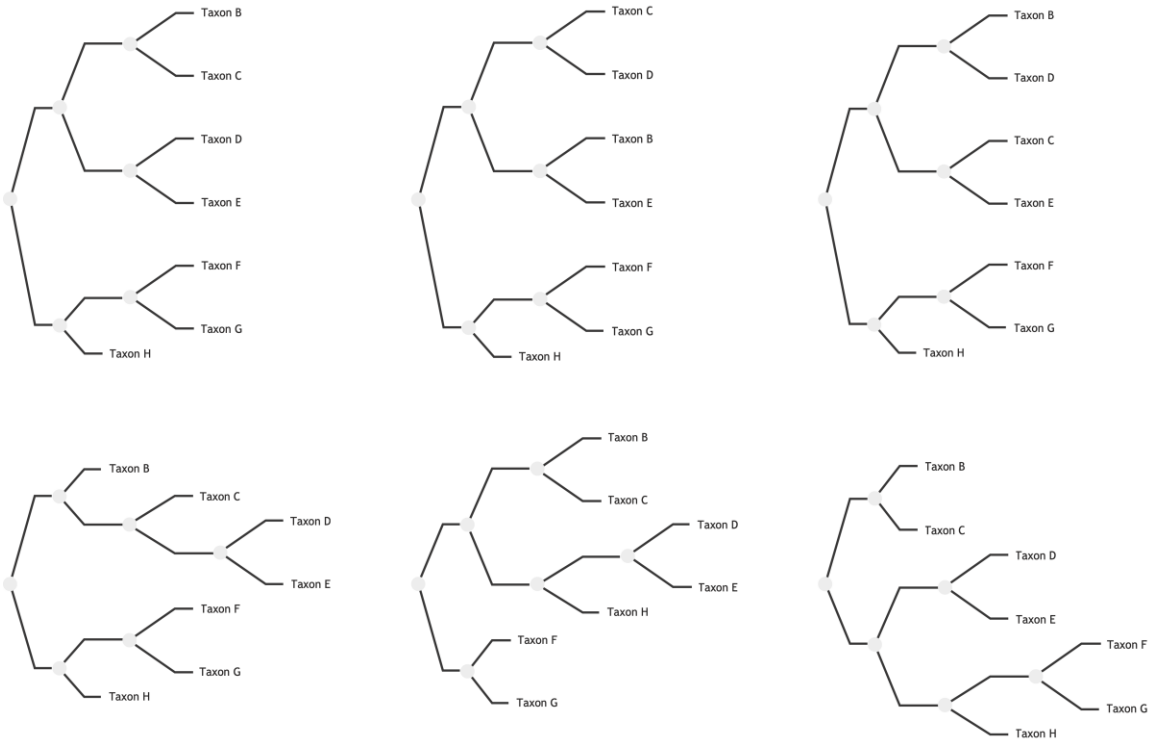
Consider the following more complex expanded example tree for taxa A-H (internal nodes, I-O, are also labeled for convenient reference):



By definition, node O (the root) and nodes I, J, and K (parents of only terminal nodes) will all receive a score of 1. Thus, nodes L, M, and N remain to be evaluated. Node N will be demonstrated here, but the procedure is the same for nodes L and M. Five taxa are in the clade under node N: taxa A-E. Consider first the stability of node N to the removal of taxon A; removing A leaves the set {B, C, D, E} at node N and set {B, C, D, E, F, G, H} at node O:



Trees with taxa B-H (the set remaining from node O, {B, C, D, E, F, G, H}) are generated with a predetermined level of replication, six in this example. To determine the frequency with which set {B, C, D, E} occurs, a node containing only those taxa B-E is searched for in each tree and tallied:



Four of these six trees contain a node with the set {B, C, D, E}, the top three and the bottom-left. The remaining two trees do not contain the requisite set; thus, the frequency of occurrence of set {B, C, D, E} is 0.67 (4/6). For the sake of this example, assume the same procedure is followed for the removal of the remaining taxa (B, C, D, and E) from node N and frequencies of occurrence were obtained. If the other frequencies were 0.5, 0.33, 0.83, and 0.67, they could then be averaged to obtain 0.6 ((0.67+0.5+0.33+0.83+0.67)/5). The same procedure can be followed for nodes L and M with the systematic removal of taxa A-C and F-H, respectively.

2.1.1 Meta-Methods

All trees shown in these examples were generated from files in Mermaid format (http://mermaid-js.github.io/mermaid) using the associated command-line interface Mermaid-CLI v8.5.3 (https://github.com/mermaid-js/mermaid-cli), which can generate diagrams and charts from text in a similar manner to Markdown (https://daringfireball.net/projects/markdown). The command to generate a vector-based image is structured like the following:

```
mmdc -b transparent -i input.mmd -o output.pdf
```

2.2 Detailed Methods

Before nodal stability scores can be calculated, the tree must be jackknifed, which is a computationally expensive process. The first step is to prepare input matrices (i.e., the alignments) for building the sampled trees, which is a simple task conceptually and computationally. The second task is to generate the $N \cdot R$ trees, where N is the number of taxa in the original tree and R is the desired level of replication (in our case, $144 \cdot 50 = 7{,}200$). Assigning nodal stability scores to every node using TANOS is computationally tractable. The implementation details will be provided after the preparatory steps are described.

2.2.1 Subsetting Alignments

Creating subset copies of the original alignment file (the input to the software used to create the tree) will vary depending upon the original file format and desired output file format. The sample Insect and related Arthropod alignment file we downloaded from Misof et. al (2014) was in PHYLIP format. Our script to parse an alignment in PHYLIP format and create new subsets in FASTA format is available with the TANOS code on GitHub. It will create N new files, each named after the taxon that has been removed to have N-1 taxa in each alignment, where N is the number of taxa in the original alignment. PHYLIP format is trickier to parse than other formats as it is designed to be more human readable than machine readable. For record-centric formats (e.g., FASTA), the following pseudocode describes the process:

```
# parse the input file
name_to_sequence_map = {}
input_file = open("some_name.txt ", 'w')
for record in input_file:
  name_to_sequence_map[ record.name ] = record.sequence
input_file.close()

# loop through each of the taxa, creating an output file for each
for name_to_exclude in name_to_sequence_map.keys():
  # write the output alignment without the taxon name_to_exclude
  output_file = open(name_to_exclude + ".fa ", 'w')
  for name in name_to_sequence_map.keys():
          if name != name_to_exclude:
                  sequence = name_to_sequence_map[ name ]
                  output_file.write('>' + name + '\n ' + sequence + '\n
')
  output_file.close()
```

For PHYLIP format, the pseudocode looks like the following:

```
# parse the input file
input_file = open("some_name.phy", 'r')

# process first line
num_taxa = input_file.getFirstLine().num_taxa
names = array[num_taxa]
sequences = array[num_taxa]
```

```
# process first section
section = input_file.getFirstAlignmentSection()
for i in range(num_taxa):
  line = section.getNextLine()
  names[i] = line.name
  sequences[i] = line.sequence

# process remaining sections
for section in input_file.getRemainingAlignmentSections():
  for i in range(num_taxa):
          line = section.getNextLine()
          names[i] += line.name
          sequences[i] += line.sequence
input_file.close()

# loop through each of the taxa, creating an output file for each
for i in range(num_taxa):
  name_to_exclude = names[i]
  # write the output alignment without the taxon name_to_exclude
  output_file = open(name_to_exclude + ".fa ", 'w')
  for j in range(num_taxa):
          name = names[j]
                  if name != name_to_exclude:
                          sequence = sequences[j]
                          output_file.write('>' + name + '\n ' + sequence
+ '\n ')
  output_file.close()
```

Of course, instead of subsetting the alignment, one could generate entirely new alignments. This could further mitigate the influence of any given taxon on the resulting trees, at the cost of increasing computational requirements.

2.2.2 Generating Trees

The primary tree was built with IQ-TREE v1.6.12 (Nguyen et al. 2015), and model selection (Kalyaanamoorthy et al 2017) resulted in GTR+F+I+G4. This same model was used as input to IQ-TREE for each of the 7,200 jackknife trees to avoid the extra computation of model selection for every tree. Of course, IQ-TREE could be substituted for any other software package preferred by someone seeking to perform a similar analysis. The command to perform model selection is the following:

```
iqtree -nt ${THREADS} \
  -mem ${MEMORY}G \
  -s ${INPUT_ALIGNMENT} \
  -pre ${OUTPUT_PREFIX} \
  -m TESTONLY
```

The command to generate the primary tree and subsequent trees was structured like the following:

```
iqtree -nt ${THREADS} \
  -mem ${MEMORY}G \
  -s ${INPUT_ALIGNMENT} \
  -pre ${OUTPUT_PREFIX} \
  -m ${MODEL}
```

In our case, all jobs were provided 16GB of RAM and 24 threads; each job finished in less than four days. Generating a single tree is a simple computational problem and can be finished in a day. However, generating thousands of trees, each requiring resources and a few days of computation, requires access to a compute cluster.

Job management was done with a pipelining software and is available with the TANOS code on GitHub. It relies on the checkpoint file created by IQ-TREE, which is how IQ-TREE keeps track of its own progress across multiple runs if it is killed early. In effect, a job is submitted if either no checkpoint file exists, or the file reports the analysis was not yet completed. When all jobs are terminated, rerunning the script will attempt any job without a checkpoint file, this is repeated until no new jobs are started and all analyses have a generated checkpoint file. At this point, all trees are successfully created.

2.2.3 Calculating Nodal Stability

Once the jackknife trees are generated, TANOS is able to calculate stability scores for each node in the primary tree. The software is implemented in Python v3.6+ (https://python.org) and is available on GitHub (https://github.com/pickettbd/TANOS) and the Python Package Index

(https://pypi.org/project/tanos). As input, TANOS requires the primary tree, the jackknife trees, and a text file providing a mapping of taxon names to file paths with trees built without that particular taxon. As output, it writes to file the tree with nodal stability scores. In our primary test case, it was able to calculate the scores for a tree with 144 taxa, which included evaluating the 7,200 trees with 143 taxa each, in a few minutes using a single thread.

The score is given individually to each node and is bounded by [0,1], where 0 and 1 respectively denote that no and all jackknife trees contain the same node. The score is the average frequency of occurrence of the node in the jackknife trees, counting the node as present if a node exists with the same taxa minus the taxon removed for that jackknife. The score for a given node can be described formulaically:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{r}\sum_{j=1}^{r} f(N_i, J_{ij})$$

Where N is the set of taxa of length n under the node in question from the primary tree with N_i denoting the subset of N without i, J is a set of jackknife trees with J_i denoting a set containing r replicates of trees made without taxon i and J_{ij} denoting the j-th replicate tree made without taxon i, and $f(N_i, J_{ij})$ is a function yielding 1 if and only if N_i exists in J_{ij}, 0 otherwise. Pseudocode for visiting each node in a primary tree and assigning a score is demonstrated here:

```
# parse the primary tree
main_tree = Tree("primary.nwk ")

# parse the taxa_to_trees mappings file and other tree files
taxa_to_trees = {}
mappings_file = open("mappings.tsv ", 'r')
for record in mappings_file:
  if not record.taxon in taxa_to_trees:
          taxa_to_trees[ record.taxon ] = []
  taxa_to_trees[ record.taxon ].append( Tree(record.path) )
mappings_file.close()
```

```
# calculate score for each node
for node in main_tree.internal_nodes():
  if main_tree.isRoot(node) or node.hasNoGrandchildren():
          node.score = 1
  else:
          score = 0
          taxa_set = node.getAllLeavesBelowMe()
          for taxon in taxa_set:
                  count = 0
                  jackknife_taxa_set = taxa_set - set(taxon)
                  for tree in taxa_to_trees[ taxon ]:
                          if tree.containsClade(jackknife_taxa_set):
                                  count++
                  score += count / length(taxa_to_trees)
          score /= length(taxa_set)
          node.score = score

# write output tree with node scores
output_file = open("output.nwk "), 'w')
main_tree.writeTreeWithScores(output_file)
output_file.close()
```

3. RESULTS

3.1 Computation

TANOS generates a single annotated tree as an output. The output tree is written in Newick format (https://evolution.genetics.washington.edu/phylip/newicktree.html), and multiple modifications to the Newick tree are possible via command-line options. By default, the score is placed in a comment for each node. Instead of placing the score in a comment, the score can be output in place of the branch length or label for a given node. For convenience, other output formats are supported with command-line options: compact or pretty-printed JSON (https://www.json.org) and Mermaid format (http://mermaid-js.github.io/mermaid). Functions for outputting a tree in ASCII art are built into the Tree class, making it relatively simple for someone to extend TANOS to output this format as well. Modifying the code that outputs JSON format to output customized JSON or XML (https://www.w3.org/TR/xml) would be relatively

straightforward, e.g., if a favorite tree imaging software accepted phyloXML (Han and Zmasek 2009) TANOS could be modified to output in this format.

3.2 Case study in higher level classification of Insects

The sample dataset, Misof et al. (2014), contained 144 hexapod taxa and analyzed 1,478 protein-coding genes; to date this remains the most comprehensive phylogeny and widely used insect classification. Published topologies were overall highly-supported at ordinal and higher taxonomic levels (Fig. 1, Misof et al. 2014). Misof et al. (2014) report 92% of nodes with a Bootstrap value of >98. In phylogenetics, standard Bootstrap values are generated from a random resampling of the data. TANOS values are generated from a systematic, non-random resampling of taxa. Thus, a direct comparison between Bootstrap and TANOS is difficult. Nonetheless, it is possible to compare well supported and less supported nodes between the approaches over the Misof et al. (2014) topology, therefore learning additional information about the topology that is not possible with a character bootstrap. Calculated TANOS values show less stability overall, with 74.6% of ordinal or higher taxonomic nodes >0.98, and 82.6% of nodes >0.75 (Figure 1). This disagreement was the clearest in two areas of the topology, the Polyneoptera and the sister groups to Holometabola.

The overall weakest TANOS values were found along the backbone for Polyneoptera. Seven of the eight nodes depicting relationships between Polyneoptera orders were recovered with TANOS scores of <50%. These nodes are specifically sensitive to the removal of a single taxon from the alignment. Further, the polyneopteran clade was shown to be quite variable given even minor changes to the included taxon sampling.

Some deep nodes were also shown to be less robust than the bootstrap support values would suggest. Specifically, nodes "104" and "105" (Misof et al. 2014, Fig 1) were again both recovered with >98% BS but were recovered with TANOS values ~0.46 demonstrating that the taxon sampling is lacking in these areas leading to instability at evolutionarily important deep nodes. These nodes are of specific importance because they depict the sister group to Holometabola (arguably one of the most successful lineages of life on Earth).

4. DISCUSSION

4.1 Case Study

Recently, molecular phylogenetics has grown from single molecular marker datasets to multiple targeted gene regions from Sanger technology to full transcriptome, genome, and/or targeted enrichment probe sets for 100s of genes that result in alignments of millions of base pairs (e.g., Cloutier et al. 2019, Misof et al. 2014, Prum et al. 2015). As this transition to genomic datasets has occurred, there has been much less of a focus on taxon sampling breadth, likely due to the obvious increase in resources required to sequence and analyze genomic scale datasets.

Nodal support as a means of assessing phylogenetic relationships has long been controversial and bootstrap values have been specifically criticized since their first usage in phylogenetics (Sanderson 1995, Soltis & Soltis 2003). Genomic level phylogenies have further exposed problems with nodal support, such as consistent maximal bootstraps (e.g., Brower 2019). We propose TANOS as a tool for the genomics era that can assess nodal stability in relation to taxon sampling rather than "support" at the node based on data agreement. This important distinction allows the researcher to access how stable the nodes are across a topology

to the taxa sampled. Specifically, this tool identifies weak areas that might be very sensitive to even minor changes in taxon sampling.

Using TANOS, several well resolved and highly supported nodes from Misof et al. (2014) were shown to be less robust than traditional support metrics might have suggested. This is not demonstrating a methodological flaw in phylogenetic reconstruction, but clearly identifies the weakest nodes and weakest portions of the overall topology with respect to the taxon sampling. One of the goals of this tool is to direct future research by highlighting which clades may benefit from increased taxon sampling, directly impacting the accuracy and predictive power of a given phylogeny. Thus, allowing for more robust investigation and discussion of the many avenues a well-supported phylogeny allows.

4.2 General implications

The steady decline in usage of the jackknifing methods (whether character or taxon based) in phylogenetics (Felsenstein 1985, Lanyon 1985) over the last two decades is not necessarily due to theoretical flaws in the statistic (e.g., compared to the Bootstrap), but instead, driven by lower values as well as the lack of tools to implement with large datasets and more modern reconstruction methods. Many of the major issues given by Wrobel (2008) in an attempt to explain the disparity between usage of jackknifing and bootstrapping are in fact larger issues present across most phylogenetic analyses. He argued that taxa are not independent due to clusters formed during phylogenetic reconstruction, violating a basic statistical assumption. Molecular data are also not independent and therefore should not be subjected to these resampling techniques. Obviously, that has not prevented thousands of research papers doing so over the last few decades. Wrobel (2008) was also in agreement with both Farris et al. (1996)

and Oxelman et al. (1999) that these tools are exploratory and more directed approaches can be used to identify weaknesses in the data or taxon sampling. Wrobel also pointed out that jackknifing methods often give overall lower support values and is likely the reason the jackknife statistic was less popular among researchers. In a time when it has been demonstrated that larger and larger datasets inflate bootstrap values (Brower 2019), methods generating overall lower support or stability values might provide resolution in those cases. Zuo et al. (2010) argued that researchers were often restricted to bootstrapping instead of jackknifing due to limitations in sampling space. With the consistent growth of phylogenetic datasets (both molecular and morphological), this criticism may no longer apply.

The taxon jackknifing methods discussed by Mariadass et al. (2012) and Denton et al. (2017) demonstrate that there is a place for these techniques in the phylogenomics era; however, we argue there is still a missing piece. In combining the traditional jackknife methods of nodal stability along with the adaptations of Mariadess et al. to apply the idea to maximum likelihood allows for a resurgence of these original methods to be used alongside other measures of support.

Nodal support and stability metrics are important when using phylogenies in every way. It is obviously preferred when using a tree to make classification or systematics changes, asking evolutionary questions, mapping characters, reconstructing ancestral distributions, or any of the other diverse tasks researchers are currently using phylogenies for, that those nodes are "well-supported". That being said, we should be cautious of artificially inflated support values. Using multiple methods, both support and stability, is now more computationally possible than ever before. With character dataset size on the order of genomes and transcriptomes our assessment of stability and support needs to shift from robustness in changes to character sampling and instead

focus on taxon sampling. Utilizing taxon jackknifing is an informative method of assessing the effect of the included taxon sampling on a given phylogenetic hypothesis.

AUTHOR CONTRIBUTIONS

SMB: Funding Acquisition; Supervision; Validation; Writing - Original Draft Preparation; Writing - Review & Editing. **GJM:** Conceptualization; Validation; Writing - Review & Editing. **BDP:** Data Curation; Formal Analysis; Methodology; Software; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **GSP:** Conceptualization; Formal analysis; Investigation; Methodology; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **PGR:** Funding Acquisition; Resources; Writing - Review & Editing. **MFW:** Validation; Writing - Review & Editing.

ACKNOWLEDGEMENTS

The authors appreciate the Brigham Young University Office of Research Computing (https://rc.byu.edu) for their continued support of our research. The authors also thank those colleagues that facilitated useful discussion on these topics and ultimately helped improve the final work. Lastly, the authors appreciate all the suggestions made by the reviewers and editors that improved this manuscript.

FUNDING

This work has been supported by funds provided by Brigham Young University and the Department of Biology.

CONFLICT OF INTEREST

None declared.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

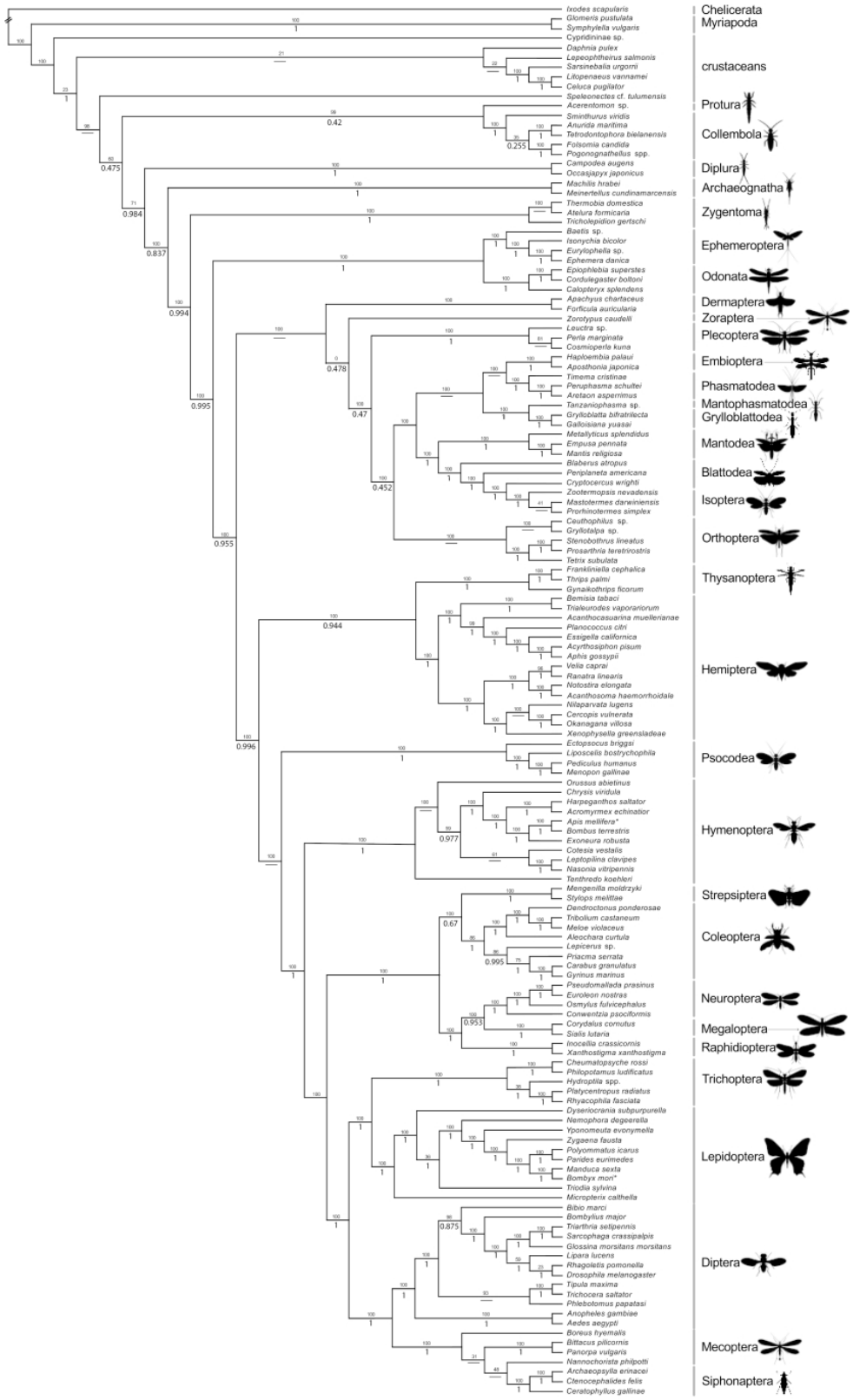


Figure 1. ML topology adapted from Misof et al. (2014) with originally reported bootstrap values (above nodes). In addition, computed TANOS values are provided (0-1, below nodes).

REFERENCES

Brower, A. V. (2019). "Maximum support"= 100% BS. *Cladistics*, **35**(3), 349-350.

Cloutier, A., Sackton, T. B., Grayson, P., Clamp, M., Baker, A. J., & Edwards, S. V. (2019). Whole-genome analyses resolve the phylogeny of flightless birds (Palaeognathae) in the presence of an empirical anomaly zone. *Systematic biology*, **68**(6), 937-955.

Doyle, J., & M. Donoghue. (1987). The importance of fossils in elucidating seed plant phylogeny and macroevolution. *Review of Palaeobotany and Palynology*. **50**, 63-95.

Efron, B. (1979). Computers and the theory of statistics: thinking the unthinkable. *SIAM review*, **21**(4), 460–480.

Farris, J. S., Albert, V. A., Källersjö, M., Lipscomb, D., & Kluge, A. G. (1996). Parsimony jackknifing outperforms neighbor-joining. *Cladistics*, **12**(2), 99-124.

Felsenstein, J. (1985). Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, **39**(4), 783-791.

Gauthier, J., A. G. Kluge, & T. Rowe. (1988). Amniote phylogeny and the importance of fossils. *Cladistics*, **4**,105–209.

Kalyaanamoorthy S, Minh BQ, Wong TKF, von Haeseler A, Jermiin LS. (2017). ModelFinder: fast model selection for accurate phylogenetic estimates. *Nature Methods*, **14**, 587-589.

Lanyon, S. M. (1985). Detecting internal inconsistencies in distance data. *Systematic Zoology*, **34**(4), 397-403.

Lanyon, S. M. (1987). Jackknifing and bootstrapping: important" new" statistical techniques for ornithologists. *The Auk*, **104**(1), 144-146.

Maria Dassou, M., Bar-Hen, A., & Kishino, H. (2012). Taxon influence index: assessing taxon-induced incongruities in phylogenetic inference. *Systematic Biology*, **61**(2), 337-345.

Miller, R. G. (1974). The jackknife-a review. *Biometrika*, **61**(1), 1-15.

Misof, B., Liu, S., Meusemann, K., Peters, R. S., Donath, A., Mayer, C., ... & Niehuis, O. (2014). Phylogenomics resolves the timing and pattern of insect evolution. *Science*, **346**(6210), 763-767.

Nguyen LT, Schmidt HA, von Haeseler A, Minh BQ. (2015). IQ-TREE: a fast and effective stochastic algorithm for estimating maximum likelihood phylogenies. *Molecular Biology and Evolution*, **32**, 268-274.

Oxelman B, Backlund M, Bremer B. (1999). Relationships of the Buddlejaceae s.l. investigated using parsimony jackknife and branch support analysis of chloroplast ndhF and rbcL sequence data. *Systematic Botany*, **24**, 164-182.

Poe, S. (1998). Sensitivity of phylogeny estimation to taxonomic sampling. *Systematic Biology*, **47**(1), 18-31.

Prum, R. O., Berv, J. S., Dornburg, A., Field, D. J., Townsend, J. P., Lemmon, E. M., & Lemmon, A. R. (2015). A comprehensive phylogeny of birds (Aves) using targeted next-generation DNA sequencing. *Nature*, **526**(7574), 569-573.

Sanderson, M. J. (1995). Objections to bootstrapping phylogenies: a critique. *Systematic Biology*, **44**(3), 299-320.

Siddall, M. E. (1995). Another monophyly index: revisiting the jackknife. *Cladistics*, **11**(1), 33-56.

Simmons, M. P., & Freudenstein, J. V. (2011). Spurious 99% bootstrap and jackknife support for unsupported clades. *Molecular Phylogenetics and Evolution*, **61**(1), 177-191.

Soltis, P. S., & Soltis, D. E. (2003). Applying the bootstrap in phylogeny reconstruction. *Statistical Science*, **18**(2), 256-267.

Tukey, J. 1958. Bias and confidence in not quite large samples. (Abstr.) *The Annals of Mathematical Statistic*, **29**(2), 614-623.

Wróbel, B. (2008). Statistical measures of uncertainty for branches in phylogenetic trees inferred from molecular sequences by using model-based methods. *Journal of Applied Genetics*, **49**(1), 49-67.

Zuo, G., Xu, Z., Yu, H., & Hao, B. (2010). Jackknife and bootstrap tests of the composition vector trees. *Genomics, Proteomics & Bioinformatics*, **8**(4), 262-267.

CHAPTER 9

Current state of and suggestions for vertebrate genome sequencing: some assembly required

Brandon D. Pickett[1], John S. K. Kauwe[1], Perry G. Ridge[1]

[1]*Department of Biology, Brigham Young University, Provo, Utah, USA*

ABSTRACT

Advancements in DNA sequencing technologies and genome informatics over the last several decades have swiftly progressed the study of genomes across the spectrum of life. The field is moving at a rapid pace, with changes to the technology causing the landscape of the field to alter significantly every few years. Keeping up with sequencing technology and its vast array of applications is a monumental challenge, especially for a single individual. In-depth reviews of specific topics, such as DNA sequencing platforms, graph-based assembly algorithms, and applications to various disciplines, are prevalent; yet, these reviews are often beyond the scope and interest of the average scientist wishing to utilize genomic data in their work. Nevertheless, many genomic analyses require genome assembly, which is a complicated and evolving process. This review and commentary aim to provide the necessary background on sequencing technologies, genome assembly methods, supplementary data types, and project planning considerations. Suggestions for new genome assembly projects are provided alongside bioinformatics best-practices and other recommendations. Additional reviews and resources are provided for interested readers. Our intention is to provide a simplified, yet thorough, primer for genome assembly to decrease the considerable barrier to entry for individuals and lab groups.

INTRODUCTION

Thirty years have passed since the Human Genome Project (HGP) began and twenty years since the first draft of the human genome was published (International Human Genome Sequencing Consortium 2001; Venter et al. 2001). The resulting progress in all related fields of research has unquestionably been remarkable, even if the research and medical communities' abilities to harness the promised power of the genome got off to a slower start than some anticipated (Nature Editors 2010). Detailed accounts of the HGP, including descriptions and examples of its impact, are well-described elsewhere (Lander 2011; Mardis 2011); one significant indicator of the impact that the availability of a reference sequence had is that it spawned entirely new fields. Researchers in these fields had to grapple with new challenges inherent to using data on a larger scale (Stein 2010), and, with time, genetic research methodologies expanded beyond single- or multi-gene studies to genome-wide analyses.

In the wake of the HGP, several model or evolutionarily-interesting organisms genomes were published (Mouse Genome Sequencing Consortium 2002; Rat Genome Sequencing Project Consortium 2004; Lindblad-Toh et al. 2005; The Chimpanzee Sequencing and Analysis Consortium 2005; Mikkelsen et al. 2007; Green et al. 2010), and they were a boon to both their own fields and our understanding of the human genome. Human microbiome function and diversity were analyzed (Gill et al. 2006; Grice et al. 2009), and common variants were identified for common diseases using genome-wide association studies (The Wellcome Trust Case Control Consortium 2007; Peter et al. 2012). Such analyses were made possible by the accessibility of the high-quality, continuously-updated human reference genome and the advent of massively-parallel sequencing (MPS) technologies; the combination of which has rapidly reduced the cost to sequence new human genomes (Fig. 1).

As more individual genomes from human and other model organisms were sequenced, the power of the reference sequences in addressing previously-unanswerable questions inspired those who study non-model organisms, or who had other niche interests, to sequence genomic DNA from many diverse organisms. Indeed, a large and continuing increase in the number of genomes submitted to NCBI began around 2009 (Fig. 2). Nevertheless, budgets, sequencing technologies, library preparations, bioinformatics methods, and quality control procedures have often limited the quality of the genomes. For example, human contamination and incorrect assembly of genes are significant problems (Denton et al. 2014; Breitwieser et al. 2019). Subsequently, while improvements in assembly algorithms, available computational power, average read length, etc. have generally improved assembly statistics over time, high-quality assembly remains a difficult task with a high barrier to entry.

As the affordability of sequencing genomes at scale continues to improve, more individuals and groups will seek to sequence the genomes of new organisms and redo the draft genomes of those previously attempted. The future utility of these genomes will depend to a great degree on their quality and accessibility in public databases, such as those in the International Nucleotide Sequence Database Collaboration (INSDC). To help protect the quality and utility of future genomes submitted to INSDC databases and reduce the barrier to entry to genome assembly, we present this report as a resource for individuals and labs seeking to begin a genome assembly project. Sequencing technologies and assembly methods will be briefly reviewed, and additional resources will be provided based on specific use-cases or interests. This report will focus principally on vertebrate genome assembly, though many principles remain the same for other groups – with special considerations being required for plant genomes that are complicated by introgression, high ploidy, et**c.** Practical lessons learned from dozens of genome

assemblies will be addressed in the discussion with the intent of answering common questions and avoiding unnecessary frustration.

REVIEW OF LITERATURE

The principal objective of genome assembly is to correctly and completely reconstruct the genetic sequence of a sample. In practice, the genetic sequences refer to the nuclear genome, possibly with the mitochondrial genome, from a multi-cellular sample; although, an assembly project could conceivably be focused on a targeted region of the genome and/or samples of mixed origin. Usually, a single organism is sequenced to represent a population or species, and the sample is often extracted from a single tissue (e.g., blood). This could cause issues for representing a population or species if the individual organism has significant genetic anomalies relative to other individuals, but the reduced complexity in the assembly process resulting from working with DNA from a single organism makes this worthwhile for now. Similarly, mosaicism could cause issues for accurate representation and possibly for the actual assembly process itself. These risks are generally expected in the assembly community, and a project requiring a truly representative sequence would be considered highly specialized. Indeed, a truly representative genome for humans has not yet been realized. Population-specific variants have been identified as part of several important projects (e.g., The International HapMap Project (The International HapMap Consortium 2010) and The 1000 Genomes Project (The 1000 Genomes Project Consortium 2015)), but work on a truly representative pangenome has begun only recently (Chaisson et al. 2020; Li et al. 2020).

In an ideal world, one could isolate individual chromosomes from individual cells and sequence each end-to-end, quickly, with zero error. This theoretical ideal is unrealized because (a) it is difficult to get enough DNA from a single cell, (b) it is difficult to isolate whole

chromosomes in an automated fashion, (c) it is difficult to keep high molecular weight (HMW) DNA (i.e., whole chromosomes) intact, (d) current sequencing technology is unable to read stretches of DNA at chromosome length, and (e) current sequencing technology is unable to read DNA with perfect accuracy. Even if one could handle each of these limitations, assigning sets of chromosomes to the correct parent and/or ancestral genome complicates the process when dealing with nonhaploid assemblies. Due to these limitations, most genome assemblies have been pseudodiploid representations where identical regions were collapsed into a single sequence, and the variable regions either remained dually represented or were partially dropped. Recent advances in sequencing technology and associated computational methods have begun to enable partial pseudohaplotype separation during or after assembly (Guan et al. 2020; Cheng et al. 2021) or genuine diploid assembly (Garg et al. 2020). Nevertheless, the aforementioned limitations (a-e) do pose significant hurdles, such that even the HGP – despite its significant resources – has not truly been completed because several gaps still remain in each chromosome.

To this end, the Telomere-to-Telomere (T2T) Consortium is seeking to sequence every chromosome from end-to-end for a human complete hydatidiform mole (CHM; Logsdon et al. 2020b; Miga et al. 2020). The complete genome is expected to be published in late 2021, but this monumental effort has required the time, minds, and resources of hundreds of people from a dozen institutions. Without such an investment, the average lab or individual can expect any new assembly to fall far short of a T2T assembly until technology improves, software is created, and expert manual curation is automated. In the meantime, reasonably high-quality genomes can be produced by non-experts, and these imperfect genome assemblies are still incredibly useful. To better understand the utility and limitations of current assembly approaches, consider how genome sequencing and assembly has progressed over time.

A (Very) Brief History

Initial approaches to assembly proceeded in a step-wise fashion (i.e., primer walking) in which a primer was designed based on a known sequence and the next portion of DNA could not be determined until a primer could be designed based on the previous portion (Sanger 1975). This was computationally trivial and could be done by hand, but it was extremely time intensive. Genome assembly at this time was effectively done manually by adding newly synthesized sequence to the end of previously determined sequence; it was no more complicated than "copy-and-paste". Shotgun sequencing (Staden 1979) – breaking DNA into many smaller pieces to be sequenced individually – paved the way for higher throughputs, but required significant computational efforts because assembly could no longer be done by hand (Simpson and Pop 2015). In this light, the term "assembly" can be somewhat confusing. In a general sense, "assembling" a genome refers to the overarching process of reconstructing genomic sequences correctly and completely. With the advent of shotgun sequencing, "assembly" sometimes holds a narrower definition (i.e., "computational assembly") in which smaller sequences are merged at areas of overlap to form longer, continuous sequences called contigs. Other steps in the overall assembly process are given distinct names (e.g., gap-filling and scaffolding). For the duration of this report, the verb "assembly" will refer exclusively to "computational assembly", and "genome assembly" will refer to the overarching process.

Recent advances in shotgun sequencing chemistry and technology, together with advances in algorithms and computational power, have radically reduced the cost and effort required to generate a genome assembly and subsequently ushered in the next generation of genome sequencing (i.e., next- or second-generation sequencing) and approach to genome

assembly. The so-called second- or next-generation sequencing (SGS/NGS) approach uses short reads (e.g., Illumina), usually paired-end (PE), as a source of low-error sequence data which is then used in combination with longer range sequence or other data (e.g., generated by mate pair (MP) libraries, long reads, physical maps, linkage maps, etc.) to fill the gaps between contigs (computationally-assembled reads), correct misassemblies, and order and orient contigs into scaffolds. The details of NGS are described in detail in the subsequent section.

Short Read Sequencing and Assembly

NGS technologies use a variety of approaches to sequence reads (segments of DNA) in a massively parallel, high-throughput manner. Read lengths vary by platform, but they are <50-600 nt (usually 100-250 nt), compared with Sanger-based sequences in the range 400-900 nt (Pettersson et al. 2009; Liu et al. 2012; Loman et al. 2012; Quail et al. 2012; El-Metwally et al. 2013; Fierst 2015). Most errors from NGS platforms are single-base substitutions, with a low error rate at ~1%. This is higher than Sanger based-sequencing at approximately 0.1%, though Illumina (San Diego, California, USA) error rates do closely resemble Sanger-based sequencing with an error rate of <0.1% (Pettersson et al. 2009; Liu et al. 2012; Quail et al. 2012; Fox et al. 2014; Fierst 2015). Additionally, short read sequencing platforms are subject to various biases, e.g., change in error rate based on position in the read (Dohm et al. 2008; Fierst 2015) and failure to sequence regions with high guanosine/cytosine (GC) content. While the read length and error properties are not as desirable as traditional Sanger-based sequencing, the improvements in throughput have reduced the cost of sequencing per megabase by four orders of magnitude (Sanger and Coulson 1975; Sanger et al. 1977; Liu et al. 2012; Fierst 2015).

Provided that sequencing was performed to sufficient coverage of the genome, the reads can be assembled *de novo* into contiguous sequences (contigs) using computer algorithms. Such algorithms are complicated; indeed, the assembly problem requires exploring an exponential number of possibilities to guarantee the optimal solution (Räihä and Ukkonen 1981; Nagarajan and Pop 2009; Kingsford et al. 2010). Furthermore, data storage, computational resources, and bioinformatics expertise are non-trivial considerations for any prospective sequencing and assembly project. The most common class of algorithms for short read assembly is based on de Bruijn graphs. In short, the sequenced reads are broken into overlapping k-mers (i.e., k-length subsections of the read overlapping by one base pair (bp)), which form vertices in the graph. Edges connect those k-mers that overlap, enabling the algorithms to "walk" through the graph to output contigs. Differences between assembly software packages lie in graph traversal, statistics, bubble resolution, error detection, etc. The fundamental , at least in theory, is that the original sequence is reconstructed because the reads (and the k-mers they are decomposed into) overlap.

Assembly fails to correctly reconstruct the real sequence when it places reads in the incorrect order (i.e., misassembly) or when it cannot determine the sequence at all (i.e., gaps). Gaps can be caused by insufficient coverage and inherent systematic sequencing biases or by repeats that are longer than the read length (Mulyukov and Pevzner 2002; Nagarajan and Pop 2009). Misassemblies and gaps occur frequently in most genome sequencing and assembly projects, resulting in fragmented assemblies with short contig N50 (the length of the contig where 50% of the contigs are longer (International Human Genome Sequencing Consortium 2001)). While a fragmented assembly of relatively low quality (e.g., N50 of <1 mb) is sufficient for some applications, a more contiguous, reliable assembly would always benefit these applications – indeed, many applications require it. Thus, short read sequencing alone is

insufficient for genome assembly, despite its high throughput and low cost (Alkan et al. 2011; El-Metwally et al. 2013; Mak et al. 2016).

Several methods exist to overcome the limitations of short read sequencing for genome assembly. Sanger-based sequencing can be used in a targeted fashion to fill gaps (Schatz et al. 2010). Some companies, such as Oxford Nanopore Technologies (ONT; Oxford, England, UK) and Pacific Biosciences (PacBio; Mountain View, California, USA) have developed single-molecule approaches that generate long reads, up to 10-30 kb (Karlsson et al. 2015; Rhoads and Au 2015; Jiao et al. 2017). Longer reads provided more unique sequence and were expected to be a significant boon to assembly by spanning repeats; however, some repeats are still be too long to bridge by reads of this length (Jiao et al. 2017). Originally, such reads were used primarily as a source of long-range information for scaffolding and gap-filling, but assembly algorithms have since been developed to try to harness the relatively high read-length in the computational assembly step itself. The principal difficulty with these longer read technologies was the relatively high error rate of 10-15% (Rhoads and Au 2015; Jiao et al. 2017), coupled with a much different error profile than Illumina reads. Several genome projects took a combined approach and used short reads to mitigate errors in the long reads, but as of the mid-2010s, the cost of long-read sequencing was still prohibitive for most projects (Quail et al. 2012; Rhoads and Au 2015). These long-read sequencing technologies did eventually usher in a third generation, but reference-guided assembly, scaffolding with short reads, and pseudo-long reads will be addressed first as they are more timeline-appropriate topics.

Reference-guided Assembly

To aid in the overall genome assembly process, one potentially helpful source of data is the genome of another organism with shared evolutionary history, the more closely related, the

better. Reads, contigs, and/or scaffolds can be aligned to the related reference genome, guiding further joining, ordering, and orienting of contigs and scaffolds. This approach is sometimes called reference-guided and has been employed several times (Schneeberger et al. 2011; Hirsch et al. 2014; Yao et al. 2015; Golicz et al. 2016). Various software packages have been developed to complete these tasks (e.g., Bao et al. (2014) and Silva et al. (2013)), often with varying purposes, such as using more than one related genome (Kolmogorov et al. 2014; Bosi et al. 2015), not requiring a tree specifying the relationships (Bosi et al. 2015), or scaffolding contigs created from ancient, degraded DNA (Rajaraman et al. 2013). Of course, this method relies heavily on assumptions about and hypotheses of (possibly incorrect) shared evolutionary history, which could lead to an incorrect assembly. Another similar method, also relying on presumed homology, uses protein sequences for comparison, instead of the nucleotide sequences (Huang et al. 2013; Zhu et al. 2016).

Scaffolding with Mate Pair Libraries

After assembling reads into contigs, the next major step was typically ordering and orienting the contigs into scaffolds (two or more contigs joined together in the correct orientation and separated by a run of ambiguous or unknown nucleotides (Ns)). This technique, often referred to as scaffolding, usually relies on information that is longer in range than the read length used for assembly. It would then use some kind of mapping information to associate contigs together when one end of the longer-range information source mapped to one contig and the other mapped to a second contig. In the era of short read-based genome sequencing and assembly, a common approach to scaffolding was mate pair (MP) libraries. A MP library produces results similar to sequencing with a paired-end (PE) library, but with a few key differences. PE and MP libraries produces reads in different orientations (Glenn 2011).

Additionally, PE libraries typically have short inserts (<500 bp), while MP libraries can have much longer insert sizes (e.g., 20 or 25 kb) (van Heesch et al. 2013). Thus, the benefit of MP libraries is that they provide a source of long-range information to assist in ordering and orienting contigs.

Many sequencing projects have included MP libraries in their sequencing projects. The best results are obtained when using multiple MP libraries with varying insert sizes. Ideally, a project will utilize at least one library with medium length (e.g., 5, 8, or 15 kb) and one with large length (e.g., 20 or 25 kb) inserts; however, using more than one of each was a common approach (Schatz et al. 2010; Gnerre et al. 2011; van Heesch et al. 2013). While MP libraries do provide a source of fairly long-range information (say 25 kb compared to read length of 150 bp), the reads are themselves still short and some may not align uniquely in the genome – making these scaffolding decisions ambiguous.

Incorporating information generated using MP technology into an assembly is computationally intractable (Huson et al. 2002), requiring additional algorithms than what the typical assembler could initially do. Some assemblers have included scaffolding modules directly into their assembly process (e.g., (Simpson et al. 2009; Gnerre et al. 2011; Luo et al. 2012; Simpson et al. 2012; Nurk et al. 2013; Jackman et al. 2017)), but beginning with Bambus (Pop et al. 2004), stand-alone programs were developed, supporting a more modular approach to assembly and scaffolding (e.g., (Assefa et al. 2009; Simpson et al. 2009; Dayarian et al. 2010; Boetzer et al. 2011; Gao et al. 2011; Koren et al. 2011; Salmela et al. 2011; Gritsenko et al. 2012; Donmez and Brudno 2013)). Naturally, some are for specific use cases, such as metagenome scaffolding (Koren et al. 2011). Hunt, et al. provide a helpful review of scaffolding methods and software through 2014 (Hunt et al. 2014). Several additional scaffolders were

subsequently released (Kajitani et al. 2014; Lindsay et al. 2014; Sahlin et al. 2014; Bodily et al. 2015; Farrant et al. 2015; Mandric and Zelikovsky 2015; Rahman and Pachter 2016; Luo et al. 2017).

MP technology improved the contiguity of SGS genome assemblies, and scaffolding software and algorithms using MP data continued to improve. Yet, some regions of the genome were still unresolvable because some repeats remain too long to be determined, even with long insert size libraries (Alkan et al. 2011; van Heesch et al. 2013). Since MP sequences are just PE sequences with a different library preparation, the reads share the same biases as those generated by standard PE sequencing. Furthermore, generating many different libraries with varying insert sizes requires additional DNA and is expensive and time consuming. Ultimately, other sources of long-range data alongside or replacing MP data is required to generate high-quality genomes with near-chromosome size pseudomolecules. Development of scaffolding programs has continued, but most have shifted focus to utilizing other sources of long-range information for the scaffolding of assemblies based on long-reads.

Scaffolding with RNA-seq Libraries

RNA-sequencing (RNA-seq) uses HTS capabilities (typically Illumina PE) to sequence cDNA created with reverse transcriptase from RNA. Note that amplification-free methods are possible on TGS platforms (Garalde et al. 2018), and they can sequence entire transcripts end-to-end. PacBio IsoSeq is a very popular choice for this technique and is a better choice than Illumina-based RNA-seq for most situations, provided the project has sufficient budget. Before long-reads became widely used for DNA or RNA, short reads were the standard. Commonly, mRNA is targeted for RNA-seq; a common application of which is differential gene expression studies. Although a few years old, the review by Wang et al. (2009) is a helpful review of the

purpose and technology of RNA-seq. From a genome assembly standpoint, RNA-seq is indispensable as an annotation tool (Yandell and Ence 2012). Considering that PE reads may appear on different exons, RNA-seq data also provides a source of long-range information – possibly enabling the merging of contigs together.

A few software tools have been written for this purpose, with somewhat varying usage possibilities. The RNAPATH module of ERANGE (Mortazavi et al. 2008) was created to demonstrate using RNA-seq data as long-range data for scaffolding and did so on the genome of a *Caenorhabditis* nematode, nearly doubling supercontig N50 (Mortazavi et al. 2010). Using an algorithm relying on BLAT (Kent 2002) for local alignments, L_RNA_scaffolder demonstrated similar results on human, pearl oyster, and zebrafish genomes (Xue et al. 2013). First, however, L_RNA_scaffolder requires the user to generate a *de novo* transcriptome assembly. TGnet also relies on transcript assemblies, but additionally requires manual inspection with their visualizer (Riba-grognuz et al. 2011). Algorithmically similar to RNAPATH, AGOUTI will update the annotations for the genome it is scaffolding (Zhang et al. 2016). While convenient for updating an old genome assembly and associated annotations, this is a limitation for new genome assembly projects that do not have annotations and/or prefer to annotate after the assembly is complete. Rascaf appears to improve upon these other methods by using a new algorithmic approach: an exon block graph to represent gene and contig relationships (Song et al. 2016). Rascaf does not depend on pre-existing annotations. Furthermore, it avoids expensive *de novo* transcript assembly by tools such as Trinity (Grabherr et al. 2011; Haas et al. 2013) by directly aligning the reads to the assembly.

Scaffolding a genome assembly (including possibly updating any pre-existing annotations) with RNA-seq data was a great idea, especially considering a many genome

assembly projects would have already been doing quality RNA-seq anyway for annotation. Yet, it was insufficient as a sole source of long-range information because it could join only contigs that would be separated by an intron. Naturally, many contigs do not meet this criterion. As other sources of long-range information for scaffolding have become widely available, RNA-seq has fallen out of favor for scaffolding – short-read RNA-seq is still a reasonable choice for annotation purposes. Scaffolding with RNA-seq is best used for updating short-read-based draft assemblies; when used with long-read-based assemblies, it is prone to introducing incorrect scaffolding joins. Spurious joins resulting from non-unique mapping of reads due to the similarity of genes are not worth the benefit of the correct joins for long-read-based genome assembly projects.

Synthetic Long Reads

One method for generating increasing read length relies on short read sequencing to create so-called read clouds or synthetic long reads (SLRs). The underlying sequencing technology is classic SGS. The real difference comes in the library preparation, in which the sample is separated into discrete reactions that occur simultaneously. After each pool is barcoded, the entire sample is sequenced. The barcoding enables recognition of which reads belong to the same subsection of the genome, enabling assembly of each subsection (subassembly) before using these "long reads" for the main assembly. The most notable commercially available read cloud option was offered by 10X Genomics (10XG). Their library preparation employed GemCode™ technology and could be completed in an extremely high throughput manner for minimal cost (Goodwin et al. 2016; Crepeau et al. 2017). 10XG referred to their reads as "linked-reads", differentiating them from the SLRs generated by Illumina's TruSeq-SLR™ (TSLR) (Voskoboynik et al. 2013; McCoy et al. 2014; 10X Genomics 2016). A

similar, lower-throughput technology that pools the genome into only 9,126 (96^2) pools is contiguity-preserving transposition sequencing (CPT-seq) (Amini et al. 2014). CPT-seq reads (Adey et al. 2014), TSLR (Kuleshov et al. 2015; Pinoli 2015; Sharon et al. 2015; Kuleshov et al. 2016; Tsai et al. 2016), and 10XG linked-reads (Mostovoy et al. 2016; Crepeau et al. 2017; Jackman et al. 2017; Weisenfeld et al. 2017; Yeo et al. 2017; Hulse-Kemp et al. 2018) have all been used to assemble, polish, and/or scaffold genome assemblies. In 2017, Illumina discontinued support for TSLR (Van Oene 2017), and 10XG did the same for their linked-reads in 2020 (10X Genomics 2020). Neither CPT-seq nor its updated single-tube protocol CPTv2-seq (Zhang et al. 2017) have been widely adopted, likely due to issues with throughput and out-of-the-box compatibility with Illumina sequencing primers (Meier et al. 2020).

Nevertheless, four new read cloud technologies have emerged: Complete Genomics' (CG) single tube long fragment reads (stLFRs) (Wang et al. 2019), Droplet Barcode Sequencing (DBS) (Redin et al. 2017), Haplotagging (Meier et al. 2020), and Universal Sequencing Technology's (UST) Transposase Enzyme Linked Long-read Sequencing (TELL-Seq™).DBS and Haplotagging are both open protocols with relatively low costs. Haplotagging in particular is inexpensive at <$3 per sample for haplotyping. As commercial products, both stLFR and TELL-Seq are very new. When considering a project, especially when haplotyping many samples is required, both are worth considering if one wishes to avoid doing the lab work in-house. TELL-seq specifically can theoretically do anything 10XG linked-reads could do. For certain applications for genome assembly, read cloud technologies are a reasonable choice, but most projects – especially if being tackled by a novice – should stick to true long reads because of the problems associated with read cloud assemblies.

One drawback with read cloud approaches is that they produce shorter "long reads" than true long reads – precluding them from resolving even more tandem repeats than the long read platforms can (Kuleshov et al. 2016). Of course, the benefit is the extremely low cost when compared with true long read technologies. Supplemented with additional long-range information, such as optical mapping or chromosome interaction maps, read cloud data was hypothesized to be sufficient for high-quality assembly (examples and discussion of this in later sections). Since true long reads also need longer-range data (e.g., optical mapping) to resolve some genomic features, read cloud technologies were an attractive option when compared with PacBio or ONT sequencing for many assembly projects. While some very impressive assemblies were created, at least in part, with read cloud approaches (primarily 10XG linked-reads), the typical project will see low- to mid-quality assemblies as a result.

One contributing factor to this is that few software packages have been developed for assembling and scaffolding genomes using read clouds. fragscaff (Adey et al. 2014) was initially developed for CPT-seq, but has also been used with read clouds from another platform (10XG) to re-scaffold the sugar pine genome (Crepeau et al. 2017). Architect (Kuleshov et al. 2016) was built to scaffold metagenomes and pooled sequences. It employs an interesting algorithmic approach to reduce the expense of subassembly: using a de Bruijn graph approach (Pevzner et al. 2001) for each pool / container and an Overlap-Layout-Consensus (OLC) approach (Myers et al. 2000) to join the subassemblies. ARCS/LINKS (Yeo et al. 2017) boasts improved performance over both fragscaff and Architect with the ability to scale to large data sets (the other two cannot realistically be used for more than 250,000 sequences). It was the first software expressly developed for 10XG read cloud assembly, excluding 10XG's in-house, push-button assembler, Supernova (Weisenfeld et al. 2017). While not an assembler itself, LRez is the most recent read

cloud software, released as a C++ API and toolkit for the now discontinued 10XG linked-reads, but it promises effectiveness with at least Haplotagging and UST's TELL-seq (Morisse et al. 2021a). As long as read cloud technologies exist, genome assembly with such reads is likely to continue, as is bioinformatics development for needed tools. Despite the likelihood of working with read cloud technologies becoming increasingly easier, the reads are still far shorter than true long reads, which are unarguably a better choice for high-quality genome assembly.

Long Read Sequencing and Assembly

The "golden goose" of genome sequencing would be to sequence molecules end-to-end with low error. Certain single-molecule technologies get significantly closer to such read length, though none have successfully come close to tens or hundreds of megabases (the length of chromosomes). Examples of such "third-generation" sequencing (TGS) platforms are PacBio SMRT™ (Single-Molecule, Real-Time) and ONT MinION™. While ONT sequencing works by measuring electric signals that change as a single DNA molecule is passing through a nanochannel, PacBio SMRT sequencing works by putting a single DNA molecule in a tiny well called a zero-mode waveguide (ZMW) and observing fluorescence as tagged nucleotides are incorporated by a polymerase. When TGS platforms were gaining popularity (~2015-2017), PacBio was generating reads in the 20-30 kb range (Karlsson et al. 2015), and ONT could reliably generate reads >10 kb (Urban et al. 2015), though some reported reads >100 kb (Goodwin et al. 2015; Madoui et al. 2015; Urban et al. 2015). The error rate for PacBio was more desirable than ONT (could be >30% (Goodwin et al. 2015; Madoui et al. 2015)), but both commonly had error rates of 10-15% (Rhoads and Au 2015; Jiao et al. 2017). Despite the error rates, long reads had been shown to be sufficient for genome assembly without additional data

types, especially in bacteria (Chin et al. 2013; Brown et al. 2014; Parker et al. 2014; Terabayashi et al. 2014; Berlin et al. 2015; Koren and Phillippy 2015; Badouin et al. 2017; Jansen et al. 2017).

One important aspect to understand of PacBio and Nanopore sequencing is the error profile. Where Illumina sequencing has an error type of systematically-biased single nucleotide substitutions at a rate of <0.1% (Fox et al. 2014), these long-read technologies' 10-15% errors were comprised of random insertions and deletions (indels). The two companies' products have since diverged enough that a separate discussion of each is warranted, but only after a discussion of the effect that "noisy" (i.e., relatively erroneous), long-reads have computational assembly. If noisy reads were dropped into a traditional assembler, especially a de Bruijn graph assembler, the errors would wreak havoc by creating excessive tangles in the graph. In practice, this would yield to an assembly with many contigs and low N50. To avoid this, the noisy reads need to be corrected. Correction can happen in one or both of two ways: (a) self-correction by calling consensus on all-vs-all alignments of the reads or (b) hybrid-correction using highly-accurate, short-read data. Hybrid-correction can be further broken down into alignment-based methods and assembly-based methods. Alignment-based methods work by calling consensus on alignments of all short reads to all long reads. Assembly-based methods work by assembling the short reads into a de Bruijn graph and correcting the long reads either by alignment to the contigs or by direct graph traversal. These processes are extremely expensive from a computational standpoint, often taking more CPU (Central Processing Unit) hours than the assembly of the corrected reads. (Zhang et al. 2020)

Oxford Nanopore Technologies Reads

ONT's long-reads commonly remain between 10-100 kb but have a much-improved accuracy of 87-98% for most reads (a small percentage of reads have relatively low accuracy at around 69%) (Logsdon et al. 2020a). Additionally, 91% of homopolymers ≥5 bp in length are accurately captured in the raw reads. Both of these error rates are lower than the error rates for PacBio's raw reads, although PacBio does not have a small percentage of reads at very low accuracy like ONT does. Where ONT data really shine for genome assembly are with a special library preparation now termed "ultra-long". By definition, these reads are mostly >100 kb and share an error profile very similar to the regular long reads. A subset of ultra-long reads, called "whales", exceed 1 mb, with the current record exceeding 2 mb (Jain et al. 2018; Logsdon et al. 2020a; Miga et al. 2020). Such reads are four orders of magnitude longer than modern short reads. Understandably, this length is an immense help for assembly and gap filling; in fact, they have been instrumental in the T2T Consortium's efforts on the CHM genome (Logsdon et al. 2020b; Miga et al. 2020).

ONT's chemistry and hardware are under active development. Their first machine was the MinION, which has a single flow cell and can be run attached to a laptop from anywhere on earth. While this has some incredible applications, it suffers from low throughput. ONT has since released the GridION and PromethION as more high-throughput options, but they are still limited by the speed that the nanopore's molecular motor can process a DNA molecule. In essence, the GridION is made of MinIONs combined into blocks, and the PromethION is, in turn, a collection of GridIONs. Ultra-long read libraries currently take two or more weeks to prepare and run, though future developments to decrease this time requirement are likely (Logsdon et al. 2020a).

Pacific Biosciences Reads

PacBio's primary instrument was the RSII (RS2), but it has subsequently upgraded through the Sequel and Sequel II (2) to the Sequel IIe (2e). Chemistry and throughput have improved dramatically to the point where a sequencing run with one SMRT Cell can generate 8M reads in the time it once took to produce 1M. PacBio has branched its long-read offerings into two main categories: continuous long reads (CLRs) and High-Fidelity (HiFi) reads.

Continuous Long Reads (CLRs)

CLR reads are PacBio's original sequencing technology. The error rates are marginally better than they once were (10-15%) at 8-15%. Unlike Nanopore reads, which can have a small percentage of reads with >30% error, PacBio CLRs all fit in the 8-15% range. Of the homopolymers 5 bp or longer, 85% are correctly recovered in the raw reads (compared to >90% for Nanopore reads). Like ONT long-reads, CLR reads have been used extensively in genome assemblies and have proven extremely useful in creating moderate- to high-quality assemblies. Because the error rates of CLR reads (and ONT reads) are so high, large and/or complex genomes require expensive high sequencing depth and/or a hybrid approach to correction, which is a significant downside to using such long reads in a genome assembly project. As such, some individuals have decided to use these long reads as a tool for only gap filling, localized reassembly, and/or scaffolding (Bashir et al. 2012; English et al. 2012; Koren et al. 2012; Boetzer and Pirovano 2014; Koren and Phillippy 2015; Rhoads and Au 2015; Warren et al. 2015; Zimin et al. 2017); however, the relative benefit of this approach is minimal compared to using the long reads directly in creation of the original assembly graph.

High-Fidelity (HiFi) Reads

Where PacBio really shines for genome assembly is with its HiFi reads. HiFi reads are generated in the exact same manner as with CLR reads, except that the sequence is circularized to enable the polymerase to re-sequence the same molecule multiple times. Similar to how SLRs require a subassembly step for each barcoded set before the main assembly, each read output of a ZMW must be split and evaluated to form a single consensus read. This process of circularization, sequencing, and consensus calling to generate HiFi reads is called Circular Consensus Sequencing (CCS) and is sometimes used interchangeably with the term "HiFi". HiFi read lengths are shorter than CLR reads because much of the sequencing time is being used to re-sequence the same molecule. Due to this and size selection during library prep, HiFi read lengths typically have a very tight distribution, whereas CLR read length distributions usually have a long right tail. Initially, HiFi reads were 10-15 kb, but recent results are showing median read lengths above 15 kb with maximum read lengths approaching 30 kb (Hon et al. 2020).

Despite the reduction in read length compared to traditional CLR reads, the increase in accuracy caused by consensus makes HiFi reads advantageous. The process of repeatedly sequencing the same molecule provides sufficient read depth to correct the random indel errors. Most importantly, the localities of the sequences are guaranteed; in other words, the multiple subreads (the sections extracted from the repeatedly-sequenced read after removal of primers and indexes) are always from the same distinct DNA molecule. As such, there is no chance of collapsing haplotypes, removing segmental duplications, removing a different gene in the same gene family, etc. through the consensus process. Accuracy varies between HiFi reads based on the length of the original DNA molecule and movie time (i.e., how long the sequencing reaction was allowed to proceed), but the majority has an accuracy at or above Q20 (i.e., 99.9% accurate) – the same quality as Illumina short reads and, as such, diminish the need for NGS reads directly

in assembly. Relatively few HiFi-based genomes have been fully published to date, but those that are show extreme promise for this datatype, especially when combined with other datatypes. Overall, HiFi reads are accurate and long enough to completely resolve human centromeres into one or a handful of contigs. Moreover, when HiFi reads are combined with other data types (as discussed later), the entire chromosome can be resolved into a single contig from telomere to telomere through the entire centromere (Logsdon et al. 2020b; Miga et al. 2020).

Long-Read Assembly Software

Assembly software has necessarily evolved rapidly over the last few years as "traditional" assemblers built for short reads were unable to handle the length and high error of long, noisy reads without modification. Understandably, a graph algorithm that expects effectively perfect reads (e.g., a 100 bp read at 99.9% accuracy has 0-1 errors) will not perform well with the messy tangles produced from noisy reads that are ~100 times more erroneous and at ~10-100 times longer. As was the case for short read assemblers like ABySS (Simpson et al. 2009; Jackman et al. 2017), ALLPATHS (Butler et al. 2008; Ribeiro et al. 2012), the Celera Assembler (Myers et al. 2000), SOAPdevovo (Li et al. 2010), and Velvet (Zerbino and Birney 2008), long-read assemblers all compete with each other and have differences in their performance, options, algorithms for bubble popping, etc. The assemblers that were created or modified to work with noisy, long-reads either use only long reads (typically pre-corrected (Fu et al. 2019; Morisse et al. 2020; Zhang et al. 2020)) or incorporate both long and short reads into the assembly graph. MaSuRCA (Zimin et al. 2013) can incorporate both types of reads, and Canu (Koren et al. 2017), PacBio's HGAP/Falcon (Chin et al. 2013; Chin et al. 2016), miniasm (Li 2016), Raven (Vaser and Šikić 2021), wtdbg2 (Ruan and Li 2019; — 2020), and others use

only long reads. Commonly, these programs have slightly different parameters for PacBio CLRs and ONT reads.

Since these long-read assemblers were created to handle noisy long-reads, they required updating – or new assemblers needed to be written – to handle HiFi data. Some assemblers folded them into the existing programs (e.g., Falcon (Wenger et al. 2019) and HiCanu (Nurk et al. 2020)), and others were entirely new or created off of forks of previous assemblers (e.g., hifiasm (Cheng et al. 2021) and Peregrine (Chin and Khalak 2019)). For the average person intending to use these softwares, the precise details of how they differ algorithmically are nonessential. Often, people will try assembly with more than one software and choose the one that looked the best. Many of these programs – especially for HiFi reads – are under active development, have limited validation, and/or have not yet been peer-reviewed. Accordingly, it is difficult to make a strong recommendation for one software over another, even for specific situations. Based on an observation of the community and reading assembly papers and preprints, my subjective recommendation would be to use Canu, miniasm, or wtdbg2 for CLR or ONT reads and HiCanu or hifiasm for HiFi reads.

Diploid Assembly

Some long-read assemblers are also beginning to address the diploid assembly problem. Two approaches are currently being explored: phasing and trio-binning. In some cases, a combination of both is employed, though the latter is still extremely new. Falcon_unzip (Chin et al. 2016) paved the way for phasing by extracting phased assemblies from the Falcon assembly graph. Similarly, hifiasm outputs a primary (i.e., the best set of paths through the graph to get a haploid representation of the genome) and an alternate assembly (i.e., everything leftover after

extracting the primary assembly). These are effectively a secondary program from the main assembler that will process the assembly graph, and similar standalone programs have been written, namely purge_haplotigs (Roach et al. 2018) and purge_dups (Guan et al. 2020). While some alternate assemblies can be fairly high in quality, most, by definition, lack the sequence that is shared between both haplotypes. The "haplotype 1" assembly (when generated) is the same as the primary assembly. The "haplotype 2" assembly (when generated) is a mixture of the primary and alternate assemblies; more specifically, it contains the parts from the primary assembly that are shared between haplotypes and the entire alternate assembly. Figure 1 from Cheng et al. (2021) provides a helpful illustration of the relationship between primary, alternate, and haplotype 1 and 2 assemblies. Additionally, "haplotype" in this case is more aptly termed a "pseudohaplotype" because there is frequent occurrence of haplotype switching. While this haplotype switching is unavoidable without additional information, methods to address this issue are beginning to be employed.

By contrast, trio binning (Koren et al. 2018)makes use of parental information (i.e., for sexually reproducing organisms, trio = mother, father, and child) to sort the reads into bins: those that come from one parent and those that come from the other. Typically, a third bin is also produced when the read could not confidently be placed in a specific bin. Binning is based on sequence similarity and could theoretically be accomplished with traditional read mapping; though, a k-mer-based method is usually employed. Trio binning is not always possible because it requires DNA from both parents of the subject and additional funds for their sequencing; although, the cost of sequencing for the parents can be relatively low because short-read sequencing can be used. Noisy long-reads do not work very well for trio binning if using a k-mer

approach, but HiFi reads would work well. Canu/HiCanu and hifiasm (and possibly others) have options for using trio binning information.

Another application of the trio binning concept is to trio bin data that will be used post-assembly for polishing and/or scaffolding, though this idea has not yet been implemented anywhere to the authors' knowledge. However, additional data types have been incorporated into assembly software to improve assembly and phasing. One example of this technique is dipasm (Garg et al. 2020), which uses chromosome conformation data (commercially available as Hi-C) with HiFi read assemblies. Currently, at least one group is exploring the incorporation of ultra-long ONT reads into a HiFi read-based assembly graph to fill gaps, but this technique and other combinations represent the bleeding edge of the discipline. The use of Hi-C and other sources of long-range information to polish or scaffold assemblies as separate steps will be discussed in subsequent sections.

Polishing Genome Assemblies

"Polishing" typically refers to the correction of errors in an assembly *after* the assembly has been produced. Presumably, the assembly software has already attempted to resolve bubbles and address prospective misassemblies. Polishing software seeks to fix sequence errors (i.e., point mutations and indels), break misassemblies, and/or fill gaps. Generally speaking, polishing falls into two categories based on the type of information used for polishing: short and long reads. Fixing point mutations and indels in a genome assembly became particularly necessary during the pre-HiFi period of long-read genome assembling because the reads were noisy, resulting in errors in the contigs. In principle, polishing occurs by mapping reads (short or long)

to the contigs, followed by processing the mapping information to make decisions and outputting new polished contigs.

The first widely-used short-read polishing software was Pilon (Walker et al. 2014). Pilon performs all types of polishing, but was, unfortunately, developed for microbial genomes and is unable to handle large genomes efficiently as the general rule is to expect 1 GB of RAM (Random Access Memory) for every 1 mb of sequence. While for microbial and bacterial genomes this generally is not an issue, it quickly becomes a problem for vertebrates and especially plants. For example, if a genome is 1 gb in size, Pilon would require approximately 1 TB of RAM. The most popular replacement/substitute for Pilon is RaCon (Vaser et al. 2017), which is extremely efficient and has undergone intense optimization, including options to run on GPUs. RaCon is a general-purpose consensus module, meaning it cannot fill gaps or explicitly detect misassemblies, but it does work with both short (including MP) and long reads. RaCon could be used for noisy read correction before assembly as well. Similarly, CONSENT (Morisse et al. 2021b) can be used for pre-assembly read correction or post-assembly contig polishing. However, CONSENT works only with long reads.

The other long-read polishers widely in-use are technology-specific. The main polisher for ONT data is called Nanopolish (Loman et al. 2015; Quick et al. 2016; Simpson et al. 2017). PacBio created their own polishing algorithms called Quiver (Chin et al. 2013; originally part of HGAP, now deprecated) and Arrow (Laird Smith et al. 2016), now part of GCpp (https://github.com/PacificBiosciences/gcpp) and the basis of the consensus algorithm for CCS. Both of these technology-specific polishers also act as variant callers; variant detection is part of the error-correction process. Another way to polish is to align reads to the assembly, call and filter variants, then change the assembly based on the variant information using an aligner and

tools like SAMtools (Li et al. 2009), BCFtools (Li 2011), and FreeBayes (Garrison and Marth 2012).

Scaffolding Genome Assemblies

Scaffolding is the process by which contigs are ordered and oriented into scaffolds with gaps between the contigs. This process requires information beyond the DNA reads used in the assembly. For the additional data to be informative, it must be longer-range than the original DNA reads. "Longer-range" need not necessarily refer to the read length of the additional data type, if said data type is even comprised of reads. As an example of scaffolding with longer-range information, one early use of both PacBio CLRs and ONT long reads was scaffolding short-read assemblies (Bashir et al. 2012; English et al. 2012; Koren et al. 2012; Boetzer and Pirovano 2014; Koren and Phillippy 2015; Rhoads and Au 2015; Warren et al. 2015; Zimin et al. 2017). Scaffolding draft assemblies with short DNA MP reads and RNA-seq reads were discussed in previous sections. The reason short MP reads or RNA-seq reads (usually 100-250 bp) can scaffold an assembly built from short DNA reads (also usually 100-250 bp) is because the important factor in length is the associating information, not the read length.

For short-read PE and MP libraries, the distance between reads (i.e., the insert size) defines how long-range the associative information will be. PE libraries have an insert size 0-500 bp, making them a poor choice for scaffolding. MP libraries frequently have an insert size between 5 and 25 kb, making them moderately informative, especially compared to the 100-250 bp length of the reads used in assembly. Biologically, if a "left" read comes from position *x* on chromosome 1, the "right" read will be *read_length* + *insert_size* bases downstream at position *x* + *read_length* + *insert_size* on chromosome 1 (note that insert sizes are approximate). *In silico*,

if the region from *x* to *x + 2(read_length) + insert_size* is contained in a single contig, the read-pair has no helpful information. If the two reads align to different contigs with less than *insert_size* combined bases downstream of the left read and upstream of the right read, those two contigs can be joined together and the gap size can be estimated. However, now that long reads are consistently longer than MP insert sizes, MP libraries have fallen out of favor for genome scaffolding purposes.

Unlike short DNA PE and MP reads, scaffold gap lengths cannot easily be determined using the insert size between short RNA-seq reads. While the distance on the mRNA molecule is known, the insert size is not likely to also be the genomic distance between the reads when spanning exon/intron boundaries. Other sources of long-range information (e.g., linkage maps and physical maps) also make estimating distance difficult, though it is possible with some of them.

Linkage Maps

Linkage maps provide the observed recombination frequencies between loci in the genome. Among other applications, they are useful for ordering, orienting, and correcting scaffolds in *de novo* genome assembly. Fierst (2015) provides an excellent review of linkage maps and their utility in genome assembly through 2015 – in short, linkage maps provide long-range information. However, at least two problems limit the feasibility of linkage maps for some genome assembly projects. First, you need an F2 mapping population (technically it can be done in some cases with only an F1), which is not trivial when not impossible, not to mention potentially quite expensive. Second, the ordering of markers is a computationally difficult task as the number of possible combinations grows exponentially as the number of markers increases. While some software has been developed to assist in this task, custom scripting is still a common

requirement for certain tasks and no software integrates completely with *de novo* assembly. Ultimately, Fierst concludes that any project that can manage a linkage map, should create one, but also points out that "undertaking a mapping project is a significant investment of resources". Since then, others have used linkage maps to assist in genome assembly, e.g., *Brassica rapa* (Markelz et al. 2017), *Arabidopsis thaliana* (Zapata et al. 2016), *Nelumbo nucifera* (Gui et al. 2018), and the domestic cat (which required genotyping 453 cats!) (Li et al. 2016).

Physical Maps

Physical maps provide information about the physical distance between locations of a marker or sequence on a given molecule; thus, a complete genome sequence is technically a physical map with single base-pair distance (O'Rourke 2014). In practice, the actual distance between loci varies by choice of marker and methods, typically providing information on megabase scales. Physical maps have often been a key component of large genome projects as they can provide long-range information to order and orient scaffolds and, in some cases, determine the size of gaps (Aston et al. 1999). The longer the reads (i.e., distance from end to end) and higher the resolution (i.e., smaller average distance between loci), the more useful the physical map will be for assisting in scaffolding a genome assembly. While optical maps, a specific type of physical map, provide information that can enable an estimation of distance between the markers, physical maps generated through chromosome conformation capture (3C) provide only information that will help determine the relative distance between pairs of markers.

Optical Maps

Optical maps are high-resolution restriction maps in which the location of the restriction enzyme sequence is determined using optics and fluorescence. Optical maps have played an

important role in validating and scaffolding genome assemblies from early assembly projects (Gardner et al. 1998; Aston et al. 1999; Jing et al. 1999; Lin et al. 1999). They are also useful for structural variant detection and analysis (Teague et al. 2010; Lam et al. 2012; Mak et al. 2016; Jaratlerdsiri et al. 2017). Early development and use of optical mapping were done by Dr. David Schwartz and his lab, paving the way towards more accessible, high-throughput methods (Dimalanta et al. 2004; Wu et al. 2009; Zhou et al. 2009). Until high-throughput methods were produced, only large and well-funded projects (e.g., rice (Zhou et al. 2007)), or projects with small genomes (Latreille et al. 2007), could realistically afford the time and money required to use optical mapping to scaffold and correct misassemblies. OpGen eventually produced the Argus™ system, making these techniques commercially available to more assembly projects (Giongo et al. 2010; Neto et al. 2011; Chen et al. 2012; Dong et al. 2012; Ganapathy et al. 2014). This technique effectively required fixing linearized DNA on a slide. This method improved throughput, but still required much time, and the utility was hampered by high error rates and the inherent difficulty in accurately measuring DNA length (Baday et al. 2012). Furthermore, optical map resolution was still constrained by the diffraction limit and common use of only a single restriction enzyme (Neely et al. 2010; Baday et al. 2012).

Improvements in nanoscopy, nanofluidics, and nickase chemistries (Dimalanta et al. 2004; Xiao et al. 2007; Das et al. 2010; Neely et al. 2011; Michaeli and Ebenstein 2012; Levy-Sakin and Ebenstein 2013) eventually led to BioNano Genomics (BNG; San Diego, California, USA) and its commercially available Irys™ and Saphyr™ systems, which feed each DNA molecule through a nanochannel in a very high-throughput manner. BNG refers to their technique as next-generation mapping (NGM). BNG NGM has higher resolution and higher throughput than the Argus system, commonly <5kb, with some reporting resolution inside SGS

read length (Baday et al. 2012; Howe and Wood 2015). With less error, lower cost, and higher throughput, NGM has proven to be a useful tool for scaffolding genome assemblies; in fact, several assembly projects demonstrated its utility in the first few years of its availability (Hastie et al. 2013; O'Bleness et al. 2014; Bickhart et al. 2016; Staňková et al. 2016; Yang et al. 2016; Daccord et al. 2017; Jiao et al. 2017; Weisenfeld et al. 2017; Gui et al. 2018; Nowoshilow et al. 2018).

Although very different, Hi-C (see the Chromosome Conformation Capture section) and NGM competed as a source of long-range information for scaffolding; circa 2017, they were approximately equal in terms of improving assembly statistics (Jiao et al. 2017; Yuan et al. 2017). Both the goat (Bickhart et al. 2016) and *Arabidopsis thaliana* (Jiao et al. 2017) genomes were assembled utilizing both technologies. Interestingly, relatively few publications had demonstrated the use of NGM for vertebrate genome assembly; though, it was speculated that this resulted from low public exposure to the technology (Howe and Wood 2015). Presently, Hi-C has far outstripped NGM for use in scaffolding. The explanation for this is likely the substantial cost difference: NGM requires purchasing a BNG Saphyr and appropriate reagents, while Hi-C requires only a library prep kit. See the review by Sedlazeck et al. (2018) for additional discussion on NGM and other mapping technologies.

Chromosome Conformation Capture (3C)

A strong background to 3C and its variants is provided in a review by Lajoie et al. (2015). Capturing chromosome conformation is useful for genome assembly projects because the information can help scaffold and phase assemblies. Hi-C, an all-vs-all variant of (3C) (Dekker et al. 2002), is one of these approaches. Unlike other C-techniques, *a priori* target selection is not

required (Lieberman-Aiden et al. 2009; Hakim and Misteli 2012). The Hi-C protocol (Belton et al. 2012) enables massively parallel sequencing (PE) on purified ligation products to generate unbiased, genome-wide chromatin interactions (Lieberman-Aiden et al. 2009). Since, read count is effectively proportional to distance (Lieberman-Aiden et al. 2009), one can "triangulate" (Lajoie et al. 2015) which sequences belong on the same chromosome and in which order they should be placed. The key idea is that the closer a locus is to another, the stronger the interaction and the subsequent signal. A strong signal does not guarantee two loci are on the same molecule; however, signal patterns between various loci can provide the necessary information to determine which loci are on the same molecule and which order they are in. This method has not yet been effective in providing accurate estimates of distance between loci, but it does provide megabase scale long-range information for scaffolding genome assemblies.

This approach to genome scaffolding has been demonstrated on human, mouse, and fruit fly data sets with the software package LACHESIS (Burton et al. 2013). Other early examples included a goat genome that also had help from BNG NGM (Bickhart et al. 2016) and a barley genome (Mascher et al. 2017). A related approach, Chicago™, is essentially Hi-C from reconstituted DNA instead of from a fresh sample. The durian fruit genome was a good early example of scaffolding with both Chicago and Hi-C (Teh et al. 2017). Both are commercially available via Dovetail Genomics (Scotts Valley, California, USA), which includes bioinformatics support with their software HiRise (Putnam et al. 2016).

Two other companies sell kits and services for Hi-C libraries: Arima Genomics (San Diego, California, USA) and Phase Genomics (Seattle, Washington, USA). Each is different in price and time required to prepare the libraries, but they do effectively the same thing as Dovetail's Hi-C product. One important consideration for Hi-C is the number of restriction

enzymes (REs) used. As with all physical maps, resolution is important, and resolution can be increased with commercially-available Hi-C products by ordering one with more REs. Two REs provide markedly more resolution than one RE, though the benefit flattens out as more REs are added. In this sense, DNase Hi-C (Ramani et al. 2016) is a significant improvement because it relies on a general purpose endonuclease instead of REs; the resulting resolution distribution is objectively superior to Hi-C with REs. The primary downside has historically been the large time requirement for completing the protocol, but Dovetail offers a much-improved (and expensive) version that it terms Omni-C. Of the two primary software packages meant for scaffolding with Omni-C, only SALSA (Ghurye et al. 2017; Ghurye et al. 2019) can handle DNase-based Hi-C (Dudchenko et al. 2017).

Other Physical Maps

A few other methods for generating physical maps exist. Older genome assembly projects generated physical maps by cloning into a vector and then probing the pieces cut by restriction enzymes (O'Rourke 2014). One other approach worth mentioning is RadMap, which is based on RAD sequencing. RadMap has been reported to outperform BNG NGM and Hi-C on highly fragmented (N50 <54 kb) human and *Arabidopsis* genome assemblies (Dou et al. 2017) and yet, does not require specialized instruments, which is a significant benefit similar to Hi-C. However, RadMap remains unvalidated because no other assembly has been published using the same technique in the five years since Dou et al. (2017) published the method.

Manual Inspection & Curation

Despite the enormous improvements made in sequencing, assembly, scaffolding, incorporation of multiple data types, etc., the algorithms are not perfect; indeed, no automated

process has produced anything resembling an error-free genome. Whenever possible, manual inspection of the assembly and any annotations is helpful. Unfortunately, if also understandably, curation techniques are difficult skills to transfer between people. Competent, professional curation is expensive and hard to come by. Genome browsers like the UCSC Genome Browser (Kent et al. 2002) or gEVAL (Chow et al. 2016) are helpful for inspecting regions of interest (at any resolution) and picking up on macro-level issues. When Hi-C data is used for scaffolding, the Hi-C contact matrix can be plotted and visualized with tools like Juicebox (Robinson et al. 2018), PretextMap/PretextView (High Performance Assembly Group - Wellcome Sanger Institute 2019; — 2020), and HiGlass (Kerpedjiev et al. 2018). Inspection of the Hi-C evidence for scaffold joins can help the curator fix misoriented or translocated contigs, detect misassemblies, etc. Details on how to effectively use a Hi-C contact matrix, use a genome browser, and perform curation tasks are well-beyond the scope of this manuscript, but instructions and tutorials are available for most of the listed softwares online. The codification of and availability of training for curation techniques is in its infancy; yet, Howe et al. (2021) provide an excellent start by describing in a helpful review their expertise in curation born from work on hundreds of assemblies and other experience.

Interoperability & Composite Softwares

Generally speaking, genome assembly is a modular process comprised of one or more of the following steps: read correction, assembly, polishing, scaffolding, and curation. One can generally switch the software for any given step without changing anything in the rest of the pipeline. This is advantageous because it allows individual steps to be replaced easily as new algorithms are designed or new sequencing types are produced. Unsurprisingly and understandably, modularity is compromised by inconsistent outputs between software packages

and a general lack of standardization. Developers should, at a minimum, provide an option to include information about how final outputs were obtained, e.g., scaffolders should provide not only the output FASTA file of scaffolds, but also an AGP (or similar) file showing how and with what evidence the contigs are arranged into the new scaffold sequences. This extra information enables subsequent re-use and evaluation by other software during quality control checks and subsequent steps.

As a practical example, consider the Hi-C scaffolder SALSA (Ghurye et al. 2017; Ghurye et al. 2019). If provided with contig-level FASTA file and a BAM file of Hi-C read alignments to the contigs, it will output scaffolds (FASTA) and an AGP file showing how the scaffolds are composed of contigs and the evidence supporting these joins. If SALSA is also provided with unitig tiling details (from the assembler), it can use the information to better make scaffolding decisions. If the chosen assembler produces the unitig tiling information and does so in the requested format, the scaffold-level assembly will improve. If a different assembler is used that does not produce the information, the scaffolds will not be as good (ignoring the fact that all assemblers have slightly different algorithms so the resulting contigs and scaffolds will inevitably be different anyway). In this case, the process is indeed modular, but the "modules" for assembly are not truly interoperable. For this reason, it is essential that all computational steps in a genome assembly project are careful considered during the project planning phase (i.e., before ordering sequencing).

Due to issues with interoperability, convenience, or precedence, many software packages are composites that do more than one thing. In their defense, the lines between assembly, polishing (including error correction, breaking misassemblies, local re-assembly, and gap filling), and scaffolding are not as cut-and-dry as have been described. If anything, this provides

further support for maximizing interoperability by providing options to output intermediate and supporting information. Similarly, it is a good reason to allow parts of a composite program to be skipped. Canu (Koren et al. 2017) is an excellent example of this; in addition to assembly, it has the ability to correct raw reads by consensus from all-vs-all read alignments (i.e., it is a composite software package). With the appropriate options and pre-corrected reads provided, the correction step can be skipped, enabling the use of an alternate correction module (e.g., RaCon (Vaser et al. 2017) or CONSENT (Morisse et al. 2021b)).

++itr (Iterate, Iterate, Iterate)

As the aforementioned lines between genome assembly steps (i.e., correction, polishing, etc.) are blurry, one would do well to not view genome assembly as a simple linear progression from reads to contigs to scaffolds to chromosomes. Iteration is a critical component. This is certainly true within steps, such as in polishing, where more than one round of polishing is common (i.e., contigs to polished contigs to more-polished contigs). Similarly, some scaffolding might be viewed as an iterative flip-flop between contigs and scaffolds, where contigs are joined into scaffolds and, after evaluation, are again separated and possibly recombined in different ways. A person or group working on a genome assembly project should expect to experiment with different software packages and create multiple iterations of the assembly. Genomics has come a long way in the last decade, but many questions are still unanswered. Even those questions that appear to have an answer, may be valid for only human genomes or particular clade. Rigorous evaluation of every intermediate assemblies will help guide the project.

Assessing Genome Assemblies

How can one determine whether more polishing rounds are necessary? Which assembler outputs the better assembly? Are these scaffold joins valid? Is this variant real? Confidence in genome assemblies is essential for gaining any biological understanding from them in subsequent studies, and that confidence begins with careful quality control and assembly assessment. As a general rule, one should follow established and/or recommended quality control procedures at every step in the sequencing and assembly process.

Traditionally, assemblies were assessed on a single metric: N50, the length of the contig in which 50% or more of the assembly is contained in contigs of equal length or longer (this metric is *not* the median). N50 is a stand-in measure for contiguity, the continuousness of contigs. Like an average, it is helpful, but it does not provide the full picture like a more-fully described distribution would. Even with one or more plots of contiguity, the length is only one aspect of the quality of a genome. Furthermore, "bigger" is not necessarily "better". Distributions of fragment lengths matching informed expectations are the best. In the end, assembly assessment falls into three categories: contiguity, completeness, and correctness. For those who are already familiar with measures of contiguity, PacBio has a helpful blog post (Pacific Biosciences 2020) exploring completeness and correctness as additional measures of assembly quality; however, short summaries of each category are herein described.

Contiguity

Contiguity is all about length, and it can be measured at the read, contig, or scaffold level. As was mentioned, N50 is a popular measure of contiguity. N50 is only one of several Nx statistics, where x refers to a percentage of the assembly size. N50 and N90 are frequently reported in prose or tables, but N1-100 can easily be calculated and plotted to show the entire

spectrum. A more representative metric than N50 is the area under the N-curve (auN; Li 2020a), and the field would be benefited by a shift towards reporting the auN alongside N50. A popular variant of Nx statistics is NGx statistics, where the "G" refers to the genome size instead of the assembly size. Like Nx statistics, NGx statistics can be easily calculated and plotted to show the distribution and area under the NG-curve (auNG). One related metric worth mentioning is Lx (and LGx), which describes the number of sequences (reads, contigs, or scaffolds, depending on the situation) needed to reach the corresponding Nx (and NGx). N(G)x and L(G)x statistics have an inverse relationship with each other: good assemblies will have high N(G)x and low L(G)x.

Of course, optimizing (i.e., seeking to maximize N(G)x and minimize L(G)x) contiguity statistics does not always produce the best outcome. Consider the following simplified example: if a genome has 10 chromosomes of length 10 mb each, the genome size is 100 mb. Half of the genome size (for the NG50) is 50 mb. If the assembly were perfectly contiguous (end-to-end for each chromosome), the N50 would be 10 mb. For a real-life (i.e., imperfect) assembly, a value <10 mb is expected. A value >10 mb would indicate the invalid joining of contigs/scaffolds because the chromosomes are not joined end-to-end in real life. For this particular example, the same statements are true for the NG10, NG20, and so on through the NG90 and NG100. The more information you have about the cellular biology of a genome, the better you will be able to assess assembly quality.

Completeness

While contiguity is important, if the assembly size is only 40% of the genome size, then the assembly is not very high-quality overall, even if the existing portion is both contiguous and correct. While there are valid reasons that support getting <95+% of the genome represented in

an assembly, the assembly team must consider the biology of the particular genome and the details of the sequencing experiments. Otherwise, low percentages are indicative of problems. Another common way to measure completeness is with single-copy orthologs that are highly-conserved across evolutionary clades. BUSCO (Simão et al. 2015) will scan an assembly for single-copy orthologs defined in OrthoDB (Kriventseva et al. 2019) and characterize the abundance and completeness of each ortholog. Three summary values are provided: the number of orthologs that are complete (C), fragmented (F), and missing (M) in/from the assembly. C is further broken into two categories: single-copy (S) and duplicated (D). The sum of S and D is C, and the sum of C, F, and M is the total number of orthologs analyzed. While the organism being assembled may have genuine variation, a high value of C and low values of F and M are expected. Similarly, duplications could have occurred, but a low value of D and high value of S are expected. These BUSCO scores are also often represented as percentages of the number of orthologs in OrthoDB for the selected clade.

Correctness

Correctness is difficult to assess when the "right" answer is unknown. In the simplified situation where a gold-standard reference exists, comparing alignments of the two assemblies can provide helpful information about how correct the assembly is. Naturally, this works only if a sufficiently high-quality reference is available. Further, if the goal is to complete a "perfect" T2T assembly, no reference is available for any species, including human. That specific case aside, how can a person tell if a SNP is a mutation or an error? Or if the SV is a scaffolding artefact or real biological rearrangement? If a reference is unavailable, each variant may have to be handled on a case-by-case basis, but SNP or indel errors, once detected, can be automatically

modified via polishing, possibly in a targeted fashion. Larger SV errors may have to be fixed manually with a text-editor or pseudo-manually with hand-made BED files and tools like BEDTools (Quinlan and Hall 2010). With a reference, it is important to mask repeats and other low-quality regions. PacBio has proposed a method for doing this and generally assessing concordance with a reference (Kingan et al. 2020).

Automating methods to determine, characterize, and report errors in genome assemblies is an active field of research. Of necessity, clever methods have been devised for individual genome projects to assess a perceived or anticipated problem. Only approximations of correctness are currently available unless significant resources are invested, such as comparing a newly-assembled genome to BACs (Vollger et al. 2020) or immortalizing a cell-line to systematically characterize and determine how to sort individual chromosomes. A k-mer analysis with Merqury (Rhie et al. 2020; Walenz et al. 2020) or Yak (Li 2020b; Cheng et al. 2021) can identify potentially erroneous k-mers that can subsequently be removed or polished. Merqury can also generate k-mer spectra plots, which can help visualize the frequency of erroneous k-mers and k-mers not present in the reads.

Annotating Genome Assemblies

Even the best genome assemblies are relatively useless without high-quality annotation. Annotation typically focuses on protein-coding genes, referring to identifying the location, ideally including exon-intron boundaries, UTRs, splice variants, etc. Furthermore, identifying the gene name, gene family, homologs in related species, function of the translated protein product, etc. are essential elements. Often genes are identified based on extra sequence information (e.g., RNA-seq), homology searches (e.g., Dunne and Kelly 2017), and/or *ab initio* gene predictors, which may involve machine learning techniques. Annotation information is stored in a variety of

file formats, depending on the exact situation. Predominantly, GFF (gmod.org/wiki/GFF3) is used, but BED format is common for use with a genome browser, such as the UCSC Genome Browser (Kent et al. 2002). Databases are another common method for storing annotation information. Naturally, other types of annotations are possible, but are not typically common (Yandell and Ence 2012) and are stored in a variety of formats.

Genome annotation software is typically an amalgamation of various softwares, compiled into a pipeline with wrapper scripts (Holt and Yandell 2011; Hoff et al. 2016), though some are completely automated, as in the NCBI Eukaryotic Genome Annotation Pipeline for NCBI genome assembly submissions (Thibaud-Nissen et al. 2013). Understanding how and when to adjust default settings for each step of the process is non-trivial and specifics will vary with each genome assembly project. In their present state, running annotation pipelines require bioinformatics expertise and an intimate understanding of sequencing technologies, bioinformatics algorithms, and the organism of interest. Ultimately, the choice of which pipeline to use will vary based on the specific situation; in some cases, organism or group-specific pipelines have been developed (Proux-Wéra et al. 2012; Campbell et al. 2014). Yandell and Ence (2012) provide a helpful review about eukaryotic genome annotation that is geared towards beginners. Ekblom and Wolf (2014) provide a helpful guide to assembly and annotation written to conservation geneticists that assumes limited background in HTS and bioinformatics. A helpful set of suggestions for submitting genome assemblies to NCBI is provided by Pirovano et al. (2015). Mudge and Harrow (2016) review structural and functional annotation and provide helpful definitions and background information; the information included in this review is critical for understanding the inherent limitations of annotation. The MAKER annotation pipeline (Holt and Yandell 2011; Yandell and Ence 2012; Campbell et al. 2014) has been the

foremost annotation pipeline for many years, but *caveat emptor*: installation is very cumbersome and difficult, even for some experienced system administrators. Liftoff (Shumate and Salzberg 2020) has gained traction recently for transferring annotations (presumably from a high-quality, trustworthy source genome) to a new genome or assembly version. Similarly, the Comparative Annotation Toolkit (CAT; Fiddes et al. 2018) is promising as a method for comparing annotations between genomes.

COMMENTARY & GUIDANCE

Entering the realm of genome assembly is extremely daunting. The technologies and methodologies have evolved so rapidly that the methods from most papers are well-behind the then new "standard". This has been particularly true since the advent of TGS as the competition to become the *de facto* best long-read platform has been fierce. The pace of research has been so breakneck that sifting through the sheer number of published (and preprint) genome assemblies alone is unpractical. Consequently, some of the lessons we learned from teaching ourselves genome assembly over the last few years were effectively irrelevant one year later. Nevertheless, these lessons (often anecdotal) could prove useful to the assembly newcomer – we certainly wish such a resource were available when we started. Accordingly, we present a series of lessons-learned, case studies, and general commentaries about genome assembly.

Assembly with Long, Noisy Reads

HiFi reads provide a distinct advantage in genome assembly, but not every project has the funds for PacBio data or access to the right sequencing machines. Further, some may have "old" CLRs that they have not yet had the chance to turn into a genome assembly – or perhaps did not

realize that HiFi reads would likely have been a better choice. Others may simply be interested in using ONT data for a variety of valid reasons. The following contain some helpful guidelines for managing genome assembly when the reads are both long and noisy.

Read Correction

Searching for a read correction software is overwhelming due to the sheer number of options. One question we faced was whether we should do self-correction or use a hybrid approach with short reads. We also wondered whether there was a cumulative benefit to trying both self-correction and then hybrid-correcting the already self-corrected reads. The right answer depends somewhat on the circumstances, but we generally found that hybrid correction is the least effective approach. One concern with hybrid correction is the aggressive collapsing of haplotypes and real duplication, which can occur when short reads map equally well to more than one location in the genome. Even ignoring issues such as low-complexity DNA regions, sequencing biases, and the non-random nature of nucleotide sequences, non-unique mappings are expected from short reads with greater probability than from a read with more bases. We recommend using self-correction only, though this is based on an important assumption; namely, we assume you have high sequencing depth. Due to the initial high cost of PacBio sequencing, hybrid correction was a cost-saving option because low-coverage CLRs (e.g., 12x) could be corrected using high-coverage short reads (e.g., 100x), and the reads would map back to a reference genome with accuracy similar to high-coverage (i.e., 50-100x), self-corrected CLRs. Budget providing, we recommend obtaining higher depths for the long reads and skipping the hybrid correction.

For completeness, we also tried "dual" correction, which is simply performing hybrid correction on already self-corrected reads. The three tested strategies (Fig. 3) were compared on

a single ~1 gb fish genome, and our results suggested that dual correction was the best option (Fig. 4). However, subsequent experiments with other genomes yielded inconsistent results. Hybrid correction was consistently worse than self-correction in terms of contiguity. We hypothesize that this is due to the unavoidable collapsing of real variation between alleles and other genomic regions. We also found that self-corrected reads generated more-contiguous assemblies than dual-corrected reads in each other case. Between these results and the general concern over the deleterious effects of hybrid correction, we recommend using a self-correction strategy. Current options for this would be with consensus modules in the assembler itself (e.g., Canu) or stand-alone consensus software (e.g., RaCon or CONSENT).

Short Read Correction

The Illumina reads that were used in the aforementioned *Albula glossodonta* correction strategies experiments were first corrected. We generally have not seen others correct Illumina reads (or at least not report that they did), and we have stopped doing so ourselves as the correction process is time-consuming (computationally), and the algorithm we liked best, Quake (Kelley et al. 2010), is implemented in old software that is cumbersome to install. That said, Quake did make corrections in our read sets, though the q-value cutoff had to be manually determined, which makes the approach difficult to replicate. We also corrected RNA-seq reads from Illumina with Rcorrector (Song and Florea 2015), but have similarly stopped using it because in every case (at least four different fish species) zero changes were made to the reads. With near-perfect accuracy for Illumina reads (>99.9%), this is not surprising as we expect zero errors in any given Illumina read. We do, however, recommend running FASTQC (Babraham Bioinformatics Group 2015) or other similar quality-control software to ensure nothing is

anomalous about your short reads. Additionally, be sure to remove sequencing adapters/indexes/etc. (e.g., with CutAdapt (Martin 2011)) if they were not already removed from the sequence dataset.

Noisy Read Correction vs. Polishing

When we first heard of polishing from others who had used Pilon, we were extremely skeptical as it was described as a method to correct SNPs and indels in the assembly without any correction in advance. What we came to learn was that they did do correction in advance, they simply did not realize that that their assembler of choice (Canu) did it for them. Unfortunately, we did not realize this until we later tried it ourselves, and our limited viewpoint of the purpose of polishing persisted for longer than we care to admit. We reasoned that correction in advance made more sense than correction after-the-fact because it would make the assembly graph less complex. Further, RaCon and CONSENT did not yet exist (though PacBio's long-read consensus module did), and the high RAM requirement of Pilon was off-putting.

With time, we came to realize that correction and polishing serve very different purposes, even if part of their function is similar. Whether you have HiFi reads (i.e., reads for which you should not run a correction step) or CLRs or ONT long-reads that have been corrected, polishing should at least be attempted post-assembly when data is available. With ONT long-reads or CLRs, polishing with GCpp (Arrow) or Nanopolish to utilize the long reads is common. Polishing with short reads (e.g., with RaCon or Pilon) is also popular. Many have used both data types for polishing in an iterative fashion, starting with the long reads. Usually, one or two rounds of polishing is done for each data type. We have not done enough evaluation of these polishing strategies to provide meaningful counsel except that (a) a polishing strategy should be utilized and (b) the resulting assemblies should be evaluated.

Genome Size Determination

To sequence a genome adequately, an appropriate sequencing depth must be selected. The depth will depend on sequencing type and may change as chemistries/error rates/ etc. improve; your sequencing provider (i.e., sequencing center) or sequencing producer (i.e., PacBio, ONT, etc.) can provide up-to-date recommendations. Assuming the desired depth has been determined, it can be utilized alongside the genome size to order the appropriate amount of sequencing. The simplest way to determine genome size for vertebrates is from the Animal Genome Size Database (Gregory 2021). Ideally, your species of interest is listed with a C-value. If a C-value is not listed for your species of interest, it can be estimated based on listed related organisms; however, a C-value estimated in this way is not guaranteed to be correct, especially if there is variation of the C-value in the clade. Provided that the C-value has been determined and assuming a GC-content of 50%, the C-value can be converted into a haploid genome size with the simple formula: S = 0.978C, where C is the C-value and S is the genome size in gigabases (Doležel et al. 2003).

If the genome size is not in the database, it can also be estimated experimentally with flow cytometry (Hare and Johnston 2012) or Feulgen microspectrophotometry (Leuchtenberger 1954; Hardie et al. 2002). If you have accurate reads (e.g., Illumina short reads), the size can also be estimated based on a k-mer analysis. Even if you have a good genome size estimate from an experiment or the Animal Genome Size Database, it is good practice to corroborate the size estimate by performing an *in silico* k-mer analysis. The k-mer analysis requires the following steps: (1) generate a k-mer coverage histogram, (2) calculate the area under the curve, and (3) identify the peak. The genome size can then be determined according to the following equation:

$a / p = s$, where a is the area under the curve, p is the number of times the k-mers occur (the x-value) at the peak, and s is the genome size. While the k-mer analysis can be done semi-manually, we recommend the much simpler approach: GenomeScope (Vurture et al. 2017; Ranallo-Benavidez et al. 2020). The input for genome scope is a "histogram" file, which is a two-column, space-separated text file containing the coverage or copy number in the first column and frequency in the second column, which can be generated using one of many programs (e.g., Jellyfish (Marcais and Kingsford 2011) or KMC (Kokot et al. 2017)). Note that in the GenomeScope profile, the value of "len" is the genome size. Also note that k-mer-based estimates of genome size can be inaccurate when the genome is unusually homozygous, the sequencing error rate is high, or the coverage is too low.

Tips for Select Software Packages

While specifics on how to run software, manage jobs in a cluster environment, etc. are outside the scope of this report, some software packages are particularly complex, and, as such, general recommendations are provided herein. Specifically, we provide experiential viewpoints about three software packages: (Hi)Canu (Koren et al. 2017; Nurk et al. 2020), MAKER (Holt and Yandell 2011; Campbell et al. 2014), and purge_dups (Guan et al. 2020).

(Hi)Canu

Canu, which is the same program used for HiCanu for HiFi reads, is an assembler that can also correct noisy reads. Canu is well-written, well-documented, and well-maintained. We mention it here only because it is a unique piece of software when running on a cluster. Specifically, Canu is capable of submitting itself to the cluster, including managing the resources

it requests for different jobs. The main Canu program assesses the cluster environment and starts an initial set of jobs while also submitting itself as an "executive" job as a dependency of these other jobs. This executive job assesses Canu's overall progress and submits new jobs, as needed, to tackle subsequent steps or redo failed steps. Then, it once again submits itself as an executive job as a dependency and the cycle continues. Notably, the initial Canu command need not be submitted as a job because it can be run quickly (i.e., <5 seconds) on an interactive node. The cluster we use is managed by SLURM (https://slurm.schedmd.com), but Canu works with other workload managers as well.

MAKER

MAKER is an annotation pipeline written primarily in Perl (https://www.perl.org). While MAKER combines a remarkable number of software packages together to accomplish a very complex and very difficult task, and despite a fair amount of guidance available in "annotation school" (Holt and Yandell 2018) and the help emails (https://groups.google.com/g/maker-devel), MAKER is notoriously difficult to run. We do not blame this on MAKER; it is a product of the enormity of the task of annotation and the age of the software. Unfortunately, many projects that use MAKER are vague about how they accomplished it. For example, little more than "and we annotated with MAKER" is sometimes stated in manuscripts. Other times, the entire annotation process is described as if it were all by hand, even though it was obvious that MAKER was used. You can save yourself extensive frustration by recognizing that MAKER is a tool to accomplish many diverse annotation tasks and is not a "push-button" solution that simply outputs reliable and usable annotations. Additionally, an extensive understanding of annotation is required; as such, it may be beneficial find a collaborator with annotation experience. For additional information about our overall annotation process (primarily using MAKER), including the exact

settings and commands run, see our *Caranx melampygus* genome paper supplement (Pickett et al. 2021). We do not claim that this is how you should annotate your assembly; it is simply a reference.

Again, another important thing to discuss about MAKER is that it is prodigiously difficult to install. Part of the issue is that it has so many dependencies, some of which are beginning to get rather old, especially some of the Perl modules. Another part of the issue is that it is difficult to manage more than one Perl installation on the same system, especially if custom modules need to be universally available (i.e., available to more than just one user). Things are even more difficult if the user attempting the install does not have or does not wish to use root privileges. While we did eventually manage a successful installation ourselves, our cluster's operating system was upgraded a few weeks later, breaking dependencies and the installation. Despite careful notes and following the exact same steps, we failed to re-install it. With extensive help from our system administrators, and after several months of work, we managed to install it a second time. We strongly encourage others to plan accordingly or determine another method of annotation. Please, note that at least one update to MAKER has been released since we had this experience (v3.01.02-beta); it is possible that the issue is helped in the update. It is also possible that our system was configured in an unusual way that interfered with the process.

purge_dups

purge_dups can be run on a contig-level genome assembly to purge duplicate contigs and generate a primary and alternate assembly. It is an excellent program, and we highly recommend it. We do provide a gentle forewarning, however. On the GitHub repository (https://github.com/dfguan/purge_dups), formal releases have been fairly far and few between considering the jump in version numbers. We used v1.0.1 because it was the most recent release, despite many

commits having been made afterward. We encountered a bug in the program that would silently replace entire contigs with Ns in certain circumstances. Gratefully, it had already been fixed in a numbered version, but that and many other versions were not formally tagged or listed in the releases. We have had success with v1.2.5, also labeled the "Chinese New Year release". We advise checking the list of commits, which have previously been named according to version number. If a new version number appears there, but not in the tags or releases, it may be worth skipping straight to it instead of using the formal release. Use your own judgement based on the content of the commit messages.

Scaffolding Scaffolds

We strongly recommend scaffolding with Hi-C data, as previously described. However, it is also possible to combine more than one data type for scaffolding. If you have the ability to generate BNG NGM data, this is also an excellent data source for scaffolding. The specifics for how combining data types works will vary between software packages, but a few principles will help. First, scaffold with data types based on the length of the long-range information they provide. For example, if you were to scaffold with Hi-C data, BNG NGM data, and read clouds (e.g., if you had old 10XG data or tried new TELL-seq), you would start with the shortest-range data (read clouds) then scaffold those scaffolds with the BNG NGM (longer-range data), and scaffold that set of scaffolds with the Hi-C data (longest-range data). Take care to avoid naming collisions as many software packages will name new scaffolds after a simple naming scheme (e.g., scaffold_1, scaffold_2, …, scaffold_N) and can complicate the situation (and may even create errors) if newly-created scaffolds from a "higher" level of scaffolding have the same name as one of the scaffolds from a previous round of scaffolding.

Moreover, when using more than one data source for scaffolding, we recommend keeping track of how to convert the contig-level assembly into the final scaffold-level assembly. If your scaffolders outputs an AGP file (or information sufficient to create one from it), you can programmatically propagate the information through each file to create a master AGP file with evidence for each type of join. This will be helpful when it comes time to submit to the assembly to NCBI. You can submit the contig-level assembly and the AGP file describing the joins. Note that any changes made during polishing to scaffolds would need to be retroactively applied to the contig-level assembly and/or master AGP file. Currently, this requires custom scripting as no software has been published to handle this.

Recommendations for New Projects

As of the time of this writing (Spring 2021), we recommend PacBio HiFi reads as the basis for the assembly. As a side note, some sequencing centers may ask you if you want the raw reads or just the HiFi reads, alternately, they may not even ask and just provide the HiFi reads. We encourage you to get and store the raw reads in addition to the HiFi reads as certain circumstances may benefit from using the underlying CLRs (e.g., if PacBio publishes an update to their consensus algorithm). Unless your project requires an assembly of every haplotype in the specimen, we recommend planning on generating a single haploid representation of the genome. Speak with your sequencing provider or PacBio about the necessary sequencing depth for your organism. We also recommend generating Hi-C data for scaffolding. If you intend to perform annotation or hope that NCBI will include your assembly in RefSeq and annotate it for you, we advise doing some form of RNA-seq (PacBio Iso-Seq being strongly recommended). While any assembler should work fine (e.g., Falcon, HiCanu, Hifiasm, and Peregrine), we recommend

Hifiasm. If your assembler outputs separate primary and alternate assemblies, use the primary assembly for the next step. Use purge_dups to split the assembly into primary and alternate assemblies. If your assembler already did this, combine the two alternate assemblies into a single file and use the primary assembly from purge_dups as your primary assembly. While you can polish at this stage, we advise waiting to polish until you have scaffolds. We do not recommend polishing with short reads. Scaffold with Hi-C data using SALSA and polish with GCpp (Arrow). For a more in-depth process, including code, and for up-to-date suggestions, see what the VGP (Vertebrate Genomes Project) is currently doing as recorded in their GitHub repository (https://github.com/VGP/vgp-assembly). This is an excellent resource, especially if you also wish to use other sources of information, such as linked-reads for polishing or BNG NGM for scaffolding. We also recommend generating short reads for genome size estimation and quality control steps.

Bioinformatics Best-practices for Genome Assembly

Any experienced bioinformatician knows how easy it is to forget what you did one week/month/year ago. Just as any wet-lab scientist should take careful notes of their experiments, bioinformaticians should do the same thing. Genome assembly, in particular, has so many moving parts and can have many iterations. So, take careful notes, use descriptive directory and file names, write down the version number and options used for each run of a program, and think twice, type once (measure twice, cut once). Be sure to record the justification (and sources, if appropriate) for decisions you make. Your future self will thank your present-day self when it comes time to justify your methods, publish a paper, or replicate the analysis on a different sample/species/project/etc.

On a related note, our experience is that sequencing details and sample information are easily lost or forgotten. Proper project planning and management will help avoid issues, but often a bioinformatician joins a project at analysis time, not having been able to provide input previously. In such cases, the prudent bioinformatician will relentlessly pursue key pieces of information at the beginning of a project. If necessary, bioinformaticians may refuse to perform any more of the analysis (in this case, genome assembly) until you acquire the requisite information. For sequencing data sets, you will need the following set of details at minimum: (a) sample collection details, (b) sample storage and transfer details (i.e., shipped on dry ice, stored at -80°C), (c) library preparation protocol, including kit names and numbers, methods for quantifying (and values of) concentrations and other quality control procedures, PCR times and temperatures (if using PCR), images of any gels, sequencing adapters, unique molecular identifiers (i.e., barcodes), etc., (d) sequencing machine (e.g., Illumina Hi-Seq 2500), (e) number of cycles (if Illumina; movie length for PacBio; run time for ONT), and (f) date of the sequencing run. For sample collection, you will need the following details: (a) species of the sample, (b) number of individuals, (c) tissue(s) collected, (d) collection date(s), (e) how long the sample was "left out" before being preserved, (f) longitude and latitude (when collected), (g) description of the collection site (i.e., collection medium (was it sandy, muddy, grassy, etc.), collection locality (e.g., near a reef (front or back), water depth, height up a tree, etc.), broad environment (e.g., ocean, tropical, rain forest, glacier, etc.)), (h) one or more of strain, isolate, cultivar, and ecotype (if none of these four, makeup a unique identifier and assign it to isolate), and (i) any other detail needed to create an NCBI BioSample for the sample(s). Do not rely on core facilities, sequencing centers, web-lab technicians, collaborators, principal investigators, or

anyone else to record, recall, or otherwise estimate these details. Again, it is your responsibility to ensure you have all the details recorded and backed up yourself.

Finally, data security is a critical task for bioinformaticians. Keep backups of all project-related documents on the cloud and/or other external drive from your primary workstation. If at all possible, automate this process. Similarly, keep backups of all original data and final results, which may be on some kind of cluster or cloud computing resource. Specific details will vary between institutions, but a common concept in high-performance computing (HPC) is a "scratch" space. Scratch spaces are typically faster storage drives (which makes computing more efficient) and are not backed up. In most situations, it is not practical to keep backups of all work and intermediate files; however, some method to keep two or more copies of raw data files (e.g., FASTQ files from a sequencing machine) and results (e.g., corrected reads, final contig- and scaffold-level assemblies, etc.) on separate drives, ideally in different physical locations, must be employed. Again, if possible, automate the backup process (copying one copy to another location). When raw data or final results are generated, copy them to the non-scratch drive. HPC centers can experience critical drive failures and can result in enormous losses of time, money, and other resources if a good data backup policy was not employed.

CONCLUSIONS & FUTURE DIRECTIONS

Every genome sequencing project is unique. Decisions about library preparations, sequencing technologies, read depth, read correction, assembly strategy, polishing, sources of long-range information for scaffolding, annotation pipelines, etc. will vary depending on the unique characteristics of the organism in question, the intended purpose of the whole genome sequence, and the available funding. Moreover, as the field and sequencing technologies

continue to rapidly advance, the ideal technology (or, more likely, combination of technologies) will change. As the changes occur, the field would be greatly benefited by formal experiments designed to test the various sequencing technologies (and combinations of technologies) for their utility in various aspects of genome assembly projects. Of the several critical questions that remain unanswered about current and emerging options, we prioritize the following questions: How well will ONT ultra-long reads perform for gap-filling in HiFi assemblies? What is the best way to incorporate these two data types together – specifically, can they be incorporated into a hybrid graph and what is the best way to do this? How can we share graph information between runs of different graph-based software packages? How can we combine multiple sources of evidence during the scaffolding process in an automated fashion, in particular, how can we combine optical mapping data (e.g., BNG NGM) and Hi-C? How can we enable non-specialists to correctly handle segmental duplications (SDs), telomeres, higher-order repeats (HORs), and centromeres? How much money and time does it realistically take to train someone to do genome assembly, assuming only a general understanding of genome biology with basic scripting and HPC skills? Answers to these and other questions will instruct future assembly and annotation projects and enable scientists to trust empirically tested sequencing and assembly strategies.

ABBREVIATIONS

API: Application Programming Interface
BED: Browser Extensible Data
BNG: BioNano Genomics
3C: Chromosome Conformation Capture
CPT-seq: Contiguity-preserving Transposition Sequencing
DG: Dovetail Genomics
DNA: Deoxyribonucleic Acid
GFF: Generic Feature Format
Hi-C: High-throughput (All-vs-all) 3C

HPC: High-performance Computing
HTS: High-throughput Sequencing
INSDC: International Nucleotide Sequence Database Collaboration
MP: Mate-pair
MPS: Massively-parallel Sequencing
NCBI: National Center for Biotechnology Information
NGM: Next-generation Mapping
NGS: Next-generation Sequencing
ONT: Oxford Nanopore Technologies
PE: Paired-end
PacBio: Pacific Biosciences
RAD: Restriction site Associated DNA
RNA: Ribonucleic Acid
RNA-seq: RNA Sequencing
SGS: Second-generation Sequencing
SLR: Synthetic Long Reads
TGS: Third-generation Sequencing
TSLR: Truseq-SLR
T2T: Telomere-to-Telomere
UTR: Untranslated Region
10XG: 10X Genomics

AUTHOR CONTRIBUTIONS

JSKK: Conceptualization; Funding Acquisition; Investigation; Supervision; Resources; Writing - Review & Editing. **BDP:** Conceptualization; Data Curation; Formal Analysis; Investigation; Methodology; Software; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing. **PGR:** Conceptualization; Funding Acquisition; Supervision; Resources; Writing - Review & Editing.

ACKNOWLEDGEMENTS

We thank the Brigham Young University (https://byu.edu) DNA Sequencing Center (https://dnasc.byu.edu) and Office of Research Computing (https://rc.byu.edu) for their

continued support of our research. We are grateful to Dennis K. Shiozawa for suggestions which improved the readability of the manuscript.

FUNDING

Not Applicable.

CONFLICT OF INTERESTS

None declared.

TABLES & FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Table 1. Reviews of sequencing, assembly, and related topics.

Reference	Description
(Pettersson et al. 2009)	Review of sequencing technologies.
(Schatz et al. 2010)	Describes how genomes can be assembled with NGS sequences if Sanger is used to fill gaps. Great section on NGS technologies. Great section on assembly.
(Earl et al. 2011)	Assemblathon 1 – Comparison of sequence assembly software.
(Quail et al. 2012)	Comparison of Ion Torrent, PacBio, and Illumina.
(Bradnam et al. 2013)	Assemblathon 2 – Comparison of sequence assembly software.
(Ekblom and Wolf 2014)	A review / field guide on sequencing, assembly, and annotation written for those with backgrounds in conservation genetics. Assumes the reader has limited background understanding.
(Fierst 2015)	Review on using linkage maps with assembly, but it has a helpful section on NGS and *de novo* assembly.
(Simpson and Pop 2015)	Review of assembly algorithms (not software performance).
(Heather and Chain 2016)	A brief history of DNA sequencing.
(Shendure et al. 2017)	Review of sequencing technologies and its current and predicted impact. Commemorates 40 years of DNA sequencing.
(Sedlazeck et al. 2018)	Review of long-range sequencing and mapping technologies and their applications.
(van Dijk et al. 2018)	Review of the third "revolution" in DNA sequencing with a discussion on the relative qualities of each technology.
(Logsdon et al. 2020a)	Review of long-read genome sequencing and its applications. Exceptional sections on the technologies and the practical implications of their respective use in *de novo* assembly. If you read any one of these, read this one.
(Howe et al. 2021)	Review of manual curation and its effects on assembly quality.
(Li 2021)	Blog post providing definitions to key terms in assembly, specifically referring to phased assembly; phased assembly without a reference is possible only because of trios and/or accurate long-reads (HiFi).

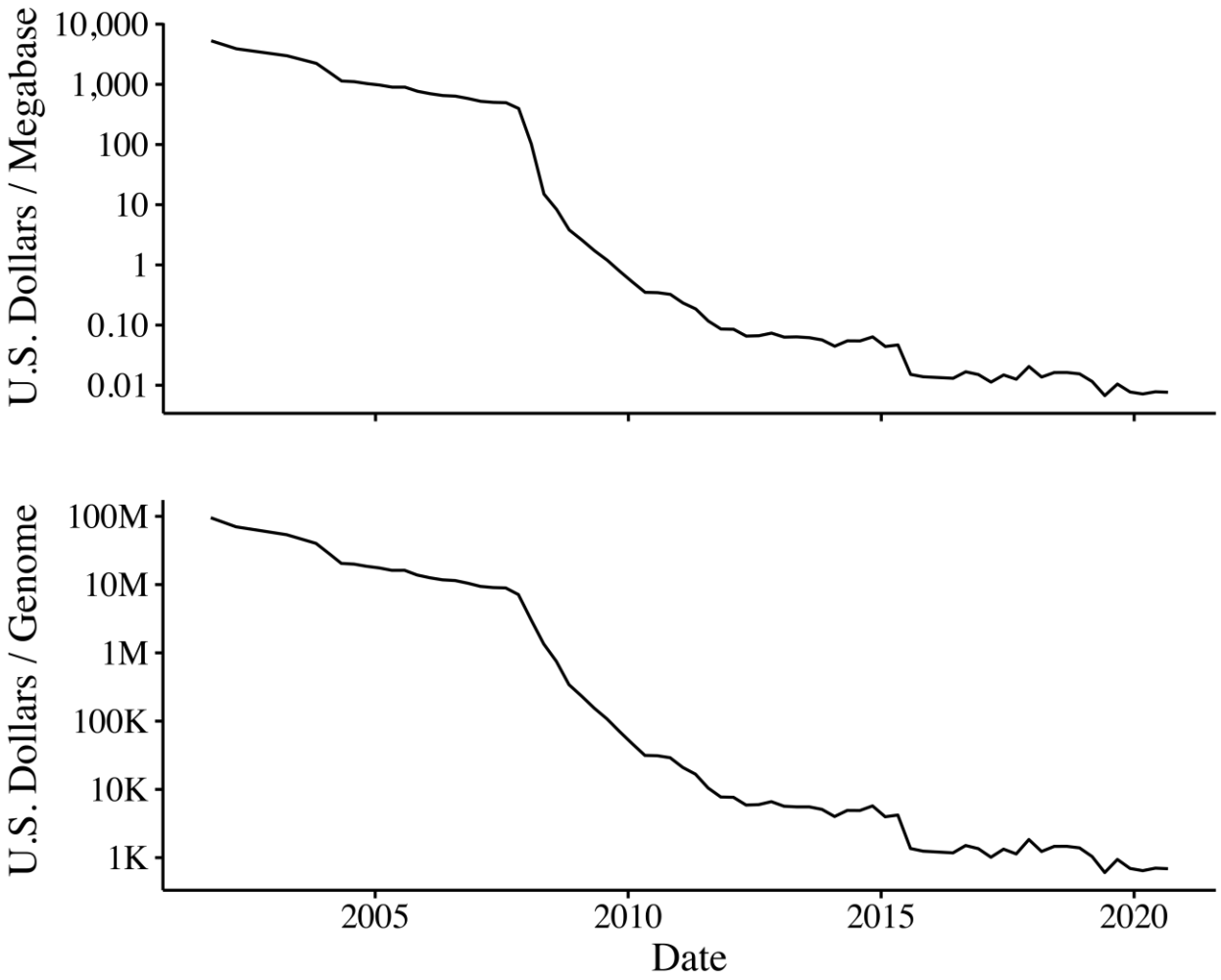


Figure 1. Cost of Genome Sequencing. The estimated cost of sequencing over time based on data reported by the U.S. National Human Genome Research Institute (NHGRI; https://www.genome.gov). The cost per genome is based on a 3 Gbp (haploid) genome. The advent of Massively Parallel Sequencing (MPS) platforms (e.g., Roche/454 systems and Illumina/Solexa systems) in the mid- to late-2000's enabled the precipitous decline in raw sequencing cost (Mardis 2011).

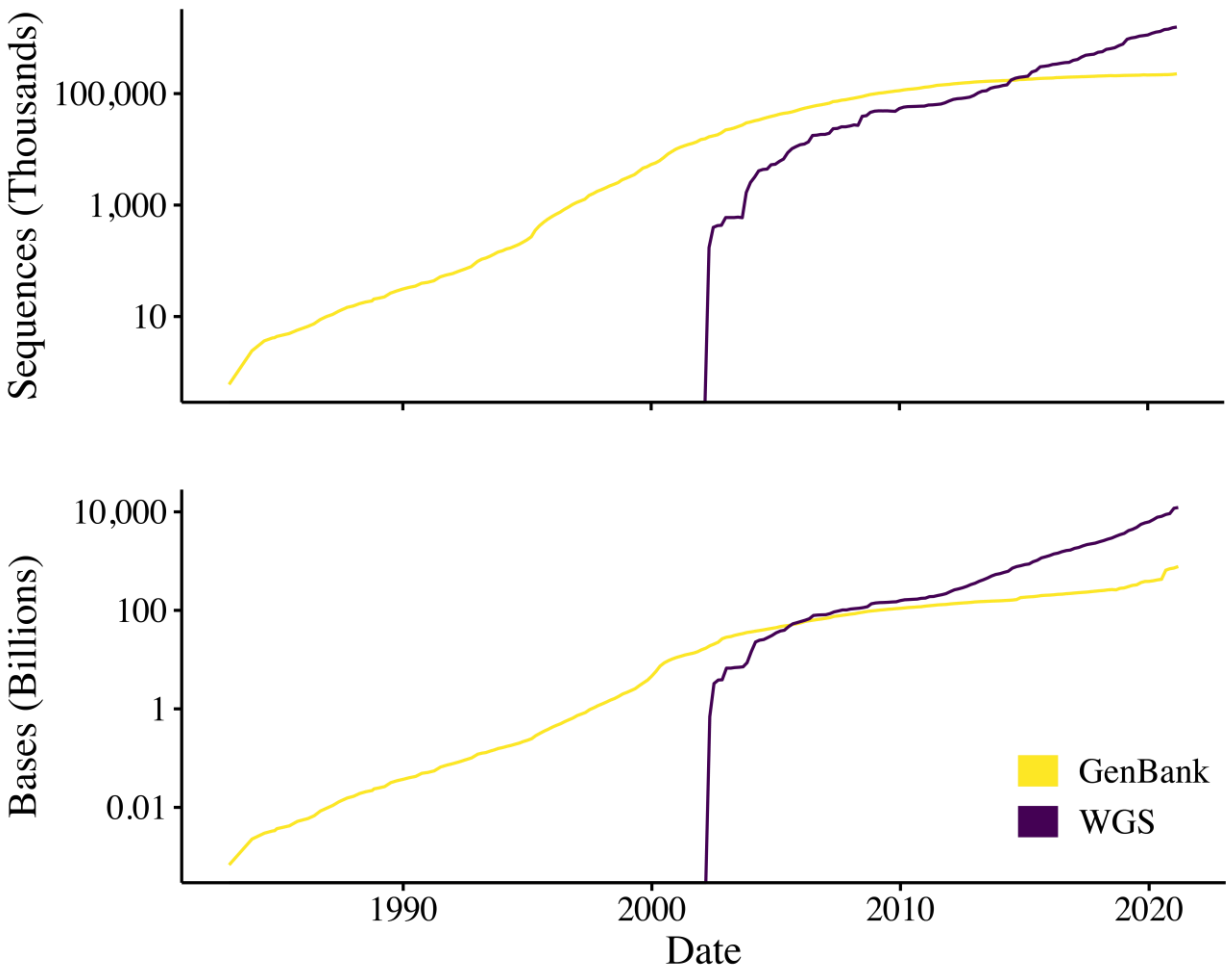


Figure 2. Genome Statistics Available on NCBI. The number of sequences and bases of the genomes available as NCBI GenBank and WGS submissions (https://www.ncbi.nlm.nih.gov/genbank/statistics). These statistics serve as a proxy for the number of genomes being sequenced over time.

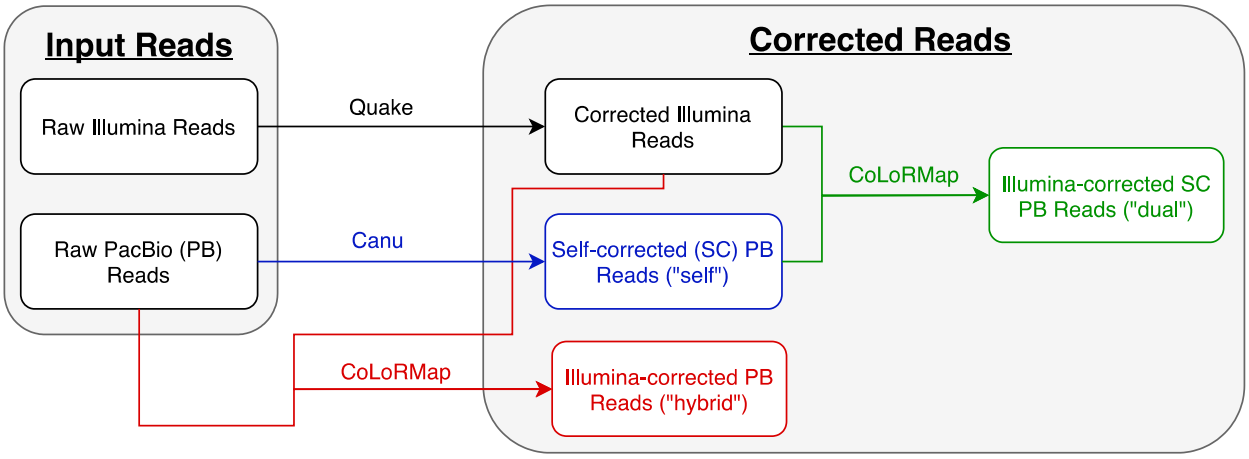


Figure 3. Flow chart showing the self-, hybrid-, and dual-correction strategies on an *Albula glossodonta* genome. Software choices are labeled above the connecting lines.

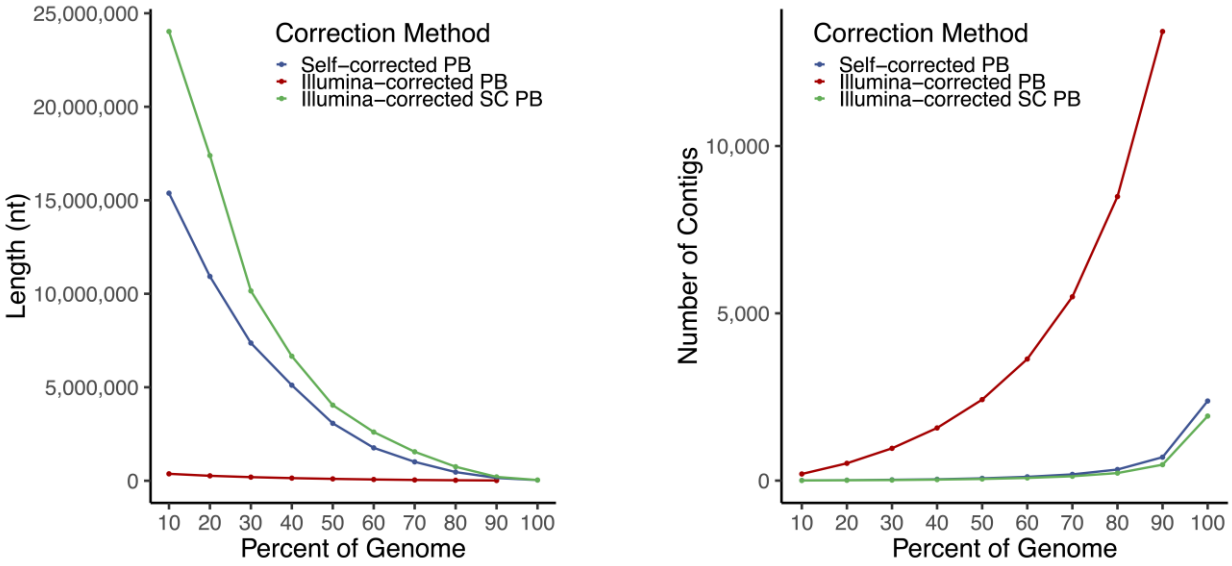


Figure 4. Comparison of self-, hybrid-, and dual-correction strategies on an *Albula glossodonta* genome. Plots of the contig-level NGx and LGx after Canu-based assembly for PacBio CLRs that were self-corrected (blue), hybrid-corrected (red), and dual-corrected (green). Note that the short reads were (probably unnecessarily) corrected before being used. Subsequent experiments with other genomes yielded inconsistent results, except that short-read only (i.e., "hybrid" correction) was the worst in terms of contiguity.

REFERENCES

10x Genomics. 2016. An Introduction to Linked-Read Technology for a More Comprehensive Genome and Exome Analysis. Pleasanton, CA: 10X Genomics, p 1-5.

10x Genomics. 2020. Discontinuation of Linked-Reads. URL: https://www.10xgenomics.com/products/linked-reads [accessed April 21].

Adey, A., J. O. Kitzman, J. N. Burton, R. Daza, A. Kumar, L. Christiansen, M. Ronaghi, S. Amini, K. L. Gunderson, F. J. Steemers, and J. Shendure. 2014. In vitro, long-range sequence information for de novo genome assembly via transposase contiguity. *Genome Research*. 24:2041-2049.

Alkan, C., S. Sajjadian, and E. E. Eichler. 2011. Limitations of next-generation genome sequence assembly. *Nature Methods*. 8:61-65.

Amini, S., D. Pushkarev, L. Christiansen, E. Kostem, T. Royce, C. Turk, N. Pignatelli, A. Adey, J. O. Kitzman, K. Vijayan, M. Ronaghi, J. Shendure, K. L. Gunderson, and F. J. Steemers. 2014. Haplotype-resolved whole-genome sequencing by contiguity-preserving transposition and combinatorial indexing. *Nature Genetics*. 46:1343-1349.

Assefa, S., T. M. Keane, T. D. Otto, C. Newbold, and M. Berriman. 2009. ABACAS: Algorithm-based automatic contiguation of assembled sequences. *Bioinformatics*. 25:1968-1969.

Aston, C., B. Mishra, and D. C. Schwartz. 1999. Optical mapping and its potential for large-scale sequencing projects. *Trends in Biotechnology*. 17:297-302.

Babraham Bioinformatics Group. 2015. FASTQC: A quality control tool for high throughput sequence data. *Babraham Institute*. URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc.

Baday, M., A. Cravens, A. Hastie, H. Kim, D. E. Kudeki, P. Y. Kwok, M. Xiao, and P. R. Selvin. 2012. Multicolor super-resolution DNA imaging for genetic analysis. *Nano Letters*. 12:3861-3866.

Badouin, H., J. Gouzy, C. J. Grassa, F. Murat, S. E. Staton, L. Cottret, C. Lelandais-Brière, G. L. Owens, S. Carrère, B. Mayjonade, L. Legrand, N. Gill, N. C. Kane, J. E. Bowers, S. Hubner, A. Bellec, A. Bérard, H. Bergès, N. Blanchet, M.-C. Boniface, D. Brunel, O. Catrice, N. Chaidir, C. Claudel, C. Donnadieu, T. Faraut, G. Fievet, N. Helmstetter, M. King, S. J. Knapp, Z. Lai, M.-C. Le Paslier, Y. Lippi, L. Lorenzon, J. R. Mandel, G. Marage, G. Marchand, E. Marquand, E. Bret-Mestries, E. Morien, S. Nambeesan, T. Nguyen, P. Pegot-Espagnet, N. Pouilly, F. Raftis, E. Sallet, T. Schiex, J. Thomas, C. Vandecasteele, D. Varès, F. Vear, S. Vautrin, M. Crespi, B. Mangin, J. M. Burke, J. Salse, S. Muños, P. Vincourt, L. H. Rieseberg, and N. B. Langlade. 2017. The sunflower genome provides insights into oil metabolism, flowering and Asterid evolution. *Nature*. 546:148-152.

Bao, E., T. Jiang, and T. Girke. 2014. AlignGraph: algorithm for secondary de novo genome assembly guided by closely related references. *Bioinformatics*. 30:i319-i328.

Bashir, A., A. A. Klammer, W. P. Robins, C.-S. Chin, D. Webster, E. Paxinos, D. Hsu, M. Ashby, S. Wang, P. Peluso, R. Sebra, J. Sorenson, J. Bullard, J. Yen, M. Valdovino, E. Mollova, K. Luong, S. Lin, B. Lamay, A. Joshi, L. Rowe, M. Frace, C. L. Tarr, M. Turnsek, B. M. Davis, A. Kasarskis, J. J. Mekalanos, M. K. Waldor, and E. E. Schadt. 2012. A hybrid approach for the automated finishing of bacterial genomes. *Nature Biotechnology*. 30:701-707.

Belton, J.-M., R. P. Mccord, J. H. Gibcus, N. Naumova, Y. Zhan, and J. Dekker. 2012. Hi-C: A comprehensive technique to capture the conformation of genomes. *Methods*. 58:268-276.

Berlin, K., S. Koren, C.-S. Chin, J. P. Drake, J. M. Landolin, and A. M. Phillippy. 2015. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology*. 33:623-630.

Bickhart, D. M., B. D. Rosen, S. Koren, B. L. Sayre, A. R. Hastie, S. Chan, J. Lee, E. T. Lam, I. Liachko, S. T. Sullivan, J. N. Burton, H. J. Huson, C. M. Kelley, J. L. Hutchison, Y. Zhou, J. Sun, A. Crisa, F. A. Ponce De Leon, J. C. Schwartz, J. A. Hammond, G. C. Waldbieser, S. G. Schroeder, G. E. Liu, M. J. Dunham, J. Shendure, T. S. Sonstegard, A. M. Phillippy, C. P. Van Tassell, and T. P. L. Smith. 2016. Single-molecule sequencing and conformational capture enable de novo mammalian reference genomes. *Journal of Chemical Information and Modeling*. 53:1689-1699.

Bodily, P. M., M. S. Fujimoto, Q. Snell, D. Ventura, and M. J. Clement. 2015. ScaffoldScaffolder: solving contig orientation via bidirected to directed graph reduction. *Bioinformatics*. 32:btv548.

Boetzer, M., C. V. Henkel, H. J. Jansen, D. Butler, and W. Pirovano. 2011. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics*. 27:578-579.

Boetzer, M. and W. Pirovano. 2014. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. *BMC Bioinformatics*. 15:211.

Bosi, E., B. Donati, M. Galardini, S. Brunetti, M. F. Sagot, P. Lió, P. Crescenzi, R. Fani, and M. Fondi. 2015. MeDuSa: A multi-draft based scaffolder. *Bioinformatics*. 31:2443-2451.

Bradnam, K. R., J. N. Fass, A. Alexandrov, P. Baranay, M. Bechner, I. Birol, S. Boisvert, J. A. Chapman, G. Chapuis, R. Chikhi, H. Chitsaz, W.-C. Chou, J. Corbeil, C. Del Fabbro, T. R. Docking, R. Durbin, D. Earl, S. Emrich, P. Fedotov, N. A. Fonseca, G. Ganapathy, R. A. Gibbs, S. Gnerre, E. Godzaridis, S. Goldstein, M. Haimel, G. Hall, D. Haussler, J. B. Hiatt, I. Y. Ho, J. Howard, M. Hunt, S. D. Jackman, D. B. Jaffe, E. D. Jarvis, H. Jiang, S. Kazakov, P. J. Kersey, J. O. Kitzman, J. R. Knight, S. Koren, T.-W. Lam, D. Lavenier, F. Laviolette, Y. Li, Z. Li, B. Liu, Y. Liu, R. Luo, I. Maccallum, M. D. Macmanes, N. Maillet, S. Melnikov, D. Naquin, Z. Ning, T. D. Otto, B. Paten, O. S. Paulo, A. M. Phillippy, F. Pina-Martins, M. Place, D. Przybylski, X. Qin, C. Qu, F. J. Ribeiro, S. Richards, D. S. Rokhsar, J. G. Ruby, S. Scalabrin, M. C. Schatz, D. C. Schwartz, A.

Sergushichev, T. Sharpe, T. I. Shaw, J. Shendure, Y. Shi, J. T. Simpson, H. Song, F. Tsarev, F. Vezzi, R. Vicedomini, B. M. Vieira, J. Wang, K. C. Worley, S. Yin, S.-M. Yiu, J. Yuan, G. Zhang, H. Zhang, S. Zhou, and I. F. Korf. 2013. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*. 2:10.

Breitwieser, F. P., M. Pertea, A. V. Zimin, and S. L. Salzberg. 2019. Human contamination in bacterial genomes has created thousands of spurious proteins. *Genome Research*. 29(6):954-960.

Brown, S. D., S. M. Utturkar, T. S. Magnuson, A. E. Ray, F. L. Poole, W. A. Lancaster, M. P. Thorgersen, M. W. W. Adams, and D. A. Elias. 2014. Complete Genome Sequence of Pelosinus sp. Strain UFO1 Assembled Using Single-Molecule Real-Time DNA Sequencing Technology. *Genome Announcements*. 2:e00881-00814-e00881-00814.

Burton, J. N., A. Adey, R. P. Patwardhan, R. Qiu, J. O. Kitzman, and J. Shendure. 2013. Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nature Biotechnology*. 31:1119-1125.

Butler, J., I. Maccallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. 2008. ALLPATHS: *De novo* assembly of whole-genome shotgun microreads. *Genome Research*. 18(5):810-820.

Campbell, M. S., M. Law, C. Holt, J. C. Stein, G. D. Moghe, D. E. Hufnagel, J. Lei, R. Achawanantakun, D. Jiao, C. J. Lawrence, D. Ware, S.-H. Shiu, K. L. Childs, Y. Sun, N. Jiang, and M. Yandell. 2014. MAKER-P: A Tool Kit for the Rapid Creation, Management, and Quality Control of Plant Genome Annotations. *Plant Physiology*. 164:513-524.

Chaisson, M., E. Eichler, and T. Marschall. 2020. Representing structural haplotypes and complex genetic variation in pan-genome graphs. Funded by *National Institutes of Health (NIH)*. Grant #1U01HG010973.

Chen, S., J. Xu, C. Liu, Y. Zhu, D. R. Nelson, S. Zhou, C. Li, L. Wang, X. Guo, Y. Sun, H. Luo, Y. Li, J. Song, B. Henrissat, A. Levasseur, J. Qian, J. Li, X. Luo, L. Shi, L. He, L. Xiang, X. Xu, Y. Niu, Q. Li, M. V. Han, H. Yan, J. Zhang, H. Chen, A. Lv, Z. Wang, M. Liu, D. C. Schwartz, and C. Sun. 2012. Genome sequence of the model medicinal mushroom Ganoderma lucidum. *Nature Communications*. 3:913.

Cheng, H., G. T. Concepcion, X. Feng, H. Zhang, and H. Li. 2021. Haplotype-resolved *de novo* assembly using phased assembly graphs with hifiasm. *Nature Methods*. 18(2):170-175.

Chin, C.-S., D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler, S. W. Turner, and J. Korlach. 2013. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods*. 10:563-569.

Chin, C.-S. and A. Khalak. 2019. Human Genome Assembly in 100 Minutes. *bioRxiv*. https://www.biorxiv.org/content/10.1101/705616v1.

Chin, C.-S., P. Peluso, F. J. Sedlazeck, M. Nattestad, G. T. Concepcion, A. Clum, C. Dunn, R. O'malley, R. Figueroa-Balderas, A. Morales-Cruz, G. R. Cramer, M. Delledonne, C. Luo, J. R. Ecker, D. Cantu, D. R. Rank, and M. C. Schatz. 2016. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature Methods*. 13(12):1050-1054.

Chow, W., K. Brugger, M. Caccamo, I. Sealy, J. Torrance, and K. Howe. 2016. gEVAL — a web-based browser for evaluating genome assemblies. *Bioinformatics*. 32(16):2508-2510.

Crepeau, M. W., C. H. Langley, and K. A. Stevens. 2017. From Pine Cones to Read Clouds: Rescaffolding the Megagenome of Sugar Pine (Pinus lambertiana). *G3: Genes, Genomes, Genetics*. 7:1563-1568.

Daccord, N., J.-M. Celton, G. Linsmith, C. Becker, N. Choisne, E. Schijlen, H. Van De Geest, L. Bianco, D. Micheletti, R. Velasco, E. A. Di Pierro, J. Gouzy, D. J. G. Rees, P. Guérif, H. Muranty, C.-E. Durel, F. Laurens, Y. Lespinasse, S. Gaillard, S. Aubourg, H. Quesneville, D. Weigel, E. Van De Weg, M. Troggio, and E. Bucher. 2017. High-quality de novo assembly of the apple genome and methylome dynamics of early fruit development. *Nature Genetics*. 49:1099-1106.

Das, S. K., M. D. Austin, M. C. Akana, P. Deshpande, H. Cao, and M. Xiao. 2010. Single molecule linear analysis of DNA in nano-channel labeled with sequence specific fluorescent probes. *Nucleic Acids Research*. 38:1-8.

Dayarian, A., T. P. Michael, and A. M. Sengupta. 2010. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*. 11:345.

Dekker, J., K. Rippe, M. Dekker, and N. Kleckner. 2002. Capturing Chromosome Conformation. *Science*. 295(5558):1306-1311.

Denton, J. F., J. Lugo-Martinez, A. E. Tucker, D. R. Schrider, W. C. Warren, and M. W. Hahn. 2014. Extensive error in the number of genes inferred from draft genome assemblies. *PLoS Computational Biology*. 10(12):e1003998.

Dimalanta, E. T., A. Lim, R. Runnheim, C. Lamers, C. Churas, D. K. Forrest, J. J. De Pablo, M. D. Graham, S. N. Coppersmith, S. Goldstein, and D. C. Schwartz. 2004. A Microfluidic System for Large DNA Molecule Arrays. *Analytical Chemistry*. 76:5293-5301.

Dohm, J. C., C. Lottaz, T. Borodina, and H. Himmelbauer. 2008. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research*. 36:e105-e105.

Doležel, J., J. Bartoš, H. Voglmayr, and J. Greilhuber. 2003. Nuclear DNA Content and Genome Size of Trout and Human. *Cytometry*. 51A(2):127-128.

Dong, Y., M. Xie, Y. Jiang, N. Xiao, X. Du, W. Zhang, G. Tosser-Klopp, J. Wang, S. Yang, J. Liang, W. Chen, J. Chen, P. Zeng, Y. Hou, C. Bian, S. Pan, Y. Li, X. Liu, W. Wang, B. Servin, B. Sayre, B. Zhu, D. Sweeney, R. Moore, W. Nie, Y. Shen, R. Zhao, G. Zhang, J. Li, T. Faraut, J. Womack, Y. Zhang, J. Kijas, N. Cockett, X. Xu, S. Zhao, J. Wang, and W. Wang. 2012. Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (Capra hircus). *Nature Biotechnology*. 31:135-141.

Donmez, N. and M. Brudno. 2013. SCARPA: Scaffolding reads with practical algorithms. *Bioinformatics*. 29:428-434.

Dou, J., H. Dou, C. Mu, L. Zhang, Y. Li, J. Wang, T. Li, Y. Li, X. Hu, S. Wang, and Z. Bao. 2017. Whole-Genome Restriction Mapping by "Subhaploid"-Based RAD Sequencing: An Efficient and Flexible Approach for Physical Mapping and Genome Scaffolding. *Genetics*. 206(3):1237-1250.

Dudchenko, O., S. S. Batra, A. D. Omer, S. K. Nyquist, M. Hoeger, N. C. Durand, M. S. Shamim, I. Machol, E. S. Lander, A. P. Aiden, and E. L. Aiden. 2017. De novo assembly of the *Aedes aegypti* genome using Hi-C yields chromosome-length scaffolds. *Science*. 356(6333):92.

Dunne, M. P. and S. Kelly. 2017. OrthoFiller: utilising data from multiple species to improve the completeness of genome annotations. *BMC Genomics*. 18:390.

Earl, D., K. Bradnam, J. St. John, A. Darling, D. Lin, J. Fass, H. O. K. Yu, V. Buffalo, D. R. Zerbino, M. Diekhans, N. Nguyen, P. N. Ariyaratne, W. K. Sung, Z. Ning, M. Haimel, J. T. Simpson, N. A. Fonseca, I. Birol, T. R. Docking, I. Y. Ho, D. S. Rokhsar, R. Chikhi, D. Lavenier, G. Chapuis, D. Naquin, N. Maillet, M. C. Schatz, D. R. Kelley, A. M. Phillippy, S. Koren, S. P. Yang, W. Wu, W. C. Chou, A. Srivastava, T. I. Shaw, J. G. Ruby, P. Skewes-Cox, M. Betegon, M. T. Dimon, V. Solovyev, I. Seledtsov, P. Kosarev, D. Vorobyev, R. Ramirez-Gonzalez, R. Leggett, D. Maclean, F. Xia, R. Luo, Z. Li, Y. Xie, B. Liu, S. Gnerre, I. Maccallum, D. Przybylski, F. J. Ribeiro, T. Sharpe, G. Hall, P. J. Kersey, R. Durbin, S. D. Jackman, J. A. Chapman, X. Huang, J. L. Derisi, M. Caccamo, Y. Li, D. B. Jaffe, R. E. Green, D. Haussler, I. Korf, and B. Paten. 2011. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research*. 21:2224-2241.

Ekblom, R. and J. B. W. Wolf. 2014. A field guide to whole-genome sequencing, assembly and annotation. *Evolutionary Applications*.n/a-n/a.

El-Metwally, S., T. Hamza, M. Zakaria, and M. Helmy. 2013. Next-Generation Sequence Assembly: Four Stages of Data Processing and Computational Challenges. *PLoS Computational Biology*. 9:e1003345.

English, A. C., S. Richards, Y. Han, M. Wang, V. Vee, J. Qu, X. Qin, D. M. Muzny, J. G. Reid, K. C. Worley, and R. A. Gibbs. 2012. Mind the Gap: Upgrading Genomes with Pacific Biosciences RS Long-Read Sequencing Technology. *PLoS ONE*. 7:1-12.

Farrant, G. K., M. Hoebeke, F. Partensky, G. Andres, E. Corre, and L. Garczarek. 2015. WiseScaffolder: an algorithm for the semi-automatic scaffolding of Next Generation Sequencing data. *BMC Bioinformatics*. 16:281.

Fiddes, I. T., J. Armstrong, M. Diekhans, S. Nachtweide, Z. N. Kronenberg, J. G. Underwood, D. Gordon, D. Earl, T. Keane, E. E. Eichler, D. Haussler, M. Stanke, and B. Paten. 2018. Comparative Annotation Toolkit (CAT)—simultaneous clade and personal genome annotation. *Genome Research*. 28(7):1029-1038.

Fierst, J. L. 2015. Using linkage maps to correct and scaffold de novo genome assemblies: Methods, challenges, and computational tools. *Frontiers in Genetics*. 6:1-8.

Fox, E. J., K. S. Reid-Bayliss, M. J. Emond, and L. A. Loeb. 2014. Accuracy of Next Generation Sequencing Platforms. *Journal of Next Generation Sequencing & Applications*. 1(1):1000106.

Fu, S., A. Wang, and K. F. Au. 2019. A comparative evaluation of hybrid error correction methods for error-prone long reads. *Genome Biology*. 20(1):26.

Ganapathy, G., J. T. Howard, J. M. Ward, J. Li, B. Li, Y. Li, Y. Xiong, Y. Zhang, S. Zhou, D. C. Schwartz, M. Schatz, R. Aboukhalil, O. Fedrigo, L. Bukovnik, T. Wang, G. Wray, I. Rasolonjatovo, R. Winer, J. R. Knight, S. Koren, W. C. Warren, G. Zhang, A. M. Phillippy, and E. D. Jarvis. 2014. High-coverage sequencing and annotated assemblies of the budgerigar genome. *GigaScience*. 3:11.

Gao, S., W.-K. Sung, and N. Nagarajan. 2011. Opera: Reconstructing Optimal Genomic Scaffolds with High-Throughput Paired-End Sequences. 18:1681-1691.

Garalde, D. R., E. A. Snell, D. Jachimowicz, B. Sipos, J. H. Lloyd, M. Bruce, N. Pantic, T. Admassu, P. James, A. Warland, M. Jordan, J. Ciccone, S. Serra, J. Keenan, S. Martin, L. Mcneill, E. J. Wallace, L. Jayasinghe, C. Wright, J. Blasco, S. Young, D. Brocklebank, S. Juul, J. Clarke, A. J. Heron, and D. J. Turner. 2018. Highly parallel direct RNA sequencing on an array of nanopores. *Nature Methods*. 15:201-206.

Gardner, M. J., H. Tettelin, D. J. Carucci, L. M. Cummings, L. Aravind, E. V. Koonin, S. Shallom, T. Mason, K. Yu, C. Fujii, J. Pederson, K. Shen, J. Jing, C. Aston, Z. Lai, D. C. Schwartz, M. Pertea, S. Salzberg, L. Zhou, G. G. Sutton, R. Clayton, O. White, H. O. Smith, C. M. Fraser, M. D. Adams, J. C. Venter, and S. L. Hoffman. 1998. Chromosome 2 sequence of the human malaria parasite Plasmodium falciparum. *Science*. 282:1126-1132.

Garg, S., A. Fungtammasan, A. Carroll, M. Chou, A. Schmitt, X. Zhou, S. Mac, P. Peluso, E. Hatas, J. Ghurye, J. Maguire, M. Mahmoud, H. Cheng, D. Heller, J. M. Zook, T. Moemke, T. Marschall, F. J. Sedlazeck, J. Aach, C.-S. Chin, G. M. Church, and H. Li. 2020. Accurate chromosome-scale haplotype-resolved assembly of human genomes. *bioRxiv*. https://biorxiv.org/content/early/2020/07/01/810341.abstract.

Garrison, E. and G. Marth. 2012. Haplotype-based variant detection from short-read sequencing. *arXiv*. https://arxiv.org/abs/1207.3907.

Ghurye, J., M. Pop, S. Koren, D. Bickhart, and C.-S. Chin. 2017. Scaffolding of long read assemblies using long range contact information. *BMC Genomics*. 18(1):1-11.

Ghurye, J., A. Rhie, B. P. Walenz, A. Schmitt, S. Selvaraj, M. Pop, A. M. Phillippy, and S. Koren. 2019. Integrating Hi-C links with assembly graphs for chromosome-scale assembly. *PLoS Computational Biology*. 15(8):e1007273.

Gill, S. R., M. Pop, R. T. Deboy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson. 2006. Metagenomic Analysis of the Human Distal Gut Microbiome. *Science*. 312(5778):1355.

Giongo, A., H. L. Tyler, U. N. Zipperer, and E. W. Triplett. 2010. Two genome sequences of the same bacterial strain, Gluconacetobacter diazotrophicus PAl 5, suggest a new standard in genome sequence submission. *Standards in Genomic Sciences*. 2:309-317.

Glenn, T. C. 2011. Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*. 11:759-769.

Gnerre, S., I. Maccallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, and D. B. Jaffe. 2011. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*. 108:1513-1518.

Golicz, A. A., P. E. Bayer, G. C. Barker, P. P. Edger, H. Kim, P. A. Martinez, C. K. K. Chan, A. Severn-Ellis, W. R. Mccombie, I. a. P. Parkin, A. H. Paterson, J. C. Pires, A. G. Sharpe, H. Tang, G. R. Teakle, C. D. Town, J. Batley, and D. Edwards. 2016. The pangenome of an agronomically important crop plant Brassica oleracea. *Nature Communications*. 7:13390.

Goodwin, S., J. Gurtowski, S. Ethe-Sayers, P. Deshpande, M. C. Schatz, and W. R. Mccombie. 2015. Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome Research*. 25:1750-1756.

Goodwin, S., J. D. Mcpherson, and W. R. Mccombie. 2016. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*. 17:333-351.

Grabherr, M. G., B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. Di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev. 2011. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology*. 29(7):644-652.

Green, R. E., J. Krause, A. W. Briggs, T. Maricic, U. Stenzel, M. Kircher, N. Patterson, H. Li, W. Zhai, M. H.-Y. Fritz, N. F. Hansen, E. Y. Durand, A.-S. Malaspinas, J. D. Jensen, T.

Marques-Bonet, C. Alkan, K. Prüfer, M. Meyer, H. A. Burbano, J. M. Good, R. Schultz, A. Aximu-Petri, A. Butthof, B. Höber, B. Höffner, M. Siegemund, A. Weihmann, C. Nusbaum, E. S. Lander, C. Russ, N. Novod, J. Affourtit, M. Egholm, C. Verna, P. Rudan, D. Brajkovic, Ž. Kucan, I. Gušic, V. B. Doronichev, L. V. Golovanova, C. Lalueza-Fox, M. De La Rasilla, J. Fortea, A. Rosas, R. W. Schmitz, P. L. F. Johnson, E. E. Eichler, D. Falush, E. Birney, J. C. Mullikin, M. Slatkin, R. Nielsen, J. Kelso, M. Lachmann, D. Reich, and S. Pääbo. 2010. A Draft Sequence of the Neandertal Genome. *Science*. 328(5979):710.

Gregory, T. R. 2021. Animal Genome Size Database. URL: http://www.genomesize.com.

Grice, E. A., H. H. Kong, S. Conlan, C. B. Deming, J. Davis, A. C. Young, G. G. Bouffard, R. W. Blakesley, P. R. Murray, E. D. Green, M. L. Turner, and J. A. Segre. 2009. Topographical and Temporal Diversity of the Human Skin Microbiome. *Science*. 324(5931):1190.

Gritsenko, A. A., J. F. Nijkamp, M. J. T. Reinders, and D. De Ridder. 2012. GRASS: A generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*. 28:1429-1437.

Guan, D., S. A. Mccarthy, J. Wood, K. Howe, Y. Wang, and R. Durbin. 2020. Identifying and removing haplotypic duplication in primary genome assemblies. *Bioinformatics*. 36(9):2896-2898.

Gui, S., J. Peng, X. Wang, Z. Wu, R. Cao, J. Salse, H. Zhang, Z. Zhu, Q. Xia, Z. Quan, L. Shu, W. Ke, and Y. Ding. 2018. Improving Nelumbo nucifera genome assemblies using high-resolution genetic maps and BioNano genome mapping reveals ancient chromosome rearrangements. *The Plant Journal*. 94:721-734.

Haas, B. J., A. Papanicolaou, M. Yassour, M. Grabherr, P. D. Blood, J. Bowden, M. B. Couger, D. Eccles, B. Li, M. Lieber, M. D. Macmanes, M. Ott, J. Orvis, N. Pochet, F. Strozzi, N. Weeks, R. Westerman, T. William, C. N. Dewey, R. Henschel, R. D. Leduc, N. Friedman, and A. Regev. 2013. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature Protocols*. 8:1494-1512.

Hakim, O. and T. Misteli. 2012. SnapShot: Chromosome Conformation Capture. *Cell*. 148:1068-1068.e1062.

Hardie, D. C., T. R. Gregory, and P. D. N. Hebert. 2002. From Pixels to Picograms: A Beginners' Guide to Genome Quantification by Feulgen Image Analysis Densitometry. *Journal of Histochemistry & Cytochemistry*. 50(6):735-749.

Hare, E. E. and J. S. Johnston. 2012. Genome Size Determination Using Flow Cytometry of Propidium Iodide-Stained Nuclei. Pages 3-12 *in* V. Orgogozo and M. V. Rockman, Editors. Molecular Methods for Evolutionary Genetics, 1 Ed. Humana Press, New York City, New York, USA.

Hastie, A. R., L. Dong, A. Smith, J. Finklestein, E. T. Lam, N. Huo, H. Cao, P.-Y. Kwok, K. R. Deal, J. Dvorak, M.-C. Luo, Y. Gu, and M. Xiao. 2013. Rapid Genome Mapping in Nanochannel Arrays for Highly Complete and Accurate De Novo Sequence Assembly of the Complex Aegilops tauschii Genome. *PLoS ONE*. 8:e55864.

Heather, J. M. and B. Chain. 2016. The sequence of sequencers: The history of sequencing DNA. *Genomics*. 107:1-8.

High Performance Assembly Group - Wellcome Sanger Institute. 2019. PretextView v0.0.1.

High Performance Assembly Group - Wellcome Sanger Institute. 2020. PretextMap v0.1.4. URL: https://github.com/wtsi-hpag/PretextMap.

Hirsch, C. N., J. M. Foerster, J. M. Johnson, R. S. Sekhon, G. Muttoni, B. Vaillancourt, F. Penagaricano, E. Lindquist, M. A. Pedraza, K. Barry, N. De Leon, S. M. Kaeppler, and C. R. Buell. 2014. Insights into the Maize Pan-Genome and Pan-Transcriptome. *The Plant Cell*. 26:121-135.

Hoff, K. J., S. Lange, A. Lomsadze, M. Borodovsky, and M. Stanke. 2016. BRAKER1: Unsupervised RNA-Seq-Based Genome Annotation with GeneMark-ET and AUGUSTUS. *Bioinformatics*. 32:767-769.

Holt, C. and M. Yandell. 2011. MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinformatics*. 12:491.

Holt, C. and M. Yandell. 2018. MAKER Tutorial for WGS Assembly and Annotation Winter School 2018. The Yandell Lab, MAKER Wiki. URL: http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER_Tutorial_for_WGS_Assembly_and_Annotation_Winter_School_2018 [accessed 1 March 2018].

Hon, T., K. Mars, G. Young, Y.-C. Tsai, J. W. Karalius, J. M. Landolin, N. Maurer, D. Kudrna, M. A. Hardigan, C. C. Steiner, S. J. Knapp, D. Ware, B. Shapiro, P. Peluso, and D. R. Rank. 2020. Highly accurate long-read HiFi sequencing data for five complex genomes. *Scientific Data*. 7(1):399.

Howe, K., W. Chow, J. Collins, S. Pelan, D.-L. Pointon, Y. Sims, J. Torrance, A. Tracey, and J. Wood. 2021. Significantly improving the quality of genome assemblies through curation. *GigaScience*. 10(1):giaa153.

Howe, K. and J. M. Wood. 2015. Using optical mapping data for the improvement of vertebrate genome assemblies. *GigaScience*. 4:10.

Huang, L., P. Chen, J. Zhuang, Y. Zhang, and S. Walt. 2013. Metabolic Cost, Mechanical Work, and Efficiency During Normal Walking in Obese and Normal-Weight Children. *Research Quarterly for Exercise and Sport*. 84:S72-S79.

Hulse-Kemp, A. M., S. Maheshwari, K. Stoffel, T. A. Hill, D. Jaffe, S. R. Williams, N. Weisenfeld, S. Ramakrishnan, V. Kumar, P. Shah, M. C. Schatz, D. M. Church, and A.

Van Deynze. 2018. Reference quality assembly of the 3.5-Gb genome of Capsicum annuum from a single linked-read library. *Horticulture Research*. 5:4.

Hunt, M., C. Newbold, M. Berriman, and T. D. Otto. 2014. A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*. 15:R42.

Huson, D. H., K. Reinert, and E. W. Myers. 2002. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM*. 49:603-615.

International Human Genome Sequencing Consortium. 2001. Initial sequencing and analysis of the human genome. *Nature*. 409(6822):860-921.

Jackman, S. D., B. P. Vandervalk, H. Mohamadi, J. Chu, S. Yeo, S. A. Hammond, G. Jahesh, H. Khan, L. Coombe, R. L. Warren, and I. Birol. 2017. ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. *Genome Research*. 27(5):768-777.

Jain, M., S. Koren, K. H. Miga, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, S. Malla, H. Marriott, T. Nieto, J. O'grady, H. E. Olsen, B. S. Pedersen, A. Rhie, H. Richardson, A. R. Quinlan, T. P. Snutch, L. Tee, B. Paten, A. M. Phillippy, J. T. Simpson, N. J. Loman, and M. Loose. 2018. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*. 36(4):338-345.

Jansen, H., R. P. Dirks, M. Liem, C. V. Henkel, G. P. H. Van Heusden, R. J. L. F. Lemmers, T. Omer, S. Shao, P. J. Punt, and H. P. Spaink. 2017. De novo whole-genome assembly of a wild type yeast isolate using nanopore sequencing. *F1000 Research*. 6:618.

Jaratlerdsiri, W., E. K. F. Chan, D. C. Petersen, C. Yang, P. I. Croucher, M. S. R. Bornman, P. Sheth, and V. M. Hayes. 2017. Next generation mapping reveals novel large genomic rearrangements in prostate cancer. *Oncotarget*. 8:23588-23602.

Jiao, W.-B., G. G. Accinelli, B. Hartwig, C. Kiefer, D. Baker, E. Severing, E.-M. Willing, M. Piednoel, S. Woetzel, E. Madrid-Herrero, B. Huettel, U. Hümann, R. Reinhard, M. A. Koch, D. Swan, B. Clavijo, G. Coupland, and K. Schneeberger. 2017. Improving and correcting the contiguity of long-read genome assemblies of three plant species using optical mapping and chromosome conformation capture data. *Genome Research*. 27:778-786.

Jing, J., Z. Lai, C. Aston, J. Lin, D. J. Carucci, M. J. Gardner, B. Mishra, T. S. Anantharaman, H. Tettelin, L. M. Cummings, S. L. Hoffman, J. C. Venter, and D. C. Schwartz. 1999. Optical mapping of Plasmodium falciparum chromosome 2. *Genome Research*. 9:175-181.

Kajitani, R., K. Toshimoto, H. Noguchi, A. Toyoda, Y. Ogura, M. Okuno, M. Yabana, M. Harada, E. Nagayasu, H. Maruyama, Y. Kohara, A. Fujiyama, T. Hayashi, and T. Itoh. 2014. Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads. *Genome Research*. 24:1384-1395.

Karlsson, E., A. Lärkeryd, A. Sjödin, M. Forsman, and P. Stenberg. 2015. Scaffolding of a bacterial genome using MinION nanopore sequencing. *Scientific Reports*. 5:11996.

Kelley, D. R., M. C. Schatz, and S. L. Salzberg. 2010. Quake: quality-aware detection and correction of sequencing errors. *Genome Biology*. 11:R116.

Kent, W. J. 2002. BLAT—The BLAST-Like Alignment Tool. *Genome Research*. 12:656-664.

Kent, W. J., C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and A. D. Haussler. 2002. The Human Genome Browser at UCSC. *Genome Research*. 12:996-1006.

Kerpedjiev, P., N. Abdennur, F. Lekschas, C. Mccallum, K. Dinkla, H. Strobelt, J. M. Luber, S. B. Ouellette, A. Azhir, N. Kumar, J. Hwang, S. Lee, B. H. Alver, H. Pfister, L. A. Mirny, P. J. Park, and N. Gehlenborg. 2018. HiGlass: web-based visual exploration and analysis of genome interaction maps. *Genome Biology*. 19(1):125.

Kingan, S. B., Z. N. Kronenberg, and A. M. Wenger. 2020. Beyond Contiguity: Evaluating the Accuracy of *de novo* Genome Assemblies. poster in *Plant and Animal Genome XXVIII Conference*, San Diego, California, USA.

Kingsford, C., M. C. Schatz, and M. Pop. 2010. Assembly complexity of prokaryotic genomes using short reads. *BMC Bioinformatics*. 11:21.

Kokot, M., M. Długosz, and S. Deorowicz. 2017. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*. 33(17):2759-2761.

Kolmogorov, M., B. Raney, B. Paten, and S. Pham. 2014. Ragout - A reference-assisted assembly tool for bacterial genomes. *Bioinformatics*. 30:302-309.

Koren, S. and A. M. Phillippy. 2015. One chromosome, one contig: Complete microbial genomes from long-read sequencing and assembly. *Current Opinion in Microbiology*. 23:110-120.

Koren, S., A. Rhie, B. P. Walenz, A. T. Dilthey, D. M. Bickhart, S. B. Kingan, S. Hiendleder, J. L. Williams, T. P. L. Smith, and A. M. Phillippy. 2018. *De novo* assembly of haplotype-resolved genomes with trio binning. *Nature Biotechnology*. 36(12):1174-1182.

Koren, S., M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. Mccombie, E. D. Jarvis, and A. M. Phillippy. 2012. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature Biotechnology*. 30:693-700.

Koren, S., T. J. Treangen, and M. Pop. 2011. Bambus 2: Scaffolding metagenomes. *Bioinformatics*. 27:2964-2971.

Koren, S., B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy. 2017. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Research*. 27(5):722-736.

Kriventseva, E. V., D. Kuznetsov, F. Tegenfeldt, M. Manni, R. Dias, F. A. Simão, and E. M. Zdobnov. 2019. OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Research*. 47(D1):D807-D811.

Kuleshov, V., C. Jiang, W. Zhou, F. Jahanbani, S. Batzoglou, and M. Snyder. 2015. Synthetic long-read sequencing reveals intraspecies diversity in the human microbiome. *Nature Biotechnology*. 34:64-69.

Kuleshov, V., M. P. Snyder, and S. Batzoglou. 2016. Genome assembly from synthetic long read clouds. *Bioinformatics*. 32:i216-i224.

Laird Smith, M., N. Delaney, N. L. Hepler, D. Alexander, D. Katzenstein, M. Brown, and E. Paxinos. 2016. An Improved Circular Consensus Algorithm with an Application to Detect HIV-1 Drug Resistance Associated Mutations (DRAMs). poster in *ASM Microbe 2016* by American Society for Microbiology, Boston, Massachusettes, USA.

Lajoie, B. R., J. Dekker, and N. Kaplan. 2015. The Hitchhiker's guide to Hi-C analysis: Practical guidelines. *Methods*. 72:65-75.

Lam, E. T., A. Hastie, C. Lin, D. Ehrlich, S. K. Das, M. D. Austin, P. Deshpande, H. Cao, N. Nagarajan, M. Xiao, and P.-Y. Kwok. 2012. Genome mapping on nanochannel arrays for structural variation analysis and sequence assembly. *Nature Biotechnology*. 30:771-776.

Lander, E. S. 2011. Initial impact of the sequencing of the human genome. *Nature*. 470(7333):187-197.

Latreille, P., S. Norton, B. S. Goldman, J. Henkhaus, N. Miller, B. Barbazuk, H. B. Bode, C. Darby, Z. Du, S. Forst, S. Gaudriault, B. Goodner, H. Goodrich-Blair, and S. Slater. 2007. Optical mapping as a routine tool for bacterial genome sequence finishing. *BMC Genomics*. 8:321.

Leuchtenberger, C. 1954. Critical Evaluation of Feulgen Microspectrophotometry for Estimating Amount of DNA in Cell Nuclei. *Science*. 120(3129):1022.

Levy-Sakin, M. and Y. Ebenstein. 2013. Beyond sequencing: Optical mapping of DNA in the age of nanotechnology and nanoscopy. *Current Opinion in Biotechnology*. 24:690-698.

Li, G., L. W. Hillier, R. A. Grahn, A. V. Zimin, V. A. David, M. Menotti-Raymond, R. Middleton, S. Hannah, S. Hendrickson, A. Makunin, S. J. O?Brien, P. Minx, R. K. Wilson, L. A. Lyons, W. C. Warren, and W. J. Murphy. 2016. A High-Resolution SNP Array-Based Linkage Map Anchors a New Domestic Cat Draft Genome Assembly and Provides Detailed Patterns of Recombination. *G3: Genes, Genomes, Genetics*. 6:1607-1616.

Li, H. 2011. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*. 27(21):2987-2993.

Li, H. 2016. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*. 32(14):2103-2110.

Li, H. 2020a. auN: a new metric to measure assembly contiguity in *Heng Li's Blog*. URL: http://lh3.github.io/2020/04/08/a-new-metric-on-assembly-contiguity [accessed 2020 April 08].

Li, H. 2020b. yak: Yet another k-mer analyzer v0.1 (r56). *GitHub*. URL: https://github.com/lh3/yak.

Li, H. 2021. Concepts in phased assemblies in *Heng Li's Blog*. URL: http://lh3.github.io/2021/04/17/concepts-in-phased-assemblies [accessed 2021/04/24].

Li, H., X. Feng, and C. Chu. 2020. The design and construction of reference pangenome graphs. *arXiv*. https://arxiv.org/abs/2003.06079.

Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and G. P. D. P. Subgroup. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*. 25(16):2078-2079.

Li, R., H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. 2010. *De novo* assembly of human genomes with massively parallel short read sequencing. *Genome Research*. 20(2):265-272.

Lieberman-Aiden, E., N. L. Van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. 2009. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science*. 326:289-293.

Lin, J., R. Qi, C. Aston, J. Jing, T. S. Anantharaman, B. Mishra, O. White, M. J. Daly, K. W. Minton, J. C. Venter, and D. C. Schwartz. 1999. Whole-Genome Shotgun Optical Mapping of *Deinococcus radiodurans*. *Science*. 285:1558-1562.

Lindblad-Toh, K., C. M. Wade, T. S. Mikkelsen, E. K. Karlsson, D. B. Jaffe, M. Kamal, M. Clamp, J. L. Chang, E. J. Kulbokas, M. C. Zody, E. Mauceli, X. Xie, M. Breen, R. K. Wayne, E. A. Ostrander, C. P. Ponting, F. Galibert, D. R. Smith, P. J. Dejong, E. Kirkness, P. Alvarez, T. Biagi, W. Brockman, J. Butler, C.-W. Chin, A. Cook, J. Cuff, M. J. Daly, D. Decaprio, S. Gnerre, M. Grabherr, M. Kellis, M. Kleber, C. Bardeleben, L. Goodstadt, A. Heger, C. Hitte, L. Kim, K.-P. Koepfli, H. G. Parker, J. P. Pollinger, S. M. J. Searle, N. B. Sutter, R. Thomas, C. Webber, J. Baldwin, A. Abebe, A. Abouelleil, L. Aftuck, M. Ait-Zahra, T. Aldredge, N. Allen, P. An, S. Anderson, C. Antoine, H. Arachchi, A. Aslam, L. Ayotte, P. Bachantsang, A. Barry, T. Bayul, M. Benamara, A. Berlin, D. Bessette, B. Blitshteyn, T. Bloom, J. Blye, L. Boguslavskiy, C. Bonnet, B.

Boukhgalter, A. Brown, P. Cahill, N. Calixte, J. Camarata, Y. Cheshatsang, J. Chu, M. Citroen, A. Collymore, P. Cooke, T. Dawoe, R. Daza, K. Decktor, S. Degray, N. Dhargay, K. Dooley, K. Dooley, P. Dorje, K. Dorjee, L. Dorris, N. Duffey, A. Dupes, O. Egbiremolen, R. Elong, J. Falk, A. Farina, S. Faro, D. Ferguson, P. Ferreira, S. Fisher, M. Fitzgerald, K. Foley, C. Foley, A. Franke, D. Friedrich, D. Gage, M. Garber, G. Gearin, G. Giannoukos, T. Goode, A. Goyette, J. Graham, E. Grandbois, K. Gyaltsen, N. Hafez, D. Hagopian, B. Hagos, J. Hall, C. Healy, R. Hegarty, T. Honan, A. Horn, N. Houde, L. Hughes, L. Hunnicutt, M. Husby, B. Jester, C. Jones, A. Kamat, B. Kanga, C. Kells, D. Khazanovich, A. C. Kieu, P. Kisner, M. Kumar, K. Lance, T. Landers, M. Lara, W. Lee, J.-P. Leger, N. Lennon, L. Leuper, S. Levine, J. Liu, X. Liu, Y. Lokyitsang, T. Lokyitsang, A. Lui, J. Macdonald, J. Major, R. Marabella, K. Maru, C. Matthews, S. Mcdonough, T. Mehta, J. Meldrim, A. Melnikov, L. Meneus, A. Mihalev, T. Mihova, K. Miller, R. Mittelman, V. Mlenga, L. Mulrain, G. Munson, A. Navidi, J. Naylor, T. Nguyen, N. Nguyen, C. Nguyen, T. Nguyen, R. Nicol, N. Norbu, C. Norbu, N. Novod, T. Nyima, P. Olandt, B. O'neill, K. O'neill, S. Osman, L. Oyono, C. Patti, D. Perrin, P. Phunkhang, F. Pierre, M. Priest, A. Rachupka, S. Raghuraman, R. Rameau, V. Ray, C. Raymond, F. Rege, C. Rise, J. Rogers, P. Rogov, J. Sahalie, S. Settipalli, T. Sharpe, T. Shea, M. Sheehan, N. Sherpa, J. Shi, D. Shih, J. Sloan, C. Smith, T. Sparrow, J. Stalker, N. Stange-Thomann, S. Stavropoulos, C. Stone, S. Stone, S. Sykes, P. Tchuinga, P. Tenzing, S. Tesfaye, D. Thoulutsang, Y. Thoulutsang, K. Topham, I. Topping, T. Tsamla, H. Vassiliev, V. Venkataraman, A. Vo, T. Wangchuk, T. Wangdi, M. Weiand, J. Wilkinson, A. Wilson, S. Yadav, S. Yang, X. Yang, G. Young, Q. Yu, J. Zainoun, L. Zembek, A. Zimmer, E. S. Lander, and M. Broad Sequencing Platform. 2005. Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*. 438(7069):803-819.

Lindsay, J., H. Salooti, I. Măndoiu, and A. Zelikovsky. 2014. ILP-based maximum likelihood genome scaffolding. *BMC Bioinformatics*. 15 Suppl 9:S9.

Liu, L., Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. 2012. Comparison of Next-Generation Sequencing Systems. *Journal of Biomedicine and Biotechnology*. 2012:1-11.

Logsdon, G. A., M. R. Vollger, and E. E. Eichler. 2020a. Long-read human genome sequencing and its applications. *Nature Reviews Genetics*. 21(10):597-614.

Logsdon, G. A., M. R. Vollger, P. Hsieh, Y. Mao, M. A. Liskovykh, S. Koren, S. Nurk, L. Mercuri, P. C. Dishuck, A. Rhie, L. G. De Lima, D. Porubsky, A. V. Bzikadze, M. Kremitzki, T. A. Graves-Lindsay, C. Jain, K. Hoekzema, S. C. Murali, K. M. Munson, C. Baker, M. Sorensen, A. M. Lewis, U. Surti, J. L. Gerton, V. Larionov, M. Ventura, K. H. Miga, A. M. Phillippy, and E. E. Eichler. 2020b. The structure, function, and evolution of a complete human chromosome 8. *bioRxiv*. https://www.biorxiv.org/content/10.1101/2020.09.08.285395v1.

Loman, N. J., R. V. Misra, T. J. Dallman, C. Constantinidou, S. E. Gharbia, J. Wain, and M. J. Pallen. 2012. Performance comparison of benchtop high-throughput sequencing platforms. *Nature Biotechnology*. 30:562-562.

Loman, N. J., J. Quick, and J. T. Simpson. 2015. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature Methods*. 12(8):733-735.

Luo, J., J. Wang, Z. Zhang, M. Li, and F.-X. Wu. 2017. BOSS: a novel scaffolding algorithm based on an optimized scaffold graph. *Bioinformatics*. 33:169-176.

Luo, R., B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, J. Tang, G. Wu, H. Zhang, Y. Shi, Y. Liu, C. Yu, B. Wang, Y. Lu, C. Han, D. W. Cheung, S.-M. Yiu, S. Peng, Z. Xiaoqian, G. Liu, X. Liao, Y. Li, H. Yang, J. Wang, T.-W. Lam, and J. Wang. 2012. SOAPdenovo2: an empirically improved memory-efficient short-read *de novo* assembler. *GigaScience*. 1(1):18.

Madoui, M.-A., S. Engelen, C. Cruaud, C. Belser, L. Bertrand, A. Alberti, A. Lemainque, P. Wincker, and J.-M. Aury. 2015. Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC Genomics*. 16:327.

Mak, A. C. Y., Y. Y. Y. Lai, E. T. Lam, T.-P. Kwok, A. K. Y. Leung, A. Poon, Y. Mostovoy, A. R. Hastie, W. Stedman, T. Anantharaman, W. Andrews, X. Zhou, A. W. C. Pang, H. Dai, C. Chu, C. Lin, J. J. K. Wu, C. M. L. Li, J.-W. Li, A. K. Y. Yim, S. Chan, J. Sibert, E. D Akula, H. Cao, S.-M. Yiu, T.-F. Chan, K. Y. Yip, M. Xiao, and P.-Y. Kwok. 2016. Genome-Wide Structural Variation Detection by Genome Mapping on Nanochannel Arrays. *Genetics*. 202:351-362.

Mandric, I. and A. Zelikovsky. 2015. ScaffMatch: Scaffolding algorithm based on maximum weight matching. *Bioinformatics*. 31:2632-2638.

Marcais, G. and C. Kingsford. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*. 27(6):764-770.

Mardis, E. R. 2011. A decade's perspective on DNA sequencing technology. *Nature*. 470(7333):198-203.

Markelz, R. J. C., M. F. Covington, M. T. Brock, U. K. Devisetty, D. J. Kliebenstein, C. Weinig, and J. N. Maloof. 2017. Using RNA-seq for Genomic Scaffold Placement, Correcting Assemblies, and Genetic Map Creation in a Common Brassica rapa Mapping Population. *G3: Genes, Genomes, Genetics*.g3.117.043000.

Martin, M. 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*. 17(1):10-12.

Mascher, M., H. Gundlach, A. Himmelbach, S. Beier, S. O. Twardziok, T. Wicker, V. Radchuk, C. Dockter, P. E. Hedley, J. Russell, M. Bayer, L. Ramsay, H. Liu, G. Haberer, X.-Q. Zhang, Q. Zhang, R. A. Barrero, L. Li, S. Taudien, M. Groth, M. Felder, A. Hastie, H. Šimková, H. Staňková, J. Vrána, S. Chan, M. Muñoz-Amatriaín, R. Ounit, S. Wanamaker, D. Bolser, C. Colmsee, T. Schmutzer, L. Aliyeva-Schnorr, S. Grasso, J. Tanskanen, A. Chailyan, D. Sampath, D. Heavens, L. Clissold, S. Cao, B. Chapman, F. Dai, Y. Han, H. Li, X. Li, C. Lin, J. K. Mccooke, C. Tan, P. Wang, S. Wang, S. Yin, G. Zhou, J. A. Poland, M. I. Bellgard, L. Borisjuk, A. Houben, J. Doležel, S. Ayling, S.

Lonardi, P. Kersey, P. Langridge, G. J. Muehlbauer, M. D. Clark, M. Caccamo, A. H. Schulman, K. F. X. Mayer, M. Platzer, T. J. Close, U. Scholz, M. Hansson, G. Zhang, I. Braumann, M. Spannagl, C. Li, R. Waugh, and N. Stein. 2017. A chromosome conformation capture ordered sequence of the barley genome. *Nature*. 544:427-433.

Mccoy, R. C., R. W. Taylor, T. A. Blauwkamp, J. L. Kelley, M. Kertesz, D. Pushkarev, D. A. Petrov, and A.-S. Fiston-Lavier. 2014. Illumina TruSeq Synthetic Long-Reads Empower De Novo Assembly and Resolve Complex, Highly-Repetitive Transposable Elements. *PLoS ONE*. 9:e106689.

Meier, J. I., P. A. Salazar, M. Kučka, R. W. Davies, A. Dréau, I. Aldás, O. B. Power, N. J. Nadeau, J. R. Bridle, C. Rolian, N. H. Barton, W. O. Mcmillan, C. D. Jiggins, and Y. F. Chan. 2020. Haplotype tagging reveals parallel formation of hybrid races in two butterfly species. *bioRxiv*. https://www.biorxiv.org/content/10.1101/2020.05.25.113688v2.

Michaeli, Y. and Y. Ebenstein. 2012. Channeling DNA for optical mapping. *Nature Biotechnology*. 30:762-763.

Miga, K. H., S. Koren, A. Rhie, M. R. Vollger, A. Gershman, A. Bzikadze, S. Brooks, E. Howe, D. Porubsky, G. A. Logsdon, V. A. Schneider, T. Potapova, J. Wood, W. Chow, J. Armstrong, J. Fredrickson, E. Pak, K. Tigyi, M. Kremitzki, C. Markovic, V. Maduro, A. Dutra, G. G. Bouffard, A. M. Chang, N. F. Hansen, A. B. Wilfert, F. Thibaud-Nissen, A. D. Schmitt, J.-M. Belton, S. Selvaraj, M. Y. Dennis, D. C. Soto, R. Sahasrabudhe, G. Kaya, J. Quick, N. J. Loman, N. Holmes, M. Loose, U. Surti, R. A. Risques, T. A. Graves Lindsay, R. Fulton, I. Hall, B. Paten, K. Howe, W. Timp, A. Young, J. C. Mullikin, P. A. Pevzner, J. L. Gerton, B. A. Sullivan, E. E. Eichler, and A. M. Phillippy. 2020. Telomere-to-telomere assembly of a complete human X chromosome. *Nature*. 585(7823):79-84.

Mikkelsen, T. S., M. J. Wakefield, B. Aken, C. T. Amemiya, J. L. Chang, S. Duke, M. Garber, A. J. Gentles, L. Goodstadt, A. Heger, J. Jurka, M. Kamal, E. Mauceli, S. M. J. Searle, T. Sharpe, M. L. Baker, M. A. Batzer, P. V. Benos, K. Belov, M. Clamp, A. Cook, J. Cuff, R. Das, L. Davidow, J. E. Deakin, M. J. Fazzari, J. L. Glass, M. Grabherr, J. M. Greally, W. Gu, T. A. Hore, G. A. Huttley, M. Kleber, R. L. Jirtle, E. Koina, J. T. Lee, S. Mahony, M. A. Marra, R. D. Miller, R. D. Nicholls, M. Oda, A. T. Papenfuss, Z. E. Parra, D. D. Pollock, D. A. Ray, J. E. Schein, T. P. Speed, K. Thompson, J. L. Vandeberg, C. M. Wade, J. A. Walker, P. D. Waters, C. Webber, J. R. Weidman, X. Xie, M. C. Zody, J. Baldwin, A. Abdouelleil, J. Abdulkadir, A. Abebe, B. Abera, J. Abreu, S. C. Acer, L. Aftuck, A. Alexander, P. An, E. Anderson, S. Anderson, H. Arachi, M. Azer, P. Bachantsang, A. Barry, T. Bayul, A. Berlin, D. Bessette, T. Bloom, J. Blye, L. Boguslavskiy, C. Bonnet, B. Boukhgalter, I. Bourzgui, A. Brown, P. Cahill, S. Channer, Y. Cheshatsang, L. Chuda, M. Citroen, A. Collymore, P. Cooke, M. Costello, K. D'aco, R. Daza, G. De Haan, S. Degray, C. Demaso, N. Dhargay, K. Dooley, E. Dooley, M. Doricent, P. Dorje, K. Dorjee, A. Dupes, R. Elong, J. Falk, A. Farina, S. Faro, D. Ferguson, S. Fisher, C. D. Foley, A. Franke, D. Friedrich, L. Gadbois, G. Gearin, C. R. Gearin, G. Giannoukos, T. Goode, J. Graham, E. Grandbois, S. Grewal, K. Gyaltsen, N. Hafez, B. Hagos, J. Hall, C. Henson, A. Hollinger, T. Honan, M. D. Huard, L. Hughes, B. Hurhula, M. E. Husby, A. Kamat, B. Kanga, S. Kashin, D. Khazanovich, P. Kisner, K.

Lance, M. Lara, W. Lee, N. Lennon, F. Letendre, R. Levine, A. Lipovsky, X. Liu, J. Liu, S. Liu, T. Lokyitsang, Y. Lokyitsang, R. Lubonja, A. Lui, P. Macdonald, V. Magnisalis, K. Maru, C. Matthews, W. Mccusker, S. Mcdonough, T. Mehta, J. Meldrim, L. Meneus, O. Mihai, A. Mihalev, T. Mihova, R. Mittelman, V. Mlenga, A. Montmayeur, L. Mulrain, A. Navidi, J. Naylor, T. Negash, T. Nguyen, N. Nguyen, R. Nicol, C. Norbu, N. Norbu, N. Novod, B. O'neill, S. Osman, E. Markiewicz, O. L. Oyono, C. Patti, P. Phunkhang, F. Pierre, M. Priest, S. Raghuraman, F. Rege, R. Reyes, C. Rise, P. Rogov, K. Ross, E. Ryan, S. Settipalli, T. Shea, N. Sherpa, L. Shi, D. Shih, T. Sparrow, J. Spaulding, J. Stalker, N. Stange-Thomann, S. Stavropoulos, C. Stone, C. Strader, S. Tesfaye, T. Thomson, Y. Thoulutsang, D. Thoulutsang, K. Topham, I. Topping, T. Tsamla, H. Vassiliev, A. Vo, T. Wangchuk, T. Wangdi, M. Weiand, J. Wilkinson, A. Wilson, S. Yadav, G. Young, Q. Yu, L. Zembek, D. Zhong, A. Zimmer, Z. Zwirko, D. B. Jaffe, P. Alvarez, W. Brockman, J. Butler, C. Chin, S. Gnerre, I. Maccallum, J. a. M. Graves, C. P. Ponting, M. Breen, P. B. Samollow, E. S. Lander, K. Lindblad-Toh, P. Broad Institute Genome Sequencing, and T. Broad Institute Whole Genome Assembly. 2007. Genome of the marsupial *Monodelphis domestica* reveals innovation in non-coding sequences. *Nature*. 447(7141):167-177.

Morisse, P., C. Lamaitre, and F. Legeai. 2021a. LRez: C++ API and tookit for analyzing and managing Linked-Reads data. *arXiv*. https://arxiv.org/abs/2103.14419.

Morisse, P., T. Lecroq, and A. Lefebvre. 2020. Long-read error correction: a survey and qualitative comparison. *bioRxiv*. https://www.biorxiv.org/content/10.1101/2020.03.06.977975v2.

Morisse, P., C. Marchet, A. Limasset, T. Lecroq, and A. Lefebvre. 2021b. Scalable long read self-correction and assembly polishing with multiple sequence alignment. *Scientific Reports*. 11(1):761.

Mortazavi, A., E. M. Schwarz, B. Williams, L. Schaeffer, I. Antoshechkin, B. J. Wold, and P. W. Sternberg. 2010. Scaffolding a Caenorhabditis nematode genome with RNA-seq. *Genome Research*. 20:1740-1747.

Mortazavi, A., B. A. Williams, K. Mccue, L. Schaeffer, and B. Wold. 2008. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*. 5:621-628.

Mostovoy, Y., M. Levy-Sakin, J. Lam, E. T. Lam, A. R. Hastie, P. Marks, J. Lee, C. Chu, C. Lin, Ž. Džakula, H. Cao, S. A. Schlebusch, K. Giorda, M. Schnall-Levin, J. D. Wall, and P.-Y. Kwok. 2016. A hybrid approach for de novo human genome sequence assembly and phasing. *Nature Methods*. 13:12-17.

Mouse Genome Sequencing Consortium. 2002. Initial sequencing and comparative analysis of the mouse genome. *Nature*. 420(6915):520-562.

Mudge, J. M. and J. Harrow. 2016. The state of play in higher eukaryote gene annotation. *Nature Reviews Genetics*. 17:758-772.

Mulyukov, Z. and P. A. Pevzner. 2002. EULER-PCR: finishing experiments for repeat resolution. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*. 7:199-210.

Myers, E. W., G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. J. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H.-H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter. 2000. A Whole-Genome Assembly of *Drosophila*. *Science*. 287(5461):2196-2204.

Nagarajan, N. and M. Pop. 2009. Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing. *Journal of Computational Biology*. 16:897-908.

Nature Editors. 2010. The human genome at ten. *Nature*. 464(7289):649-650.

Neely, R. K., P. Dedecker, J.-I. Hotta, G. Urbanavičiūtė, S. Klimašauskas, and J. Hofkens. 2010. DNA fluorocode: A single molecule, optical map of DNA with nanometre resolution. *Chemical Science*. 1:453.

Neely, R. K., J. Deen, and J. Hofkens. 2011. Optical mapping of DNA: Single-molecule-based methods for mapping genomes. *Biopolymers*. 95:298-311.

Neto, M., G. Skorski, D. Thevenot, and E. Loukiadis. 2011. Optical maps : methodology and applications in microbiology. *Euro Reference*. 5:38-46.

Nowoshilow, S., S. Schloissnig, J.-F. Fei, A. Dahl, A. W. C. Pang, M. Pippel, S. Winkler, A. R. Hastie, G. Young, J. G. Roscito, F. Falcon, D. Knapp, S. Powell, A. Cruz, H. Cao, B. Habermann, M. Hiller, E. M. Tanaka, and E. W. Myers. 2018. The axolotl genome and the evolution of key tissue formation regulators. *Nature*. 554:50-55.

Nurk, S., A. Bankevich, D. Antipov, A. Gurevich, A. Korobeynikov, and Lapidus. 2013. Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads. Research in Computational Molecular Biology. Beijing, China: Springer Berlin Heidelberg, p 158-170.

Nurk, S., B. P. Walenz, A. Rhie, M. R. Vollger, G. A. Logsdon, R. Grothe, K. H. Miga, E. E. Eichler, A. M. Phillippy, and S. Koren. 2020. HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *Genome Research*.

O'rourke, J. A. 2014. Genetic and Physical Map Correlation. Pages 1-7. eLS. John Wiley & Sons, Ltd, Chichester, UK.

O'bleness, M., V. B. Searles, C. Dickens, D. Astling, D. Albracht, A. C. Y. Mak, Y. Y. Y. Lai, C. Lin, C. Chu, T. Graves, P.-Y. Kwok, R. K. Wilson, and J. M. Sikela. 2014. Finished

sequence and assembly of the DUF1220-rich 1q21 region using a haploid human genome. *BMC Genomics*. 15:387.

Pacific Biosciences. 2020. Beyond Contiguity – Assessing the Quality of Genome Assemblies with the 3 C's in *Pacific Biosciences Blog*. URL: https://www.pacb.com/blog/beyond-contiguity [accessed 2021/04/23].

Parker, D., A. Narechania, R. Sebra, G. Deikus, S. Larussa, C. Ryan, H. Smith, A. Prince, B. Mathema, A. J. Ratner, B. Kreiswirth, and P. J. Planet. 2014. Genome Sequence of Bacterial Interference Strain Staphylococcus aureus 502A. *Genome Announcements*. 2:e00284-00214-e00284-00214.

Peter, Matthew, Mark, and J. Yang. 2012. Five Years of GWAS Discovery. *The American Journal of Human Genetics*. 90(1):7-24.

Pettersson, E., J. Lundeberg, and A. Ahmadian. 2009. Generations of sequencing technologies. *Genomics*. 93:105-111.

Pevzner, P. A., H. Tang, and M. S. Waterman. 2001. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*. 98:9748-9753.

Pickett, B. D., J. R. Glass, P. G. Ridge, and J. S. K. Kauwe. 2021. *De novo* genome assembly of the marine teleost, Bluefin Trevally (*Caranx melampygus*). *G3: Genes, Genomes, Genetics*.

Pinoli, P. a. C. D. a. M. M. a. a. M. a. B. C. a. a. B. J. a. a. B. D. a. B. H. a. C. J. 2015. Computational algorithms to predict Gene Ontology annotations. *BMC Bioinformatics*. 16(Suppl 6):S4.

Pirovano, W., M. Boetzer, M. F. L. Derks, and S. Smit. 2015. NCBI-compliant genome submissions: tips and tricks to save time and money: Table 1. *Briefings in Bioinformatics*. 18:bbv104.

Pop, M., D. S. Kosack, and S. L. Salzberg. 2004. Hierarchical scaffolding with Bambus. *Genome Research*. 14:149-159.

Proux-Wéra, E., D. Armisén, K. P. Byrne, and K. H. Wolfe. 2012. A pipeline for automated annotation of yeast genome sequences by a conserved-synteny approach. *BMC Bioinformatics*. 13:237.

Putnam, N. H., B. L. O'connell, J. C. Stites, B. J. Rice, M. Blanchette, R. Calef, C. J. Troll, A. Fields, P. D. Hartley, C. W. Sugnet, D. Haussler, D. S. Rokhsar, and R. E. Green. 2016. Chromosome-scale shotgun assembly using an in vitro method for long-range linkage. *Genome Research*. 26:342-350.

Quail, M., M. E. Smith, P. Coupland, T. D. Otto, S. R. Harris, T. R. Connor, A. Bertoni, H. P. Swerdlow, and Y. Gu. 2012. A tale of three next generation sequencing platforms:

comparison of Ion torrent, pacific biosciences and illumina MiSeq sequencers. *BMC Genomics*. 13:341.

Quick, J., N. J. Loman, S. Duraffour, J. T. Simpson, E. Severi, L. Cowley, J. A. Bore, R. Koundouno, G. Dudas, A. Mikhail, N. Ouédraogo, B. Afrough, A. Bah, J. H. J. Baum, B. Becker-Ziaja, J. P. Boettcher, M. Cabeza-Cabrerizo, Á. Camino-Sánchez, L. L. Carter, J. Doerrbecker, T. Enkirch, I. G. Dorival, N. Hetzelt, J. Hinzmann, T. Holm, L. E. Kafetzopoulou, M. Koropogui, A. Kosgey, E. Kuisma, C. H. Logue, A. Mazzarelli, S. Meisel, M. Mertens, J. Michel, D. Ngabo, K. Nitzsche, E. Pallasch, L. V. Patrono, J. Portmann, J. G. Repits, N. Y. Rickett, A. Sachse, K. Singethan, I. Vitoriano, R. L. Yemanaberhan, E. G. Zekeng, T. Racine, A. Bello, A. A. Sall, O. Faye, O. Faye, N. F. Magassouba, C. V. Williams, V. Amburgey, L. Winona, E. Davis, J. Gerlach, F. Washington, V. Monteil, M. Jourdain, M. Bererd, A. Camara, H. Somlare, A. Camara, M. Gerard, G. Bado, B. Baillet, D. Delaune, K. Y. Nebie, A. Diarra, Y. Savane, R. B. Pallawo, G. J. Gutierrez, N. Milhano, I. Roger, C. J. Williams, F. Yattara, K. Lewandowski, J. Taylor, P. Rachwal, D. J. Turner, G. Pollakis, J. A. Hiscox, D. A. Matthews, M. K. O. Shea, A. M. Johnston, D. Wilson, E. Hutley, E. Smit, A. Di Caro, R. Wölfel, K. Stoecker, E. Fleischmann, M. Gabriel, S. A. Weller, L. Koivogui, B. Diallo, S. Keïta, A. Rambaut, P. Formenty, S. Günther, and M. W. Carroll. 2016. Real-time, portable genome sequencing for Ebola surveillance. *Nature*. 530(7589):228-232.

Quinlan, A. R. and I. M. Hall. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*. 26(6):841-842.

Rahman, A. and L. Pachter. 2016. SWALO: scaffolding with assembly likelihood optimization. *bioRxiv*.1-17.

Räihä, K.-J. and E. Ukkonen. 1981. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*. 16:187-198.

Rajaraman, A., E. Tannier, and C. Chauve. 2013. FPSAC: Fast phylogenetic scaffolding of ancient contigs. *Bioinformatics*. 29:2987-2994.

Ramani, V., D. A. Cusanovich, R. J. Hause, W. Ma, R. Qiu, X. Deng, C. A. Blau, C. M. Disteche, W. S. Noble, J. Shendure, and Z. Duan. 2016. Mapping 3D genome architecture through in situ DNase Hi-C. *Nature Protocols*. 11(11):2104-2121.

Ranallo-Benavidez, T. R., K. S. Jaron, and M. C. Schatz. 2020. GenomeScope 2.0 and Smudgeplot for reference-free profiling of polyploid genomes. *Nature Communications*. 11(1):1432.

Rat Genome Sequencing Project Consortium. 2004. Genome sequence of the Brown Norway rat yields insights into mammalian evolution. *Nature*. 428(6982):493-521.

Redin, D., E. Borgström, M. He, H. Aghelpasand, M. Käller, and A. Ahmadian. 2017. Droplet Barcode Sequencing for targeted linked-read haplotyping of single DNA molecules. *Nucleic Acids Research*. 45(13):e125.

Rhie, A., B. P. Walenz, S. Koren, and A. M. Phillippy. 2020. Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *Genome Biology*. 21(1):245.

Rhoads, A. and K. F. Au. 2015. PacBio Sequencing and Its Applications. *Genomics, Proteomics & Bioinformatics*. 13:278-289.

Riba-Grognuz, O., L. Keller, L. Falquet, I. Xenarios, and Y. Wurm. 2011. Visualization and quality assessment of de novo genome assemblies. *Bioinformatics*. 27:3425-3426.

Ribeiro, F. J., D. Przybylski, S. Yin, T. Sharpe, S. Gnerre, A. Abouelleil, A. M. Berlin, A. Montmayeur, T. P. Shea, B. J. Walker, S. K. Young, C. Russ, C. Nusbaum, I. Maccallum, and D. B. Jaffe. 2012. Finished bacterial genomes from shotgun sequence data. *Genome Research*. 22:2270-2277.

Roach, M. J., S. A. Schmidt, and A. R. Borneman. 2018. Purge Haplotigs: allelic contig reassignment for third-gen diploid genome assemblies. *BMC Bioinformatics*. 19(1):460.

Robinson, J. T., D. Turner, N. C. Durand, H. Thorvaldsdóttir, J. P. Mesirov, and E. L. Aiden. 2018. Juicebox.js Provides a Cloud-Based Visualization System for Hi-C Data. *Cell Systems*. 6(2):256-258.

Ruan, J. and H. Li. 2019. Fast and accurate long-read assembly with wtdbg2. *bioRxiv*. https://www.biorxiv.org/content/10.1101/530972v1.

Ruan, J. and H. Li. 2020. Fast and accurate long-read assembly with wtdbg2. *Nature Methods*. 17(2):155-158.

Sahlin, K., F. Vezzi, B. Nystedt, J. Lundeberg, and L. Arvestad. 2014. BESST--efficient scaffolding of large fragmented assemblies. *BMC Bioinformatics*. 15:281.

Salmela, L., V. Mäkinen, N. Välimäki, J. Ylinen, and E. Ukkonen. 2011. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*. 27:3259-3265.

Sanger, F. 1975. The Croonian Lecture, 1975: Nucleotide Sequences in DNA. *Proceedings of the Royal Society B: Biological Sciences*. 191:317-333.

Sanger, F. and A. R. Coulson. 1975. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*. 94:441-448.

Sanger, F., S. Nicklen, and A. R. Coulson. 1977. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*. 74:5463-5467.

Schatz, M. C., A. L. Delcher, and S. L. Salzberg. 2010. Assembly of large genomes using second-generation sequencing. *Genome Research*. 20:1165-1173.

Schneeberger, K., S. Ossowski, F. Ott, J. D. Klein, X. Wang, C. Lanz, L. M. Smith, J. Cao, J. Fitz, N. Warthmann, S. R. Henz, D. H. Huson, and D. Weigel. 2011. Reference-guided

assembly of four diverse Arabidopsis thaliana genomes. *Proceedings of the National Academy of Sciences*. 108:10249-10254.

Sedlazeck, F. J., H. Lee, C. A. Darby, and M. C. Schatz. 2018. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*. 19:329-346.

Sharon, I., M. Kertesz, L. A. Hug, D. Pushkarev, T. A. Blauwkamp, C. J. Castelle, M. Amirebrahimi, B. C. Thomas, D. Burstein, S. G. Tringe, K. H. Williams, and J. F. Banfield. 2015. Accurate, multi-kb reads resolve complex populations and detect rare microorganisms. *Genome Research*. 25:534-543.

Shendure, J., S. Balasubramanian, G. M. Church, W. Gilbert, J. Rogers, J. A. Schloss, and R. H. Waterston. 2017. DNA sequencing at 40: past, present and future. *Nature*. 550:345-353.

Shumate, A. and S. L. Salzberg. 2020. Liftoff: accurate mapping of gene annotations. *Bioinformatics*.

Silva, G. G., B. E. Dutilh, T. Matthews, K. Elkins, R. Schmieder, E. A. Dinsdale, R. A. Edwards, M. Imelfort, D. Edwards, M. Boetzer, C. Henkel, H. Jansen, D. Butler, W. Pirovano, R. Edwards, G. Olsen, S. Maloy, B. Chevreux, T. Pfisterer, B. Drescher, A. Driesel, W. Müller, T. Wetter, S. Suhai, M. Pop, A. Phillippy, A. Delcher, S. Salzberg, S. Gnerre, E. Lander, K. Lindblad-Toh, D. Jaffe, Y. Boucher, O. Cordero, A. Takemura, D. Hunt, K. Schliep, E. Bapteste, P. Lopez, C. Tarr, M. Polz, T. Matthews, R. Edwards, S. Maloy, T. Matthews, S. Maloy, S. Gao, W.-K. Sung, N. Nagarajan, M. Barton, H. Barton, M. Galardini, E. Biondi, M. Bazzicalupo, A. Mengoni, S. V. Hijum, A. Zomer, O. Kuipers, J. Kok, S. Assefa, T. Keane, T. Otto, C. Newbold, M. Berriman, F. Vezzi, F. Cattonaro, A. Policriti, M. Margulies, M. Egholm, W. Altman, S. Attiya, J. Bader, L. Bemben, J. Berka, M. Braverman, Y.-J. Chen, Z. Chen, S. Dewell, L. Du, J. Fierro, X. Gomes, B. Godwin, W. He, S. Helgesen, C. Ho, C. Ho, G. Irzyk, S. Jando, M. Alenquer, T. Jarvie, K. Jirage, J.-B. Kim, J. Knight, J. Lanza, J. Leamon, S. Lefkowitz, M. Lei, R. Overbeek, T. Begley, R. Butler, J. Choudhuri, H.-Y. Chuang, M. Cohoon, V. D. Crécy-Lagard, N. Diaz, T. Disz, R. Edwards, M. Fonstein, E. Frank, S. Gerdes, E. Glass, A. Goesmann, A. Hanson, D. Iwata-Reuyl, R. Jensen, N. Jamshidi, L. Krause, M. Kubal, N. Larsen, B. Linke, A. Mchardy, F. Meyer, H. Neuweger, G. Olsen, R. Olson, A. Osterman, V. Portnoy, K. Mcelroy, F. Luciani, T. Thomas, S. Kurtz, A. Phillippy, A. Delcher, M. Smoot, M. Shumway, C. Antonescu, S. Salzberg, S. Needleman, C. Wunsch, S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman, R. Edwards, J. Haggerty, N. Cassman, J. Busch, K. Aguinaldo, S. Chinta, M. Vaughn, R. Morey, T. Harkins, C. Teiling, K. Fredrikson, E. Dinsdale, E. Schadt, S. Turner, A. Kasarskis, S. V. Hijum, A. Zomer, O. Kuipers, J. Kok, R. Helm, S. Maloy, A. Darling, B. Mau, and N. Perna. 2013. Combining de novo and reference-guided assembly with scaffold_builder. *Source Code for Biology and Medicine*. 8:23.

Simão, F. A., R. M. Waterhouse, P. Ioannidis, E. V. Kriventseva, and E. M. Zdobnov. 2015. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*. 31(19):3210-3212.

Simpson, J. T., R. Durbin, D. R. Zerbino, E. Birney, K. Wong, and S. D. Jackman. 2012. Efficient de novo assembly of large genomes using compressed data structures sequence data. *Genome Research*.549-556.

Simpson, J. T. and M. Pop. 2015. The Theory and Practice of Genome Sequence Assembly. *Annual Review of Genomics and Human Genetics*. 16:153-172.

Simpson, J. T., K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. 2009. ABySS: A parallel assembler for short read sequence data. *Genome Research*. 19(6):1117-1123.

Simpson, J. T., R. E. Workman, P. C. Zuzarte, M. David, L. J. Dursi, and W. Timp. 2017. Detecting DNA cytosine methylation using nanopore sequencing. *Nature Methods*. 14(4):407-410.

Song, L. and L. Florea. 2015. Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads. *GigaScience*. 4(48)

Song, L., D. S. Shankar, and L. Florea. 2016. Rascaf: Improving Genome Assembly with RNA Sequencing Data. *The Plant Genome*. 9(3):1-12.

Staden, R. 1979. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research*. 6:2601-2610.

Staňková, H., A. R. Hastie, S. Chan, J. Vrána, Z. Tulpová, M. Kubaláková, P. Visendi, S. Hayashi, M. Luo, J. Batley, D. Edwards, J. Doležel, and H. Šimková. 2016. BioNano genome mapping of individual chromosomes supports physical mapping and sequence assembly in complex plant genomes. *Plant Biotechnology Journal*. 14:1523-1531.

Stein, L. D. 2010. The case for cloud computing in genome informatics. *Genome Biology*. 11(5):207.

Teague, B., M. S. Waterman, S. Goldstein, K. Potamousis, S. Zhou, S. Reslewic, D. Sarkar, A. Valouev, C. Churas, J. M. Kidd, S. Kohn, R. Runnheim, C. Lamers, D. Forrest, M. A. Newton, E. E. Eichler, M. Kent-First, U. Surti, M. Livny, and D. C. Schwartz. 2010. High-resolution human genome structure by single-molecule analysis. *Proceedings of the National Academy of Sciences*. 107:10848-10853.

Teh, B. T., K. Lim, C. H. Yong, C. C. Y. Ng, S. R. Rao, V. Rajasegaran, W. K. Lim, C. K. Ong, K. Chan, V. K. Y. Cheng, P. S. Soh, S. Swarup, S. G. Rozen, N. Nagarajan, and P. Tan. 2017. The draft genome of tropical fruit durian (*Durio zibethinus*). *Nature Genetics*. 49:1633-1641.

Terabayashi, Y., A. Juan, H. Tamotsu, N. Ashimine, K. Nakano, M. Shimoji, A. Shiroma, K. Teruya, K. Satou, and T. Hirano. 2014. First Complete Genome Sequence of Salmonella enterica subsp. enterica Serovar Typhimurium Strain ATCC 13311 (NCTC 74), a Reference Strain of Multidrug Resistance, as Achieved by Use of PacBio Single-Molecule Real-Time Technology. *Genome Announcements*. 2:e00986-00914-e00986-00914.

The 1000 Genomes Project Consortium. 2015. A global reference for human genetic variation. *Nature*. 526(7571):68-74.

The Chimpanzee Sequencing and Analysis Consortium. 2005. Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*. 437(7055):69-87.

The International Hapmap Consortium. 2010. Integrating common and rare genetic variation in diverse human populations. *Nature*. 467(7311):52-58.

The Wellcome Trust Case Control Consortium. 2007. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*. 447(7145):661-678.

Thibaud-Nissen, F., A. Souvorov, T. Murphy, M. Dicuccio, and P. Kitts. 2013. Eukaryotic Genome Annotation Pipeline. Pages 135-158 *in* The NCBI Handbook, 2 Ed. National Center for Biotechnology Information, Bethesda, MD.

Tsai, K. J., M.-Y. J. Lu, K.-J. Yang, M. Li, Y. Teng, S. Chen, M. S. B. Ku, and W.-H. Li. 2016. Assembling the Setaria italica L. Beauv. genome into nine chromosomes and insights into regions affecting growth and drought tolerance. *Scientific Reports*. 6:35076.

Urban, J. M., J. Bliss, C. E. Lawrence, and S. A. Gerbi. 2015. Sequencing ultra-long DNA molecules with the Oxford Nanopore MinION. *bioRxiv*.1-26.

Van Dijk, E. L., Y. Jaszczyszyn, D. Naquin, and C. Thermes. 2018. The Third Revolution in Sequencing Technology. *Trends in Genetics*. 34(9):666-681.

Van Heesch, S., W. P. Kloosterman, N. Lansu, F.-P. Ruzius, E. Levandowsky, C. C. Lee, S. Zhou, S. Goldstein, D. C. Schwartz, T. T. Harkins, V. Guryev, and E. Cuppen. 2013. Improving mammalian genome scaffolding using large insert mate-pair next-generation sequencing. *BMC Genomics*. 14:257.

Van Oene, M. 2017. Product Obsolescence Notification. Illumina. URL: https://mkt.illumina.com/rs/600-XEX-927/images/PON0216%20NeoPrep%20Final.pdf [accessed 2021/04/21].

Vaser, R. and M. Šikić. 2021. Raven: a *de novo* genome assembler for long reads. *bioRxiv*. http://biorxiv.org/content/early/2021/02/22/2020.08.07.242461.abstract.

Vaser, R., I. Sović, N. Nagarajan, and M. Šikić. 2017. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Research*. 27(5):737-746.

Venter, J. C., M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. Mckusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K.

Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. D. Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R.-R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Y. Wang, A. Wang, X. Wang, J. Wang, M.-H. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. C. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. Mccawley, T. Mcintosh, I. Mcmullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y.-H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigó, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y.-H. Chiang, M. Coyne, C. Dahlke, A. D. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. Mcdaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen, D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu. 2001. The Sequence of the Human Genome. *Science*. 291(5507):1304.

Vollger, M. R., G. A. Logsdon, P. A. Audano, A. Sulovari, D. Porubsky, P. Peluso, A. M. Wenger, G. T. Concepcion, Z. N. Kronenberg, K. M. Munson, C. Baker, A. D. Sanders, D. C. J. Spierings, P. M. Lansdorp, U. Surti, M. W. Hunkapiller, and E. E. Eichler. 2020. Improved assembly and variant detection of a haploid human genome using single-molecule, high-fidelity long reads. *Annals of Human Genetics*. 84(2):125-140.

Voskoboynik, A., N. F. Neff, D. Sahoo, A. M. Newman, D. Pushkarev, W. Koh, B. Passarelli, H. C. Fan, G. L. Mantalas, K. J. Palmeri, K. J. Ishizuka, C. Gissi, F. Griggio, R. Ben-Shlomo, D. M. Corey, L. Penland, R. A. White, I. L. Weissman, and S. R. Quake. 2013. The genome sequence of the colonial chordate, Botryllus schlosseri. *eLife*. 2:1-24.

Vurture, G. W., F. J. Sedlazeck, M. Nattestad, C. J. Underwood, H. Fang, J. Gurtowski, and M. C. Schatz. 2017. GenomeScope: fast reference-free genome profiling from short reads. *Bioinformatics*. 33(14):2202-2204.

Walenz, B., A. Rhie, S. Nurk, S. Koren, and A. M. Phillippy. 2020. A genomic k-mer counter (and sequence utility) with nice features. Github. URL: https://github.com/marbl/meryl [accessed 7 February 2020].

Walker, B. J., T. Abeel, T. Shea, M. Priest, A. Abouelliel, S. Sakthikumar, C. A. Cuomo, Q. Zeng, J. Wortman, S. K. Young, and A. M. Earl. 2014. Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement. *PLoS ONE*. 9:e112963.

Wang, O., R. Chin, X. Cheng, M. K. Y. Wu, Q. Mao, J. Tang, Y. Sun, E. Anderson, H. K. Lam, D. Chen, Y. Zhou, L. Wang, F. Fan, Y. Zou, Y. Xie, R. Y. Zhang, S. Drmanac, D. Nguyen, C. Xu, C. Villarosa, S. Gablenz, N. Barua, S. Nguyen, W. Tian, J. S. Liu, J. Wang, X. Liu, X. Qi, A. Chen, H. Wang, Y. Dong, W. Zhang, A. Alexeev, H. Yang, J. Wang, K. Kristiansen, X. Xu, R. Drmanac, and B. A. Peters. 2019. Efficient and unique cobarcoding of second-generation sequencing reads from long DNA molecules enabling cost-effective and accurate sequencing, haplotyping, and *de novo* assembly. *Genome Research*. 29(5):798-808.

Wang, Z., M. Gerstein, and M. Snyder. 2009. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*. 10:57-63.

Warren, R. L., C. Yang, B. P. Vandervalk, B. Behsaz, A. Lagman, S. J. Jones, and I. Birol. 2015. LINKS: Scalable, alignment-free scaffolding of draft genomes with long reads. *GigaScience*. 4:35.

Weisenfeld, N. I., V. Kumar, P. Shah, D. M. Church, and D. B. Jaffe. 2017. Direct determination of diploid genome sequences. *Genome Research*. 27:757-767.

Wenger, A. M., P. Peluso, W. J. Rowell, P.-C. Chang, R. J. Hall, G. T. Concepcion, J. Ebler, A. Fungtammasan, A. Kolesnikov, N. D. Olson, A. Töpfer, M. Alonge, M. Mahmoud, Y. Qian, C.-S. Chin, A. M. Phillippy, M. C. Schatz, G. Myers, M. A. Depristo, J. Ruan, T. Marschall, F. J. Sedlazeck, J. M. Zook, H. Li, S. Koren, A. Carroll, D. R. Rank, and M. W. Hunkapiller. 2019. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature Biotechnology*. 37(10):1155-1162.

Wu, C.-W., T. M. Schramm, S. Zhou, D. C. Schwartz, and A. M. Talaat. 2009. Optical mapping of the Mycobacterium avium subspecies paratuberculosis genome. *BMC Genomics*. 10:25.

Xiao, M., A. Phong, C. Ha, T.-F. Chan, D. Cai, L. Leung, E. Wan, A. L. Kistler, J. L. Derisi, P. R. Selvin, and P.-Y. Kwok. 2007. Rapid DNA mapping by fluorescent single molecule detection. *Nucleic Acids Research*. 35:e16-e16.

Xue, W., J.-T. Li, Y.-P. Zhu, G.-Y. Hou, X.-F. Kong, Y.-Y. Kuang, and X.-W. Sun. 2013. L_RNA_scaffolder: scaffolding genomes with transcripts. *BMC Genomics*. 14:604.

Yandell, M. and D. Ence. 2012. A beginner's guide to eukaryotic genome annotation. *Nature Reviews Genetics*. 13:329-342.

Yang, J., D. Liu, X. Wang, C. Ji, F. Cheng, B. Liu, Z. Hu, S. Chen, D. Pental, Y. Ju, P. Yao, X. Li, K. Xie, J. Zhang, J. Wang, F. Liu, W. Ma, J. Shopan, H. Zheng, S. A. Mackenzie, and M. Zhang. 2016. The genome sequence of allopolyploid Brassica juncea and analysis of differential homoeolog gene expression influencing selection. *Nature Genetics*. 48:1225-1232.

Yao, W., G. Li, H. Zhao, G. Wang, X. Lian, and W. Xie. 2015. Exploring the rice dispensable genome using a metagenome-like assembly strategy. *Genome Biology*. 16:187.

Yeo, S., L. Coombe, J. Chu, R. L. Warren, and I. Birol. 2017. ARCS: Assembly Roundup by Chromium Scaffolding. *bioRxiv*.1-13.

Yuan, Y., P. E. Bayer, J. Batley, and D. Edwards. 2017. Improvements in Genomic Technologies: Application to Crop Genomics. *Trends in Biotechnology*. 35:547-558.

Zapata, L., J. Ding, E.-M. Willing, B. Hartwig, D. Bezdan, W.-B. Jiao, V. Patel, G. Velikkakam James, M. Koornneef, S. Ossowski, and K. Schneeberger. 2016. Chromosome-level assembly of Arabidopsis thaliana L er reveals the extent of translocation and inversion polymorphisms. *Proceedings of the National Academy of Sciences*. 113:E4052-E4060.

Zerbino, D. R. and E. Birney. 2008. Velvet: Algorithms for *de novo* short read assembly using de Bruijn graphs. *Genome Research*. 18(5):821-829.

Zhang, F., L. Christiansen, J. Thomas, D. Pokholok, R. Jackson, N. Morrell, Y. Zhao, M. Wiley, E. Welch, E. Jaeger, A. Granat, S. J. Norberg, A. Halpern, M. C Rogert, M. Ronaghi, J. Shendure, N. Gormley, K. L. Gunderson, and F. J. Steemers. 2017. Haplotype phasing of whole human genomes using bead-based barcode partitioning in a single tube. *Nature Biotechnology*. 35(9):852-857.

Zhang, H., C. Jain, and S. Aluru. 2020. A comprehensive evaluation of long read error correction methods. *BMC Genomics*. 21(6):889.

Zhang, S. V., L. Zhuo, and M. W. Hahn. 2016. AGOUTI: improving genome assembly and annotation using transcriptome data. *GigaScience*. 5:31.

Zhou, S., M. C. Bechner, M. Place, C. P. Churas, L. Pape, S. A. Leong, R. Runnheim, D. K. Forrest, S. Goldstein, M. Livny, and D. C. Schwartz. 2007. Validation of rice genome sequence by optical mapping. *BMC Genomics*. 8:278.

Zhou, S., F. Wei, J. Nguyen, M. Bechner, K. Potamousis, S. Goldstein, L. Pape, M. R. Mehan, C. Churas, S. Pasternak, D. K. Forrest, R. Wise, D. Ware, R. A. Wing, M. S. Waterman, M. Livny, and D. C. Schwartz. 2009. A Single Molecule Scaffold for the Maize Genome. *PLoS Genetics*. 5:e1000711.

Zhu, B. H., Y. N. Song, W. Xue, G. C. Xu, J. Xiao, M. Y. Sun, X. W. Sun, and J. T. Li. 2016. PEP-scaffolder: Using (homologous) proteins to scaffold genomes. *Bioinformatics*. 32:3193-3195.

Zimin, A. V., G. Marcais, D. Puiu, M. Roberts, S. L. Salzberg, and J. A. Yorke. 2013. The MaSuRCA genome assembler. *Bioinformatics*. 29:2669-2677.

Zimin, A. V., K. A. Stevens, M. W. Crepeau, D. Puiu, J. L. Wegrzyn, J. A. Yorke, C. H. Langley, D. B. Neale, and S. L. Salzberg. 2017. An improved assembly of the loblolly pine mega-genome using long-read single-molecule sequencing. *GigaScience*. 6:1-4.

APPENDIX 1

Chapter 1 – Supplementary File 1

This is Supplementary File 1 from "Lingering Taxonomic Challenges Hinder Conservation and Management of Global Bonefishes". The following is a tree in Newick format, with bootstrap support values provided when greater than 90. No branch lengths are specified. For more information, see the main manuscript and Wallace (2014), from which this tree was taken.

(((('Albula pacifica','Albula nemoptera'):92,((('Albula argentea','Albula virgata'):100,'Albula oligolepis'):99,('Albula koreana',(('Albula gilberti',('Albula sp. cf. vulpes','Albula esuncula')):92,('Albula goreensis',('Albula vulpes','Albula glossodonta')))))):100,('Anguilla rostrata','Pterothrissus gissu'):100);

APPENDIX 2

Chapter 2 – Additional File 1

SUPPLEMENTARY BIOINFORMATICS METHODS

An overview of the methods used in this study was provided in the main manuscript. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided here.

S.1 – Tissue Collection and Preservation

Not applicable.

S.2 – Sequencing

Not applicable.

S.3 – Read Error Correction

S.3.1 – Illumina DNA

An estimate of the number of k-mers present in the reads is required to run BFCounter. This number is really just a simple math problem based on the number of reads, the length of the reads, and k-mer size according to this equation:

$$T = n(l - k + 1)$$

Where n is the number of reads, l is the read length, and k is the k-mer size, and T is the total number of k-mers (not necessarily unique or distinct) present in the reads. Of course, this

assumes a uniform read length. If the reads are paired-end, n is still the number of reads, not the number of pairs of reads. Since ntCard v1.0.1 (Hamid et al. 2017) was used to quickly get a picture for the k-mer coverage histogram, its reported value F0 was used instead of the equation as it is an estimate for T. ntCard was run according to the following command:

```
ntcard \
    -k 19\
    -t ${THREADS} \
    -p ${OUTPUT_FILE_BASE_NAME} \
    ${INPUT_FASTQ_FILES[@]}
```

To generate q-mer counts BFCounter v0.2 (Melsted and Pritchard 2011) was used to count and dump the q-mers according to the following commands:

```
BFCounter count\
    -k 19\
-n ${TOTAL_NUMBER_OF_KMERS} \
-s ${RANDOM_SEED} \
-t ${THREADS} \
-o ${COUNTS_FILE_NAME} \
--quake \
--quality-scale=33 \
${INPUT_FASTQ_FILES[@]}

BFCounter dump\
    -k 19\
-i ${COUNTS_FILE_NAME} \
-o ${OUTPUT_FILE_NAME} \
--quake
```

Quake v0.3.5 (Kelley et al. 2010) was run in two stages where the first identifies a q-mer cutoff and the second corrects the reads based on that cutoff. The suggested q-mer cutoff was 2.33, which was subsequently used by the correction phase of Quake. The two steps were executed according to the following commands:

```
cov_model.py \
${BFCOUNTER_DUMP_FILE}
```

```
correct \
    -k 19 -q 33 \
-m ${QMER_COUNTS_FILE} \
-o ${OUTPUT_FILE_NAME} \
-f ${INPUT_FASTQ_FILES[@]} \
-p ${THREADS} \
-c ${CUTOFF} \
-u --headers --log
```

Quake was developed quite some time ago, and the installation process was made difficult as dependencies were updated and function calls were broken. Multiple solutions likely exist to remedy the problem, but we found success by installing Quake with R v3.4.0 (https://www.r-project.org) with package VGAM v0.7-8 (https://CRAN.R-project.org/package=VGAM) (Yee and Wild 1996).

S.3.2 – Illumina RNA

Since no corrections were made by Rcorrector v1.0.2 (Song and Florea 2015) and the command is fairly straightforward, little additional detail is necessary. Recall that BFCounter was used instead of the built-in Jellyfish to generate the counts. Also note that this process was run separately for each tissue. The commands used are the following:

```
BFCounter count\
    -k 19\
-n ${TOTAL_NUMBER_OF_KMERS} \
-s ${RANDOM_SEED} \
-t ${THREADS} \
-o ${COUNTS_FILE_NAME} \
--quality-scale=33 \
${INPUT_FASTQ_FILES[@]}

BFCounter dump\
    -k 19\
-i ${COUNTS_FILE_NAME} \
-o ${DUMP_FILE_NAME} \

rcorrector \
    -k 19 \
-c ${DUMP_FILE_NAME} \
-od ${OUTPUT_DIR_NAME} \
-p ${INPUT_FASTQ_FILES[@]} \
-t ${THREADS}
```

S.3.3 – PacBio CLRs

First the process to correct the PacBio CLRs will be described. Next, the experiments with other correction strategies will be briefly described.

S.3.3.1 – Dual Correction Strategy

Typically, a "hybrid" correction strategy is defined as one in which more than one data type (i.e., PacBio CLRs and Illumina short reads) are employed. This differs from a "self" correction strategy in which only the PacBio CLRs are used to correct themselves. We employed a strategy that is "hybrid", but that is not fully described by the word "hybrid". We have referred to this strategy as "dual" correction. First, "self" correction is completed. Second, "hybrid" correction is done on the already self-corrected reads. The self-corrected reads were generated using Canu v1.6 (Koren et al. 2017) with the following command:

```
canu -correct \
    -s ${SETTINGS_FILE} \
-d ${OUTPUT_DIR_NAME} \
-p ${OUTPUT_PREFIX} \
-pacbio-raw \
${INPUT_PACBIO_READS[@]}
```

The relevant lines of the setting file are included here:

```
genomeSize=932813000
ovsMethod=sequential
gridEngine=slurm
```

The self-corrected reads were provided to CoLoRMap downloaded April 2018 (Haghshenas et al. 2016) as the "uncorrected" input reads. Please note that you will need to combine and interleave all Illumina short reads into a single file. All PacBio reads will also need to be in a single file, and the headers will need to be unique up to the first space, so some modification to the headers may be necessary. CoLoRMap is really a pipeline with a very basic wrapper script. In practice, it makes more sense to run each step in the wrapper script as separate jobs to avoid re-computing if a failure (e.g., too much RAM or time) occurs in a downstream step. If nothing else, a simple addition of logical checks can be added to the wrapper script to ensure subsequent steps aren't run if the previous step failed. If run without any such modifications, the commands to run CoLoRMap are the following:

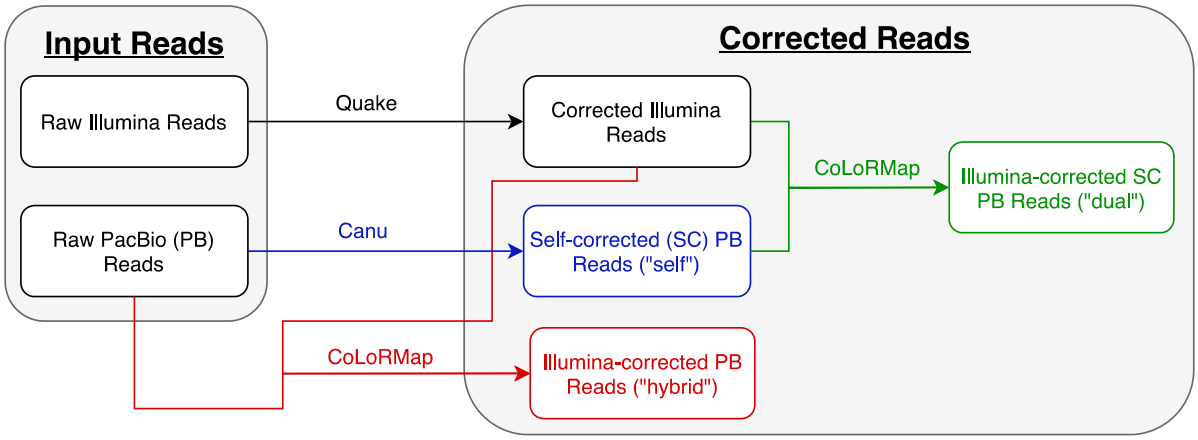
```
runCorr.sh \
    ${INPUT_SELF_CORRECTED_PACBIO_READS} \
${INPUT_ILLUMINA_READS} \
${OUTPUT_CORRECTED_PACBIO_READS_DIR} \
${OUTPUT_CORRECTED_PACBIO_READS_PREFIX} \
${THREADS}

runOEA.sh \
    ${INPUT_COLORMAP_CORRECTED_READS} \
${INPUT_ILLUMINA_READS} \
${OUTPUT_CORRECTED_PACBIO_READS_DIR} \
${OUTPUT_CORRECTED_PACBIO_READS_PREFIX} \
${THREADS}
```

Once the correction and overlap error extension assembly phases are completed, the now "dual" corrected reads are ready for assembly.

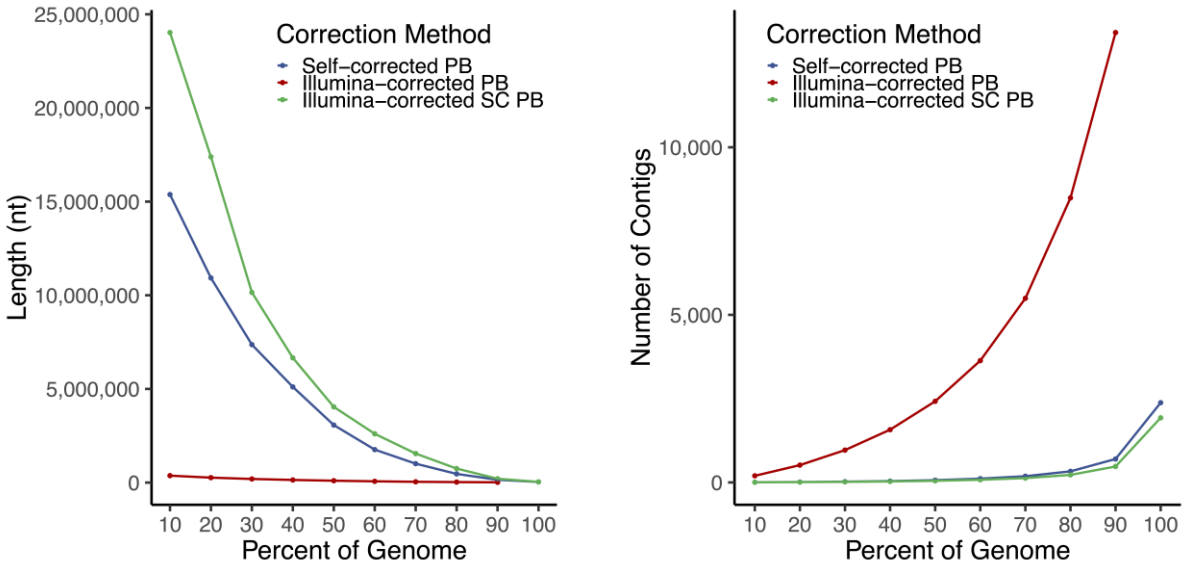
S.3.3.2 – Correction Experiments

We explored the effects on assembly continuity of several correction strategies before settling on the chosen strategy. Ignoring failed strategies due to software failures, three strategies were employed: (a) "self" correction (only PacBio CLRs, (b) "hybrid" correction (using only Illumina reads to correct the PacBio CLRs), and (c) "dual" correction (using Illumina reads to correct already self-corrected PacBio CLRs). These correction strategies are described visually in the following flow chart:



The table and two plots show the NGx and LGx plots where x is a number between 0 and 100 representing the percentage of the genome size. NGx and LGx statistics are similar to the Nx and Lx statistics except they are scaled to the genome size instead of the assembly size. In theory, assemblies improve by maximizing and minimizing the areas under the NG and LG curves, respectively. Plainly, the "dual" correction strategy is superior in terms of continuity.

	Self-corrected PacBio		Illumina-corrected PacBio		Dual-corrected PacBio	
	NG	LG	NG	LG	NG	LG
10	15,377,867	6	370,666	198	24,024,878	4
20	10,924,741	13	264,154	517	17,391,607	10
30	7,365,265	24	193,486	965	10,151,733	17
40	5,108,341	40	140,579	1,574	6,659,461	30
50	3,071,409	68	100,138	2,421	4,042,832	49
60	1,760,100	113	67,721	3,634	2,604,415	80
70	1,010,581	186	42,692	5,494	1,549,398	130
80	463,094	331	26,676	8,489	749,230	225
90	149,314	700	14,858	13,423	205,177	476
100	35,787	2,379	NA	NA	38,443	1,928



S.4 – Genome Size Estimation

ntCard v1.0.1 (Hamid et al. 2017) was used to estimate the k-mer coverage histogram using the following command:

```
ntcard \
    -k 19 \
    -t ${THREADS} \
    -p ${OUTPUT_FILE_BASE_NAME} \
    ${INPUT_FASTQ_FILES[@]}
```

The equation described in the main manuscript was used to determine the genome size from the ntCard output, implemented as a simple AWK program. First, the k-mer coverage histogram must be processed to match the output format of Jellyfish's `histo` command (Marcais and Kingsford 2011).

```
tail -n +3 ${NTCARD_OUTPUT_FILE} \
         | tr -d "f" \
         > ${HISTO_FILE}

awk -f ${AWK_SCRIPT} ${HISTO_FILE}
```

Where the AWK program referred to as `${AWK_SCRIPT}` is the following:

```
BEGIN {
   x = 0; # x at max y
   y = 0; # max y
   s = 0; # genome size
}
{
   if ($2 >= y) {
       y = $2;
       x = NR;
   }
   s += $1 * $2
}
END {
   print "peak: " x "," y "; sum: " s "; size: " s / x;
}
```

S.5 – Genome Assembly, Polishing, and Scaffolding

The individual steps of genome assembly, polishing, and scaffolding will each be described separately. Calculation of assembly summary statistics will also be described.

S.5.1 – Genome Assembly

The assembly was created with Canu v1.6 (Koren et al. 2017) using the already reads from the "dual" correction strategy using the following command:

```
canu -trim-assemble \
    -s ${SETTINGS_FILE} \
    -d ${OUTPUT_DIR_NAME} \
    -p ${OUTPUT_PREFIX} \
    -pacbio-corrected \
    ${INPUT_DUAL_CORRECTED_PACBIO_READS_FILE}
```

S.5.2 – Polishing

Before polishing the contigs, the corrected Illumina WGS reads required slight modification of the headers because spaces were not allowed. The exact modifications required to make sequence headers match RaCon's expectations may vary, but the following AWK program worked in our case:

```
BEGIN {
    FS = " ";
}
{
    if (NR % 4 == ) {
        print $1 "-" substr($2, 1, 1);
    } else {
        print $0;
    }
}
```

RaCon also required mapping these short reads to the contigs before it would run. The alignments were performed with BWA v0.7.17-r1998 (Li 2013) and converted from SAM format to BAM format using SAMtools v1.6 (Li et al. 2009):

```
bwa index \
    -p ${CONTIGS_INDEX_PREFIX} \
    ${CONTIGS_FASTQ_FILE}

bwa mem \
    -t ${THREADS} \
    -p ${CONTIGS_INDEX_PREFIX} \
    ${ILLUMINA_SHORT_READS_FILE} \
    > ${ALIGNMENT_SAM_FILE}

samtools view \
    -buS ${ALIGNMENT_SAM_FILE} \
    | samtools sort \
        -@ ${THREADS} \
        > ${ALIGNMENT_BAM_FILE}
```

Polishing with the corrected Illumina WGS reads using RaCon v1.3.1 (Vaser et al. 2017) was accomplished using the following command:

```
racon \
    --include-unpolished \
    --threads ${THREADS} \
    ${ILLUMINA_SHORT_READS_FILE} \
    ${ILLUMINA2CONTIGS_ALIGNMENTS_BAM} \
    ${CONTIGS_FILE} \
    > ${POLISHED_CONTIGS_FILE}
```

This process of alignment and polishing was repeated for a second round with the polished output contigs from the first round acting as "unpolished" contigs for the second round.

S.5.3 – Scaffolding

The polished contigs were scaffolded in a stepwise fashion using two types of long-range information: Hi-C and RNA-seq reads.

S.5.3.1 – Hi-C Scaffolding

The Hi-C data alignments were performed following the Arima Genomics (San Diego, California, USA; https://arimagenomics.com) Mapping Pipeline (https://github.com/ArimaGenomics/mapping_pipeline), which relied on bwa v0.7.17-r1998 (Li 2013), Picard v2.19.2 (Broad Institute 2019), and SAMtools v1.6 (Li et al. 2009). As the pipeline is reasonably well-documented, it will be only summarized here:

1. The assembly (polished contigs) is indexed using SAMtools `faidx`.
2. The assembly is indexed with bwa `index` and the Hi-C reads are mapped to the assembly with bwa `mem`.
3. The alignments are converted from SAM to BAM format with SAMtools `view`.
4. The 5' ends are filtered using SAMtools `view` and the Arima Genomics Perl (https://www.perl.org) script `filter_five_end.pl`.

5. Paired-end reads are combined into a single file with the Arima Genomics Perl script `two_read_bam_combiner.pl` and sorted with SAMtools `sort`. These reads will be treated as single-end hereafter.
6. Read groups are added to the BAM file using Picard `AddOrReplaceReadGroups`.
7. Merge technical replicates. This step was skipped because no such replicates existed.
8. Duplicates in the BAM file were marked using Picard `MarkDuplicates`.
9. Merge biological replicates. This step was skipped because no such replicates existed.
10. The final BAM file was indexed with SAMtools `index`.
11. Stats were reported with the Arima Genomics Perl script `get_stats.pl`.

Scaffolding was performed on the polished contigs using the final BAM file from the Arima Genomics Mapping Pipeline with SALSA downloaded 29 May 2019 (Ghurye et al. 2017; Ghurye et al. 2019). First, some pre-processing was required with BEDTools v2.28.0 (Quinlan and Hall 2010) to convert the final BAM file from the mapping pipeline to BED format; this was then sorted. The BEDTools, sorting, and SALSA commands are listed here (note that the `${RESTRICTION_ENZYME_SEQ}` was `GATC`):

```
bedtools bamtobed \
    -i ${FINAL_ARIMA_BAM_FILE} \
    > ${HIC_BED_FILE}

sort -k 4 \
    ${HIC_BED_FILE} \
    > ${SORTED_HIC_BED_FILE}

run_pipeline.py \
    -a ${POLISHED_CONTIGS_FILE} \
    -l ${POLISHED_CONTIGS_FAIDX_FILE} \
    -b ${SORTED_HIC_BED_FILE} \
    -e ${RESTRICTION_ENZYME_SEQ} \
    -s ${GENOME_SIZE} \
    -m yes \
    -o ${OUTPUT_SALSA_DIR}
```

Note that all newly-created gaps from SALSA will all be assigned a length of 500 nucleotides (i.e., 500 Ns in a row). Assuming these are gaps of unknown size, these will ideally be changed to 100 nucleotides for any submissions to GenBank. If you have multiple sources of evidence for gaps (e.g., Hi-C and RNA-seq), you will want to keep track of which gaps were supported by each type of evidence.

S.5.3.2 – RNA-seq Scaffolding

The RNA-seq data were aligned using HiSat v0.1.6-beta (Kim et al. 2015), and the alignments were converted from SAM to BAM format and sorted using SAMtools v1.6 (Li et al. 2009). First, the assembly (scaffolds from Hi-C) was indexed with HiSat. For each tissue (i.e., heart, gill, and liver), HiSat aligned reads to the assembly, SAMtools sorted and compressed the output alignments, and Rascaf downloaded June 2018 (Song et al. 2016) computed how scaffolding could be done. The actual scaffolding was done with Rascaf in a single step after all steps had been completed for each tissue. The process is described in the following script:

```
hisat-build \
    ${HISAT_IDX_PREFIX} \
    ${HIC_SCAFFOLDS}


for TISSUE in {gill,heart,liver}
do
    RNASEQ_READS_LEFT=${TISSUE}_L.fq.gz
    RNASEQ_READS_RIGHT=${TISSUE}_R.fq.gz
    ALIGNMENT_SAM=${TISSUE}_aln.sam

    hisat \
        -p ${THREADS} \
        --phred33 -q -t \
        -x ${HISAT_IDX_PREFIX} \
        -1 ${RNASEQ_READS_LEFT} \
        -2 ${RNASEQ_READS_RIGHT} \
        -S ${ALIGNMENT_SAM}
```

```
    samtools view \
        -buh ${ALIGNMENT_SAM} \
        | samtools sort \
            -@ ${THREADS} \
            -m ${MEMORY}M \
            -O BAM \
            -o ${ALIGNMENT_BAM}

    rascaf \
        -breakN 1 \
        -b ${ALIGNMENT_BAM} \
        -f ${HIC_SCAFFOLDS} \
        -o ${TISSUE}.out
done

rascaf-join \
    -r gill.out \
    -r heart.out \
    -r liver.out \
    -o ${OUTPUT_FILE_PREFIX}
```

Note that the `-breakN 1` option breaks all scaffolds at gaps of any size (1 or more Ns) while it determines which sequences it can join. Broken gaps are then restored to their original length and location when additional gaps are added based on the RNA-seq read pairs. If the RNA-seq evidence disagrees with any pre-existing gaps, it will remove them. Also note that newly-created gaps from Rascaf will all be assigned a length of 17 nucleotides (i.e., 17 Ns in a row). For submission to GenBank, these will ideally be changed to 100 nucleotides. If you have multiple sources of evidence (e.g., Hi-C and RNA-seq), you will want to keep track of which gaps were supported by each type of evidence.

S.5.4 – Assembly Statistics

Assembly continuity statistics, e.g., N50 and auN (Li 2020), were calculated with caln50 downloaded April 2020 (https://github.com/lh3/calN50) and a custom Python (https://www.python.org) script. caln50 is run using the following simple command:

```
caln50 \
    -s 0.01 \
    -L ${GENOME_SIZE} \
    ${CONTIGS_OR_SCAFFOLDS_FILE} \
    > ${STATISTICS_FILE}
```

The custom Python script is not efficient, but it does calculate Nx, Lx, NGx, and LGx, as well as a few other interesting points about sequences in a fasta file. This script is too long to realistically represent when embedded in the text; it is available on GitHub at https://github.com/pickettbd/basicAsmStatsCalcInPy.

Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 (Simão et al. 2015) and OrthoDB v10 (Kriventseva et al. 2019). The BUSCO config file was the not modified from the default aside from the locations of OrthoDB v10 and the binary executables for BUSCO. It was run based on the following command structure:

```
busco \
    --offline \
    --config ${BUSCO_CONFIG_FILE} \
    --cpu ${THREADS} \
    --in ${CONTIGS_OR_SCAFFOLDS_FASTA} \
    --out_path ${OUTPUT_DIR} \
    --out ${OUTPUT_FILE_PREFIX} \
    --mode genome \
    --lineage actinopterygii \
    --augustus_species zebrafish
```

S.6 – Transcriptome Assembly

The transcripts were assembled using Trinity v2.6.6 (Grabherr et al. 2011), which depended on Bowtie v2.3.4.3 (Langmead and Salzberg 2012), Jellyfish v2.2.10 (Marcais and Kingsford 2011), salmon v0.12 (Patro et al. 2017), and SAMtools v1.6 (Li et al. 2009):

```
trinity \
    --no_version_check \
    --max_memory ${MEMORY} \
    --CPU ${THREADS} \
    --long_reads ${DUAL_CORRECTED_PACBIO_READS} \
    --seqType fq \
    --left ${RNASEQ_READS_LEFT} \
    --right ${RNASEQ_READS_RIGHT} \
    --SS_lib_type FR \
    --normalize_max_read_cov 50 \
    --normalize_by_read_set \
    --min_contig_length 200 \
    --output ${TRINITY_OUTPUT_DIR}
```

Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 (Simão et al. 2015) and OrthoDB v10 (Kriventseva et al. 2019). The command and config file were a match to how BUSCO was run to assess genome assembly correctness, except that the `--mode` option was `transcriptome` instead of `genome`.

S.7 – Computational Annotation

The MAKER v3.01.02-beta (Holt and Yandell 2011) pipeline was used to annotate the assembly. With a large enough cluster with MPI support, MAKER runs relatively quickly for each round. The general process was described in prose in the main manuscript, but it can be summarized in outline form here:

I. MAKER round #1

II. *ab initio* gene predictors

   a. AUGUSTUS

   b. GeneMark-ES

   c. SNAP

III. MAKER round #2

IV. *ab initio* gene predictors

a. AUGUSTUS

b. SNAP

V. MAKER round #3

VI. MAKER post-processing & functional annotation

As each round of MAKER was run in a nearly identical fashion, the process will be described once, followed by differences between the rounds. Similarly, AUGUSTUS and SNAP will also be described once.

S.7.1 – MAKER Round #1

The command to run MAKER is straight-forward, though may vary slightly depending on the implementation of MPI employed by the cluster. The MAKER documentation says to run MAKER with the `mpiexec` command, but `mpirun` was successful for our setup. Running MAKER from a working directory on an NFS drive will almost certainly result in failure unless MAKER is directed where to do its work in a non-NFS temporary directory. This required some extra attention to job cleanup on our cluster, but it was successful when we pointed MAKER to the local drives on the nodes on which it was run, which were mounted at `/tmp`. When calling MAKER from the directory in which the control files exist, the command to start MAKER looks like this:

```
mpirun maker \
    -cpus ${CPUS} \
    -TMP ${MAKER_TMP_DIR}
```

The truly critical parts are in the MAKER control files. Assuming one has a successfully installed and configured version of MAKER available, default control files can be generated in the working directory by running the following command: `maker -CTL`. No modifications were made to the `maker_evm.ctl` file. The `maker_bopt.ctl` file was left unchanged as well. Note that `use_rapsearch` was set to `0` and `blast_type` was set to `ncbi+`. The `maker_exe.ctl` file

was modified as needed only to set correct paths to the executables for MAKER's dependencies.

The following shows the modified or otherwise relevant lines from the `maker_opts.ctl` file:

```
# genome
genome=/path/to/scaffolds.fa
organism_type=eukaryotic

#re-annotation
maker_gff=
est_pass=0
protein_pass=0
rm_pass=0
model_pass=0
pred_pass=0
other_pass=0

# est/rna-seq
est=/path/to/Trinity/transcripts.fa
est_gff=

# protein homology
protein=/path/to/uniprot_sprot.fa
protein_gff=

# repeat masking
model_org=all
rmlib=/path/to/RepeatModeler/results/assembly-db-families.fa
repeat_protein=/path/to/maker-install-dir/data/te_proteins.fa
rm_gff=
softmask=1

# gene prediction
snaphmm=
gmhmm=
augustus_species=
pred_gff=
model_gff=
run_evm=0
est2genome=1
protein2genome=1
trna=0

# maker behavior
max_dna_len=1000000
min_contig=20000
```

```
pred_flank=200
pred_stats=0
AED_threshold=1
min_protein=0
alt_splice=0
always_complete=0
map_forward=0
keep_preds=0

split_hit=10000
min_intron=20
single_exon=0
single_length=250
correct_est_fusion=0
```

Once MAKER has completed, a few MAKER accessory scripts can be run to extract the results from its datastore located at `${PROJECT_DIR}/maker/rnd1/*.datastore`. Additional modifications (shown), can also be employed to make output names more palatable. For sake of demonstration, we assume the master datastore index log file is prefixed with `scaffolds`, and the output base (`-o` option for `fasta_merge`) is `agloss-rnd1` (*A. glossodonta* round 1)):

```
cd maker/rnd1/scaffolds.maker.output

fasta_merge \
    -o agloss-rnd1 \
    -d scaffolds_master_datastore_index.log

gff3_merge \
    -n -s \
    -d scaffolds_master_datastore_index.log \
    > agloss-rnd1_noSeq.gff

cd scaffolds_datastore

rename 's/.all.maker./_/' *.fasta # Perl rename, not Linux util
rename 's/fasta/fa/' *.fasta      # Perl rename, not Linux util

awk '{if ($2 == "est2genome") print $0}' \
    agloss-rnd1_noSeq.gff \
    > agloss-rnd1_est2genome.gff

awk '{if ($2 == "protein2genome") print $0}' \
    agloss-rnd1_noSeq.gff \
    > agloss-rnd1_protein2genome.gff
```

```
awk '{if ($2 ~ "repeat") print $0}' \
    agloss-rnd1_noSeq.gff \
    > agloss-rnd1_repeats.gff

mv agloss-rnd1*.fa agloss-rnd1*.gff ../..

cd ../../../..
```

S.7.2 – ab initio Gene Prediction

Three *ab initio* gene prediction programs were run between MAKER rounds 1 and 2. AUGUSTUS and SNAP can take gene models as input, and they are thus able to be run with new models after rounds 1 and 2 of MAKER in preparation for rounds 2 and 3, respectively. GeneMark-ES does not take gene models as input, and it thus needs to be run only one time.

S.7.2.1 – GeneMark-ES

GeneMark-ES required a software key to be run, which can be obtained or re-obtained for free for academic use at any time. GeneMark-ES also requires a configuration file to be run; the default configuration file was used. The following command demonstrates how to run GeneMark-ES:

```
gmes_petap.pl \
    --ES \
    --usr_cfg ${COPY_OF_DEFAULT_CONFIG_FILE} \
    --cores ${THREADS} \
    --sequence ${SCAFFOLDS_ASSEMBLY_FILE}
```

S.7.2.2 – AUGUSTUS

AUGUSTUS training can be handled with BUSCO. Before AUGUSTUS can be trained, configuration files and data from AUGUSTUS and BUSCO will need to be copied to the working directory for this part of the analysis, and the relevant environment variables will need to be reset (which assumes they are properly set in the first place):

```
cp  -r ${AUGUSTUS_CONFIG_PATH} ${PROJECT_DIR}/augustus_config
export AUGUSTUS_CONFIG_PATH=${PROJECT_DIR}/augustus_config

cp  ${BUSCO_CONFIG_FILE} ${PROJECT_DIR}/busco_config.ini
export BUSCO_CONFIG_FILE=${PROJECT_DIR}/busco_config.ini
```

No changes were made to the AUGUSTUS files. The only change made to the BUSCO configuration file was to set `download_path=/path/to/odb10` instead of `./busco_download`. This is assuming OrthoDB v10 has already been downloaded to that location and that the `--offline` flag will be used when running BUSCO. Before training AUGUSTUS, candidate gene regions need to be extracted. This was done with a custom Python script (available at https://github.com/pickettbd/albula-glossodonta_assembly-paper_misc-scripts) and BEDTools v2.28.0 (Quinlan and Hall 2010).

```
python3 generateBedForMrnaExtraction.py \
    maker/rnd1/agloss-rnd1_noSeq.gff \
    scaffolds.fa \
    candidates-rnd1.bed

bedtools getfasta \
    -fi scaffolds.fa \
    -bed candidates-rnd1.bed \
    -fo candidates-rnd1.fa
```

AUGUSTUS was trained by running BUSCO with the same command described in the section S.5.4 (i.e., `mode=genome`, `lineage=actinopterygii`, `augustus_species=zebrafish`). To make the AUGUSTUS training parameters generated after running BUSCO available to the next round of MAKER, some post-processing is required:

```
# make dir for final results
mkdir augustus_config/species/agloss

# move to results location
cd "busco-augustus/agloss-rnd1/
    run_actinopterygii_odb10/augustus_output/
    retraining_parameters/BUSCO_agloss-rnd1"
```

```
# rename some files and their references to eachother
rename \ # Perl rename, not Linux util
    's/BUSCO_(agloss-rnd1_)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/BUSCO_(agloss-rnd1_)/\1/' \
    ./agloss-rnd1_parameters.cfg*

# do it again, removing the rnd info
rename \ # Perl rename, not Linux util
    's/(agloss)-rnd1)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/(agloss)-rnd1/\1/' \
    ./*

# copy the files to final results location
cp -f ./* ../../../../../../augustus_config/species/agloss/

# move back to main project dir
cd -
```

S.7.2.3 – SNAP

Training with SNAP is much less resource intensive than training AUGUSTUS. Most, if not all, of the commands can reasonably be run "locally" on a login node or other machine. The final output file, `genome.hmm`, is what will be provided to the next round of MAKER. Inspection of the log files was performed after each step. The process of training SNAP can be described by the following commands:

```
mkdir -p snap/rnd1

ln -s \
    ../../maker/rnd1/agloss-rnd1_withSeq.gff \
    snap/rnd1/genome.gff

cd snap/rnd1

maker2zff genome.gff
```

```
fathom \
    genome.ann genome.dna \
    -gene-stats \
    > gene-stats.log

fathom \
    genome.ann genome.dna \
    -validate \
    > validate.log

fathom \
    genome.ann genome.dna \
    -categorize 1000 \
    > categorize.log

fathom \
    uni.ann uni.dna \
    -export 1000 -plus \
    > export.log

forge \
    export.ann export.dna \
    > forge.log

hmm-assembler.pl \
    genome params \
    > genome.hmm
```

S.7.3 – MAKER Round #2

The second round of MAKER was run much the same way as the first, with a few modifications. First, the second round was run in a separate directory: `maker/rnd2`. The `run_evm` flag was set to enable MAKER to run EVidenceModeler v1.1.1 (Haas et al. 2008). The control files were copied from the first round and the following changes were made to `maker_opts.ctl`:

```
# est/rna-seq
est=
est_gff=/path/to/project/maker/rnd1/agloss-rnd1_est2genome.gff

# protein homology
protein=
protein_gff=/path/to/project/maker/rnd1/agloss-rnd1_protein2genome.gff
```

```
# repeat masking
model_org=
rmlib=
repeat_protein=
rm_gff=/path/to/project/maker/rnd1/agloss-rnd1_repeats.gff

# gene prediction
snaphmm=/path/to/project/snap/rnd1/genome.hmm
gmhmm=/path/to/project/gmes/output/gmhmm.mod
augustus_species=agloss
run_evm=1
est2genome=0
protein2genome=0
```

Additionally, the same accessory scripts, renaming, etc. was performed after this second round of MAKER as with the first round. The only differences being that `rnd1` was replaced with `rnd2` in all the commands and names and the `awk` commands were skipped.

S.7.4 – ab initio Gene Prediction

Since GeneMark-ES does not take gene models as input, only SNAP and AUGUSTUS could be re-run after MAKER's second round. Before training them, the models from MAKER were filtered using gFACs v1.1.1 (Caballero and Wegrzyn 2019).

S.7.4.1 – gFACs Filtering

In an attempt to improve the quality of gene models being used for this final round of training with AUGUSTUS and SNAP, gFACs was employed to filter out models with single-exon genes, introns shorter than 20bp, etc. The gFACs command and relevant supporting commands (e.g., creating working directories) are shown here:

```
mkdir -p gfacs/rnd2

ln -s \
    ../../maker/rnd2/agloss-rnd2_noSeq.gff \
    gfacs/rnd2/orig_noSeq.gff
```

```
ln -s \
    ../../assembly/scaffolds.fa \
    gfacs/rnd2/assembly.fa

awk \
    'BEGIN{x=0;}/^##FASTA/{x=1;}{if(x){print $0;}}' \
    maker/rnd2/agloss-rnd2_withSeq.gff \
    > gfacs/rnd2/orig_onlySeq.gff

cd gfacs/rnd2

gFACs.pl \
    -f "maker_2.31.9_gff" \
    -p  ./output/agloss-rnd2_noSeq \
    --statistics-at-every-step \
    --statistics \
    --rem-monoexonics \
    --min-exon-size 20 \
    --min-intron-size 20 \
    --min-CDS-size 74 \
    --fasta assembly.fa \
    --splice-table \
    --nt-content \
    --canonical-only \
    --rem-genes-without-stop-codon \
    --allowed-inframe-stop-codons 0 \
    --create-gff3 \
    --get-fasta-with-introns \
    --get-fasta-without-introns \
    --get-protein-fasta \
    --distributions \
    exon_lengths \
    intron_lengths \
    CDS_lengths \
    gene_lengths \
    exon_position \
    exon_position_data \
    intron_position \
    intron_position_data \
    -O ./output \
    orig_noSeq.gff
```

```
ln -s \
    agloss-rnd2_noSeq_out.gff3 \
    output/agloss-rnd2_noSeq.gff

cat \
    output/agloss-rnd2_noSeq.gff orig_onlySeq.gff \
    > output/agloss-rnd2_withSeq.gff

cd ../..
```

S.7.4.2 – AUGUSTUS

Training AUGUSTUS after the second round of MAKER in preparation for the third round occurred in the same manner as the first time. The exceptions were that (a) the input GFF3 file came from gFACs instead of directly from MAKER, (b) `augustus_species=agloss` was used instead of `augustus_species=zebrafish`, and (c) the occurrences of `rnd1` in the commands and names were changed to `rnd2`. The commands are replicated (and appropriately modified) again here:

```
python3 generateBedForMrnaExtraction.py \
    gfacs/rnd2/output/agloss-rnd2_noSeq.gff \
    scaffolds.fa \
    candidates-rnd2.bed

bedtools getfasta \
    -fi scaffolds.fa \
    -bed candidates-rnd2.bed \
    -fo candidates-rnd2.fa
```

AUGUSTUS was trained by running BUSCO with the same command described in the section S.5.4 (i.e., `mode=genome` and `lineage=actinopterygii`) except that `augustus_species=agloss` instead of `zebrafish`. To make the AUGUSTUS training parameters generated after running BUSCO available to the next round of MAKER, some post-processing is required:

```
# move to results location
cd "busco-augustus/agloss-rnd2/
    run_actinopterygii_odb10/augustus_output/
    retraining_parameters/BUSCO_agloss-rnd2"

# rename some files and their references to each other
rename \ # Perl rename, not Linux util
    's/BUSCO_(agloss-rnd2_)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/BUSCO_(agloss-rnd2_)/\1/' \
    ./agloss-rnd1_parameters.cfg*

# do it again, removing the rnd info
rename \ # Perl rename, not Linux util
    's/(agloss)-rnd2)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/(agloss)-rnd2/\1/' \
    ./*

# copy the files to final results location
cp -f ./* ../../../../../../augustus_config/species/agloss/

# move back to main project dir
cd -
```

S.7.4.3 – SNAP

Training SNAP after the second round of MAKER in preparation for the third round occurred in the same manner as the first time. The exceptions were that (a) the input GFF3 file came from gFACs instead of directly from MAKER, (b) the `maker2zff` command had to be modified, and (c) the occurrences of `rnd1` in the commands and names were changed to `rnd2`. The `maker2zff` script provided by MAKER that was modified is referred to as `maker2zff_v2`. The only change required was to use `exon` instead of `CDS` on line 142. The commands are replicated (and appropriately modified) again here:

```
mkdir -p snap/rnd2

ln -s \
    ../../gfacs/rnd2/output/agloss-rnd2_withSeq.gff \
    snap/rnd2/genome.gff

cd snap/rnd2

maker2zff_v2 -n genome.gff

fathom \
    genome.ann genome.dna \
    -gene-stats \
    > gene-stats.log

fathom \
    genome.ann genome.dna \
    -validate \
    > validate.log

fathom \
    genome.ann genome.dna \
    -categorize 1000 \
    > categorize.log

fathom \
    uni.ann uni.dna \
    -export 1000 -plus \
    > export.log

forge \
    export.ann export.dna \
    > forge.log

hmm-assembler.pl \
    genome params \
    > genome.hmm
```

S.7.5 – MAKER Round #3

The third round of MAKER was run much the same way as the second, with a few modifications. First, the third round was run in a separate directory: `maker/rnd3`. The `trna` flag was used to ensure MAKER ran tRNAscan-SE v1.3.1 (Chan and Lowe 2019). The control files were copied from the second round and the following changes were made to `maker_opts.ctl`:

```
# gene prediction
snaphmm=/path/to/project/snap/rnd2/genome.hmm
trna=1
```

Additionally, the same accessory scripts, renaming, etc. was performed after this third round of MAKER as with the second round. The only difference being `rnd2` replaced with `rnd3` in all the commands and names (the `awk` commands were again skipped).

S.7.6 – MAKER Post-processing and Functional Annotation

The structural annotations created by MAKER required some modest post-processing before adding functional annotations. MAKER accessory scripts were used to update sequence names from the long MAKER names to friendlier ones. Other MAKER scripts were used to update the fasta and/or gff3 files with functional annotations found with the BLAST+ Suite v2.9.0 (Altschul et al. 1990; Camacho et al. 2009) and InterProScan v5.45-80.0 (Jones et al. 2014; Mitchell et al. 2019).

```
# create and move to a working dir
mkdir -p maker/post
cd maker/post

# copy the requisite output files
cp ../rnd3/*.gff ../rnd3/*.fa .
cp ../rnd1/agloss-rnd1_{repeats,{est,protein}2genome}.gff .

# remove the rnd info
rename \ # Perl version, not Linux util
    's/-rnd[1-3]//' \
    *.fa *.gff

# map new ids to MAKER names
NUM_SEQS=`grep -Ev '^#' agloss_noSeq.gff \
    | cut -d "\t" -f 9 | tr ';' '\n' \
    | cut -d '=' -f 2 | sort -u | wc -l`
```

```
maker_map_ids \
    --initial=1 \
    --prefix=Albula-glossodonta \
    --suffix='-?%' \
    --iterate=1 \
    --justify=${#NUM_SEQS} \
    agloss_withSeq.gff \
    > identifiers_map.tsv

# rename based on new ids
for FASTA in *.fa
do
    cp -f "${FASTA}" "${FASTA%.fa}_renamed.fa"
    map_fasta_ids identifiers_map.tsv "${FASTA%.fa}_renamed.fa"
done

for GFF in *.gff
do
    cp -f "${GFF}" "${GFF%.gff}_renamed.gff"
    map_gff_ids identifiers_map.tsv "${GFF%.gff}_renamed.gff"
done

# prep for functional annotation
cd /path/to/swissprot

makeblastdb \
    -dbtype prot \
    -in uniprot_sprot.fa \
    -input_type fasta \
    -title uniprot_sprot \
    -hash_index \
    -out uniprot_sprot \
    -logfile uniprot_sprot_makeblastdb.log

cd -

# do the alignment for func. annot.
blastp \
    -task blastp \
    -query proteins_renamed.fa \
    -db /path/to/swissprot/uniprot_sprot \
    -num_threads ${THREADS} \
    -max_target_seqs 1 \
    -max_hsps 1 \
    -evalue 1e-6 \
    -outfmt 6 \
    -out proteins-x-uniprotSprot_fmt6.tsv
```

```
# update the fasta and gff files with func. annots.
for FASTA in *_renamed.fa
do
    maker_functional_fasta \
        /path/to/swissprot/unitprot_sprot.fa \
        proteins-x-uniprotSprot_fmt6.tsv \
        ${FASTA} \
        > ${FASTA%.fa}_putative-function.fa
done

for GFF in *_renamed.gff
do
    maker_functional_gff \
        /path/to/swissprot/unitprot_sprot.fa \
        proteins-x-uniprotSprot_fmt6.tsv \
        ${GFF} \
        > ${GFF%.gff}_putative-function.gff
done

# run interproscan for more func. annots.
interproscan.sh \
    -m "standalone" \
    -cpu ${THREADS} \
    -T "${TMP}" \
    -appl "pfam" \
    -dp \
    -f "TSV" \
    -goterms \
    -iprlookup \
    -pa \
    -t "p" \
    -i proteins_renamed.fa\
    -o proteins-interproscan.tsv

# update the gff files with interproscan results
for GFF in {with,no}Seq_renamed_putative-function.gff
do
    ipr_update_gff \
        ${GFF} \
        proteins-interproscan.tsv \
        > ${GFF%.gff}_domain-added.gff
done

for GFF in {with,no}Seq_renamed.gff
do
    iprscan2gff3 \
        proteins-interproscan.tsv \
        ${GFF} \
        > ${GFF%.gff}_visible-iprscan-domains.gff
done
```

```
cd ../..
```

SUPPLEMENTAL REFERENCES

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. 1990. Basic Local Alignment Search Tool. Journal of Molecular Biology. 215:403-410.

Broad Institute. 2019. Picard Toolkit. Broad Institute, GitHub repository: Broad Institute.

Caballero, M. and J. Wegrzyn. 2019. gFACs: Gene Filtering, Analysis, and Conversion to Unify Genome Annotations Across Alignment and Gene Prediction Frameworks. Genomics, Proteomics & Bioinformatics. 17(3):305-310.

Camacho, C., G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden. 2009. BLAST+: architecture and applications. BMC Bioinformatics. 10:421.

Chan, P. P. and T. M. Lowe. 2019. tRNAscan-SE: Searching for tRNA Genes in Genomic Sequences. Methods in Molecular Biology. 1962:1-14.

Ghurye, J., M. Pop, S. Koren, D. Bickhart, and C.-S. Chin. 2017. Scaffolding of long read assemblies using long range contact information. BMC Genomics. 18(1):1-11.

Ghurye, J., A. Rhie, B. P. Walenz, A. Schmitt, S. Selvaraj, M. Pop, A. M. Phillippy, and S. Koren. 2019. Integrating Hi-C links with assembly graphs for chromosome-scale assembly. PLoS Computational Biology. 15(8):1–19.

Grabherr, M. G., B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. Di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev. 2011. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nature Biotechnology. 29(7):644-652.

Haas, B. J., S. L. Salzberg, W. Zhu, M. Pertea, J. E. Allen, J. Orvis, O. White, C. R. Buell, and J. R. Wortman. 2008. Automated eukaryotic gene structure annotation using EVidenceModeler and the Program to Assemble Spliced Alignments. Genome Biology. 9(1):R7.

Haghshenas, E., F. Hach, S. C. Sahinalp, and C. Chauve. 2016. CoLoRMap: Correcting Long Reads by Mapping short reads. Bioinformatics. 32:i545-i551.

Hamid, M., H. Khan, and I. Birol. 2017. ntCard: a streaming algorithm for the cardinality estimation of genomics data. Bioinformatics. 33(9):1324-1330.

Holt, C. and M. Yandell. 2011. MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. BMC Bioinformatics. 12:491.

Jones, P., D. Binns, H.-Y. Chang, M. Fraser, W. Li, C. Mcanulla, H. Mcwilliam, J. Maslen, A. Mitchell, G. Nuka, S. Pesseat, A. F. Quinn, A. Sangrador-Vegas, M. Scheremetjew, S.-Y.

Yong, R. Lopez, and S. Hunter. 2014. InterProScan 5: genome-scale protein function classification. Bioinformatics. 30(9):1236-1240.

Kelley, D. R., M. C. Schatz, and S. L. Salzberg. 2010. Quake: quality-aware detection and correction of sequencing errors. Genome Biology. 11:R116.

Kim, D., B. Langmead, and S. L. Salzberg. 2015. HISAT: a fast spliced aligner with low memory requirements. Nature Methods. 12(4):357-360.

Koren, S., B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy. 2017. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome research. 27(5):722-736.

Kriventseva, E. V., D. Kuznetsov, F. Tegenfeldt, M. Manni, R. Dias, F. A. Simão, and E. M. Zdobnov. 2019. OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. Nucleic acids research. 47(D1):D807-D811.

Langmead, B. and S. L. Salzberg. 2012. Fast gapped-read alignment with Bowtie 2. Nature Methods. 9(4):357-359.

Li, H. 2013. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv.

Li, H. 2020. auN: a new metric to measure assembly contiguity. Heng Li's Blog.

Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and G. P. D. P. Subgroup. 2009. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 25(16):2078-2079.

Marcais, G. and C. Kingsford. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 27(6):764-770.

Melsted, P. and J. K. Pritchard. 2011. Efficient counting of k-mers in DNA sequences using a bloom filter. BMC Bioinformatics. 12(333)

Mitchell, A. L., T. K. Attwood, P. C. Babbitt, M. Blum, P. Bork, A. Bridge, S. D. Brown, H.-Y. Chang, S. El-Gebali, M. I. Fraser, J. Gough, D. R. Haft, H. Huang, I. Letunic, R. Lopez, A. Luciani, F. Madeira, A. Marchler-Bauer, H. Mi, D. A. Natale, M. Necci, G. Nuka, C. Orengo, A. P. Pandurangan, T. Paysan-Lafosse, S. Pesseat, S. C. Potter, M. A. Qureshi, N. D. Rawlings, N. Redaschi, L. J. Richardson, C. Rivoire, G. A. Salazar, A. Sangrador-Vegas, C. J. A. Sigrist, I. Sillitoe, G. G. Sutton, N. Thanki, P. D. Thomas, S. C. E. Tosatto, S.-Y. Yong, and R. D. Finn. 2019. InterPro in 2019: improving coverage, classification and access to protein sequence annotations. Nucleic acids research. 47(D1):D351-D360.

Patro, R., G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford. 2017. Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods. 14(4):417-419.

Quinlan, A. R. and I. M. Hall. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 26(6):841-842.

R Core Team. 2017. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing.

Simão, F. A., R. M. Waterhouse, P. Ioannidis, E. V. Kriventseva, and E. M. Zdobnov. 2015. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics. 31(19):3210-3212.

Song, L. and L. Florea. 2015. Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads. GigaScience. 4(48)

Song, L., D. S. Shankar, and L. Florea. 2016. Rascaf: Improving Genome Assembly with RNA Sequencing Data. The Plant Genome. 9(3):1-12.

Vaser, R., I. Sović, N. Nagarajan, and M. Šikić. 2017. Fast and accurate de novo genome assembly from long uncorrected reads. Genome research. 27(5):737-746.

Yee, T. W. and C. J. Wild. 1996. Vector Generalized Additive Models. Journal of Royal Statistical Society, Series B. 58(3):481-493.

APPENDIX 3

Chapter 2 – Additional File 2

SUPPLEMENTAL TABLES

— for —

Genome Assembly of the Roundjaw Bonefish (*Albula glossodonta*), a Vulnerable Circumtropical Sportfish

Table S1. Sampling sites for *A. glossodonta* for population genomic analyses. The number of individuals (N) after data filtering are displayed for each atoll and island group.

Island Group	Atoll	N (Atoll)	N (Island group)
Amirantes	St Joseph	17	17
Farquhar	Farquhar	8	17
	Providence	9	
Aldabra	Aldabra	8	14
	Cosmoledo	6	
Mauritius	St. Brandon	18	18

Table S2. BUSCO statistics for the RNA transcripts and genomic assemblies

	Complete (%)	Complete Single-Copy (%)	Complete Duplicated (%)	Fragmented (%)	Missing (%)	Total
Transcriptome						
Trinity Transcripts	3,144 (86.4)	1,241 (34.1)	1,903 (52.3)	128 (3.5)	368 (10.1)	3,640
Genome						
Canu Contigs	3,485 (95.7)	3,081 (84.6)	404 (11.1)	22 (0.6)	133 (3.7)	3,640
RaCon Polished Contigs	3,484 (95.7)	3,076 (84.5)	408 (11.2)	22 (0.6)	134 (3.7)	3,640
SALSA Scaffolds	3,480 (95.6)	3,074 (84.5)	406 (11.2)	27 (0.7)	133 (3.7)	3,640
SALSA + Rascaf Scaffolds	3,481 (95.6)	3,076 (84.5)	405 (11.1)	25 (0.7)	134 (3.7)	3,640

Table S3. Input parameters for *ipyrad* used to assemble ddRAD data to the *A. glossodonta* reference genome

Parameter	Description	Input
assembly_method	Assembly method	reference
datatype	Datatype	ddrad
restriction_overhang	Restriction overhang (cut1,) or (cut1, cut2)	TGCAG, CCG
max_low_qual_bases	Max low quality base calls (Q<20) in a read	5
phred_Qscore_offset	phred Q score offset	33
mindepth_statistical	Min depth for statistical base calling	6
mindepth_majrule	Min depth for majority-rule base calling	6
maxdepth	Max cluster depth within samples	10000
clust_threshold	Clustering threshold for de novo assembly	0.9
max_barcode_mismatch	Max number of allowable mismatches in barcodes	0
filter_adapters	Filter for adapters/primers	2
filter_min_trim_len	Min length of reads after adapter trim	35
max_alleles_consens	Max alleles per site in consensus sequences	2
max_Ns_consens	Max N's (uncalled bases) in consensus	0.05
max_Hs_consens	Max Hs (heterozygotes) in consensus	0.05
min_samples_locus	Min # samples per locus for output	10
max_SNPs_locus	Max # SNPs per locus	0.2
max_Indels_locus	Max # of indels per locus	8
max_shared_Hs_locus	Max # heterozygous sites per locus	0.5
trim_reads	Trim raw read edges (R1>, <R1, R2>, <R2)	0, 0, 0, 0
trim_loci	Trim locus edges (R1>, <R1, R2>, <R2)	0, 0, 0, 0

Table S4. Data filtering steps implemented in VCFtools and PLINK after assembly in *ipyrad*

SNP Quality Filters	
Genotype Calls	Remove individuals missing > 98% genotype calls
Indels	Remove indels
Read Depth	Remove loci with mean depth > 100
Singletons and minor alleles	Retain sites with a minor allele frequency > 0.05 and minor allele count ≥ 2
Biallelic SNPs	Max alleles = 2
Missing Data	
	Remove loci with genotype call rate < 40%
	Remove individuals missing > 60% genotype calls
	Remove loci with genotype call rate < 60%
	Remove individuals missing > 50% genotype calls
	Remove loci with genotype call rate < 75%
Hardy-Weinberg Equilibrium	Remove loci out of HWE (0.05)
Linkage Disequilibrium	Remove loci within 1kb windows with r2 > 0.6

Table S5. Observed heterozygosity (H_O) and expected heterozygosity (H_S) for each island group

Island Group	H_O	H_S
Amirantes	0.2800	0.2915
Farquhar	0.2901	0.2946
Aldabra	0.2589	0.2862
Mauritius	0.2829	0.2923

APPENDIX 4

Chapter 3 – Supplementary File 1

SUPPLEMENTARY BIOINFORMATICS METHODS

An overview of the methods used in this study was provided in the main manuscript. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided here.

Read Error Correction

The self-corrected reads were generated using Canu v1.6 (Koren et al. 2017) with the following command:

```
canu -correct \
    -s ${SETTINGS_FILE} \
    -d ${OUTPUT_DIR_NAME} \
    -p ${OUTPUT_PREFIX} \
    -pacbio-raw \
    ${INPUT_PACBIO_READS[@]}
```

The relevant lines of the setting file are included here:

```
genomeSize=782400000
ovsMethod=sequential
gridEngine=slurm
```

Genome Assembly and Scaffolding

The individual steps of genome assembly and scaffolding will each be described separately. Calculation of assembly summary statistics will also be described.

Genome Assembly

The assembly was created with Canu v1.6 (Koren et al. 2017) using the already corrected reads from the "self" correction strategy using the following command:

```
canu -trim-assemble \
    -s ${SETTINGS_FILE} \
    -d ${OUTPUT_DIR_NAME} \
    -p ${OUTPUT_PREFIX} \
    -pacbio-corrected \
    ${INPUT_SELF_CORRECTED_PACBIO_READS_FILE}
```

Scaffolding

The RNA-seq data were aligned using HiSat v0.1.6-beta (Kim et al. 2015), and the alignments were converted from SAM to BAM format and sorted using SAMtools v1.6 (Li et al. 2009). First, the assembly (contigs from Canu) was indexed with HiSat. For each tissue (i.e., brain, eye, fin, gill, heart, kidney, liver, and muscle), HiSat aligned reads to the assembly, SAMtools sorted and compressed the output alignments, and Rascaf downloaded June 2018 (Song et al. 2016) computed how scaffolding could be done. The actual scaffolding was done with Rascaf in a single step after all steps had been completed for each tissue. The process is described in the following script:

```
hisat-build \
    ${HISAT_IDX_PREFIX} \
    ${HIC_SCAFFOLDS}


for TISSUE in {brain,eye,fin,gill,heart,kidney,liver,muscle}
do
    RNASEQ_READS_LEFT=${TISSUE}_L.fq.gz
    RNASEQ_READS_RIGHT=${TISSUE}_R.fq.gz
    ALIGNMENT_SAM=${TISSUE}_aln.sam

    hisat \
        -p ${THREADS} \
        --phred33 -q -t \
        -x ${HISAT_IDX_PREFIX} \
        -1 ${RNASEQ_READS_LEFT} \
        -2 ${RNASEQ_READS_RIGHT} \
        -S ${ALIGNMENT_SAM}
```

```
    samtools view \
        -buh ${ALIGNMENT_SAM} \
        | samtools sort \
            -@ ${THREADS} \
            -m ${MEMORY}M \
            -O BAM \
            -o ${ALIGNMENT_BAM}

    rascaf \
        -breakN 1 \
        -b ${ALIGNMENT_BAM} \
        -f ${HIC_SCAFFOLDS} \
        -o ${TISSUE}.out
done

rascaf-join \
    -r gill.out \
    -r heart.out \
    -r liver.out \
    -o ${OUTPUT_FILE_PREFIX}
```

Assembly Statistics

Assembly continuity statistics, e.g., N50 and auN (Li 2020), were calculated with caln50 downloaded April 2020 (https://github.com/lh3/calN50) and a custom Python (https://www.python.org) script. caln50 is run using the following simple command:

```
caln50 \
    -s 0.01 \
    -L ${GENOME_SIZE} \
    ${CONTIGS_OR_SCAFFOLDS_FILE} \
    > ${STATISTICS_FILE}
```

The custom Python script is not efficient, but it does calculate Nx, Lx, NGx, and LGx, as well as a few other interesting points about sequences in a fasta file. This script is too long to realistically represent when embedded in the text; it is available on GitHub at https://github.com/pickettbd/basicAsmStatsCalcInPy.

Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 (Simão et al. 2015) and OrthoDB v10 (Kriventseva et al. 2019). The BUSCO config file was the

not modified from the default aside from the locations of OrthoDB v10 and the binary executables for BUSCO. It was run based on the following command structure:

```
busco \
    --offline \
    --config ${BUSCO_CONFIG_FILE} \
    --cpu ${THREADS} \
    --in ${CONTIGS_OR_SCAFFOLDS_FASTA} \
    --out_path ${OUTPUT_DIR} \
    --out ${OUTPUT_FILE_PREFIX} \
    --mode genome \
    --lineage actinopterygii \
    --augustus_species zebrafish
```

Transcriptome Assembly

The transcripts were assembled using Trinity v2.6.6 (Grabherr et al. 2011), which depended on Bowtie v2.3.4.3 (Langmead and Salzberg 2012), Jellyfish v2.2.10 (Marcais and Kingsford 2011), salmon v0.12 (Patro et al. 2017), and SAMtools v1.6 (Li et al. 2009):

```
trinity \
    --no_version_check \
    --max_memory ${MEMORY} \
    --CPU ${THREADS} \
    --long_reads ${DUAL_CORRECTED_PACBIO_READS} \
    --seqType fq \
    --left ${RNASEQ_READS_LEFT} \
    --right ${RNASEQ_READS_RIGHT} \
    --SS_lib_type FR \
    --normalize_max_read_cov 50 \
    --normalize_by_read_set \
    --min_contig_length 200 \
    --output ${TRINITY_OUTPUT_DIR}
```

Assembly correctness was assessed using single-copy orthologs with BUSCO v4.0.6 (Simão et al. 2015) and OrthoDB v10 (Kriventseva et al. 2019). The command and config file were a match to how BUSCO was run to assess genome assembly correctness, except that the `--mode` option was `transcriptome` instead of `genome`.

Computational Annotation

The MAKER v3.01.02-beta (Holt and Yandell 2011) pipeline was used to annotate the assembly. With a large enough cluster with MPI support, MAKER runs relatively quickly for each round. The general process was described in prose in the main manuscript, but it can be summarized in outline form here:

I. MAKER round #1
II. *ab initio* gene predictors
    a. AUGUSTUS
    b. GeneMark-ES
    c. SNAP
III. MAKER round #2
IV. *ab initio* gene predictors
    d. AUGUSTUS
    e. SNAP
V. MAKER round #3
VI. MAKER post-processing & functional annotation

As each round of MAKER was run in a nearly identical fashion, the process will be described once, followed by differences between the rounds. Similarly, AUGUSTUS and SNAP will also be described once.

MAKER Round #1

The command to run MAKER is straight-forward, though may vary slightly depending on the implementation of MPI employed by the cluster. The MAKER documentation says to run MAKER with the `mpiexec` command, but `mpirun` was successful for our setup. Running MAKER from a working directory on an NFS drive will almost certainly result in failure unless MAKER is directed where to do its work in a non-NFS temporary directory. This required some

extra attention to job cleanup on our cluster, but it was successful when we pointed MAKER to the local drives on the nodes on which it was run, which were mounted at `/tmp`. When calling MAKER from the directory in which the control files exist, the command to start MAKER looks like this:

```
mpirun maker \
    -cpus ${CPUS} \
    -TMP ${MAKER_TMP_DIR}
```

The truly critical parts are in the MAKER control files. Assuming one has a successfully installed and configured version of MAKER available, default control files can be generated in the working directory by running the following command: `maker -CTL`. No modifications were made to the `maker_evm.ctl` file. The `maker_bopt.ctl` file was left unchanged as well. Note that `use_rapsearch` was set to `0` and `blast_type` was set to `ncbi+`. The `maker_exe.ctl` file was modified as needed only to set correct paths to the executables for MAKER's dependencies. The following shows the modified or otherwise relevant lines from the `maker_opts.ctl` file:

```
# genome
genome=/path/to/scaffolds.fa
organism_type=eukaryotic

#re-annotation
maker_gff=
est_pass=0
protein_pass=0
rm_pass=0
model_pass=0
pred_pass=0
other_pass=0

# est/rna-seq
est=/path/to/Trinity/transcripts.fa
est_gff=

# protein homology
protein=/path/to/uniprot_sprot.fa
protein_gff=
```

```
# repeat masking
model_org=all
rmlib=/path/to/RepeatModeler/results/assembly-db-families.fa
repeat_protein=/path/to/maker-install-dir/data/te_proteins.fa
rm_gff=
softmask=1

# gene prediction
snaphmm=
gmhmm=
augustus_species=
pred_gff=
model_gff=
run_evm=0
est2genome=1
protein2genome=1
trna=0

# maker behavior
max_dna_len=1000000
min_contig=20000

pred_flank=200
pred_stats=0
AED_threshold=1
min_protein=0
alt_splice=0
always_complete=0
map_forward=0
keep_preds=0

split_hit=10000
min_intron=20
single_exon=0
single_length=250
correct_est_fusion=0
```

Once MAKER has completed, a few MAKER accessory scripts can be run to extract the results from its datastore located at `${PROJECT_DIR}/maker/rnd1/*.datastore`. Additional modifications (shown) can also be employed to make output names more palatable. For sake of demonstration, we assume the master datastore index log file is prefixed with `scaffolds`, and the output base (`-o` option for `fasta_merge`) is `cmel-rnd1` (*C. melampygus* round 1)):

```
cd maker/rnd1/scaffolds.maker.output
```

```
fasta_merge \
    -o cmel-rnd1 \
    -d scaffolds_master_datastore_index.log

gff3_merge \
    -n -s \
    -d scaffolds_master_datastore_index.log \
    > cmel-rnd1_noSeq.gff

cd scaffolds_datastore

rename 's/.all.maker./_/' *.fasta # Perl rename, not Linux util
rename 's/fasta/fa/' *.fasta      # Perl rename, not Linux util

awk '{if ($2 == "est2genome") print $0}' \
    cmel-rnd1_noSeq.gff \
    > cmel-rnd1_est2genome.gff

awk '{if ($2 == "protein2genome") print $0}' \
    cmel-rnd1_noSeq.gff \
    > cmel-rnd1_protein2genome.gff

awk '{if ($2 ~ "repeat") print $0}' \
    cmel-rnd1_noSeq.gff \
    > cmel-rnd1_repeats.gff

mv cmel-rnd1*.fa cmel-rnd1*.gff ../..

cd ../../../..
```

ab initio Gene Prediction

Three *ab initio* gene prediction programs were run between MAKER rounds 1 and 2. AUGUSTUS and SNAP can take gene models as input, and they are thus able to be run with new models after rounds 1 and 2 of MAKER in preparation for rounds 2 and 3, respectively. GeneMark-ES does not take gene models as input, and it thus needs to be run only one time.

GeneMark-ES

GeneMark-ES required a software key to be run, which can be obtained or re-obtained for free for academic use at any time. GeneMark-ES also requires a configuration file to be run;

the default configuration file was used. The following command demonstrates how to run GeneMark-ES:

```
gmes_petap.pl \
    --ES \
    --usr_cfg ${COPY_OF_DEFAULT_CONFIG_FILE} \
    --cores ${THREADS} \
    --sequence ${SCAFFOLDS_ASSEMBLY_FILE}
```

AUGUSTUS

AUGUSTUS training can be handled with BUSCO. Before AUGUSTUS can be trained, configuration files and data from AUGUSTUS and BUSCO will need to be copied to the working directory for this part of the analysis, and the relevant environment variables will need to be reset (which assumes they are properly set in the first place):

```
cp  -r ${AUGUSTUS_CONFIG_PATH} ${PROJECT_DIR}/augustus_config
export AUGUSTUS_CONFIG_PATH=${PROJECT_DIR}/augustus_config

cp  ${BUSCO_CONFIG_FILE} ${PROJECT_DIR}/busco_config.ini
export BUSCO_CONFIG_FILE=${PROJECT_DIR}/busco_config.ini
```

No changes were made to the AUGUSTUS files. The only change made to the BUSCO configuration file was to set `download_path=/path/to/odb10` instead of `./busco_download`. This is assuming OrthoDB v10 has already been downloaded to that location and that the `--offline` flag will be used when running BUSCO. Before training AUGUSTUS, candidate gene regions need to be extracted. This was done with a custom Python script (available at https://github.com/pickettbd/caranx-melampygus_assembly-paper_misc-scripts) and BEDTools v2.28.0 (Quinlan and Hall 2010).

```
python3 generateBedForMrnaExtraction.py \
    maker/rnd1/cmel-rnd1_noSeq.gff \
    scaffolds.fa \
    candidates-rnd1.bed

bedtools getfasta \
    -fi scaffolds.fa \
    -bed candidates-rnd1.bed \
    -fo candidates-rnd1.fa
```

AUGUSTUS was trained by running BUSCO with the same command described in the Assembly Statistics section (i.e., `mode=genome`, `lineage=actinopterygii`, `augustus_species=zebrafish`). To make the AUGUSTUS training parameters generated after running BUSCO available to the next round of MAKER, some post-processing is required:

```
# make dir for final results
mkdir augustus_config/species/cmel

# move to results location
cd "busco-augustus/cmel-rnd1/
    run_actinopterygii_odb10/augustus_output/
    retraining_parameters/BUSCO_cmel-rnd1"

# rename some files and their references to eachother
rename \ # Perl rename, not Linux util
    's/BUSCO_(cmel-rnd1_)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/BUSCO_(cmel-rnd1_)/\1/' \
    ./cmel-rnd1_parameters.cfg*

# do it again, removing the rnd info
rename \ # Perl rename, not Linux util
    's/(cmel)-rnd1)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/(cmel)-rnd1/\1/' \
    ./*

# copy the files to final results location
cp -f ./* ../../../../../../augustus_config/species/cmel/
```

```
# move back to main project dir
cd -
```

SNAP

Training with SNAP is much less resource intensive than training AUGUSTUS. Most, if not all, of the commands can reasonably be run "locally" on a login node or other machine. The final output file, `genome.hmm`, is what will be provided to the next round of MAKER. Inspection of the log files was performed after each step. The process of training SNAP can be described by the following commands:

```
mkdir -p snap/rnd1

ln -s \
    ../../maker/rnd1/cmel-rnd1_withSeq.gff \
    snap/rnd1/genome.gff

cd snap/rnd1

maker2zff genome.gff

fathom \
    genome.ann genome.dna \
    -gene-stats \
    > gene-stats.log

fathom \
    genome.ann genome.dna \
    -validate \
    > validate.log

fathom \
    genome.ann genome.dna \
    -categorize 1000 \
    > categorize.log

fathom \
    uni.ann uni.dna \
    -export 1000 -plus \
    > export.log

forge \
    export.ann export.dna \
    > forge.log
```

```
hmm-assembler.pl \
    genome params \
    > genome.hmm
```

MAKER Round #2

The second round of MAKER was run much the same way as the first, with a few modifications. First, the second round was run in a separate directory: `maker/rnd2`. The `run_evm` flag was set to enable MAKER to run EVidenceModeler v1.1.1 (Haas et al. 2008). The control files were copied from the first round and the following changes were made to `maker_opts.ctl`:

```
# est/rna-seq
est=
est_gff=/path/to/project/maker/rnd1/cmel-rnd1_est2genome.gff

# protein homology
protein=
protein_gff=/path/to/project/maker/rnd1/cmel-rnd1_protein2genome.gff

# repeat masking
model_org=
rmlib=
repeat_protein=
rm_gff=/path/to/project/maker/rnd1/cmel-rnd1_repeats.gff

# gene prediction
snaphmm=/path/to/project/snap/rnd1/genome.hmm
gmhmm=/path/to/project/gmes/output/gmhmm.mod
augustus_species=cmel
run_evm=1
est2genome=0
protein2genome=0
```

Additionally, the same accessory scripts, renaming, etc. was performed after this second round of MAKER as with the first round. The only differences being that `rnd1` was replaced with `rnd2` in all the commands and names and the `awk` commands were skipped.

ab initio Gene Prediction

Since GeneMark-ES does not take gene models as input, only SNAP and AUGUSTUS could be re-run after MAKER's second round. Before training them, the models from MAKER were filtered using gFACs v1.1.1 (Caballero and Wegrzyn 2019).

gFACs Filtering

In an attempt to improve the quality of gene models being used for this final round of training with AUGUSTUS and SNAP, gFACs was employed to filter out models with single-exon genes, introns shorter than 20bp, etc. The gFACs command and relevant supporting commands (e.g., creating working directories) are shown here:

```
mkdir -p gfacs/rnd2

ln -s \
    ../../maker/rnd2/cmel-rnd2_noSeq.gff \
    gfacs/rnd2/orig_noSeq.gff

ln -s \
    ../../assembly/scaffolds.fa \
    gfacs/rnd2/assembly.fa

awk \
    'BEGIN{x=0;}/^##FASTA/{x=1;}{if(x){print $0;}}' \
    maker/rnd2/cmel-rnd2_withSeq.gff \
    > gfacs/rnd2/orig_onlySeq.gff

cd gfacs/rnd2

gFACs.pl \
    -f "maker_2.31.9_gff" \
    -p  ./output/cmel-rnd2_noSeq \
    --statistics-at-every-step \
    --statistics \
    --rem-monoexonics \
    --min-exon-size 20 \
    --min-intron-size 20 \
    --min-CDS-size 74 \
    --fasta assembly.fa \
    --splice-table \
    --nt-content \
    --canonical-only \
    --rem-genes-without-stop-codon \
    --allowed-inframe-stop-codons 0 \
```

```
    --create-gff3 \
    --get-fasta-with-introns \
    --get-fasta-without-introns \
    --get-protein-fasta \
    --distributions \
    exon_lengths \
    intron_lengths \
    CDS_lengths \
    gene_lengths \
    exon_position \
    exon_position_data \
    intron_position \
    intron_position_data \
    -O ./output \
    orig_noSeq.gff

ln -s \
    cmel-rnd2_noSeq_out.gff3 \
    output/cmel-rnd2_noSeq.gff

cat \
    output/cmel-rnd2_noSeq.gff orig_onlySeq.gff \
    > output/cmel-rnd2_withSeq.gff

cd ../..
```

AUGUSTUS

Training AUGUSTUS after the second round of MAKER in preparation for the third round occurred in the same manner as the first time. The exceptions were that (a) the input GFF3 file came from gFACs instead of directly from MAKER, (b) `augustus_species=cmel` was used instead of `augustus_species=zebrafish`, and (c) the occurrences of `rnd1` in the commands and names were changed to `rnd2`. The commands are replicated (and appropriately modified) again here:

```
python3 generateBedForMrnaExtraction.py \
    gfacs/rnd2/output/cmel-rnd2_noSeq.gff \
    scaffolds.fa \
    candidates-rnd2.bed
```

```
bedtools getfasta \
    -fi scaffolds.fa \
    -bed candidates-rnd2.bed \
    -fo candidates-rnd2.fa
```

AUGUSTUS was trained by running BUSCO with the same command described in the Assembly Statistics section (i.e., `mode=genome` and `lineage=actinopterygii`) except that `augustus_species=cmel` instead of `zebrafish`. To make the AUGUSTUS training parameters generated after running BUSCO available to the next round of MAKER, some post-processing is required:

```
# move to results location
cd "busco-augustus/cmel-rnd2/
    run_actinopterygii_odb10/augustus_output/
    retraining_parameters/BUSCO_cmel-rnd2"

# rename some files and their references to each other
rename \ # Perl rename, not Linux util
    's/BUSCO_(cmel-rnd2_)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/BUSCO_(cmel-rnd2_)/\1/' \
    ./cmel-rnd1_parameters.cfg*

# do it again, removing the rnd info
rename \ # Perl rename, not Linux util
    's/(cmel)-rnd2)/$1/' \
    ./*

sed \ # gnu sed
    -i -r \
    's/(cmel)-rnd2/\1/' \
    ./*

# copy the files to final results location
cp -f ./* ../../../../../../augustus_config/species/cmel/

# move back to main project dir
cd -
```

SNAP

Training SNAP after the second round of MAKER in preparation for the third round occurred in the same manner as the first time. The exceptions were that (a) the input GFF3 file came from gFACs instead of directly from MAKER, (b) the `maker2zff` command had to be modified, and (c) the occurrences of `rnd1` in the commands and names were changed to `rnd2`. The `maker2zff` script provided by MAKER that was modified is referred to as `maker2zff_v2`. The only change required was to use `exon` instead of `CDS` on line 142. The commands are replicated (and appropriately modified) again here:

```
mkdir -p snap/rnd2

ln -s \
    ../../gfacs/rnd2/output/cmel-rnd2_withSeq.gff \
    snap/rnd2/genome.gff

cd snap/rnd2

maker2zff_v2 -n genome.gff

fathom \
    genome.ann genome.dna \
    -gene-stats \
    > gene-stats.log

fathom \
    genome.ann genome.dna \
    -validate \
    > validate.log

fathom \
    genome.ann genome.dna \
    -categorize 1000 \
    > categorize.log

fathom \
    uni.ann uni.dna \
    -export 1000 -plus \
    > export.log

forge \
    export.ann export.dna \
    > forge.log
```

```
hmm-assembler.pl \
    genome params \
    > genome.hmm
```

MAKER Round #3

The third round of MAKER was run much the same way as the second, with a few modifications. First, the third round was run in a separate directory: `maker/rnd3`. The `trna` flag was used to ensure MAKER ran tRNAscan-SE v1.3.1 (Chan and Lowe 2019). The control files were copied from the second round and the following changes were made to `maker_opts.ctl`:

```
# gene prediction
snaphmm=/path/to/project/snap/rnd2/genome.hmm
trna=1
```

Additionally, the same accessory scripts, renaming, etc. was performed after this third round of MAKER as with the second round. The only difference being `rnd2` replaced with `rnd3` in all the commands and names (the `awk` commands were again skipped).

MAKER Post-processing and Functional Annotation

The structural annotations created by MAKER required some modest post-processing before adding functional annotations. MAKER accessory scripts were used to update sequence names from the long MAKER names to friendlier ones. Other MAKER scripts were used to update the fasta and/or gff3 files with functional annotations found with the BLAST+ Suite v2.9.0 (Camacho et al. 2009; Altschul et al. 1990) and InterProScan v5.45-80.0 (Jones et al. 2014; Mitchell et al. 2019). BLAST was run using the annotated protein sequences as the query and UniProt/Swissprot as the subject database. The following options were used: `-task blastp -max_target_seqs 1 -max_hsps 1 -evalue 1e-6 -outfmt 6`. InterProScan was run using annotated proteins as input (same as BLAST) with the following options: `-appl pfam -dp -f TSV -goterms -iprlookup -pa -t p`.

```
# create and move to a working dir
mkdir -p maker/post
cd maker/post

# copy the requisite output files
cp ../rnd3/*.gff ../rnd3/*.fa .
cp ../rnd1/cmel-rnd1_{repeats,{est,protein}2genome}.gff .

# remove the rnd info
rename \ # Perl version, not Linux util
    's/-rnd[1-3]//' \
    *.fa *.gff

# map new ids to MAKER names
NUM_SEQS=`grep -Ev '^#' cmel_noSeq.gff \
    | cut -d "\t" -f 9 | tr ';' '\n' \
    | cut -d '=' -f 2 | sort -u | wc -l`

maker_map_ids \
    --initial=1 \
    --prefix=Caranx-melampygus\
    --suffix='-?%' \
    --iterate=1 \
    --justify=${#NUM_SEQS} \
    cmel_withSeq.gff \
    > identifiers_map.tsv

# rename based on new ids
for FASTA in *.fa
do
    cp -f "${FASTA}" "${FASTA%.fa}_renamed.fa"
    map_fasta_ids identifiers_map.tsv "${FASTA%.fa}_renamed.fa"
done

for GFF in *.gff
do
    cp -f "${GFF}" "${GFF%.gff}_renamed.gff"
    map_gff_ids identifiers_map.tsv "${GFF%.gff}_renamed.gff"
done

# prep for functional annotation
cd /path/to/swissprot

makeblastdb \
    -dbtype prot \
    -in uniprot_sprot.fa \
    -input_type fasta \
    -title uniprot_sprot \
    -hash_index \
    -out uniprot_sprot \
    -logfile uniprot_sprot_makeblastdb.log
```

```
cd -

# do the alignment for func. annot.
blastp \
    -task blastp \
    -query proteins_renamed.fa \
    -db /path/to/swissprot/uniprot_sprot \
    -num_threads ${THREADS} \
    -max_target_seqs 1 \
    -max_hsps 1 \
    -evalue 1e-6 \
    -outfmt 6 \
    -out proteins-x-uniprotSprot_fmt6.tsv

# update the fasta and gff files with func. annots.
for FASTA in *_renamed.fa
do
    maker_functional_fasta \
        /path/to/swissprot/unitprot_sprot.fa \
        proteins-x-uniprotSprot_fmt6.tsv \
        ${FASTA} \
        > ${FASTA%.fa}_putative-function.fa
done

for GFF in *_renamed.gff
do
    maker_functional_gff \
        /path/to/swissprot/unitprot_sprot.fa \
        proteins-x-uniprotSprot_fmt6.tsv \
        ${GFF} \
        > ${GFF%.gff}_putative-function.gff
done

# run interproscan for more func. annots.
interproscan.sh \
    -m "standalone" \
    -cpu ${THREADS} \
    -T "${TMP}" \
    -appl "pfam" \
    -dp \
    -f "TSV" \
    -goterms \
    -iprlookup \
    -pa \
    -t "p" \
    -i proteins_renamed.fa\
    -o proteins-interproscan.tsv

# update the gff files with interproscan results
for GFF in {with,no}Seq_renamed_putative-function.gff
do
    ipr_update_gff \
```

```
        ${GFF} \
        proteins-interproscan.tsv \
        > ${GFF%.gff}_domain-added.gff
done

for GFF in {with,no}Seq_renamed.gff
do
    iprscan2gff3 \
        proteins-interproscan.tsv \
        ${GFF} \
        > ${GFF%.gff}_visible-iprscan-domains.gff
done

cd ../..
```

Demographic History

The scripts to perform this analysis are available on GitHub (https://github.com/pickettbd/msmc-slurmPipeline) with supporting documentation.

SUPPLEMENTAL REFERENCES

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, 1990 Basic Local Alignment Search Tool. *J. Mol. Biol*. 215:403-410.

Caballero, M., and J. Wegrzyn, 2019 gFACs: Gene Filtering, Analysis, and Conversion to Unify Genome Annotations Across Alignment and Gene Prediction Frameworks. Genomics Proteomics *Bioinformatics* 17 (3):305-310.

Camacho, C., G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos *et al.*, 2009 BLAST+: architecture and applications. *BMC Bioinform*. 10:421.

Chan, P. P., and T. M. Lowe, 2019 tRNAscan-SE: Searching for tRNA Genes in Genomic Sequences. *Methods Mol. Biol*. 1962:1-14.

Grabherr, M. G., B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson *et al.*, 2011 Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat. Biotechnol.* 29 (7):644-652.

Haas, B. J., S. L. Salzberg, W. Zhu, M. Pertea, J. E. Allen *et al.*, 2008 Automated eukaryotic gene structure annotation using EVidenceModeler and the Program to Assemble Spliced Alignments. *Genome Biol.* 9 (1):R7.

Holt, C., and M. Yandell, 2011 MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinform*. 12:491.

Jones, P., D. Binns, H.-Y. Chang, M. Fraser, W. Li *et al.*, 2014 InterProScan 5: genome-scale protein function classification. *Bioinformatics* 30 (9):1236-1240.

Kim, D., B. Langmead, and S. L. Salzberg, 2015 HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12 (4):357-360.

Koren, S., B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman *et al.*, 2017 Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* 27 (5):722-736.

Kriventseva, E. V., D. Kuznetsov, F. Tegenfeldt, M. Manni, R. Dias *et al.*, 2019 OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Res.* 47 (D1):D807-D811.

Langmead, B., and S. L. Salzberg, 2012 Fast gapped-read alignment with Bowtie 2. *Nat. Methods* 9 (4):357-359.

Li, H., 2020 auN: a new metric to measure assembly contiguity in *Heng Li's Blog*.

Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan *et al.*, 2009 The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25 (16):2078-2079.

Marcais, G., and C. Kingsford, 2011 A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 27 (6):764-770.

Mitchell, A. L., T. K. Attwood, P. C. Babbitt, M. Blum, P. Bork *et al.*, 2019 InterPro in 2019: improving coverage, classification and access to protein sequence annotations. *Nucleic Acids Res.* 47 (D1):D351-D360.

Patro, R., G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford, 2017 Salmon provides fast and bias-aware quantification of transcript expression. *Nat. Methods* 14 (4):417-419.

Quinlan, A. R., and I. M. Hall, 2010 BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26 (6):841-842.

Simão, F. A., R. M. Waterhouse, P. Ioannidis, E. V. Kriventseva, and E. M. Zdobnov, 2015 BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* 31 (19):3210-3212.

Song, L., D. S. Shankar, and L. Florea, 2016 Rascaf: Improving Genome Assembly with RNA Sequencing Data. *Plant Genome* 9 (3):1-12.

APPENDIX 5

Chapter 4 – Supplementary File 1

SUPPLEMENTARY BIOINFORMATICS METHODS

An overview of the methods used in this study was provided in the main manuscript. Where appropriate, additional details, such as the code for custom scripts and the commands used to run software, are provided here.

Read Error Correction

The self-corrected reads were generated using Canu v1.8 [1] with the following command:

```
canu -correct \
    -s ${SETTINGS_FILE} \
    -d ${OUTPUT_DIR_NAME} \
    -p ${OUTPUT_PREFIX} \
    -pacbio-raw \
    ${INPUT_PACBIO_READS[@]}
```

The relevant lines of the setting file are included here:

```
genomeSize=625920000
ovsMethod=sequential
gridEngine=slurm
```

Genome Assembly and Scaffolding

The individual steps of genome assembly and scaffolding will each be described separately. Calculation of assembly summary statistics will also be described.

Genome Assembly

The assembly was created with Canu v1.8 [1] using the already corrected reads from the correction process using the following command:

```
canu -trim-assemble \
    -s ${SETTINGS_FILE} \
    -d ${OUTPUT_DIR_NAME} \
    -p ${OUTPUT_PREFIX} \
    -pacbio-corrected \
    ${INPUT_SELF_CORRECTED_PACBIO_READS_FILE}
```

Scaffolding and Mis-assembly Detection with Hi-C Data

Part of the scaffolding process with Hi-C data employed by SALSA is a mis-assembly detection step. The set of contigs created during this process will be pointed out as the scaffolding process is described. The Hi-C data (in this case, Dovetail Genomics Omni-C library using general endonucleases instead of site-specific restriction enzymes) alignments were performed following the Arima Genomics (San Diego, California, USA; https://arimagenomics.com) Mapping Pipeline commit #2e74ea4 (https://github.com/ArimaGenomics/mapping_pipeline), which relied on BWA-MEM2 v2.1 [2,3], Picard v2.19.2 [4], and SAMtools v1.9 [5]. As the pipeline is reasonably well-documented, it will be only summarized here:

1. The assembly (Canu contigs) is indexed using SAMtools `faidx`.
2. The assembly is indexed with bwa `index` and the Hi-C reads are mapped to the assembly with bwa `mem` (I used BWA-MEM2 instead).
3. The alignments are converted from SAM to BAM format with SAMtools `view`.
4. The 5' ends are filtered using SAMtools `view` and the Arima Genomics Perl (https://www.perl.org) script `filter_five_end.pl`.
5. Paired-end reads are combined into a single file with the Arima Genomics Perl script `two_read_bam_combiner.pl` and sorted with SAMtools `sort`. These reads will be treated as single-end hereafter.
6. Read groups are added to the BAM file using Picard `AddOrReplaceReadGroups`.

7. Merge technical replicates. This step was skipped because no such replicates existed.
8. Duplicates in the BAM file were marked using Picard `MarkDuplicates`.
9. Merge biological replicates. This step was skipped because no such replicates existed.
10. The final BAM file was indexed with SAMtools `index`.
11. Stats were reported with the Arima Genomics Perl script `get_stats.pl`.

Scaffolding was performed on the Canu contigs using the final BAM file from the Arima Genomics Mapping Pipeline with SALSA commit #974589f [6,7]. First, some pre-processing was required with BEDTools v2.28.0 [8] to convert the final BAM file from the mapping pipeline to BED format; this was then sorted. The BEDTools, sorting, and SALSA commands are listed here (note that the `${RESTRICTION_ENZYME_SEQ}` was `DNASE`):

```
bedtools bamtobed \
    -i ${FINAL_ARIMA_BAM_FILE} \
    > ${HIC_BED_FILE}

sort -k 4 \
    ${HIC_BED_FILE} \
    > ${SORTED_HIC_BED_FILE}

run_pipeline.py \
    -a ${CANU_CONTIGS_FILE} \
    -l ${CANU_CONTIGS_FAIDX_FILE} \
    -b ${SORTED_HIC_BED_FILE} \
    -e ${RESTRICTION_ENZYME_SEQ} \
    -s ${GENOME_SIZE} \
    -m yes \
    -o ${OUTPUT_SALSA_DIR}
```

Note that all newly-created gaps from SALSA will all be assigned a length of 500 nucleotides (i.e., 500 Ns in a row). Assuming these are gaps of unknown size, these will ideally be changed to 100 nucleotides for any submissions to GenBank. If you have multiple sources of evidence for gaps, you will want to keep track of which gaps were supported by each type of evidence. The final command in that set (i.e., `run_pipeline.py`) iteratively scaffolds with the

Hi-C evidence after fixing mis-assemblies. The fixed contigs will be found in a file called `assembly.cleaned.fasta` and the final iteration of scaffolds will be located in `scaffolds_FINAL.fasta`. The tiling of contigs (from `assembly.cleaned.fasta` to create `scaffolds_FINAL.fasta`) will be in `scaffolds_FINAL.agp`.

Scaffolding with RNA-seq Data

The RNA-seq data were aligned using HiSat v0.1.6-beta [9], and the alignments were converted from SAM to BAM format and sorted using SAMtools v1.11 [5]. First, the assembly (scaffolds from SALSA) was indexed with HiSat. For each tissue (i.e., brain, eye, fin, gill, heart, kidney, liver, and muscle), HiSat aligned reads to the assembly, SAMtools sorted and compressed the output alignments, and Rascaf v1.0.2 commit #690f618 [10] computed how scaffolding could be done. The actual scaffolding was done with Rascaf in a single step after all steps had been completed for each tissue. The process is described in the following script:

```
hisat-build \
    ${HISAT_IDX_PREFIX} \
    ${HIC_SCAFFOLDS}


for TISSUE in {brain,eye,fin,gill,heart,kidney,liver,muscle}
do
    RNASEQ_READS_LEFT=${TISSUE}_L.fq.gz
    RNASEQ_READS_RIGHT=${TISSUE}_R.fq.gz
    ALIGNMENT_SAM=${TISSUE}_aln.sam

    hisat \
        -p ${THREADS} \
        --phred33 -q -t \
        -x ${HISAT_IDX_PREFIX} \
        -1 ${RNASEQ_READS_LEFT} \
        -2 ${RNASEQ_READS_RIGHT} \
        -S ${ALIGNMENT_SAM}
```

```
    samtools view \
        -buh ${ALIGNMENT_SAM} \
        | samtools sort \
            -@ ${THREADS} \
            -m ${MEMORY}M \
            -O BAM \
            -o ${ALIGNMENT_BAM}

    rascaf \
        -breakN 600 \
        -b ${ALIGNMENT_BAM} \
        -f ${HIC_SCAFFOLDS} \
        -o ${TISSUE}.out
done

rascaf-join \
    -r brain.out \
    -r eye.out \
    -r fin.out \
    -r gill.out \
    -r heart.out \
    -r kidney.out \
    -r liver.out \
    -r muscle.out \
    -o ${OUTPUT_FILE_PREFIX}
```

Note that all newly-created gaps from Rascaf will all be assigned a length of 17 nucleotides (i.e., 17 Ns in a row). Assuming these are gaps of unknown size, these will ideally be changed to 100 nucleotides for any submissions to GenBank. If you have multiple sources of evidence for gaps, you will want to keep track of which gaps were supported by each type of evidence. Also, note that the `-breakN` option of Rascaf was set to 600 because the gaps from SALSA were 500 bases long. The choice of 600 was arbitrary, it just needed to be longer than 500 (i.e., 501 would have been sufficient). The goal here was to prevent Rascaf from undoing the work SALSA had already done.

Unfortunately, Rascaf does not produce an AGP file like SALSA does. For simplicity in submission to GenBank, such a file is necessary because you would submit the contig-level assembly (contigs made with Canu and fixed with SALSA in this case) and provide an AGP file with scaffold joins and relevant evidence. The information needed to create an AGP file from the

Rascaf scaffolds is available in the ancillary output file ending in ".info". A custom Python script was written to take the contigs file, SALSA AGP file, SALSA scaffolds file, Rascaf scaffolds file, and Rascaf .info file to create two sets of two output files (4 total files). Each set is a fasta and AGP pair where the fasta file is the scaffold level sequence and the AGP file is the description of how to obtain that file from the contig-level file (provided as input). The first set of these files leaves the gaps as they are provided (500 Ns from SALSA and 17 Ns from Rascaf), the second converts them all to 100 Ns. This script is too long to be readable in a document, but the code is available in the file `combineHicRna.py` on GitHub at https://github.com/pickettbd/caranx-ignobilis_assembly-paper_misc-scripts. During the NCBI submission process, contaminants were identified in the submitted fasta file. These sequences were removed, and appropriate adjustments to the AGP file were also made before resubmission. To create a new scaffold-level fasta file, another custom script was written. It will take an AGP file and input contigs and output scaffolds in fasta format. It is also available in the same GitHub repository in the file `agp2fa.py`.

Assembly Statistics

Assembly continuity statistics, e.g., N50 and auN [11], were calculated with caln50 commit #3e1b2be (https://github.com/lh3/calN50) and a custom Python (https://www.python.org) script. caln50 is run using the following simple command:

```
caln50 \
    -s 0.01 \
    -L ${GENOME_SIZE} \
    ${CONTIGS_OR_SCAFFOLDS_FILE} \
    > ${STATISTICS_FILE}
```

The custom Python script is not efficient, but it does calculate Nx, Lx, NGx, and LGx, as well as a few other interesting points about sequences in a fasta file. This script is too long to

realistically represent when embedded in the text; it is available on GitHub at https://github.com/pickettbd/basicAsmStatsCalcInPy.

Assembly completeness was assessed using single-copy orthologs with BUSCO v4.0.6 [12] and OrthoDB v10 [13]. The BUSCO config file was the not modified from the default aside from the locations of OrthoDB v10 and the binary executables for BUSCO. It was run based on the following command structure:

```
busco \
    --offline \
    --config ${BUSCO_CONFIG_FILE} \
    --cpu ${THREADS} \
    --in ${CONTIGS_OR_SCAFFOLDS_FASTA} \
    --out_path ${OUTPUT_DIR} \
    --out ${OUTPUT_FILE_PREFIX} \
    --mode genome \
    --lineage actinopterygii \
    --augustus_species zebrafish
```

Genome Comparisons with Single-copy Orthologs

Single-copy orthologs were identified from the Actinopterygii set of OrthoDB v9 (same process as for assessing the assembly, but with OrthoDB v9 instead of v10) and BUSCO v3.0.6. These versions of BUSCO and OrthoDB were used, despite being older, because the plotting technique provided by ChrOrthLink depends on the output file structure from BUSCO v3, and BUSCO v4 has changed the format. The commands between BUSCO versions have changed slightly, but they are the same in essence. The command for each genome was based on the following structure:

```
run_busco.py \
    --cpu ${THREADS} \
    --in ${ASSEMBLY_FASTA} \
    --out_path ${OUTPUT_DIR} \
    --out ${OUTPUT_FILE_PREFIX} \
    --mode genome \
    --lineage_path odb9/actinopterygii \
    --species zebrafish
```

The ChrOrthLink scripts have not yet been prepared for production, so manual editing of the files was necessary to repurpose the code for this analysis. The four scripts (three Python, one R[14]) accept no command-line arguments, so the only way to make it work without adding that functionality is to edit file names and things directly. The simplest way to recreate my analysis or repurpose the ChrOrthLink code for your own analysis in a similar manner would be to clone the repository, edit according to the process described below (substituting your species/filenames/etc. over those described here), and copy your input files into the directory tree. We omitted all sequences that were shorter than 1mb for the plot.

1. Clone the repository. Let's assume the repo is cloned into a directory called `project_dir`. Enter the directory and only subdirectory (`cd project_dir/VGP_fig5a`).
2. Cleanup the stuff you don't need.

```
rm -rf \
       work/output/* \
       work/BUSCO_genoPlotR_input \
       work/*.csv \
       work/input/BUSCO/*.txt \
       work/input/chr.assign/*.csv \
       work/input/chrsize/*.txt
```

3. Make a note to yourself of some handy abbreviations to use for the genome names. For ours, we used the first letter of the genus and the first 3 letters of the species (e.g., Enau, Cign, Tova, etc.). The rest of these comments will refer to the species name and be meaning this shortened code name as `${SPECIES}` (in shell scripts).

4. *Copy* the BUSCO output into `work/input/BUSCO`. Do not *move* the original output files because these copies will get edited by the scripts; if you made a mistake, it would be annoying to undo the changes when you could have simply re-copied over them. There should be one file per genome included in the analysis (for us, that was eight). The output files from BUSCO are located in the respective BUSCO output directories. The filename is `full_table_${SPECIES}.tsv`. When copied into `work/input/BUSCO`, it will need to match the following pattern `BUSCO_${SPECIES}.txt`.
5. Create the `chrsize` files. These are formatted as a tab-separated file with the first column being the sequence identifier (from the fasta file, without the `>`) and the second column being the length of the sequence. The simplest way to obtain this, if you don't already have it, is to create a fasta index using SAMtools faidx: `samtools faidx ${SPECIES}.fa`. This will create the index file at `${SPECIES}.fa.fai`. The first two columns of this file are what you need. They can be extracted with cut:

```
cut -d \t -f 1-2 \
        path/to/${SPECIES}.fa.fai \
        > work/input/chrsize/chrsize_${SPECIES}.txt
```

6. Create the `chr.assign` files. These are formatted as comma-separated files with the first column being the sequence identifier (from the fasta file, without the `>`), the second column being the assigned chromosome number, and the third column being "y" or "n". If you have curated genomes with assigned chromosome numbers, they can be used. Otherwise, you can make something up. For our plot, we simply assigned chromosome numbers 1-n, where n was the number of sequences in the file. We ordered it based on length of the sequence. We also assigned "y" for the third column for each entry. This can be done with a simple sort and awk command:

```
sort -t \t -n -r -k 2 \
        work/input/chrsize/chrsize_${SPECIES}.txt \
        | awk 'BEGIN{FS="\t"; OFS=",";}{print $1, NR, "y";}'
        > work/input/chr.assign/${SPECIES}.csv
```

7. Edit script #1 in the `bin` directory (if needed). I changed the location of the `work` directory, so I had to change the paths, but otherwise this shouldn't need any fixing. This script will edit the files in `work/input/BUSCO` and `work/input/chrsize` based on the files in `work/input/chr.assign`. Run the script. If a mistake is made when run, you'll have to re-do steps 4-6 here.
8. Edit script #2 in the `bin` directory. This script creates `*.csv` files in `work`. Change the value of `Ref_BUSCO` on line 16; we set it to `BUSCO_Enau.txt`. Run the script.
9. Edit script #3 in the `bin` directory. This script creates the input for the plot. Change the value of `RefID_list` on line 19. We set it to `["Enau"]`. Change the value of `sID_LIST` on line 21 to all the species codes. We set it to `["Enau", "Cign", "Cmel", "Tova", "Ttra", "Sdum", "Squi ", "Sriv"]`. Change the value of `target_chr_name` to on line 23 to `"All"`. I suggest changing the system calls for `mkdir` around line 570 to include the `-p` option; this will prevent errors from being unable to create directories that already exist if you re-run these scripts. Run the script.
10. Edit script #4 in the `bin` directory. Change the value of `RefID` on line 32. We set it to `"Enau"`. Change the list starting on line 72 to the same names in the same order for `sID_LIST` as described in step #9. Do the same for the items starting on line 94. Add or remove items for the list starting on line 112 until there are numbers 1-(n-1), with n being the number of species used. In our case, we had 1-7. Run the script. The output should be in `work/output`. The species names were manually edited in Adobe Illustrator for the final figure.

SUPPLEMENTAL REFERENCES

1 Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.* **27**, 722-736, doi:10.1101/gr.215087.116 (2017).

2 Vasimuddin, M., Misra, S., Li, H. & Aluru, S. in *2019 IEEE IPDPS.* 314-324 (Institute of Electrical and Electronics Engineers (IEEE), 2019).

3 Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Preprint at https://arxiv.org/abs/1303.3997 (2013).

4 Broad Institute. Picard Toolkit. *GitHub* http://broadinstitute.github.io/picard (2019).

5 Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078-2079, doi:10.1093/bioinformatics/btp352 (2009).

6 Ghurye, J., Pop, M., Koren, S., Bickhart, D. & Chin, C.-S. Scaffolding of long read assemblies using long range contact information. *BMC Genomics* **18**, 1-11, doi:10.1186/s12864-017-3879-z (2017).

7 Ghurye, J. *et al.* Integrating Hi-C links with assembly graphs for chromosome-scale assembly. *PLoS Comput. Biol.* **15**, e1007273, doi:0.1371/journal.pcbi.1007273 (2019).

8 Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841-842, doi:10.1093/bioinformatics/btq033 (2010).

9 Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* **12**, 357-360, doi:10.1038/nmeth.3317 (2015).

10 Song, L., Shankar, D. S. & Florea, L. Rascaf: Improving Genome Assembly with RNA Sequencing Data. *Plant Genome* **9**, 1-12, doi:10.3835/plantgenome2016.03.0027 (2016).

11 Li, H. auN: a new metric to measure assembly contiguity. *Heng Li's Blog* http://lh3.github.io/2020/04/08/a-new-metric-on-assembly-contiguity (2020).

12 Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V. & Zdobnov, E. M. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* **31**, 3210-3212, doi:10.1093/bioinformatics/btv351 (2015).

13 Kriventseva, E. V. *et al.* OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic Acids Res.* **47**, D807-D811, doi:10.1093/nar/gky1053 (2019).

14 R Core Team. R: A language and environment for statistical computing. *R Foundation for Statistical Computing* https://www.r-project.org (2021).

APPENDIX 6

Chapter 5 – Supplement

SUPPLEMENTARY TEXTS

Supplementary Text 1. Suffix and Longest Common Prefix Arrays

A suffix array is an array of character positions representing a list of all possible suffixes of a string, ordered lexicographically. Consider the sequence "CAGAGA$". A proper suffix array implementation would not enumerate a list of suffixes, but viewing the list helps conceptualize suffix array construction (see Supplementary Fig. 1A and B). The suffix and longest common prefix arrays (with zero-based indexing) for this sequence are shown in Supplementary Fig. 1C. The 6 in position 0 of the suffix array (Supplementary Fig. 1B and C) informs us that the suffix beginning at position 6 (i.e., "$") is lexicographically first. The 5 in position 1 of the suffix array informs us that the suffix beginning at position 5 (i.e., "A$") is lexicographically second. Likewise, the 2 in position 6 of the suffix array informs us that the suffix beginning at position 2 (i.e., "GAGA$") is lexicographically last.

Longest common prefix arrays are arrays of the lengths of the longest common prefix of each adjacent suffix in the suffix array. To illustrate, consider position 3 in the suffix and longest common prefix arrays in Supplementary Fig. 1C. The longest common prefix at this position is 3 (highlighted in red text in Supplementary Fig. 1C), meaning there are three common nucleotides at the beginning of the suffixes starting at positions 1 and 3 (i.e., "AGA"). The longest common prefix array stores the length of the longest common prefix, and the positions of the two suffixes in the original sequence are obtained by looking at the same position in the suffix array (in this

example position 3), and the prior position in the suffix array (in this example position 2). This longest common prefix is represented in red nucleotides in Supplementary Fig. 1B. Although the sequence is the same, they are adjacent in the original sequence. These relationships are the basis for our algorithm to find SSRs in a sequence. The longest common prefix array is constructed while creating the suffix array.

Supplementary Text 2. Calculating SSR Length and Position from Suffix and Longest Common Prefix Arrays

Let *k* equal the length of an SSR repeating unit or period size, *r* equal the number of times it repeats after the original occurrence, and *p* equal the position of the first nucleotide of the first period of the SSR. For example, consider the repeating unit "ACG" in the sequence "ACGACGACG". The length of the repeating unit is 3 (*k*), there are three instances of the unit (*r* + 1), and the SSR begins at position 0 in the sequence (*p*). So, in this example, $k = 3$, $r = 2$ ($r + 1$ is the total number of repeats in the SSR), and $p = 0$. SSRs are identified by calculating *k*, *p*, and *r* from the suffix and longest common prefix arrays. Let *i* equal the index of any entry in the suffix array (except the first position), where SA and LCPA are the suffix and longest common prefix arrays, respectively:

$$k = |SA_i - SA_{i-1}| \quad (1)$$

$$r = \left\lfloor \frac{LCPA_i}{k_i} \right\rfloor \quad (2)$$

$$p = MIN(SA_{i-1}, SA_i) \quad (3)$$

If r > 0, an SSR of length k * (r + 1) exists at position p in the original sequence, otherwise if r = 0 there is no SSR at position p. The base unit (e.g., AG in the SSR AGAGAG) of

the SSR starts at position p and ends at position p + (k − 1). Thus, by comparing each adjacent element in the suffix array we can find SSRs in a sequence.

Extending the previous example, Fig. 1C shows the values of k, r, and p calculated from the suffix and longest common prefix arrays for “CAGAGA$”. Two SSRs, each of length 4, exist at positions 1 and 2 in the original sequence (i.e., “AGAG” and “GAGA”) and their locations are shown in Fig. 1D.

SUPPLEMENTARY FIGURES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

A

	Before Sorting						
0	C	A	G	A	G	A	$
1	A	G	A	G	A	$	
2	G	A	G	A	$		
3	A	G	A	$			
4	G	A	$				
5	A	$					
6	$						

B

	After Sorting						
6	$						
5	A	$					
3	A	G	A	$			
1	A	G	A	G	A	$	
0	C	A	G	A	G	A	$
4	G	A	$				
2	G	A	G	A	$		

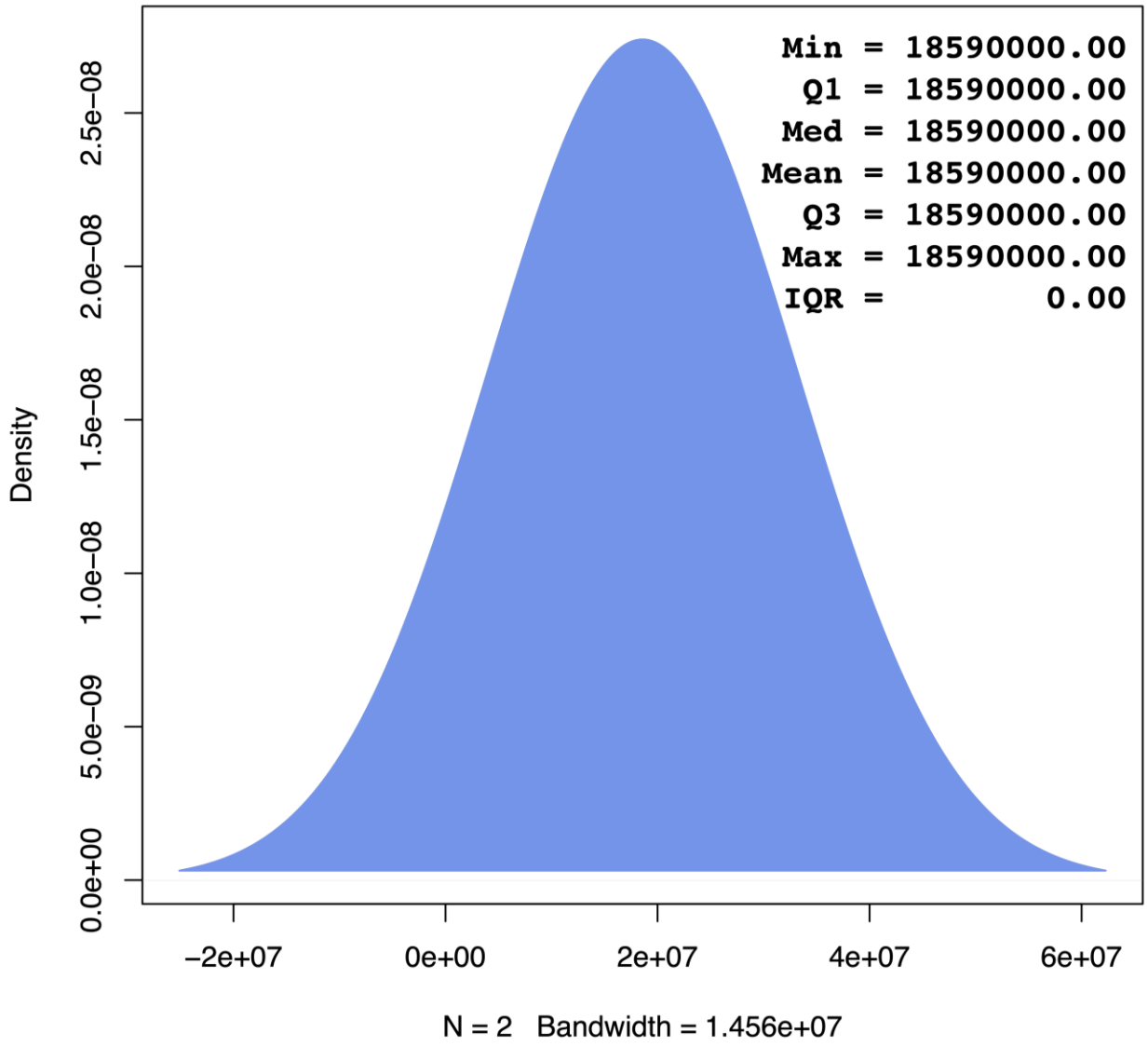
C

Array Indices:	0	1	2	3	4	5	6
Suffix Array:	6	5	3	1	0	4	2
Longest Common Prefix Array:	-	0	1	3	0	0	2
k:	-	1	2	2	1	4	2
r:	-	0	0	1	0	0	1
p:	-	5	3	1	0	0	2

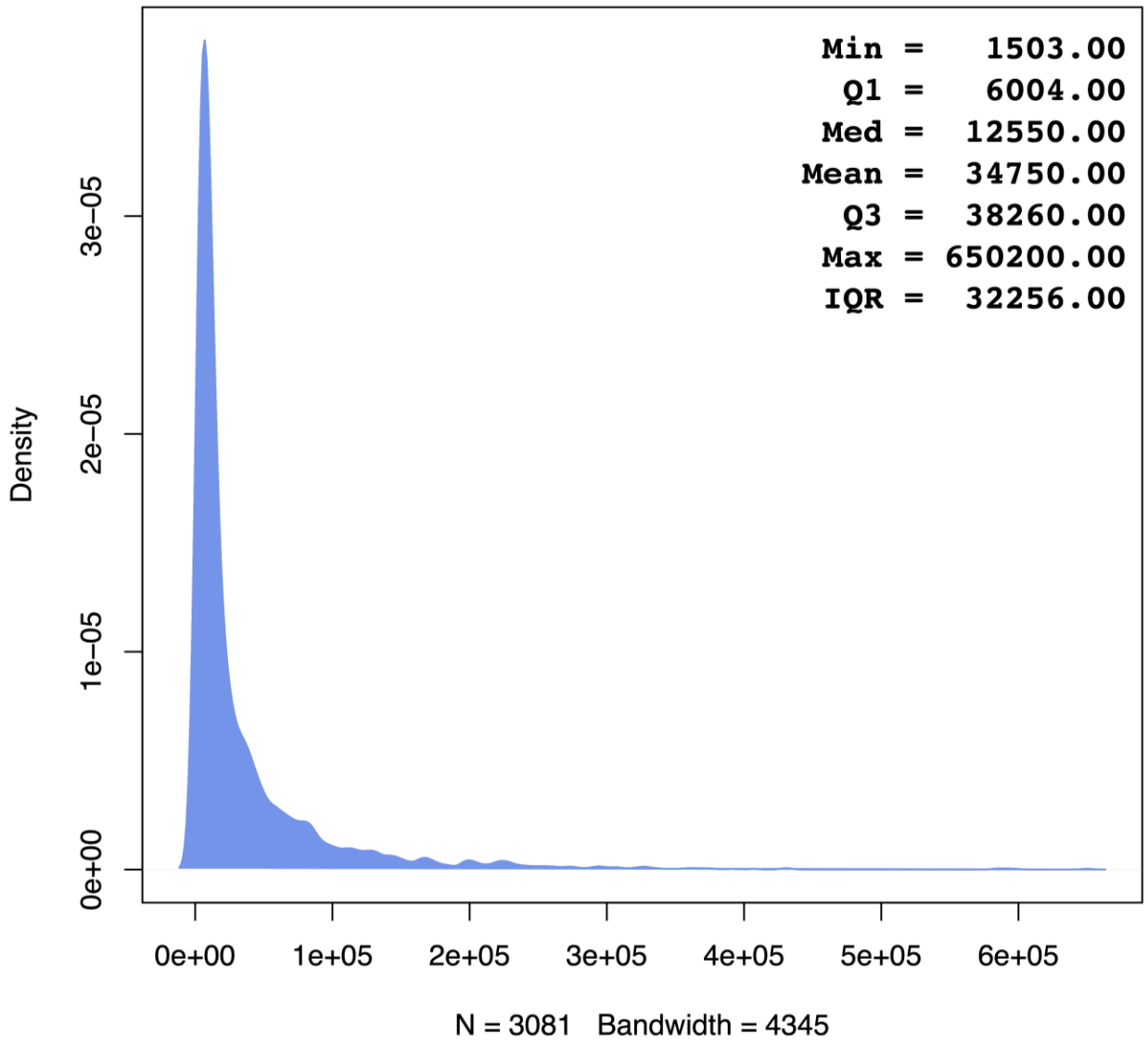
D

Position:	0	1	2	3	4	5	6
SSR 1:	C	A	G	A	G	A	$
SSR 2:	C	A	G	A	G	A	$

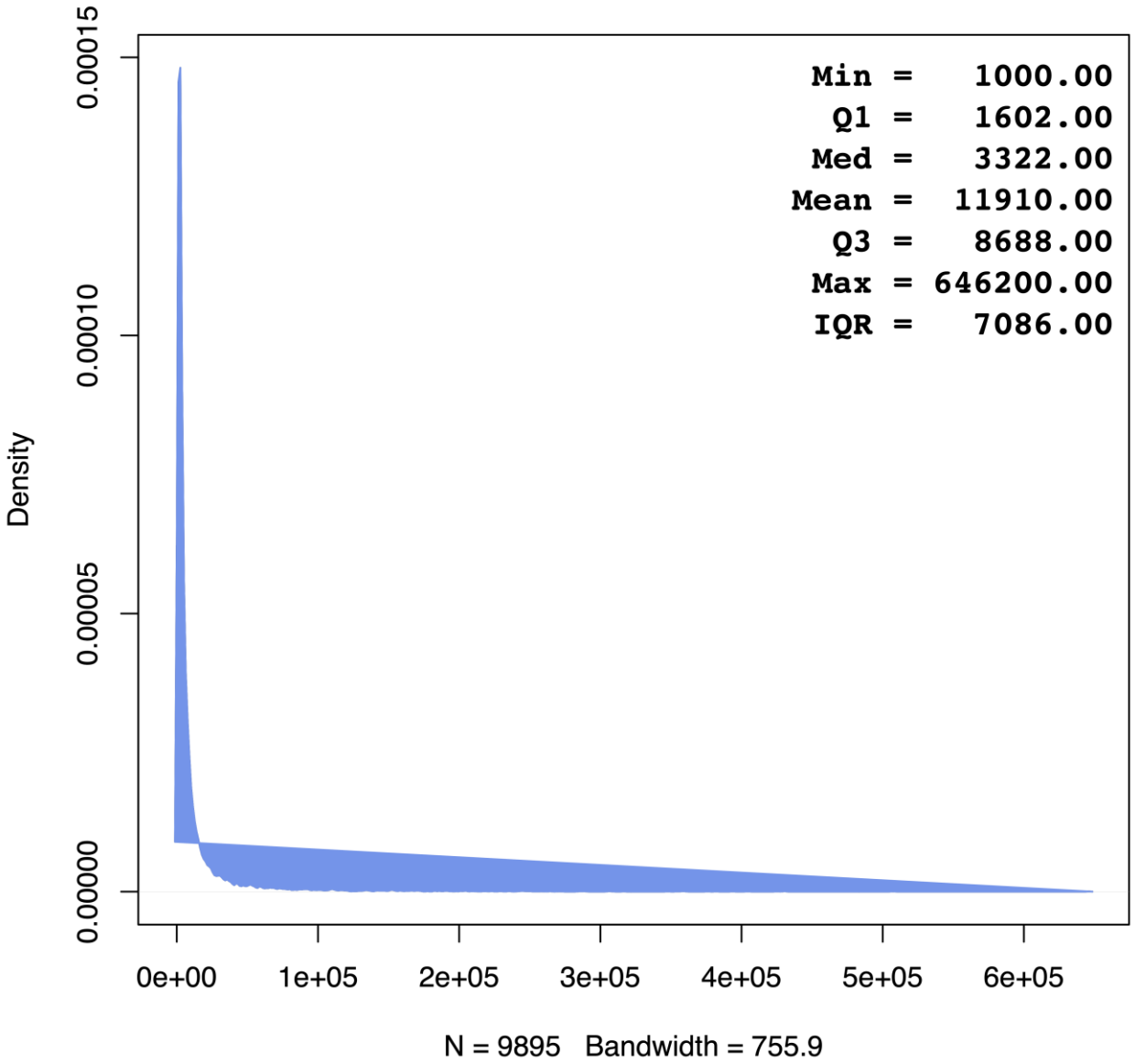
Supplementary Figure 1. Suffix and Longest Common Prefix Arrays Example. In this figure we demonstrate how to construct a suffix array and its use to identify SSRs. (A) First, all suffixes of "CAGAGA$", are shown here and marked by their beginning position in the original sequence. (B) Next, the set of possible suffixes (part A) are ordered lexicographically, where '$' is the first character in the alphabet, and maintain their start positions in the original sequence. The start positions are the numbers to the left of each suffix. The new ordering of these start positions is the suffix array. (C) Here we show the suffix array, longest common prefix array, and three parameters: *k*, *r*, and *p* (explained in the text). The suffix array stores the ordered start positions determined by ordering possible suffixes (shown in part B). (D) This particular sequence has two SSRs: "AGAG" and "GAGA". In part D we show each of the two SSRs in the original sequence. SSR1 is highlighted blue, and SSR2 is highlighted green. The repeating units of the two SSRs are AG and GA, respectively, and a vertical bar separates each repeating unit in the sequence.



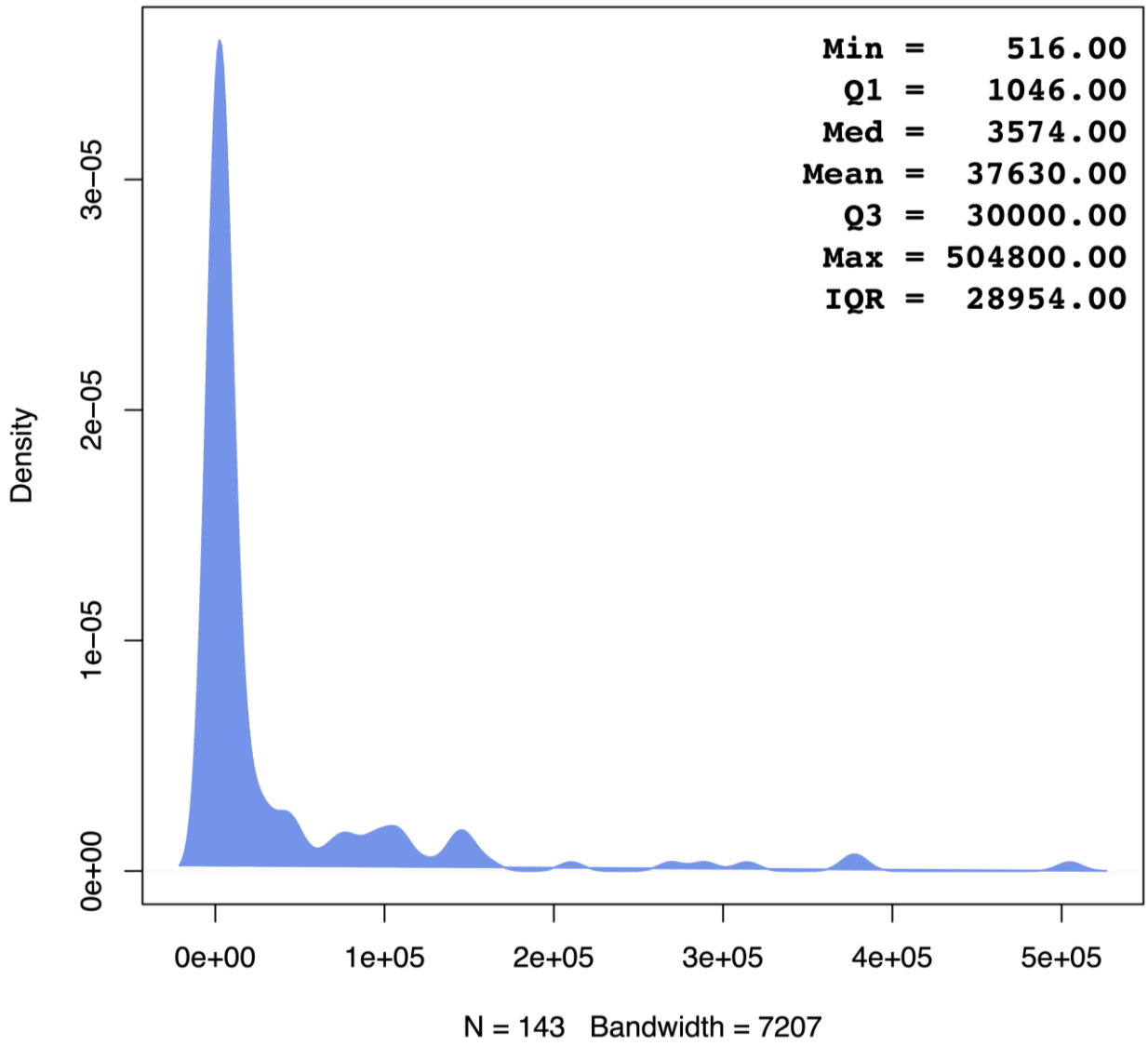
Supplementary Figure 2. *Arabidopsis thaliana* Sequence Length Density Plot. Density plot showing the distribution of sequence lengths for the *Arabidopsis thaliana* chromosome 4. A summary is included in the upper, right-hand corner.



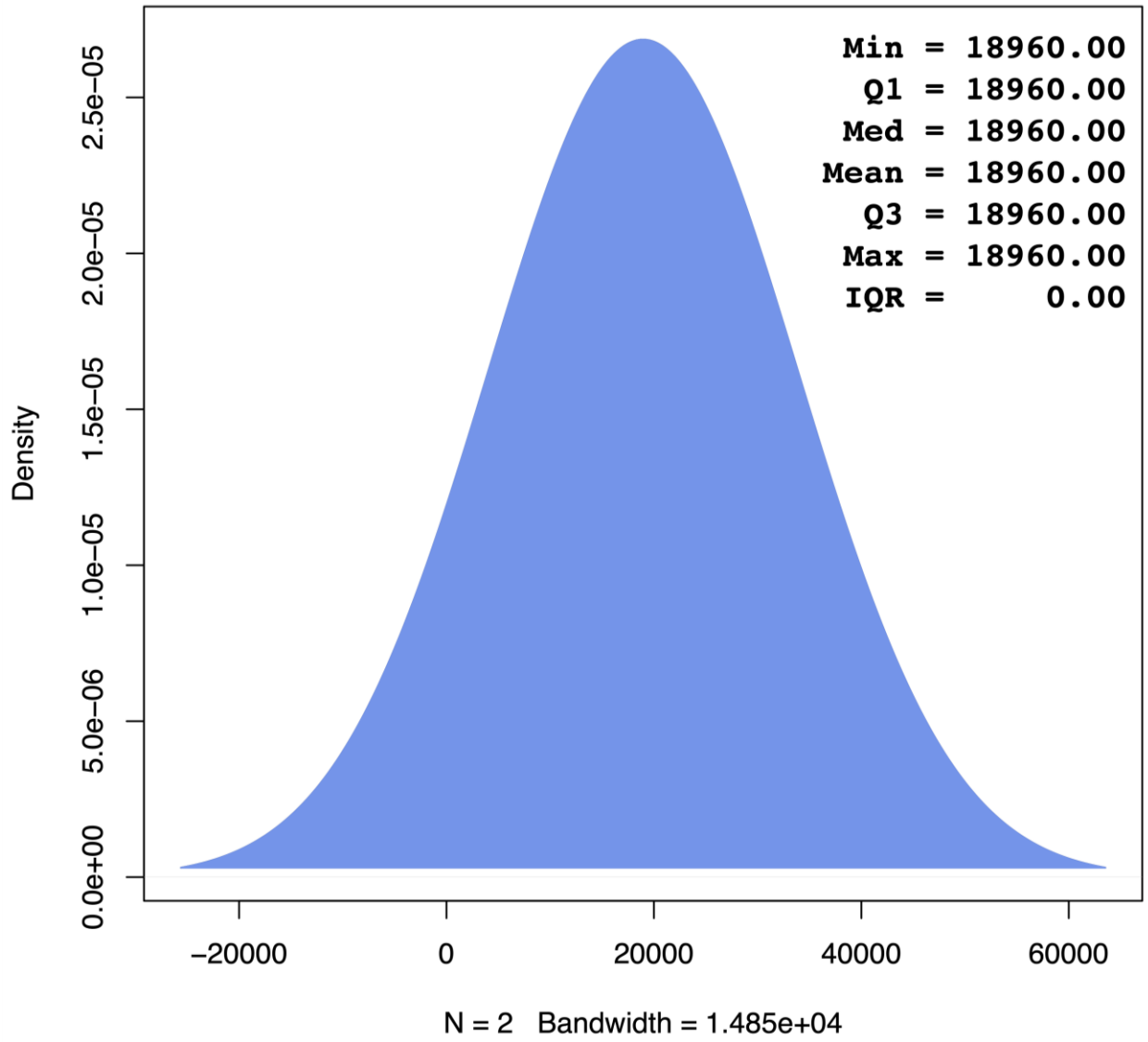
Supplementary Figure 3. *Caenorhabditis elegans* Sequence Lengths Density Plot. Density plot showing the distribution of sequence lengths for the *Caenorhabditis elegans* genome. A summary is included in the upper, right-hand corner.



Supplementary Figure 4. *Drosophila melanogaster* Sequence Lengths Density Plot. Density plot showing the distribution of sequence lengths for the *Drosophila melanogaster* genome. A summary is included in the upper, right-hand corner.



Supplementary Figure 5. *Escherichia coli* Sequence Lengths Density Plot. Density plot showing the distribution of sequence lengths for the *Escherichia coli* genome. A summary is included in the upper, right-hand corner.



Supplementary Figure 6. *Zaire ebolavirus* Sequence Lengths Density Plot. Density plot showing the distribution of sequence lengths for the *Zaire ebolavirus* genome. A summary is included in the upper, right-hand corner.

SUPPLEMENTARY TABLES

Supplementary Table 1. Algorithms Included in Comparisons. We compared our algorithm to existing algorithms that (a) were capable of processing the *Drosophila melanogaster genome* dataset (see the main text), (b) had a non-interactive, Linux, command-line interface, (c) were freely available for immediate download, and (d) had 10 or more citations per year (based on publication date and Google Scholar citation count) or were published in the last three years. A few other algorithms met our requirements, but were rendered unusable due to antiquated shared libraries, compile- or run-time errors, or other issues.

Algorithm
GMATo (Wang, et al., 2013)
MREPS (Kolpakov, et al., 2003)
PRoGeRF (Lopes, et al., 2015)
QDD (Meglécz, et al., 2014)
SSR-Pipeline (Miller, et al., 2013)
SSRIT (Temnykh, et al., 2001)
TRF (Benson, 1999)

Supplementary Table 2. Performance Comparisons.

[a] MREPS timing includes the pre- and post-processing time for each genome necessary to adjust positions to account for removing "incorrect symbols" and Ns. The additional times are an average of multiple approaches.

[b] We only considered SSRs with period sizes 1-7 (inclusive) and lengths of at least 16 nucleotides (nt). The difference between the number of SSRs in range and reported is due exclusively to SSR length (less than 16 nt) and period size (greater than 7).

[c] Whenever possible, we salvaged correct SSRs that were inside incorrect SSRs reported by other software packages. For example, in *Drosophila melanogaster*, we recovered three for PRoGeRF and 8,408 for TRF. To illustrate, in sequence JXOZ01000043.1, TRF reports a CT repeated 36 times at position 2,171. While TRF does correctly identify a low-complexity region with many CT repeats, there are not 36 perfect repeats in a row. In this case, we salvaged two perfect CT regions, each repeating 8 times.

[d] Detailed pairwise comparisons can be found in Supplementary Tables 4-31.

								Comparison with SA-SSR		
		CPU Time (mm:ss)	Real Time (mm:ss)	SSRs Reported	SSRs In Range	Number Correct	Percent Correct	SSRs Unique to Software	SSRs Unique to SA-SSR	SSRs Shared
Arabidopsis thaliana (chr 4)	GMATo	312:29	312:29	4,004,812	1,854	1,854	100	5	713	1,550
	MREPS	386:15	386:15	4,201	2,270	2,270	100	11	0	2,259
	PRoGeRF	9:23	9:23	4,116,484	2,247	2,247	100	11	26	2,233
	QDD	2:02	2:02	3,965	1,100	1,100	100	2	1,165	1,098
	SA-SSR	28,066:12	2,338:47	2,265	2,265	2,265	100	NA	NA	NA
	SSR-Pipeline	1,395:04	1,395:04	4,754,929	2,242	2,242	100	11	66	2,193
	SSRIT	0:10	0:10	900	900	900	100	6	1,365	894
	TRF	0:47	0:47	135,135	9,275	2,167	23.36	10	152	2,107
Caenorhabditis elegans	GMATo	9:39	9:39	22,889,822	6,068	6,068	100	27	2,685	5,236
	MREPS	4:34	4:34	18,958	7,962	7,962	100	53	0	7,909
	PRoGeRF	744:21	744:21	531,822	105	105	100	0	7,818	105
	QDD	10:32	10:32	11,720	3,379	3,379	100	8	4,552	3,369
	SA-SSR	645:54	60:31	7,923	7,923	7,923	100	NA	NA	NA
	SSR-Pipeline	13:14	13:14	26,475,821	7,827	7,827	100	32	204	7,715
	SSRIT	0:57	0:57	2,374	2,374	2,374	100	12	5,555	2,362
	TRF	7:20	7:20	1,029,051	39,378	6,663	16.92	23	1,578	6,336

								Comparison with SA-SSR		
		CPU Time (mm:ss)	Real Time (mm:ss)	SSRs Reported	SSRs In Range	Number Correct	Percent Correct	SSRs Unique to Software	SSRs Unique to SA-SSR	SSRs Shared
Drosophila melanogaster	GMATo	6:31	6:31	30,386,038	23,218	23,171	99.80	78	7,970	19,900
	MREPS	1:47	1:47	52,346	28,008	28,008	100	163	0	27,845
	PRoGeRF	2,436:55	2,436:55	470,382	571	562	98.42	2	27,318	560
	QDD	11:11	11:11	37,525	12,931	12,931	100	39	14,978	12,883
	SA-SSR	52:58	4:52	27,880	27,880	27,880	100	NA	NA	NA
	SSR-Pipeline	1:47	1:47	29,015,430	27,513	27,513	100	96	726	27,138
	SSRIT	1:02	1:02	9,943	9,943	9,943	100	37	17,956	9,906
	TRF	4:01	4:01	856,363	105,179	25,940	24.66	85	2,770	25,084
Escherichia coli	GMATo	0:39	0:39	1,127,792	14	14	100	0	9	11
	MREPS	0:26	0:26	46	20	20	100	0	0	20
	PRoGeRF	3:36	3:36	334,091	4	4	100	0	16	4
	QDD	0:32	0:32	38	8	8	100	0	12	8
	SA-SSR	55:07	12:21	20	20	20	100	NA	NA	NA
	SSR-Pipeline	1:15	1:15	1,309,541	20	20	100	0	0	20
	SSRIT	0:03	0:03	0	0	0	NA	0	20	0
	TRF	0:06	0:06	15,107	224	20	8.93	0	0	20
Zaire ebolavirus	GMATo	0:00	0:00	4,180	0	0	NA	0	0	0
	MREPS	0:00	0:00	0	0	0	NA	0	0	0
	PRoGeRF	0:03	0:03	4,350	0	0	NA	0	0	0
	QDD	0:00	0:00	0	0	0	NA	0	0	0
	SA-SSR	0:01	0:01	0	0	0	NA	NA	NA	NA
	SSR-Pipeline	0:01	0:01	4,862	0	0	NA	0	0	0
	SSRIT	0:00	0:00	0	0	0	NA	0	0	0
	TRF	0:00	0:00	59	0	0	NA	0	0	0
Combined	GMATo	329:18	329:18	58,412,644	31,154	31,107	99.85	110	11,377	26,697
	MREPS	393:02	393:02	75,551	38,260	38,260	100	227	0	38,033
	PRoGeRF	3,194:18	3,194:18	5,457,129	2,927	2,918	99.69	13	35,178	2,902
	QDD	24:17	24:17	53,248	17,418	17,418	100	49	20,707	17,358
	SA-SSR	28,820:12	2,416:32	38,088	38,088	38,088	100	NA	NA	NA
	SSR-Pipeline	1,411:21	1,411:21	61,560,583	37,602	37,602	100	139	996	37,066
	SSRIT	2:12	2:12	13,217	13,217	13,217	100	55	24,896	13,162
	TRF	12:14	12:14	2,035,715	154,056	34,790	22.58	118	4,500	33,547

Supplementary Table 3. Features of Software for Finding SSRs.

	Op. Sys.					Format					Complexity						
	MS Win	Mac OS X	Linux	CLI	GUI	Input	Output	Language	Algorithm	Type	Time	Space	Period	Repeats	Multi-threaded	Ignore Characters	Search for Specific SSRs
SA-SSR			X	X		FASTA	TSV	C++	Combinatorial	Exact	O(n)	O(n)	1+	2+	X	Yes (Configurable)	X
GMATo	X	X	X	X	X	FASTA	TSV	Perl & Java	Regular Expressions	Exact	?	?	1-10	2+		Yes (default)	
MREPS			X	X		FASTA	Text	C	Combinatorial	Inexact	$O(nk \cdot \log(n/k) + S)$	?	1+	2+		Yes (only some Ns)	
PRoGeRF			X	X	Web	FASTA	TSV	Perl	?	Inexact	?	?	1-12	2+		Yes (default)	
QDD	X		X	X		FASTA	SCSV	Perl	?	Exact	?	?	?	2+		Yes (default)	
SSR-Pipeline	X	X	X	X		FASTA	FASTA	Python	?	Exact	?	?	2-25	2+		Yes (default)	
SSRIT			X	X		FASTA	TSV	Perl	Regular Expressions	Exact	?	?	?	2+		Yes (default)	
TRF	X	X	X	X	X	FASTA	Text	?	Heuristic	Inexact	$O(n^2 \cdot \text{polylog}(n))$	?	1+	2+		Yes (default)	

Supplementary Table 4. SA-SSR compared with GMATo for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to GMATo, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
GMATo	0	0	0	0	0	0	0	**0**
SA-SSR	660	721	343	126	60	245	110	**2265**
Shared	0	0	0	0	0	0	0	**0**

Supplementary Table 5. SA-SSR compared with MREPS for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to MREPS, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 11 SSRs that MREPS found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 9 of the 11 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 2 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	MREPS	0	5	1	2	1	2	0	**11**
	SA-SSR	660	5	1	1	0	1	0	**668**
	Shared	0	716	342	125	60	244	110	**1597**
Overlap	MREPS	0	0	0	0	1	1	0	**2**
	SA-SSR	2742	6064	2171	322	134	553	535	**12521**
	Shared	0	721	343	127	60	245	110	**1606**
Exhaustive	MREPS	0	0	0	0	0	0	0	**0**
	SA-SSR	2752	8824	3761	9867	1029	10115	1194	**37542**
	Shared	0	721	343	127	61	246	110	**1608**

Supplementary Table 6. SA-SSR compared with ProGeRF for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to ProGeRF, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 32 SSRs that ProGeRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 16 of the 32 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. 14 of the remaining 16 SSRs were also found by SA-SSR, but SA-SSR correctly reported shorter period lengths than ProGeRF did. Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times. The last 2 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	ProGeRF	0	5	6	3	2	16	0	**32**
	SA-SSR	660	7	21	5	1	4	0	**698**
	Shared	0	714	322	121	59	241	110	**1567**
Overlap	ProGeRF	0	0	0	0	1	15	0	**16**
	SA-SSR	2742	6066	2186	325	134	556	535	**12544**
	Shared	0	719	328	124	60	242	110	**1583**
Exhaustive	ProGeRF	0	0	0	0	0	0	0	**0**
	SA-SSR	2752	8826	3776	9870	1029	10104	1194	**37551**
	Shared	0	719	328	124	61	257	110	**1599**

Supplementary Table 7. SA-SSR compared with QDD for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to QDD, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 2 SSRs that QDD found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. Both were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result.

		1	2	3	4	5	6	7	Total
Normal	QDD	0	2	0	0	0	0	0	**2**
	SA-SSR	660	2	1	99	55	240	110	**1167**
	Shared	0	719	342	27	5	5	0	**1098**
Overlap	QDD	0	0	0	0	0	0	0	**0**
	SA-SSR	2742	6064	2172	422	189	793	645	**13027**
	Shared	0	721	342	27	5	5	0	**1100**

Supplementary Table 8. SA-SSR compared with SSR-Pipeline for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to SSR-Pipeline, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 84 SSRs that SSR-Pipeline found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 81 of the 84 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. One of the remaining 3 SSRs was just a different SSR base, but covering essentially the same SSR (AATAAA vs AAAATA). The remaining 2 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	SSR-Pipeline	0	47	16	7	1	7	6	**84**
	SA-SSR	660	59	26	9	0	8	7	**769**
	Shared	0	662	317	117	60	237	103	**1496**
Overlap	SSR-Pipeline	0	0	0	0	1	2	0	**3**
	SA-SSR	2742	6076	2181	325	134	556	536	**12550**
	Shared	0	709	333	124	60	242	109	**1577**
Exhaustive	SSR-Pipeline	0	0	0	0	0	0	0	**0**
	SA-SSR	2752	8836	3771	9870	1029	10117	1195	**37570**
	Shared	0	709	333	124	61	244	109	**1580**

Supplementary Table 9. SA-SSR compared with SSRIT for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to SSRIT, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 7 SSRs that SSRIT found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. All 7 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	SSRIT	0	5	1	1	0	0	0	**7**
	SA-SSR	660	198	1	98	60	245	110	**1372**
	Shared	0	523	342	28	0	0	0	**893**
Overlap	SSRIT	0	0	0	0	0	0	0	**0**
	SA-SSR	2742	6257	2171	420	194	798	645	**13227**
	Shared	0	528	343	29	0	0	0	**900**

Supplementary Table 10. SA-SSR compared with TRF for *Arabidopsis thaliana*. The number of SSRs in the *Arabidopsis thaliana* chromosome 4 found unique to TRF, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 18600000 -m 1 -M 7 -n 16 -r 1 –i D,M,N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 124 SSRs that TRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 111 of the 124 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 13 SSRs were also found by SA-SSR and they fall into three different categories. The categories are overstated period size, finding different numbers of repeats, and special cases requiring the exhaustive approach by SA-SSR. 6 of the 13 are cases where TRF overstated the period size (e.g., calling ATATATAT a 4-mer instead of a 2-mer). Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times. Of the remaining 7, the 6 that were not found even under the exhaustive approach were actually found by SA-SSR, but SA-SSR correctly reported a larger number of repeats. So, while it appeared that SA-SSR didn't find them, it actually did. For these 6, both are correct, but SA-SSR is more complete. Finally, the last of the 7 was found during the exhaustive approach and is a special, rare case involving the specific sequence and suffix sort. Of course, the number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	TRF	0	48	26	9	2	14	25	**124**
	SA-SSR	660	67	41	13	3	42	36	**862**
	Shared	0	654	302	113	57	203	74	**1403**
Overlap	TRF	0	1	3	5	0	3	1	**13**
	SA-SSR	2742	6084	2189	332	135	584	547	**12613**
	Shared	0	701	325	117	59	214	98	**1514**
Exhaustive	TRF	0	1	3	0	0	1	1	**6**
	SA-SSR	2752	8844	3779	9872	1031	10145	1206	**37629**
	Shared	0	701	325	122	59	216	98	**1521**

Supplementary Table 11. SA-SSR compared with GMATo for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to GMATo, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 2291 SSRs that GMATo found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 2270 of the 2291 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. One of the remaining 21 SSRs were also found by SA-SSR, but SA-SSR correctly reported a greater number of repeats than GMATo did. Finally, the last 20 were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	GMATo	0	687	220	248	55	807	274	**2291**
	SA-SSR	522	866	428	601	130	1551	565	**4663**
	Shared	0	1032	415	393	50	1097	273	**3260**
Overlap	GMATo	0	3	0	5	0	12	1	**21**
	SA-SSR	1862	13378	2802	4084	661	16224	5361	**44372**
	Shared	0	1716	635	636	105	1892	546	**5530**
Exhaustive	GMATo	0	0	0	0	0	0	0	**0**
	SA-SSR	1862	15261	3803	21089	1258	32453	5858	**81584**
	Shared	0	1719	635	641	105	1904	547	**5551**

Supplementary Table 12. SA-SSR compared with MREPS for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to MREPS, unique to SA-SSR, and shared between the two using the three different parameter sets. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 84 SSRs that MREPS found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 54 of the 84 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. Four of the remaining 30 SSRs were also found by SA-SSR, but SA-SSR reported a different repeating unit than MREPS did (e.g., GT vs TG). Finally, the last 26 were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	MREPS	0	11	3	16	0	39	15	**84**
	SA-SSR	522	6	0	8	0	22	9	**567**
	Shared	0	1892	843	986	180	2626	829	**7356**
Overlap	MREPS	0	5	1	8	0	14	2	**30**
	SA-SSR	1862	13196	2592	3726	586	15465	5065	**42492**
	Shared	0	1898	845	994	180	2651	842	**7410**
Exhaustive	MREPS	0	0	0	0	0	0	0	**0**
	SA-SSR	1862	15077	3592	20728	1183	31692	5561	**79695**
	Shared	0	1903	846	1002	180	2665	844	**7440**

Supplementary Table 13. SA-SSR compared with ProGeRF for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to ProGeRF, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -l -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 2 SSRs that ProGeRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 1 of the 2 was also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining SSR was also found by SA-SSR, but SA-SSR correctly reported shorter period lengths than ProGeRF did. Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times.

		1	2	3	4	5	6	7	Total
Normal	ProGeRF	0	0	0	0	0	1	1	**2**
	SA-SSR	522	1871	833	971	179	2620	830	**7826**
	Shared	0	27	10	23	1	28	8	**97**
Overlap	ProGeRF	0	0	0	0	0	1	0	**1**
	SA-SSR	1862	15067	3427	4697	765	18088	5898	**49804**
	Shared	0	27	10	23	1	28	9	**98**

Supplementary Table 14. SA-SSR compared with QDD for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to QDD, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 16 SSRs that QDD found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 9 of the 16 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. One of the remaining 7 was a case where the two programs correctly reported different repeating units (e.g., GT vs TG). The remaining 6 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	QDD	0	8	1	4	0	3	0	**16**
	SA-SSR	522	4	0	715	141	2340	838	**4560**
	Shared	0	1894	843	279	39	308	0	**3363**
Overlap	QDD	0	5	1	0	0	1	0	**7**
	SA-SSR	1862	13197	2594	4437	727	17806	5907	**46530**
	Shared	0	1897	843	283	39	310	0	**3372**
Exhaustive	QDD	0	0	0	0	0	0	0	**0**
	SA-SSR	1862	15078	3594	21447	1324	34046	6405	**83756**
	Shared	0	1902	844	283	39	311	0	**3379**

Supplementary Table 15. SA-SSR compared with SSR-Pipeline for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to SSR-Pipeline, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 286 SSRs that SSR-Pipeline found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 259 of the 286 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. Three of the remaining 27 were cases where the two programs correctly reported different repeating units (e.g., GT vs TG). The remaining 24 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	SSR-Pipeline	0	116	31	38	1	87	13	**286**
	SA-SSR	522	141	53	56	3	115	14	**904**
	Shared	0	1757	790	938	177	2533	824	**7019**
Overlap	SSR-Pipeline	0	5	1	5	0	14	2	**27**
	SA-SSR	1862	13226	2617	3749	588	15510	5072	**42624**
	Shared	0	1868	820	971	178	2606	835	**7278**
Exhaustive	SSR-Pipeline	0	0	0	0	0	0	0	**0**
	SA-SSR	1862	15107	3617	20754	1185	31737	5568	**79830**
	Shared	0	1873	821	976	178	2620	837	**7305**

Supplementary Table 16. SA-SSR compared with SSRIT for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to SSRIT, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 17 SSRs that SSRIT found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 14 of the 17 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 3 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	SSRIT	0	8	3	6	0	0	0	**17**
	SA-SSR	522	662	0	716	180	2648	838	**5566**
	Shared	0	1236	843	278	0	0	0	**2357**
Overlap	SSRIT	0	2	1	0	0	0	0	**3**
	SA-SSR	1862	13852	2592	4436	766	18116	5907	**47531**
	Shared	0	1242	845	284	0	0	0	**2371**
Exhaustive	SSRIT	0	0	0	0	0	0	0	**0**
	SA-SSR	1862	15736	3592	21446	1363	34357	6405	**84761**
	Shared	0	1244	846	284	0	0	0	**2374**

Supplementary Table 17. SA-SSR compared with TRF for *Caenorhabditis elegans*. The number of SSRs in the *Caenorhabditis elegans* genome found unique to TRF, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 700000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 900 SSRs that TRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 851 of the 900 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 49 SSRs were also found by SA-SSR, and they fall into three different categories. The categories are overstated period size, finding different numbers of repeats, and special cases requiring the exhaustive approach by SA-SSR. 10 of the 49 are cases where TRF overstated the period size (e.g., calling ATATATAT a 4-mer instead of a 2-mer). Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times. Of the remaining 38, the 26 that were not found even under the exhaustive approach were actually found by SA-SSR. For 25 of the 26, SA-SSR correctly reported a larger number of repeats. So, while it appeared that SA-SSR didn't find them, it actually did. For these 25, both are correct, but SA-SSR is more complete. The last of the 26 was also found by SA-SSR, but SA-SSR correctly stated a shorter period size (another example where ATATATAT should be a 2-mer, not a 4-mer). This leaves us with 13 unaccounted for. 7 were more cases where TRF and SA-SSR either reported different SSRs (e.g., GT vs TG) or reported different number of repeats. Finally, the last 6 were found during the exhaustive approach and is a special, rare case involving the specific sequence and suffix sort. Of course, the number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	TRF	0	99	46	77	11	537	130	**900**
	SA-SSR	522	144	66	165	26	1443	283	**2649**
	Shared	0	1754	777	829	154	1205	555	**5274**
Overlap	TRF	0	9	8	10	3	17	2	**49**
	SA-SSR	1862	13250	2622	3824	604	16391	5224	**43777**
	Shared	0	1844	815	896	162	1725	683	**6125**
Exhaustive	TRF	0	8	7	2	3	5	1	**26**
	SA-SSR	1862	15135	3622	20826	1201	32620	5721	**80987**
	Shared	0	1845	816	904	162	1737	684	**6148**

Supplementary Table 18. SA-SSR compared with GMATo for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to GMATo, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 467 SSRs that GMATo found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 450 of the 467 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 17 SSRs were also found by SA-SSR, but SA-SSR correctly reported longer SSRs than GMATo did (e.g., in sequence JXOZ01000280.1, SA-SSR reported CAGGGAC repeated 7 times beginning at position 73168 while GMATo reported the same repeating only 4 times).

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	GMATo	0	0	0	15	20	151	281	**467**
	SA-SSR	4734	8094	3286	3328	1088	5557	1207	**27294**
	Shared	0	0	0	15	25	228	318	**586**
Overlap	GMATo	0	0	0	1	1	6	9	**17**
	SA-SSR	31700	47110	16452	14537	4328	25154	6006	**145287**
	Shared	0	0	0	29	44	373	590	**1036**

Supplementary Table 19. SA-SSR compared with MREPS for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to MREPS, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 232 SSRs that MREPS found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 188 of the 232 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. 43 of the remaining 44 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The last SSR was a case where SA-SSR and MREPS simply reported a slightly different SSR (e.g., AT vs TA). The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	MREPS	6	21	33	56	17	90	9	**232**
	SA-SSR	1	11	19	16	10	42	5	**104**
	Shared	4733	8083	3267	3327	1103	5743	1520	**27776**
Overlap	MREPS	2	2	0	36	3	1	0	**44**
	SA-SSR	26963	39008	13152	11219	3255	19695	5067	**118359**
	Shared	4737	8102	3300	3347	1117	5832	1529	**27964**
Exhaustive	MREPS	0	0	0	0	0	0	0	**0**
	SA-SSR	26963	70718	36560	90090	21713	91821	21709	**359574**
	Shared	4739	8104	3300	3383	1120	5833	1529	**28008**

Supplementary Table 20. SA-SSR compared with ProGeRF for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to ProGeRF, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 10 SSRs that ProGeRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 6 of the 10 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 4 SSRs were also found by SA-SSR, but SA-SSR correctly reported shorter period lengths than ProGeRF did (e.g., in sequence JXOZ01000073.1, SA-SSR reported A repeated 19 times beginning at position 136707 while ProGeRF reported AAA repeating 6 times at the same position). Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	ProGeRF	1	1	4	0	1	3	0	**10**
	SA-SSR	4651	7930	3233	3271	1095	5659	1485	**27324**
	Shared	83	164	53	72	18	126	40	**556**
Overlap	ProGeRF	0	1	2	0	0	1	0	**4**
	SA-SSR	31616	46946	16397	14494	4353	25399	6556	**145761**
	Shared	84	164	55	72	19	128	40	**562**

Supplementary Table 21. SA-SSR compared with QDD for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to QDD, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 63 SSRs that QDD found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 59 of the 63 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 4 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	QDD	0	25	22	8	6	2	0	**63**
	SA-SSR	4734	18	15	2246	880	5594	1525	**15012**
	Shared	0	8076	3271	1097	233	191	0	**12868**
Overlap	QDD	0	2	0	2	0	0	0	**4**
	SA-SSR	31700	39011	13159	13463	4133	25334	6596	**133396**
	Shared	0	8099	3293	1103	239	193	0	**12927**
Exhaustive	QDD	0	0	0	0	0	0	0	**0**
	SA-SSR	31702	70721	36567	92368	22594	97461	23238	**374651**
	Shared	0	8101	3293	1105	239	193	0	**12931**

Supplementary Table 22. SA-SSR compared with SSR-Pipeline for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to SSR-Pipeline, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 987 SSRs that SSR-Pipeline found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 944 of the 987 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. 42 of the remaining 43 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The last SSR was a case where SA-SSR and SSR-Pipeline simply reported a slightly different SSR (e.g., AT vs TA). The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
Normal	SSR-Pipeline	6	386	207	152	45	166	25	**987**
	SA-SSR	1	473	271	190	70	298	51	**1354**
	Shared	4733	7621	3015	3153	1043	5487	1474	**26526**
Overlap	SSR-Pipeline	2	2	0	36	2	1	0	**43**
	SA-SSR	26963	39105	13230	11297	3286	19875	5097	**118853**
	Shared	4737	8005	3222	3269	1086	5652	1499	**27470**
Exhaustive	SSR-Pipeline	0	0	0	0	0	0	0	**0**
	SA-SSR	26963	70815	36638	90168	21745	92001	21739	**360069**
	Shared	4739	8007	3222	3305	1088	5653	1499	**27513**

Supplementary Table 23. SA-SSR compared with SSRIT for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to SSRIT, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 56 SSRs that SSRIT found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 54 of the 56 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 2 SSRs were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	SSRIT	0	12	32	12	0	0	0	**56**
	SA-SSR	4734	2570	18	2248	1113	5785	1525	**17993**
	Shared	0	5524	3268	1095	0	0	0	**9887**
Overlap	SSRIT	0	0	0	2	0	0	0	**2**
	SA-SSR	31700	41574	13152	13461	4372	25527	6596	**136382**
	Shared	0	5536	3300	1105	0	0	0	**9941**
Exhaustive	SSRIT	0	0	0	0	0	0	0	**0**
	SA-SSR	31702	73286	36560	92366	22833	97654	23238	**377639**
	Shared	0	5536	3300	1107	0	0	0	**9943**

Supplementary Table 24. SA-SSR compared with TRF for *Drosophila melanogaster*. The number of SSRs in the *Drosophila melanogaster* genome found unique to TRF, unique to SA-SSR, and shared between the two using three different sets of parameters for SA-SSR. The normal parameter set was as follows: -L 1000000 -m 1 -M 7 -n 16 -r 1 –i N. The overlap set was identical to normal with the following addition: -o. The exhaustive set was identical to overlap with the following addition: -e. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 2187 SSRs that TRF found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. 2018 of the 2187 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result. The remaining 169 SSRs were also found by SA-SSR and they fall into three different categories. The categories are overstated period size, finding different numbers of repeats, and special cases requiring the exhaustive approach by SA-SSR. 60 of the 169 are cases where TRF overstated the period size (e.g., in sequence JXOZ01000843.1, TRF reports an AGAG repeating 4 times at position 109312 while SA-SSR correctly reports an AG repeated 8 times at the same position). 2 of these appear again in the 103 that SA-SSR didn't appear to find using the exhaustive parameter set, but SA-SSR did find them, it just reported the correct period size. Obviously, reporting a longer period length than is strictly necessary to describe the SSR is misleading and certainly incorrect. AAAAAAAAA has a period size of one repeated nine times, not three repeated three times. Likewise, ATATATAT has a period size of two repeated four times, not four repeated two times.

The remaining 111 cases fall into the other two categories. 104 of the 169 are cases where TRF and SA-SSR reported different SSRs (e.g., AT vs TA) or TRF reported less repeats of the same SSR (e.g., in sequence JXOZ01001169.1, TRF reports a TTTCGA repeated 3 times at position 83483 while SA-SSR reports the same repeated 4 times). 101 of these also appear not to be found using the exhaustive parameter set because SA-SSR correctly reported SSRs with more repeats. The remaining 5 were also found by SA-SSR, but only when using the exhaustive approach because of a special, rare case involving the specific sequence and suffix sort order. The number of unique SSRs found by SA-SSR as reported using the exhaustive parameter set is also inflated.

		1	2	3	4	5	6	7	Total
Normal	TRF	5	769	373	323	61	528	128	**2187**
	SA-SSR	22	1042	551	544	210	1224	318	**3911**
	Shared	4712	7052	2735	2799	903	4561	1207	**23969**
Overlap	TRF	1	53	14	54	9	36	2	**169**
	SA-SSR	26984	39342	13358	11498	3417	20474	5263	**120336**
	Shared	4716	7768	3094	3068	955	5053	1333	**25987**
Exhaustive	TRF	0	52	13	15	8	13	2	**103**
	SA-SSR	26985	71053	36765	90366	21877	92578	21905	**361529**
	Shared	4717	7769	3095	3107	956	5076	1333	**26053**

Supplementary Table 25. SA-SSR compared with GMATo for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to GMATo, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 8 SSRs that GMATo found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. All 8 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result.

		1	2	3	4	5	6	7	Total
Normal	GMATo	0	0	0	0	0	7	1	**8**
	SA-SSR	1	0	0	0	0	13	1	**15**
	Shared	0	0	0	1	0	4	0	**5**
Overlap	GMATo	0	0	0	0	0	0	0	**0**
	SA-SSR	5	0	0	2	0	287	36	**330**
	Shared	0	0	0	1	0	11	1	**13**

Supplementary Table 26. SA-SSR compared with MREPS for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to MREPS, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
MREPS	0	0	0	0	0	0	0	**0**
SA-SSR	1	0	0	0	0	0	0	**1**
Shared	0	0	0	1	0	17	1	**19**

Supplementary Table 27. SA-SSR compared with ProGeRF for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to ProGeRF, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
ProGeRF	0	0	0	0	0	0	0	**0**
SA-SSR	1	0	0	1	0	13	1	**16**
Shared	0	0	0	0	0	4	0	**4**

Supplementary Table 28. SA-SSR compared with QDD for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to QDD, unique to SA-SSR, and shared between the two using two different sets of parameters for SA-SSR. The normal parameter set was as follows: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. The overlap set was identical to normal with the following addition: -o. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

Why did SA-SSR not find the 8 SSRs that QDD found uniquely? By default, SA-SSR reports only one SSR when multiple may be found in an overlapping location. All 8 were also found by SA-SSR when this default behavior is changed to report every SSR, even though they overlap. Naturally, the number of unique SSRs found by SA-SSR as reported using the overlap parameter set is inflated as a result.

		1	2	3	4	5	6	7	Total
Normal	QDD	0	0	0	0	0	8	0	**8**
	SA-SSR	1	0	0	1	0	17	1	**20**
	Shared	0	0	0	0	0	0	0	**0**
Overlap	QDD	0	0	0	0	0	0	0	**0**
	SA-SSR	5	0	0	3	0	290	37	**335**
	Shared	0	0	0	0	0	8	0	**8**

Supplementary Table 29. SA-SSR compared with SSR-Pipeline for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to SSR-Pipeline, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
SSR-Pipeline	0	0	0	0	0	0	0	**0**
SA-SSR	1	0	0	1	0	17	1	**20**
Shared	0	0	0	0	0	0	0	**0**

Supplementary Table 30. SA-SSR compared with SSRIT for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to SSRIT, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
SSRIT	0	0	0	0	0	0	0	**0**
SA-SSR	1	0	0	1	0	17	1	**20**
Shared	0	0	0	0	0	0	0	**0**

Supplementary Table 31. SA-SSR compared with TRF for *Escherichia coli*. The number of SSRs in the *Escherichia coli* genome found unique to TRF, unique to SA-SSR, and shared between the two using the following parameter set: -l 1 -L 600000 -m 1 -M 7 -n 16 -r 1. Any SSRs with period size greater than 7, with total length less than 16nt, or that were incorrect were excluded from this comparison.

	1	2	3	4	5	6	7	Total
TRF	0	0	0	0	0	0	0	**0**
SA-SSR	1	0	0	0	0	0	0	**1**
Shared	0	0	0	1	0	17	1	**19**

SUPPLEMENTAL REFERENCES

Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences, *Nucleic acids research*, **27**, 573.

Kolpakov, R., Bana, G. and Kucherov, G. (2003) mreps: efficient and flexible detection of tandem repeats in DNA, *Nucleic acids research*, **31**, 3672-3678.

Lopes, R.d.S.*, et al.* (2015) ProGeRF: Proteome and Genome Repeat Finder Utilizing a Fast Parallel Hash Function, *BioMed research international*, **2015**.

Meglécz, E.*, et al.* (2014) QDD version 3.1: a user- friendly computer program for microsatellite selection and primer design revisited: experimental validation of variables determining genotyping success rate, *Molecular ecology resources*, **14**, 1302-1313.

Miller, M.P.*, et al.* (2013) SSR_pipeline: A bioinformatic infrastructure for identifying microsatellites from paired-end Illumina high-throughput DNA sequencing data, *Journal of Heredity*, est056.

Temnykh, S.*, et al.* (2001) Computational and experimental analysis of microsatellites in rice (Oryza sativa L.): frequency, length variation, transposon associations, and genetic marker potential, *Genome research*, **11**, 1441-1452.

Wang, X., Lu, P. and Luo, Z. (2013) GMATo: A novel tool for the identification and analysis of microsatellites in large genomes, *Bioinformation*, **9**, 541-544.

APPENDIX 7

Chapter 7 – File S1

SUPPLEMENTARY BIOINFORMATICS METHODS

This document contains an explanation of the bioinformatics methods required for incompatibility group/replicon typing and plasmid characterization. It is expanded from our paper in Genome. This document will begin with an overview of the process and will be followed by a detailed description of the methods.

Overview

The process begins with one fasta file and multiple GenBank files. The fasta file is the local download of the PlasmidFinder database referenced in our paper. The GenBank files come from the Entrez search strategy also described in the paper. The ultimate output is a CSV file and text-based report file for each input GenBank file. The CSV file contains basic information (e.g., plasmid length), the incompatibility group(s) the plasmid best aligns to, accession numbers of identical plasmids, some gene/function annotation based on key term searches of the GenBank file's CDS regions, and some other metadata extracted from the GenBank files. The text-based report is a file containing various information and statistics about each group of plasmids from the various input GenBank files. We also generated a tree to help visualize the identical plasmids.

Our process occurs stepwise, with most steps requiring the output from the previous steps. As our project developed, additional steps were inserted or modified. While most steps do

depend on the output of the previous step(s), the order is in many instances arbitrary. The code is published online in this GitHub repository (https://github.com/ridgelab/plasmidCharacterization). Each output CSV file requires the following input processed from the "raw" input data (in no particular order): (a) a list of identical plasmids for each accession, (b) extracted metadata from the GenBank files, (c) gene/function annotations extracted from the GenBank files, and (d) a list of incompatibility groups. Each output statistics report file is created based on each CSV file just described.

Identical Plasmids

First, a blast database was created with makeblastdb. Each plasmid sequence (which would have to be extracted from the GenBank files) is aligned with blastn to each other plasmid sequence in a pairwise fashion. Hits were kept only if the percent identity was >=98%. Plasmids were considered identical if the hits covered >=98% of both the query and the subject sequence. We created a tree using a simple distance metric to help visualize the identical plasmids. The distance metric is the sum of the query and subject covered bases divided by the sum of the length of the query and subject sequences (see step 25 for details). The Newick formatted tree was made from the distance matrix using the makeNewick.py script from CAM (Miller et al. 2019) and is available on GitHub at https://github.com/ridgelab/cam. makeblastdb and blastn are part of the BLAST+ Suite (Altschul et al. 1990; Camacho et al. 2009).

GenBank Metadata

The sequencing technology used to sequence each plasmid was identified with GNU AWK. The remaining metadata was also obtained from the GenBank files using GNU AWK.

The remaining data points are as follows: country of origin for the plasmid, isolation source for the plasmid, plasmid collection data, and source organism.

GenBank Annotations

This is by far the most complicated part of the process. First, search regions were extracted from the GenBank files. The search regions were the function, gene, note, and product sections of the CDS features. We then identified matches in these regions to key terms (these key terms were obtained as described in our paper). The search occurred under the following strategy:

The search terms are each part of one or more categories. Each can belong to multiple categories, but only if the categories are subsets of each other. Five principal categories exist, two of which have subcategories. The category structure is as follows:

- Antimicrobial Resistance
    - Beta-lactamase
        - Beta-lactamase Special
- Toxin/Antitoxin System
- DNA Maintenance/Modification
    - DNA Maintenance/Modification Special
- Mobile Genetic Elements
- Hypothetical Genes

The strategy could be described as top-to-bottom, in-to-out; i.e., Antimicrobial Resistance is more important that Toxin/Antitoxin System and Beta-lactamase Special is more important than Beta-lactamase and Antimicrobial Resistance. The reason these are

shown nested instead of simply above their parents is because a match for a Beta-lactamase Special search term will increment the count for not only itself, but also its parents. If no matches are found, the CDS region being searched is classified as "Other". Some CDS regions will never be searched for these terms if they first match a term in a special "Ignored" category. Provided a CDS region is not to be ignored, it will be searched with Beta-lactamase Special terms, then Beta-lactamase terms, then Antimicrobial Resistance Terms, then Toxin/Antitoxin System terms, and so-forth, until a match is found (thus halting the search on this CDS region) or no more search terms remain, in which case it is assigned to the "Other" category. All CDS regions are converted to lowercase before being searched as described. These terms are listed, with their associated Python regular expressions, in the doc directory of the online repository.

Incompatibility Groups

The incompatibility fasta sequences were downloaded from the PlasmidFinder database as previously described. This was turned into a database using makeblastdb. Each plasmid sequence was then aligned to the database using blastn and hits were retained only if the percent identity was >=80%. Hits were further dropped if the subject (the sequences in the database) coverage was <60%. The “best” hits were then used to determine which incompatibility group(s) applied to each plasmid. “Best” is defined as the result(s) with the highest percent identity and those that have percent identities within only 1 percent of the highest one. makeblastdb and blastn are part of the BLAST+ Suite (Altschul et al. 1990; Camacho et al. 2009).

Detailed Methods

This section is a more detailed explanation of the bioinformatics methods required for incompatibility group/replicon typing and plasmid characterization. Please note that most of these steps will be simple data formatting. Also note that it would have been easier in some cases to combine multiple steps into one. The choice to separate each piece of the process was for clarity and to enable another to modify this process for their own purposes. Additionally, some steps might have made better sense in different orders. This process evolved as the project changed; we recognize alternate orders are plausible. For our work, all steps could be run interactively, i.e., not requiring a high-performance computing (HPC) architecture. Our work was completed on a machine running Red Hat Enterprise Linux.

Summary

This process begins with one fasta file and multiple GenBank files. The formats for these files are described in steps 1 and 3, respectively. The fasta file contains the incompatibility group sequences. In our work, this was a download of the PlasmidFinder v1.3 *Enterobacteriaceae* database (Carattoli et al. 2014). The GenBank files contain one or more GenBank records in them, where each record could itself be considered a GenBank file for a single accession number. Thus, these GenBank files are concatenations of multiple GenBank records. Effectively, this is how we grouped accessions of interest. The same accession may appear in multiple groupings. Note, if you attempt to re-use our process with your own data and have GenBank files as a single file per accession, combining them into groups will feel unnecessary. We began this way because that is what we had to start with.

The results of the entire process are CSV files with information about each plasmid in a group and a text file with summary statistics about each group. The file contains basic information (e.g., plasmid length), the incompatibility group(s) the plasmid best aligns to, and

some gene/function annotation based on key term searches of the GenBank file's CDS regions. To accomplish this, each (input) group GenBank file is split into a single GenBank file per accession and the sequences are extracted as fasta files. The sequence lengths are recorded, and these sequences are individually aligned (using the NCBI BLAST+ Suite (Altschul et al. 1990; Camacho et al. 2009)) to the incompatibility group sequences. After filtering out the "best" alignments, the incompatibility group is determined and saved for later assimilation into the final outputs. The CDS regions are extracted from the GenBank files and searched for key terms using regular expressions. Each key term belongs to one or more categories. Matches in each category are counted and summarized in the final output. For more details on this searching strategy, please see step #14. The key terms are listed with their Python regular expression in the supplement of our paper. Additional information, e.g., sequencing platforms, country, etc., is also available in the final outputs.

This summary concludes with an outline of the steps. Each step will then be addressed in detail. The code in the detailed steps has, in many cases, been simplified. In other cases, the code is several pages long and would be difficult to copy and paste effectively. Especially with the Python code, readability suffers as lines wrap because a standard page is not wide enough to contain some code statements on a single line. Accordingly, we encourage you to visit the online repository for the code: https://github.com/ridgelab/plasmidCharacterization.

Outline of Steps

Step 1. Format Incompatibility Groups Fasta File
Step 2. Create Incompatibility Groups BLAST Database
Step 3. Split Multi-Accession GenBank Files
Step 4. Extract ORIGIN Sequence from GB to Fasta
Step 5. Extract Group Lists
Step 6. BLAST Incompatibility Groups

Step 7. Subset BLAST Results by Coverage Cutoff of 60%
Step 8. Add Incompatibility Group Family as Column to BLAST Results
Step 9. Filter Best Matches in BLAST Results
Step 10. Extract Incompatibility Families
Step 11. Extract Sequencing Technologies
Step 12. Extract Source Information
Step 13. Extract Plasmid Search Regions
Step 14. Identify Plasmid Matches
Step 15. Summarize Plasmid Matches
Step 16. Drop Plasmids
Step 17. Create Plasmid BLAST Database
Step 18. BLAST Plasmid
Step 19. Extract Identical Plasmids with BLAST Result Coverage Cutoff of 98%
Step 20. Fix Identical Plasmid Non-concordance
Step 21. Generate Plasmid CSVs
Step 22. Create Group CSVs from Plasmid CSVs
Step 23. Create Group Matches from Plasmid Matches
Step 24. Calculate Group Statistics from Group CSV
Step 25. Create Distance Matrix
Step 26. Create Distance Tree
Step 27. Add Leaf Labels to Tree
Step 28. Add Color to Leaf Labels

Step 1. Format Incompatibility Groups Fasta File

Input: Fasta file with incompatibility group sequences. Each sequence may be on one or more lines. The headers might start with "Inc".

Output: Same fasta file as the input, but sequences occur on only one line. Headers without "Inc" now have "Inc" prepended.

Code:

Bash Command

```
awk -f formatIncGroupFasta.awk \
         original_incomp-grp.fasta \
         > incomp-grp.fasta
```

AWK Script (formatIncGroupFasta.awk)

```
#! /bin/awk -f

{
  if ( $0 ~ /^>.+$/ ) {

         if ( NR != 1 ) {
                printf "\n";
         }

         if ( $0 ~ /^>Inc.+$/ ) {
                print $0;
         }
         else {
                printf "%s%s\n", ">Inc", substr($0, 2);
         }
  }
  else {
         printf "%s", $0;
  }
}

END {
  printf "\n";
}
```

Step 2. Create Incompatibility Groups BLAST database

Input: Fasta file with incompatibility group sequences. Each sequence is on only one line. The headers start with ">Inc".

Output: BLAST database of the incompatibility group sequences.

Code:

Bash Command

```
makeblastdb \
          -dbtype nucl \
          -in  incompatibility.fasta \
          -input_type fasta \
          -title incompatibility \
          -parse_seqids \
          -hash_index \
          -out incompatibility \
          -max_file_sz 2GB \
          -logfile makeBlastDB.log
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (Altschul et al. 1990; Camacho et al. 2009).

Step 3. Split Multi-Accession GenBank Files

Input: 1+ GenBank files, each with 1+ records. Each record is itself a GenBank file for a single Accession. Thus, the multi-accession GenBank files are simply concatenations of multiple single-accession GenBank files. Assume that these GenBank files are in a directory called `original_gb`.

Output: One GenBank file for each accession. If the same accession exists in more than one multi-accession file, assume they are the same and overwrite it. Assume that the output GenBank files will be in a directory called `plasmid_gb`.

Code:

Bash Command

```
cd plasmid_gb

while read ifn
do
        awk -f splitMultiGB.awk "${ifn}"

done < <(ls -1 original_gb/*.gb)
```

AWK Script (splitMultiGB.awk)

```
#! /bin/awk -f

BEGIN {
  FS="[ ]+";
  accession="";
  ofn="";
}

{
  if ($0 == "//" || $0 == "")
  {
        accession = "";
        ofn = "";
  }
  else if ($1 == "LOCUS")
  {
        accession = $2;
        ofn = accession ".gb";
        print $0 > ofn;
  }
  else
  {
        print $0 >> ofn;
  }
}
```

```
END {
  print "done splitting " FILENAME " by accession";
}
```

Step 4. Extract ORIGIN Sequence from GB to Fasta

Input: One GenBank file with a single accession in it. Assume it is in the directory `plasmid_gb` and it is named after the pattern `${ACCESSION}.gb`.

Output: One Fasta file with the sequence from the ORIGIN section of the GenBank file. The Fasta file has sequences that are each on only one line. It will be in the directory `plasmid_fasta`.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" ".gb"`

        awk -f extractOriginSeqFromGBtoFasta.awk \
                "plasmid_gb/${ACCESSION}.gb" \
                > "plasmid_fasta/${ACCESSION}.fasta"

done < <(ls -1 plasmid_gb/*.gb)
```

AWK Script (extractOriginSeqFromGBtoFasta.awk)

```
#! /bin/awk -f

BEGIN {
  FS = "[ ]+";
  origin_found = 0; # false
}

{
  if (origin_found)
  {
        sub(/ *[0-9]+ /, "", $0);
        gsub(/ +/, "", $0);
        printf toupper($0);
  }
  else if ($1 == "ORIGIN")
  {
        origin_found = 1; # true

        print ">" gensub(/^(.+)\.gb$/, "\\1", "-1", gensub(/^.*\//,
"", "-1", FILENAME));
  }
}
```

```
END {
  printf "\n";
  print "done extracting ORIGIN seq from " FILENAME " to fasta" >
"/dev/stderr";
}
```

Step 5. Extract Group Lists

Input: One GenBank file with multiple accessions in it. Assume it is in the directory `original_gb` and it is named after the pattern `${GROUP}.gb`.

Output: Multiple text files, each with the extension ".list". Each file is a line separated list of accession numbers that make up the group. The files will be in a directory called `groups` with the name `${GROUP}.list`.

Code:

Bash Command

```
while read ifn
do
        awk -f extractGroupLists.awk \
                "${ifn}"

done < <(ls -1 original_gb/*.gb)
```

AWK Script (extractGroupLists.awk)

```
#! /bin/awk -f

BEGIN {
  FS="[ ]+";
  accession="";
  ofn="";
}

{
  if (NR == 1)
  {
        ofn = gensub(/^(.+)\.gb$/, "\\1", "-1", gensub(/^.*\//, "", "-
1", FILENAME)) ".list";
  }

  if ($1 == "LOCUS")
  {
        accession = $2;
        print accession >> ofn;
  }
}

END {
  print "done extracting accessions from " FILENAME;
}
```

Step 6. BLAST Incompatibility Groups

Input: Fasta files. Each contains the sequence from a single accession. Assume they are in the directory `plasmid_fasta` and they are named after the pattern `${ACCESSION}.fasta`.

Input: The incompatibility groups BLAST database created in step #1. It is named `incompatibility`.

Output: One tab-separated value file for each input file. Each file is a modified version of the BLAST output format 6. The format is specified as seen using the -outfmt option with blastn. The columns are as follows: qseqid, sseqid, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, qseq, and sseq. The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c.tsv`. Note that a match was not included in the output if the percent identity was <80%.

Code:

Bash Command

```
THREADS=8

while read ifn
do
        ACCESSION=`basename "${ifn}" ".fasta"`

        blastn \
                -query "${ifn}" \
                -strand both \
                -task blastn \
                -db icompatibility \
                -out blast_results/${ACCESSION}_fmt6c.tsv \
                -outfmt "6 qseqid sseqid pident length evalue qframe
qlen qstart qend sframe slen sstart send qseq sseq" \
                -num_threads ${THREADS} \
                -perc_identity 80

done < <(ls -1 plasmid_fasta/*.fasta)
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (Altschul et al. 1990; Camacho et al. 2009).

Step 7. Subset BLAST Results by Coverage Cutoff of 60%

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. Assume they are in the directory `blast_results` and they are named after the pattern `${ACCESSION}_fmt6c.tsv`.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except some results may be omitted if the coverage was less than 60%. The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c_cov60.tsv`. Note that a new column was inserted as column number 14 (1-based indexing). The columns will now be as follows: qseqid, sseqid, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" "_fmt6c.tsv"`

        awk -f subCovCutoff60.awk \
                "${ifn}" \
                > "blast_results/${ACCESSON}_fmt6c_cov60.tsv"

done < <(ls -1 blast_results/*_fmt6c.tsv)
```

AWK Script (subCovCutoff60.awk)

```
#! /bin/awk -f

BEGIN {
    FS="\t";
    OFS="\t";
    ORS="\n";
    count=0;
}

{
    # 4 = length, 11 = slen, scov = length / slen
    scov = $4 / $11;
    if (scov >= 0.6)
    {
        count += 1

        # keep 1-13, add new column, keep 14-15 (will become 15-16)
        for (i = 1; i <= 13; i++)
        {
                printf "%s", $i OFS;
        }
```

```
          printf "%f", scov OFS;

          for (i = 14; i <= NF; i++)
          {
                  printf "%s", $i (i == NF ? ORS : OFS);
          }
     }
}

END {
     print FILENAME ": " count > "/dev/stderr";
}
```

Step 8. Add Incompatibility Group as Column to BLAST Results

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. It has an added column with the subject coverage and has only records with coverage >60%. Assume they are in the directory `blast_results` and they are named after the pattern `${ACCESSION}_fmt6c_cov60.tsv`.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except that an additional column is added. This column has the family or root of the incompatibility group from column #2 (sseqid). The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c_cov60_fam.tsv`. Note that a new column was inserted as column number 3 (1-based indexing). The columns will now be as follows: qseqid, sseqid, fam, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" "_fmt6c_cov60.tsv"`

        awk -f addFamCol.awk \
                "${ifn}" \
                > "blast_results/${ACCESSON}_fmt6c_cov60_fam.tsv"

done < <(ls -1 blast_results/*_fmt6c_cov60.tsv)
```

AWK Script (addFamCol.awk)

```
#! /bin/awk -f

BEGIN {
  FS="\t";
  OFS="\t";
  ORS="\n";
}

{
  # 2 = subject_id, keep 1-2, add new column,
  #keep 3-16 (will become 4-17)
  for (i = 1; i <= 2; i++)
  {
        printf "%s", $i OFS;
  }

  printf "%s", gensub(/^([^(_]+).*$/, "\\1", "-1", $2) OFS;
```

```
  for (i = 3; i <= NF; i++)
  {
          printf "%s", $i (i == NF ? ORS : OFS);
  }
}
```

Step 9. Filter Best Matches in BLAST Results

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. It has two added columns with the subject coverage (and has only records with coverage >60%) and family. Assume they are in the directory `blast_results` and are named after the pattern `${ACCESSION}_fmt6c_cov60_fam.tsv`.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except that some results are omitted. The "best" results are retained. "Best" is defined as the result(s) with the highest percent identity and those that have percent identities within only 1 percent of the highest one. The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c_cov60_fam_best.tsv`. As in the input file, the columns will be as follows: qseqid, sseqid, fam, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" "_fmt6c_cov60_fam.tsv"`

        python3 filterBestResults.py \
                "${ifn}" \
                > "blast_results/${ACCESSON}_fmt6c_cov60_fam_best.tsv"

done < <(ls -1 blast_results/*_fmt6c_cov60_fam.tsv)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (filterBestResults.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 10. Extract Incompatibility Families

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. It has two added columns with the subject coverage (and has only records with coverage >60%) and family. Only the "best" results remain. Assume they are in the directory `blast_results` and are named after the pattern `${ACCESSION}_fmt6c_cov60_fam_best.tsv`.

Output: One file for each input file. Each file is a line-delimited list of incompatibility group roots/families. The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_families.list`.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}"
"_fmt6c_cov60_fam_best.tsv"`

        cut -f 3 "${ifn}" \
                | sort \
                | uniq \
                > blast_results/"${ACCESSON}_families.list"

done < <(ls -1 blast_results/*_fmt6c_cov60_fam_best.tsv
```

Step 11. Extract Sequencing Technologies

Input: GenBank files for each plasmid. We assume they are in the directory `plasmid_gb` and they are named after the pattern `${ACCESSION}.gb`.

Output: One tab-separated value file. The file has one column for the accession number, one column containing the sequencing technology string taken from the GenBank file, and several columns containing counts for the various sequencing technologies and groups of technologies. The file is assumed to be called `seqTechs.tsv` in the `plasmid_seqTech` directory. The columns are as follows: accession, sequencing_technologies, num_total, num_short, num_long, num_illumina, num_454, num_abi, num_sanger, num_torrent, num_pacbio, and num_nanopore.

Code:

Bash Command

```
printf "%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n" \
  'accession' \
  'sequencing_technologies' \
  "num_total" \
  "num_short" \
  "num_long" \
  "num_illumina" \
  "num_454" \
  "num_abi" \
  "num_sanger" \
  "num_torrent" \
  "num_pacbio" \
  "num_nanopore" \
  > "plasmid_seqTech/seqTech.tsv"

while read ifn
do
         ACCESSION=`basename "${ifn}" ".gb"`

         printf '%s\t' "${ACCESSION}" >> "plasmid_seqTech/seqTech.tsv"

         awk -f sequesterSeqTech.awk \
                "${ifn}" \
                >> "plasmid_seqTech/seqTech.tsv"

done < <(ls -1 plasmid_gb/*.gb)
```

AWK Script (sequesterSeqTech.awk)

This script is too long to reasonably represent in this document. Please view it in the freely-accessible online repository.

Step 12. Extract Source Information

Input: GenBank files for each plasmid. We assume they are in the directory `plasmid_gb` and they are named after the pattern `${ACCESSION}.gb`.

Output: One tab-separated value file. The file has one column for the accession number and one column for each of these subsections of the GenBank file source section: organism, isolation source, country, and collection_date. The file is assumed to be called `sourceInfo.tsv` in the `plasmid_sourceInfo` directory. The columns are as follows: accession, organism, isolation_source, country, and collection_date.

Code:

Bash Command

```
printf "%s\t%s\t%s\t%s\t%s\n" \
  'accession' \
  'organism' \
  'isolation_source' \
  'country' \
  'collection_date' \
  > "plasmid_sourceInfo/sourceInfo.tsv"

while read ifn
do
         ACCESSION=`basename "${ifn}" ".gb"`

         printf '%s\t' "${ACCESSION}" >>
"plasmid_sourceInfo/sourceInfo.tsv"

         awk -f snagSourceInfo.awk \
                "${ifn}" \
                >> " plasmid_sourceInfo/sourceInfo.tsv "

done < <(ls -1 plasmid_gb/*.gb)
```

AWK Script (snagSourceInfo.awk)

This script is too long to reasonably represent in this document. Please view it in the freely-accessible online repository.

Step 13. Extract Plasmid Search Regions

Input: This Python program requires 3 inputs. 1- The accession number of the plasmid it will extract the search regions from. 2- The directory where the output will be placed. 3- The directory where the GenBank file is located for that plasmid. We assume the GenBank file is named after the pattern `${ACCESSION}.gb`.

Output: One text file containing the lines from input GenBank file that will be searched using the key terms. We assume the output file will be named after the following pattern: `${ACCESSION}_searchRegions.txt`. For convenience, it will also generate a copy of the input GenBank file with shell color codes, marking the CDS and source regions in blue, the portions of the CDS and source regions that will be included in green, and the portion of the CDS and source regions that will not be searched in red. The FEATURE line will be blue. This file will have the same name as the `.txt` file but will have the extension `.gb` instead of `.txt`. Note that intended search space is to consider each CDS region as a separate entity. However, only the following subsections of each CDS region are to be considered: `/function`, `/gene`, `/note`, and `/product`.

Code:

Bash Command

```
while read ifn
do
          ACCESSION=`basename "${ifn}" ".gb"`

          python3 extractPlasmidSearchRegions.py \
                  "${ACCESSION}" \
                  plasmid_searchRegions \
                  plasmid_gb

done < <(ls -1 plasmid_gb/*.gb)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (extractPlasmidSearchRegions.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 14. Identify Plasmid Matches

Input: This Python program requires 3 inputs. 1- The accession number of the plasmid in which it will identify matches. 2- The directory where the input search regions file is located. 3- The directory where the output matches will be placed. We assume the input search regions file is named after the pattern `${ACCESSION}_searchRegions.txt`.

Output: One text file containing the lines from input GenBank file that will be searched using One tab-separated value file containing matches. We assume the output file will be named after the following pattern: `${ACCESSION}_matches.tsv`. The columns of the file are as follows:

1. Ignored (True/False)
2. Categories (c1[,c2,…,cN])
3. Search Term
4. CDS Region

Column 1 is a simple flag denoting if the term was to be ignored. This could also be determined based on the second column, but it was convenient to have a simple flag as its own column. Column 2 contains the category (categories) that the search term belonged to. Column 3 contains the regular expression used. Column 4 contains the CDS region that was searched (all tabs and newlines were converted to `\t` (backslash and a t, not a tab) and `\n` (backslash and an n, not a newline) to not interfere with the tab-separated value file format and keep each record on a single line).

Search Strategy: The search terms are each part of one or more categories. It can belong to multiple categories only if the categories are subsets of each other. Five principal categories exist, two of which have subcategories. The category structure is as follows:

- Antimicrobial Resistance
  - Beta-lactamase
    - Beta-lactamase Special
- Toxin/Antitoxin System
- DNA Maintenance/Modification
  - DNA Maintenance/Modification Special
- Mobile Genetic Elements
- Hypothetical Genes

The strategy could be described as top-to-bottom, in-to-out; i.e., Antimicrobial Resistance is more important that Toxin/Antitoxin System and Beta-lactamase Special is more important than Beta-lactamase and Antimicrobial Resistance. The reason these are shown nested instead of simply above their parents is because a match for a Beta-lactamase Special search term will increment the count for not only itself, but also its parents. If no matches are found, the CDS region being searched is classified as "Other". Some CDS regions will never be searched for these terms if they first match a term in a special "Ignored" category. Provided a CDS region is not to be ignored, it will be searched with Beta-lactamase Special terms, then

Beta-lactamase terms, then Antimicrobial Resistance Terms, then Toxin/Antitoxin System terms, and so-forth, until a match is found (thus halting the search on this CDS region) or no more search terms remain (it is assigned to the "Other" category). All CDS regions are converted to lowercase before being searched as described. See our paper for a table of search terms.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" "_searchRegions.txt"`

        python3 identifyPlasmidMatches.py \
                "${ACCESSION}" \
                plasmid_searchRegions \
                plasmid_matches

done < <(ls -1 plasmid_searchRegions/*_searchRegions.txt)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (identifyPlasmidMatches.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 15. Summarize Plasmid Matches

Input: This Python program requires 2 inputs. 1- The accession number of the plasmid in which it will summarize matches. 2- The directory where the input matches are to be found and the output summarized matches will be placed. We assume the input matches file is named after the pattern `${ACCESSION}_matches.tsv` in a directory called `plasmid_matches`.

Output: One tab-separated value file containing summarized matches. It will have two lines only. The first is a header line; the second the data. We assume the output file will be named after the following pattern: `${ACCESSION}_matches-summary.tsv` in a directory called `plasmid_matches`. The columns of the file are as follows:

1. Accession #
2. Antimicrobial Resistance CDS
3. Antimicrobial Resistance CDS %
4. Beta-lactamase CDS
5. Beta-lactamase CDS %
6. Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy #
7. Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy # % of Beta-lactamase
8. Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy # % of Total
9. Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Absent (Yes/No)
10. Plasmid Transfer CDS
11. Plasmid Transfer CDS %
12. Toxin/Antitoxin System CDS
13. Toxin/Antitoxin System CDS %
14. Toxin/Antitoxin System Present (Yes/No)
15. DNA Maintenance/Modification CDS
16. DNA Maintenance/Modification CDS %
17. DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy #
18. DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy # % of DNA Maintenance/Modification
19. DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy # % of Total
20. DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Present (Yes/No)
21. Mobile Genetic Elements CDS
22. Mobile Genetic Elements CDS %
23. Hypothetical Genes CDS
24. Hypothetical Genes CDS %
25. Other CDS
26. Other CDS %
27. Total CDS

This data, with the exception of the first column, will be copied into the plasmid csv file created later. Column number 6 will also be used to drop plasmids.

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" "_sorted_matches.tsv"`

        python3 summarizePlasmidMatchInfo.py \
                "${ACCESSION}" \
                plasmid_matches

done < <(ls -1 plasmid_matches/*_sorted_matches.tsv)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (summarizePlasmidMatchInfo.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 16. Drop Plasmids

Input: This script acts on all the plasmids directly (i.e., not calling on a subroutine in Python or AWK for each of the plasmids). It requires no user input directly as it ascertains the plasmid accession numbers from file names. It also relies on the directory structure to find the files named after the pattern `${ACCESSION}_matches-summary.tsv` in a directory called `plasmid_matches`.

Output: Two new directories in the `groups` directory: `keep` and `discard`. Inside the `discard` directory will be a file called `discard.list`. It will contain the accession numbers (one per line) that are to be excluded from the rest of the analysis. The same is true in the `keep` directory, except the accession numbers are the ones that will be retained for the rest of the analysis and the file will be called `keep.list`. Also in the `keep` directory is a new group list file for each of the groups found in `groups` directory. These lists are the same as the originals except that the discarded accessions have been removed.

Code:

Bash Command

```
while read ifn
do
       GROUP=`basename "${ifn}" ".list"`

       while read ACCESSION
       do
            COUNT=`tail -n 1 \
                "plasmid_matches/${ACCESSION}_matches-summary.tsv" \
                | cut -d '\t' -f 6 \
                | tr -d '"'`

            if [ $COUNT -ge 1 ] && [ $COUNT -le 6 ]
            then
                printf "${ACCESSION}\n" >> "groups/keep/${GROUP}.list"
                printf "${ACCESSION}\n" >> "groups/keep/keep.list"
            else
                printf "${ACCESSION}\n" >> "groups/discard/discard.list"
            fi

       done < "${ifn}"

done < <(ls -1 groups/*.list)
```

Step 17. Create Plasmid BLAST Database

Input: Fasta files for each plasmid. We assume they are in the directory `plasmid_fasta` and they are named after the pattern `${ACCESSION}.fasta`.

Output: One BLAST database. We are creating this so we can do pairwise BLAST between the plasmid fastas. The objective is to identify plasmids that are "identical". Identical will, for our purposes, be defined as >=98% percent identity and >=98% query *and* subject coverage.

Code:

Bash Command

```
cat plasmid_fasta/*.fasta > plasmid_blast_results/plasmids.fasta

cd plasmid_blast_results

makeblastdb \
          -dbtype nucl \
          -in plasmids.fasta \
          -input_type fasta \
          -title plasmids \
          -parse_seqids \
          -hash_index \
          -out plasmids \
          -max_file_sz 2GB \
          -logfile makeBlastDB.log
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (Altschul et al. 1990; Camacho et al. 2009).

Step 18. BLAST Plasmid

Input: Fasta files. Each contains the sequence from a single accession. Assume they are in the directory `plasmid_fasta` and they are named after the pattern `${ACCESSION}.fasta`.

Input: The plasmids BLAST database created in step #12. It is named `plasmids`.

Output: One tab-separated value file for each input file. Each file is a modified version of the BLAST output format 6. The format is specified as seen using the -outfmt option with blastn. The columns are as follows: qseqid, sseqid, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, qseq, and sseq. The files will be in a directory called `plasmid_blast_results` and named after the pattern `${ACCESSION}_fmt6c.tsv`. Note that a match was not included in the output if the percent identity was <98%.

Code:

Bash Command

```
THREADS=8

while read ifn
do
     ACCESSION=`basename "${ifn}" ".fasta"`

     blastn \
        -query "${ifn}" \
        -strand both \
        -task blastn \
        -db plasmids \
        -out plasmid_blast_results/${ACCESSION}_fmt6c.tsv \
        -outfmt "6 qseqid sseqid pident length evalue qframe qlen
qstart qend sframe slen sstart send qseq sseq" \
        -num_threads ${THREADS} \
        -perc_identity 98

done < <(ls -1 plasmid_fasta/*.fasta)
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (Altschul et al. 1990; Camacho et al. 2009).

Step 19. Extract Identical Plasmids with BLAST Result Coverage Cutoff of 98%

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the plasmids BLAST database. Assume they are in the directory `plasmids_blast_results` and they are named after the pattern `${ACCESSION}_fmt6c.tsv`. Note that these BLAST results all have >=98% sequence identity.

Output: One file for each input file. Each file is a line-delimited list of accessions associated with "identical" plasmids. The files will be in a directory called `plasmid_blast_results` and named after the pattern `${ACCESSION}_identicalPlasmids.list`. The BLAST results are further filtered based on the query and subject coverage; each must be >= 98%. Coverage is determined based on number of bases covered by the other sequence. This coverage can come from one or more BLAST hits, as long as the total number of covered bases is >=98% of the number of possible bases.

Code:

Bash Command

```
while read ifn
do
          ACCESSION=`basename "${ifn}" "_fmt6c.tsv"`

          python3 queryAndSubCovCutoff98-multiHit.py \
                  "${ifn}" \
                  >
"plasmid_blast_results/${ACCESSON}_identicalPlasmids.list"

done < <(ls -1 blast_results/*_fmt6c.tsv)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (queryAndSubCovCutoff98.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 20. Fix Identical Plasmid Non-concordance

Input: This Python program requires 6 inputs. 1- the path of the coverage information files. 2- the path of the identical plasmid files. Inputs 3-6 are suffixes to file names; the assumed base of the name is the accession number. 3- the suffix of the input coverage info file. 4- the suffix of the output coverage info file. 5- the suffix of the input identical plasmids file. 6- the suffix of the output identical plasmids file.

Output: Two text files. The first will be the output coverage info file. It will be the concordant version of its respective input file. The second will be the output identical plasmids file. It will be the concordant version of its respective input file. We assume they are both in the `plasmid_blast_results` directory and have the suffixes `_covInfo_concordant.tsv` and `_identicalPlasmids_concordant.list`, respectively. Another term for concordance might be reciprocal. This step accounts for inconsistencies in BLAST outputs. One might get hits from sequence A to B with >=98% identity and >=98% query and subject coverage, yet get no hits from B to A. This non-concordance is "fixed" in this step to force reciprocity of the BLAST hits. These hits are not updated in the BLAST output file, though the outcome is affected in the two output files from this step.

Code:

Bash Command

```
python3 fixIdenticalPlasmidsNonConcordance.py \
         plasmid_blast_results \
         plasmid_blast_results \
         "_covInfo.tsv" \
         "_covInfo_concordant.tsv" \
         "_identicalPlasmids.list" \
         "_identicalPlasmids_concordant.list"
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (fixIdenticalPlasmidsNonConcordance.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 21. Generate Plasmid CSVs

Input: This Python program requires 8 inputs. 1- The accession number of the plasmid it will generate a CSV file for. 2- The directory where the output CSV file is to be placed. 3- The directory where the plasmid fasta file is located. We assume it is named after the pattern `${ACCESSION}.fasta`. 4- The directory where the input plasmid matches file is located. We assume it is named after the pattern `${ACCESSION}_matches-summary.tsv`. 5- The directory where the input incompatibility groups (derived from the BLAST results) are located. We assume it is named after the pattern `${ACCESSION}_families.list`. 6- The filename of the source info. We assume it is named `sourceInfo.tsv` in the `plasmid_sourceInfo` directory. 7- The directory of the plasmid BLAST results. We assume it is called `plasmid_blast_results`. 8- The filename of the sequence technologies information. We assume it is at `plasmid_seqTech/seqTech.tsv`.

Output: One comma-separated value file. It will be placed in the directory specified in the input position 2. We assume the output file will be named after the following pattern: `${ACCESSION}.csv`. The columns of the file are as follows:

> "Accession #","Identical Plasmids","Source: Organism","Source: Isolation Source","Source: Country","Source: Collection Date","Sequencing Technologies","Sequencing Technologies Count","Short Read Count","Long Read Count","Illumina Count","Roche 454 Count","ABI Solid Count","Sanger Count","Ion Torrent Count","PacBio Count","ONT Count","Plasmid Length","Antimicrobial Resistance CDS","Antimicrobial Resistance CDS %","Beta-lactamase CDS","Beta-lactamase CDS %","Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy #","Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy # % of Beta-lactamase","Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Copy # % of Total","Beta-lactamase Special (Carbapenem*,IMP,KPC,NDM,VIM) Absent (Yes/No)","Plasmid Transfer CDS","Plasmid Transfer CDS %","Toxin/Antitoxin System CDS","Toxin/Antitoxin System CDS %","Toxin/Antitoxin System Present (Yes/No)","DNA Maintenance/Modification CDS","DNA Maintenance/Modification CDS %","DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy #","DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy # % of DNA Maintenance/Modification","DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Copy # % of Total","DNA Maintenance/Modification Special (mucA,mucB,polymerase,umuC,umuD) Present (Yes/No)","Mobile Genetic Elements CDS","Mobile Genetic Elements CDS %","Hypothetical Genes CDS","Hypothetical Genes CDS %","Other CDS","Other CDS %","Total CDS","Incompatibility Groups"

Code:

Bash Command

```
while read ifn
do
        ACCESSION=`basename "${ifn}" ".fasta"`

        python3 generatePlasmidCSV.py \
                "${ACCESSION}" \
                plasmid_csv \
                plasmid_fasta \
                plasmid_matches \
                blast_results \
                plasmid_sourceInfo/sourceInfo.tsv \
                plasmid_blast_results \
                plasmid_seqTech/seqTech.tsv

done < <(ls -1 plasmid_fasta/*.fasta)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (generatePlasmidCSV.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 22. Create Group CSVs from Plasmid CSVs

Input: The inputs required are the group list files that contain the plasmids in each group (see step #4) and the individual plasmid CSVs (see step #12). The group list files are assumed to be in the directory `groups` and named after the pattern `${GROUP}.list`. The plasmid CSVs are assumed to be in the `plasmid_csv` directory and named after the pattern `${ACCESSION}.csv`.

Output: One comma-separated value file containing the same header line as all the plasmid CSVs and a concatenation of the non-header lines from the plasmid CSVs. We assume the output file will be in the directory `group_csv` and will be named after the following pattern: `${GROUP}.csv`.

Code:

Bash Command

```
while read ifn
do
        GROUP=`basename "${ifn}" ".list"`
        ofn="group_csv/${GROUP}.csv"

        # get and write a header
        hfn=plasmid_csv/`head -q -n 1 "${ifn}"`".csv"
        head -q -n 1 "${hfn}" > "${ofn}"

        # get and write the non-headers lines
        nhfns=`cat "${ifn}" | sed -r 's,^(.+)$,plasmid_csv/\1.csv,' |
tr '\n' ' '`
        tail -q -n +2 ${nhfns} >> "${ofn}"

done < <(ls -1 groups/*.list)
```

sed Note

sed must be GNU (https://www.gnu.org) sed. -r does not enable extended regular expression syntax with BSD (http://www.bsd.org) sed.

Step 23. Create Group Matches from Plasmid Matches

Note that this step is not technically necessary to generate the desired output (the group CSV files (step #13) and the group statistics files (step #15)). This is really for convenience in inspecting results.

Input: The inputs required are the group list files that contain the plasmids in each group (see step #4) and the individual plasmid matches (see step #11). The group list files are assumed to be in the directory `groups` and named after the pattern `${GROUP}.list`. The plasmid matches are assumed to be in the `plasmid_matches` directory and named after the pattern `${ACCESSION}_matches.tsv`.

Output: One text file containing the matches for the group. We assume the output file will be in the directory `group_matches` and will be named after the following pattern: `${GROUP}_matches.tsv`.

Code:

Bash Command

```
while read ifn
do
     GROUP=`basename "${ifn}" ".list"`
     ofn="group_matches/${GROUP}_matches.tsv"

     fns=`cat "${ifn}" \
          | sed -r 's,^(.+)$,plasmid_matches/\1_matches.tsv,' \
          | tr '\n' ' '`
     head -q -n 1 ${fns} | head -n 1 > "${ofn}"
     tail -q -n +2 ${fns} >> "${ofn}"

done < <(ls -1 groups/*.list)
```

sed Note

sed must be GNU (https://www.gnu.org) sed. -r does not enable extended regular expression syntax with BSD (http://www.bsd.org) sed.

Step 24. Calculate Group Statistics from Group CSV

Input: This Python program requires 2 inputs. 1- The CSV file for a group. Here, we show the CSV files in the directory `group_csv`, named after the pattern `${GROUP}.csv`. 2- The output statistics file for the group. Here, we show the statistics files in the directory `group_stats`, named after the pattern `${GROUP}.stats`.

Output: One text file named as described in position 2 of the input to the Python program. That file is formatted as follows:

```
GROUP_NAME
==========
Total # of Plasmids: ##

Incompatibility Groups Structure:
  Inc.          Plasmid   Size          Size
  Group         Count     Mean          St. Dev.
  IncGrp1       #         #.###         #.###
  IncGrp2       #         ######.###    #####.###
  .
  .
  .
  IncGrpN       #         #####.###     ####.###

Plasmid Lengths Summary:
       Min: ####
       Max: ######
    Median: #####
      Mean: ######.###
  St. Dev.: ######.###

Key Words Structure:
  Key                       Plasmid   Size          Size
  Word                      Count     Mean          St. Dev.
  anti_microb_resist        ##        ######.###    ######.###
  anti_microb_resist_not    #         ######.###    ######
  beta_lact                 ##        ######.###    ######.###
  beta_lact_not             #         ######.###    ######
  plasmid_transfer          ##        ######.###    ######.###
  plasmid_transfer_not      #         #####.###     #####.###
  toxin                     ##        ######.###    #####.###
  toxin_not                 ##        #####.###     ######.###
  dna_maint                 ##        ######.###    ######.###
  dna_maint_not             #         ######.###    ######
  mob_gen_elem              ##        ######.###    ######.###
  mob_gen_elem_not          #         ######.###    ######.###
  hypo_genes                ##        ######.###    ######.###
  hypo_genes_not            #         ######.###    ######
  other                     ##        ######.###    ######.###
  other_not                 #         ######.###    ######.###

Plasmid Structure:
  This information is already reported in the CSV file: GROUP_NAME.csv
```

```
Sequencing Technologies:
  Sequencing       Num        Occurances per   Percent Total   Percent
Known
  Technology       Plasmids   Plasmid          Plasmids        Plasmids
  Known            ##         NA               ##.###          ###.###
  Unknown          ##         NA               ##.###          #.###
  Illumina         ##         #.###            ##.###          ##.###
  Roche ###        #          #.###            #.###           ##.###
  ABI Solid        #          #.###            #.###           #.###
  Sanger           #          #.###            #.###           #.###
  Ion Torrent      #          #.###            #.###           #.###
  PacBio           #          #.###            ##.###          ##.###
  ONT              #          #.###            #.###           #.###
  Short            ##         #.###            ##.###          ##.###
  Long             #          #.###            ##.###          ##.###
  Multiple Short   #          #.###            #.###           #.###
  Multiple Long    #          #.###            #.#             #.#
  Short Only       ##         #.###            ##.###          ##.###
  Long Only        #          #.###            #.###           #.###
  Short & Long     #          #.###            #.###           ##.###

Identical Plasmids Summary:
                            Plasmids (GROUP_NAME): ##
                               Discrete Plasmids: ##
          Indiscrete Plasmids (inside GROUP_NAME): ##
         Indiscrete Plasmids (outside GROUP_NAME): ##
                             Indiscrete Plasmids: ##
                  Groups of Indiscrete Plasmids: ##
                         Group Member Count Min: ##
                         Group Member Count Max: ##
                     Group Member Count Median: ##
                       Group Member Count Mean: ##.###
                   Group Member Count St. Dev.: ##.###

Identical Plasmids Groups:
       Discrete (GROUP_NAME):
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########

          Indiscrete Group #1:
                 ########  ########  ########
                 ########  ########  ########
                 ########  ########  ########
                 ########

           ...
           ...
           ...
```

```
          Indiscrete Group #n:
                 ########  ########
```

Code:

Bash Command

```
while read gfn
do
          GROUP=`basename "${gfn}" ".list"`

          ifn="group_csv/${GROUP}.csv"
          ofn="group_stats/${GROUP}.stats"

          python3 calcGroupCSVstats.py \
                  "${ifn}" \
                  "${ofn}"

done < <(ls -1 groups/*.list)
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (calcGroupCSVstats.py

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 25. Create Distance Matrix

Input: This script acts on all the files directly (i.e., not calling on a subroutine in Python or AWK for each of the accession numbers). It requires no user input directly as it ascertains the plasmid accession numbers from file names. It also relies on the directory structure to find the files named after the pattern `${ACCESSION}_identicalPlasmids_concordant.list` in a directory called `plasmid_blast_results`.

Output: One file per each accession. Each file is effectively a single row in the distance matrix. Once they are all created, they are combined into an additional file, the full distance matrix. The distance matrix is a full matrix (not only the bottom or upper halves); it is a csv file. The format looks like this:

Accession	A	B	C	D
A	0	x	y	a
B	x	0	i	j
C	y	i	0	k
D	a	j	k	0

Code:

Bash Command

This script is too long to reasonably represent in this document. Please view it in the freely-accessible online repository.

Distance Metric Definition and Examples:

Definition

The distance metric is the sum of the query and subject covered bases divided by the sum of the query and subject sequences. A covered base is defined as a base covered by (i.e., included in) the alignment. Given that d is the distance between a query and subject sequence, c is the coverage (i.e., bases included in the alignment) from a given sequence, and l is the length of a given sequence, the distance metric can be expressed in equation notation:

$$d = \frac{c_{query} + c_{subject}}{l_{query} + l_{subject}}$$

This metric is calculated for each pair of query and subject sequences; in other words, it is calculated in an all-vs-all fashion between the sequence for each plasmid.

Example

Given a query sequence that is 10 bases long and a subject sequence that is 20 bases long, consider an alignment that has a length of 6 bases and looks like this:

```
Query:      AAAAACGGGG
Subject:        A-GGGGTTTTTGGGGGCCCCC
```

The length of the alignment is 6. The number of covered bases (i.e., bases in the alignment) from the query sequence is 6. For the subject sequence, the number is 5. The distance can be found using the equation:

$$d = \frac{c_{query} + c_{subject}}{l_{query} + l_{subject}} = \frac{6 + 5}{10 + 20} = \frac{11}{30} \cong 0.367$$

This example is for a single pairwise comparison and would need to be repeated for every pair of plasmids.

Step 26. Create Distance Tree

Input: The input is the distance matrix from the previous step. We assume it is called `dist_matrix.csv` in the `tree` directory.

Output: One text file called `dist_tree.newick` in the `tree` directory. It is in the Newick tree format.

Code:

Bash Command

```
makeNewick.py \
          -i "tree/dist_matrix.csv" \
          -o "tree/dist_tree.newick"
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (makeNewick.py)

This script is not part of this package. It must be downloaded and installed separately. The only substantive requirement is Python 3.5+. makeNewick.py comes from a software package called CAM - **C**odon **A**version **M**otifs for Alignment-free Phylogenies (Miller et al. 2019). CAM is freely-available on GitHub at https://github.com/ridgelab/cam.

Step 27. Add Leaf Labels to Tree

Input: The input is the distance tree in Newick format from the previous step. We assume it is called `dist_tree.newick` in the `tree` directory. It also requires the location of source information (e.g., the country of origin of the plasmid) file and the name of the output file.

Output: This step appends additional information to the accession numbers that are the leaf labels in the tree. It creates a new tree, also in Newick format. We assume the output tree is in the `tree` directory and is called `dist_tree_labels.newick`.

Code:

Bash Command

```
python3 modifyLeafLabels.py \
         "plasmid_sourceInfo/sourceInfo.tsv" \
         "tree/dist_tree.newick" \
         "tree/dist_tree_labels.newick"
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (modifyLeafLabels.py)

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

Step 28. Add Color to Leaf Labels

Input: The input is the labeled distance tree in Newick format from the previous step. We assume it is called `dist_tree_label.newick` in the `tree` directory. Additional input is a colors mapping file. We assume it is called `colors.tsv`. The format is assumed to be one entry per line, where each entry has one column for the group and another column for the hex color (without the # symbol). The file we used is as follows:

```
IMP   FF0000
KPC   0000FF
NDM   00B600
VIM   000000
```

As you can see, we were looking for four groups for this tree figure in our analysis: IMP, KPC, NDM, and VIM. The final inputs required are a list of associated accession numbers for each group. We assume the files are named after the pattern `${GROUP}.list` in the directory `groups/keep`.

Output: This step includes the Newick-formatted tree from the input in a new Nexus file. It relies on a taxa block to specify colors for the leaf labels. As an example, the leaf label will have the label (e.g., `\t'some label here'`) followed by the color specification (e.g., `[&!color=#6789AB]`). This Nexus file will be available for directly opening with FigTree (https://github.com/rambaut/figtree) and, presumably, by other tree viewing/editing software. We assume the output tree is in the `tree` directory and is called `dist_tree_labels_colors.nexus`.

Code:

Bash Command

```
python3 convertNewick2NexusAndAddColor.py \
         "tree/dist_tree_labels_colors.nexus" \
         "colors.tsv" \
         "tree/dist_tree_labels.newick" \
         "groups/keep/IMP.list" \
         "groups/keep/KPC.list" \
         "groups/keep/NDM.list" \
         "groups/keep/VIM.list"
```

Python Version

Python 3.6.4 (https://www.python.org).

Python Script (convertNewick2NexusAndAddColor.py)

This script has at least one line that is too long to represent in this document without sacrificing readability. Please view it in the freely-accessible online repository.

A comment on data availability

The version of the PlasmidFinder database that we downloaded is no longer available. Accordingly, we release the fasta file we downloaded for reproducibility purposes. However, we advise a fresh download for any new experiments. This file may be found in the repository at the following path: `data/original_incompatibility_groups/incompatibility.fasta`. Similarly, many GenBank files have been updated since our download on 1 March 2018. We likewise release the versions we downloaded here for reproducibility purposes. However, we recommend fresh downloads of these files for new analyses. A script (labelled as "Step 0") is released with the online code repository for such a purpose. Please note that additional plasmids could now (and should) be included if the Entrez search strategy were to be re-done. The script would not reflect such changes as it downloads the specific GenBank groupings we used via accession numbers, completely ignoring the Entrez strategy. This is appropriate for reproducing our results, but it would probably not be ideal for a future study.

SUPPLEMENTAL REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic Local Alignment Search Tool. Journal of Molecular Biology **215**: 403-410. doi:10.1016/S0022-2836(05)80360-2.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T.L. 2009. BLAST+: architecture and applications. BMC Bioinformatics **10**: 421. doi:Artn 421. doi:10.1186/1471-2105-10-421.

Carattoli, A., Zankari, E., Garcia-Fernandez, A., Voldby Larsen, M., Lund, O., Villa, L., Moller Aarestrup, F., and Hasman, H. 2014. In silico detection and typing of plasmids using PlasmidFinder and plasmid multilocus sequence typing. Antimicrob Agents Chemother **58**(7): 3895-3903. doi:10.1128/aac.02412-14.

Miller, J.B., McKinnon, L.M., Whiting, M.F., and Ridge, P.G. 2019. CAM: an alignment-free method to recover phylogenies using codon aversion motifs. PeerJ **7**: e6984. doi:10.7717/peerj.6984.

APPENDIX 8

Chapter 7 – File S2

SUPPLEMENTARY TABLES

========== REMAINDER OF THIS PAGE INTENTIONALLY LEFT BLANK ==========

Supplementary Table 1. Full Dataset. This dataset is available online at the journal website as a spreadsheet. It is too wide to display meaningfully in this document.

Supplementary Table 2. Percent of plasmids belonging to each incompatibility group. Note: IncHI2 and IncHI2A were always found together, IncY replicon was only found in conjunction with other replicons.

Percent of plasmids	Inc Group	Percent of plasmids	Inc Group
0.22%	IncA/C	0.67%	IncN3
10.08%	IncA/C2	0.22%	IncP1
0.22%	IncB/O/K/Z	0.90%	IncP6
0.67%	Col	0.67%	IncQ1
1.12%	Col440I	0.22%	IncQ2
1.12%	ColRNAI	3.81%	IncR
2.02%	IncFIA	2.24%	repA
8.74%	IncFIB	1.12%	IncU
13.00%	IncFII	12.11%	IncX3
0.22%	IncHI1B	0.22%	IncX4
0.45%	IncHI2,HI2A	0.90%	IncX5
0.22%	IncI1	0.67%	IncX6
0.90%	IncI2	0.00%	IncY
2.47%	IncL/M	13.90%	Multi-replicon
12.56%	IncN	7.62%	NA
0.67%	IncN2		

Supplementary Table 3. Relative abundance of incompatibility groups among carbapenemase-carrying plasmids. Note: IncHI2 and IncHI2A were always found together, IncY replicon was only found in conjunction with other replicons.

Carbapenemase Family	Incompatibility Groups (Percent of plasmids)										
	IncA/C	IncA/C2	IncB/O/K/Z	IncCol	IncCol440I	IncColRNAI	IncFIA	IncFIB	IncFII	IncHI1B	IncHI2/HI2A
KPC	0.00%	3.10%	0.00%	0.00%	2.00%	2.60%	1.50%	15.80%	8.20%	0.00%	0.00%
NDM	0.00%	15.10%	0.60%	0.00%	0.00%	0.00%	3.60%	4.20%	25.30%	0.60%	0.00%
IMP	0.00%	22.40%	0.00%	0.00%	0.00%	0.00%	0.00%	2.00%	2.00%	0.00%	4.10%
VIM	3.40%	16.10%	0.00%	9.70%	3.20%	0.00%	0.00%	6.50%	3.20%	0.00%	0.00%

	IncI1	IncI2	IncL/M	IncN	IncN2	IncN3	IncP1	IncP6	IncQ1	IncQ2	IncR
KPC	0.00%	2.00%	2.60%	15.80%	0.00%	1.00%	0.50%	1.50%	1.50%	0.50%	5.60%
NDM	0.00%	0.00%	1.20%	3.00%	1.80%	0.00%	0.00%	0.00%	0.00%	0.00%	1.80%
IMP	2.00%	0.00%	8.20%	32.70%	0.00%	2.00%	0.00%	0.00%	0.00%	0.00%	0.00%
VIM	0.00%	0.00%	0.00%	13.80%	0.00%	0.00%	0.00%	3.40%	0.00%	0.00%	10.30%

	repA	IncU	IncX3	IncX4	IncX5	IncX6	IncY	Multi-replicon	NA
KPC	5.10%	1.00%	3.50%	0.00%	1.50%	1.50%	0.00%	17.30%	5.60%
NDM	0.00%	0.00%	28.30%	0.60%	0.00%	0.00%	0.00%	11.40%	2.40%
IMP	0.00%	6.10%	0.00%	0.00%	2.00%	0.00%	0.00%	6.10%	16.30%
VIM	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	37.90%

APPENDIX 9

Chapter 7 – File S3

SUPPLEMENTARY FIGURES

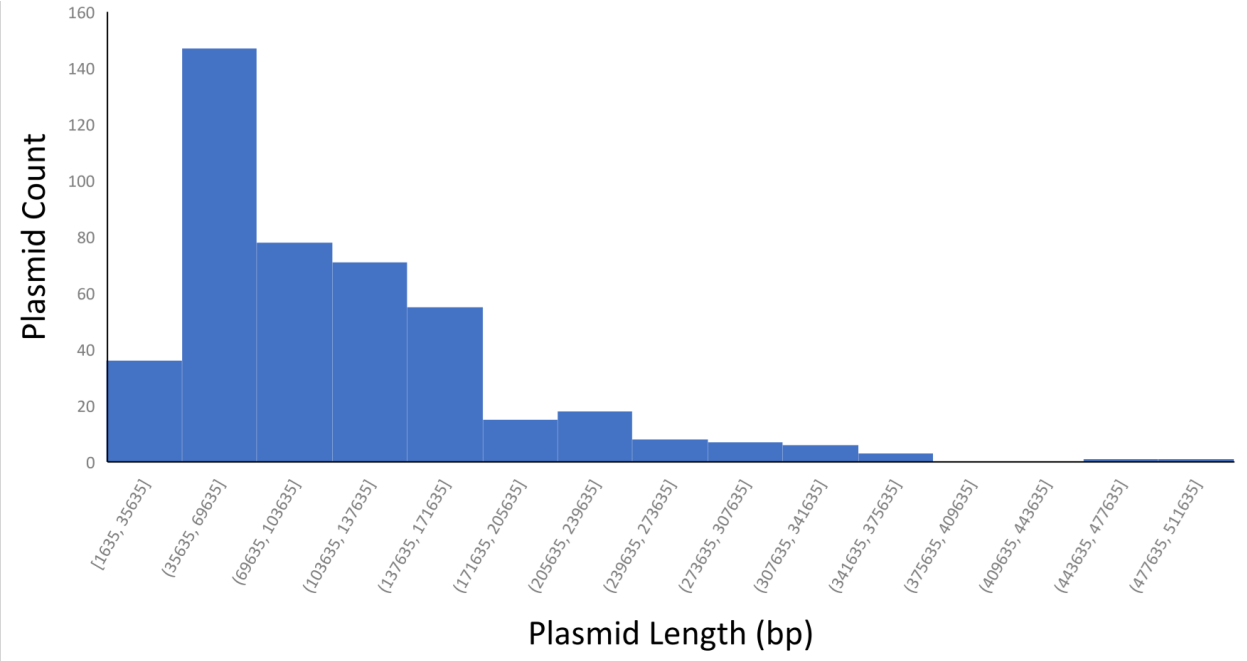


Figure S1. Distribution of length for all 446 plasmid sequences in this study.

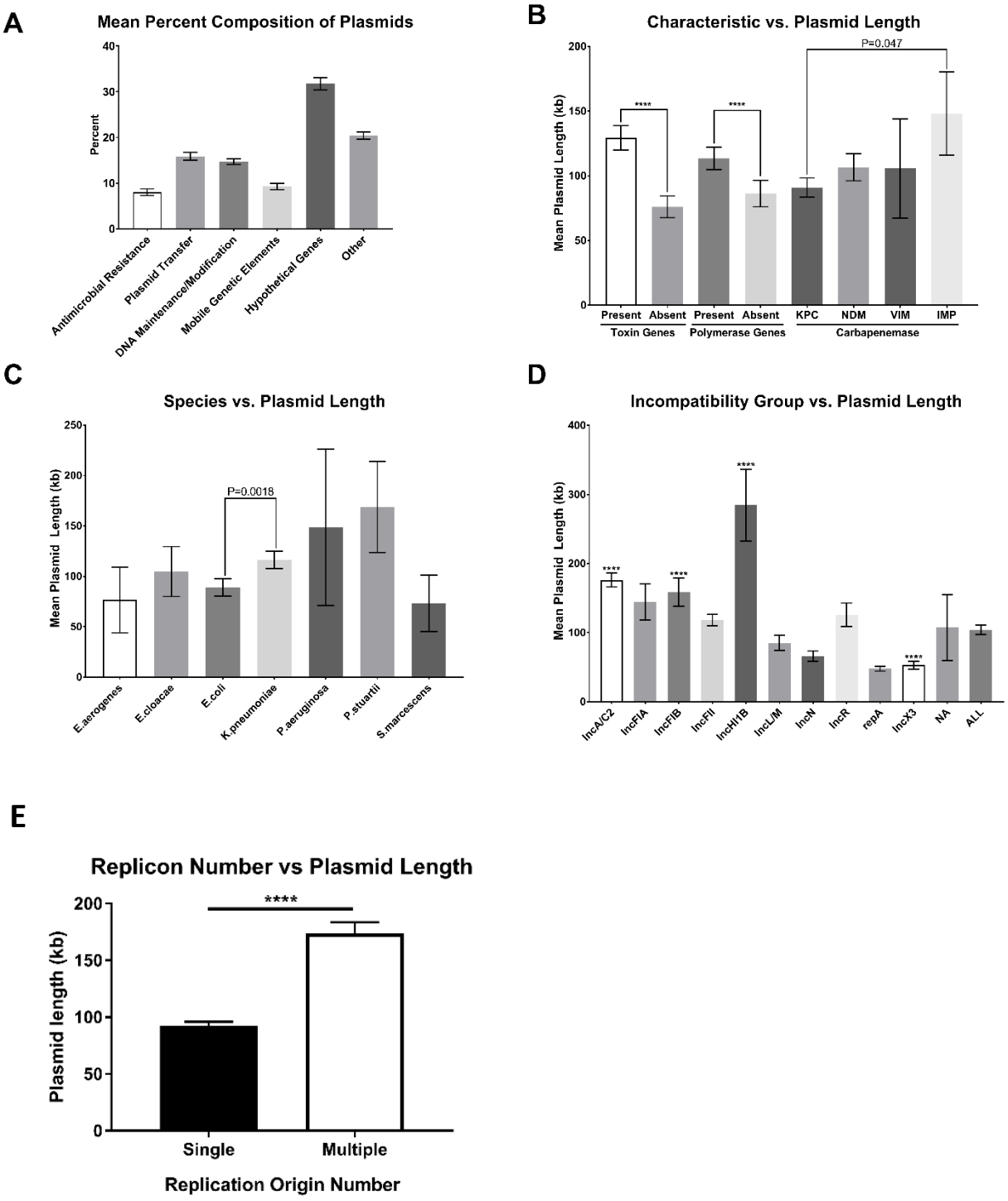


Figure S2. Various characteristics of carbapenemase carrying plasmids. A) Average gene content of plasmids by gene ontology. **B)** Average length of plasmids by characteristic of interest (Presence or absence of toxin-antitoxin system, polymerase genes, and carbapenemase carried. **C)** Determination of species on plasmid length. **D)** Determination of replicon type on plasmid length. **E)** Determination of multi-replicon content on plasmid length.