Comput Econ (2007) 30:19–40
DOI 10.1007/s10614-007-9084-4

Reproducible research in computational economics: guidelines, integrated approaches, and open source software

Giovanni Baiocchi

Received: 26 December 2005 / Accepted: 31 January 2007 / Published online: 22 March 2007

Abstract Traditionally, computer and software applications have been used by economists to off-load otherwise complex or tedious tasks onto technology, freeing up time and intellect to address other, intellectually more rewarding, aspects of research. On the negative side, this increasing dependence on computers has resulted in research that has become increasingly difficult to replicate. In this paper, we propose some basic standards to improve the production and reporting of computational results in economics for the purpose of accuracy and reproducibility. In particular, we make recommendations on four aspects of the process: computational practice, published reporting, supporting documentation, and visualization. Also, we reflect on current developments in the practice of computing and visualization, such as integrated dynamic electronic documents, distributed computing systems, open source software, and their potential usefulness in making computational and empirical research in economics more easily reproducible.

Keywords Economic methodology · Econometric software · Other computer software

JEL classification B4 · C87 · C88

1 Introduction

Econometrics and other traditionally empirically oriented economic models such as input–output analysis are inherently computational. More recently, the

G. Baiocchi (✉)
Department of Economics and Finance, University of Durham, Durham DH1 3HY, UK
e-mail: giovanni.baiocchi@durham.ac.uk

use of ever more powerful computers and the development of increasingly sophisticated software applications has allowed economists to explore economic models with less restrictive assumptions, estimate and test richer behavioral models, experiment with different complex methodologies, compare different estimation methods, etc. All these approaches have become part of the cross-disciplinary subject we now refer to as *computational economics*. In general terms, the goal of computational economics is to advance the subjects of economics, mainly through the analysis of mathematical economic models by the application of advanced computing techniques. To appreciate the wide range of economic issues were computational methods have been brought to bear, one just needs to glance at the table of content of issues of this Journal, the *Journal of Applied Econometrics (JAE), Journal of Economic Dynamics and Control*, or at the papers collected in books, such as, Varian (1996), Amman, Kendrick, and Rust (1996), and Judd and Tesfatsion (2006). Many of these applications rely rather heavily on computing.[1] Increasingly often, economists use computers, not only for computations on data and model simulations, but also for simple, mechanical operations such as searching for information, collecting and storing data, changing the format of data, validating data, post-processing output from statistical applications, writing reports, handling tedious and complex algebraic manipulations, collaborating with other researchers, and in disseminating the final results. The use of computers has also benefited learning and research by suggesting conjectures and enriching our understanding of abstract economic and econometric concepts by means of examples and visualizations.

On the negative side, this increasing dependence of economists on computers has resulted in research that has become increasingly difficult to replicate. Implementation details of computations in economics are left out of traditional printed publication. Reproducibility relies on a plethora of implementation details that are difficult to communicate through conventional printed publications. As Dewald, Thursby, and Anderson (1986) pointed out, this lack of information can result in months of effort by researchers trying to replicate a study yielding inconclusive results regarding the validity of the original study. Program written in specialized languages such as GAUSS are often not easily portable between different platforms and versions of the program. Programs written in conventional programming languages such as FORTRAN or C++ also depend on implementation details including the vendor, version of the compiler, and the specific platform on which they run. All these factors can amount to insurmountable obstacles in the replication of computational-based results in economics, as extensively reported by Dewald et al. (1986). For instance, Dewald et al. (1986) report that they had to abandon attempts to reproduce

[1] We could say that these methods are computationally intensive, however this expression is rather fluid as yesterday's computationally intensive methods become today's standard approaches. As an example, Leontief (1966) recounts that in 1939 to solve a system of 42 equations in 42 unknown, in what was the first effort to analyze a large economics model through computers, required several months of programming and 56 hours of computing on the Harvard Mark II computer, one of the most powerful computers available at the time. Today the same calculations can be done in a fraction of a second on a standard PC after comparatively very little programming effort.

results from a large macroeconometric model because of difficulties in transferring programs and data across computer systems. McCullough and Renfro (1999), in a survey of GARCH estimation procedure implemented in various packages, found that often important information that affects computed results, such as parameter initialization, was not available. Buckheit and Donoho (1995) pointed out that in the field of computational experiments researchers often cannot reproduce their own work, even only a few months after its completion, that research students have difficulties in presenting their problems to their academic advisers, and that researchers cannot reproduce computational results of other researchers and other published work. There is substantial evidence that analogous problems occur also in economics (see, e.g., Dewald et al., 1986).

Computational and empirical results in economics require independent verification in order to contribute to the advancement of the subject of economics. An important step in that direction is that published computational results should be reproducible by other researchers. Ideally, reproducibility implies that identical computational results should be obtainable in a short amount of time, without requiring expensive computational resources, proprietary data, licensed software, and any application-specific knowledge. Of course, insisting on "bit-by-bit" reproducibility of computational results in economics is not always practical and the definition must be interpreted in the light of the specific context of application.[2] In applied work, it is quite frequent that a particular commercial software, dataset, or expensive equipment makes research results difficult to reproduce.[3] In practice, obtaining qualitatively similar results might be sufficient to claim that a computational result has been reproduced.

There has been an increasing interest in making research in empirical economics reproducible since the alarm raised by Dewald et al. (1986), in their Journal of Money, Credit, and Banking (JMCB) project, in which they attempted and failed to replicate most empirical results published or submitted to the same journal. Based on their recommendation several journal introduced publicly available Internet archives and required the submission of data and programs from the authors of the empirical papers submitted. In a more recent investigation, Vinod (2001) found that ~70 per cent of articles from prestigious economic journals were not reproducible. He attributed this problem to sloppy record keeping, inaccurate software, and the lack of maintenance of software and data, in particular, after publication. McCullough, McGeary and Harrison (2006) take stock of 23 years experience of the JMCB data and code archive. They convincingly argue that, though most empirical work could still not be reproduced, the requirement of a data and code archive should be adopted by more journals and that stricter rules that ensure compliance from the

[2] Gentle (2003) talks of Monte Carlo computation being *strictly* reproducible if the software and the seeds used for the random number generators are preserved.

[3] Stokes (2004) discusses the potential advantages of using different software to solve the same problem.

author should be introduced. Based on the experience, they provide guidelines to facilitated the reproduction of empirical research in economics.

If progress in making empirical research reproducible has been slow, progress computational computational economics has been even slower. Of the three journals that specialize in computational economics and econometrics mentioned above only the JAE has a mandatory archive, but requires the submission of the data only. The Journal of Economic Dynamics and Control upon acceptance of a paper offers the authors the option to submit data and computer code used in the paper. This implies that in even in principle most computational results cannot be verified. This state of affairs is regrettable and every effort should be made to correct it. Besides allowing for verification, the reproducibility of computational economics research would have many other important benefits. Research in economics is often a process of iterative refinement. Reproducible computational results are easier to improve upon. Supervision can also benefit from reproducibility at least in two ways. First, by using computational results that are easily reproducible and modifiable to solve other economic problems, research students can learn and get started with their own research. Second, reproducible results can be better monitored for quality.[4]

In this paper, we propose some basic standards to improve the production and reporting of computational results in economics for the purpose of accuracy and reproducibility. In particular, we will make recommendations on four aspects of the process: computational practice, published reporting, supporting documentation, and visualization. Also, we reflect on current developments in the practice of computing and visualization, such as integrated dynamic electronic documents, distributed computing systems, open source software, and their potential usefulness in making computational and empirical research in economics more easily reproducible. In particular, we will highlight the potential role of these innovative computational developments in organizing computational research and in mediating researcher's interaction with each other, Ph.D. students, and journal editors, by streamlining operations such as replication, validation, and supporting students' participation in the research process. We also show how open source software, software whose source code is made freely available to the public, enabling anyone to copy, modify and redistribute the source, is naturally conducive to reproducibility. In particular, we illustrate how scripts using only open source applications, can be used to collect commands that can reproduce the computational results of a project in its entirety from data collection and preparation to analysis and dissemination.

In Sect. 2, we introduce guidelines to improve the reproducibility of computational results in economics. In Sect. 3, we consider several innovative approaches that have been recently proposed to preserve all steps of a computation in an electronic medium and their potential application to computational economics in order to facilitate reproducibility. In Sect. 4, we introduce open source software and its uses in economics. Section 5 discusses the advantages and disadvantaged of open source software. In Sect. 6, we illustrate how open

[4] For an extensive review of the benefits of reproducibility see McCullough et al. (2006).

source software can be used to facilitate the reproduction of computational and empirical results in Economics. In particular, we illustrate how scripts using only open source applications, can be used to collect commands that can reproduce the computational results of a project in its entirety from data collection and preparation to analysis and dissemination. Section 7 concludes.

2 Guidelines for reproducibility in computational economics

In this section, we propose some basic standards to improve the production and reporting of computational results in economics for the purpose of accuracy and reproducibility. Results based on computational methods in economics should be reported as carefully as any other computation results from other disciplines. McCullough et al. (2006) provided guidelines for reporting reproducible research in empirical economics. In the field of statistics, Hoaglin and Andrews (1975) supplied a useful list of items that should accompany any statistical computation-based result. No equivalent guidelines are available for obtaining and reporting computational results in economics. To be of use to economists, it is important that guidelines are tailored to the specific problems of computational economics and that they take into account the recent experience in making empirical research in economics and other disciplines reproducible (see, e.g., Anderson, Greene, McCullough and Vinod, 2005, McCullough, McGeary and Harrison, 2006). However, given the wide scope of computational economics, recommendation must be interpreted in the light of the specific context of application. In this section, we make recommendations on four aspects of the process we think require special attention: computational practice, published reporting, supporting documentation, and visualization.

2.1 Computational practice

Computational practice in economics should conform to best practice adopted in other scientific disciplines relying heavily on computation to further the development of their respective fields. The algorithms used in computational economic should be state of the art and fully adequate for the needs of the study. It is important to avoid reinventing the wheel when writing software code. Whenever possible, preference should be given to well-known and thoroughly tested algorithms available in the public domain as subroutines code, libraries, packages, software applications, and systems. Computational routines that are not in the public domain or that have not being tested before, should be thoroughly tested before use.

It is often the case that computational economists have to craft their own code. In fact, the algorithm and its implementation often constitute the main contribution of a computational economic paper. To facilitate reproducibility, code should be easy to read and to maintain. As an example, Kernighan and Plauger (1978) and Kernighan and Pike (1999) provided basic elements of programming style including a guide for variable names and commenting styles.

2.2 Published reporting

Theoretical results in economics often rely on deep properties of the real number system. Computer numbers, referred to as "floating-point numbers" and often denoted by $\mathbb{F}$, are a finite subset of the reals, $\mathbb{R}$. Operations in $\mathbb{F}$ do not follow the familiar properties of the corresponding operations in $\mathbb{R}$ (see, e.g., Ueberhuber, 1997).[5] Computer numbers with their associated arithmetic are merely a convenient approximation of the real number system. Economists engaging heavily in computation hope that, by accounting for more complex relationships into their models, they can reduce the inevitable gap between economic models and reality. By doing so however, they introduce errors of a different nature. Computational results can be at best approximations to the desired problem solution. More often, computational results are the solution of a qualitatively different problem from the original. Because of this, every effort should be made to reassure the reader that best practice has been applied in obtaining the results. In principle, any information useful to assess the accuracy of the results, to help with their interpretation, and to facilitate their reproduction, should be supplied with the study. In particular, any published result should include details on:

- the estimated accuracy of the computed results,
- the extent of agreement with known theoretical results and certified benchmarks, including, whenever possible, an analysis of the discrepancies,
- details on any measure employed to speed up computations such as discretization, truncation, convergence criterion, etc.,[6]
- the numerical algorithms used (rootfinding, minimization, etc.), which should be fully adequate for the needs of the study, convergence properties, error bounds, starting values selection strategy, etc.,
- robustness checks, whenever possible, with respect to the choice of alternative computational algorithms,
- the programming languages or software applications used, and vendor, version, serial number, alternative platforms on which they run, etc.,
- the computer used, including details on the CPU, and operating system.[7]

All the items listed above provide information to help assess the accuracy of the computer-based results and facilitate their reproduction.

[5] Lack of basic knowledge of computer numbers and algebra can lead to a considerable waste of time and embarrassment. I was once asked by a colleague why, in the calculation a numerical derivative of a function, $\frac{f(x+h)-f(x)}{h}$, the accuracy seemed to decrease with decreasing h. On another occasion, I was told that in order to achieve very accurate computational results, the convergence criteria of a numerical algorithm for a computational economics study was set to less than 10^{-308}.

[6] Often these measures convert a computationally intractable problem into a tractable one (see, e.g., Judd and Tesfatsion, 2006).

[7] It is worth remembering that in the fall of 1994, a serious design flaw was discovered in the Intel Pentium processor, commonly referred to as the "Pentium floating-point-division bug" or "Pentium bug" in short. As a consequence, certain floating-point division operations performed by the Pentium processor produced incorrect results.

2.3 Code, data, and other supporting documentation

Authors should submit code and data plus any other supporting documentation that can facilitate replication and help in the assessment of the quality of the results. Data and code archives have been used with limited success in empirical economics so far. McCullough et al. (2006), in their review of the JMCB data and code archive, found that even when code and data were provided, it proved often difficult or impossible to replicate empirical results. For instance, they found that in many cases data was provided in a different format from the one required as an input to the code supplied, data or subroutines files were missing, the code was poorly organized, and so on. Among other things, they recommend the use of data dictionary, were variable and data sources are described, and the use of readme files listing all the files needed for replication with a brief description of each. For the reproduction of computational econometrics results Koenker (2006) proposes that all the code and software environment together with relevant documentation necessary to reproduce tables and figures should be made available through the Internet. We propose that authors of computational studies should provide the following items:

- any data used in the study in its original form and in the form required by the computational code,
- all code used to produce the computational results presented in tables and figures,
- output from a computer run that makes use of the code and data supplied and contains the computational results included in the paper, and
- a *readme file* list all the replication files with a brief description of each.

The code should written following basic programming style guidelines. In particular, it should be easy to read, modular, and adequately documented. Code and data in its original form should consist of textual (ASCII) data which can be readily transferred from one system to another. The code should be well organized and easy to run, all files should be listed and documented. We recommend that a data dictionary be included in the replication files, as part of a readme file. We recommend that the readme file list all the replication files with a brief description of each.

2.4 Visualization

Traditionally, results in empirical economics are presented in the form of tables. The advantage of tables is that information can be clearly organized and they show exact numerical values. However, tables can only be practically used only when the computational results can be represented or summarized as a small finite set of numbers. Often, to manage large numbers of results resulting from changes in experimental conditions, response surfaces. More often, computational results can be communicated accurately and clearly only by means of graphs. Because of the nature of the computed results visualization has becomes an essential part of computational economics. However no attention

has been given to best practices in the visualization of computational results. Visualization should display data accurately and clearly, and should help to highlight important characteristics. A well designed graph should be able to facilitate exploration, communication, as well as calculation and processing of the computational results.

Some methods of visualization, such as kernel density estimation used to present monte carlo results in econometrics, are themselves computational methods and depend on a plethora of implementation details that can be built-in the software application, fixed as default settings, or determined by the researcher. Given the importance of visualization in computational economics, the same standard for obtaining the computational result should be applied to the production of figures.

Visualization methods can draw on the relevant literature in the fields of scientific visualization, psychology, and computer graphics. Several graphics parameter can affect the presentation of computational results. Excellent reference for guidelines on how to produce good quality graphs. In particular, Cleveland (1993, 1994) or Tufte (2001) should serve as a useful guides. In view of the above considerations, we feel that the computational results that are displayed graphically should be accompanied by detailed information on any interpolating, smoothing, or other algorithm used for the display of the results.

3 Integrated approaches to reproducibility of computations

Traditionally, the process of performing economic computations is being done separately from the preparation of documents and reports in which they are presented. The results from economic computation are often inserted into Word or LaTeX documents manually and in ad hoc ways. Papers are then traditionally presented in a printed format. The documents are not easily reproducible. Printed papers are static and "closed" in that the reader cannot update the contents of the document directly or by introducing new or additional data. Even electronic documents, mostly in pdf format, disseminated through the internet, are qualitatively no different from the printed version.

Several approaches have been recently proposed to preserve all steps of a computation in an electronic medium in order to facilitate reproducibility. Two main approaches have emerged recently. Scripting Languages and dynamic interactive documents

3.1 Scripting languages

The use of scripting languages,[8] that can connect diverse software tools and applications to accomplish a sequence of computations and data processing, has been suggested in the computing literature. Schwab, Karrenbach and Claerbout

[8] Also known, especially in the past, as batch languages, glue languages, or job control languages.

(2000) of the Stanford Exploration Project (SEP) group,[9] developed the concept of reproducible electronic document based on the GNU *make* utility. Programs written for the make utility are known as *makefiles*. A makefile contains all the commands needed to build a software application and is a standard UNIX utility for software maintenance. In Sect. 6, we show how a complete computational project with all the data and commands needed for its reproduction can be compactly preserved and effectively communicated through a Perl script.[10]

3.2 Dynamic documents and environments

Recent advancements in technology and computer science have permitted the development of software tools that can integrate computational processes into both the document preparation and display process (see, e.g., Gentleman, 2005; Gentleman & Lang, 2004; Leisch, 2002; Sawitzki, 1999; Sawitzki, 2005). These electronic documents are *dynamic*, in that computations presented as figures and tables, can be recalculated each time the document is viewed, and *interactive*, in that computations included in the document can be manipulated and controlled directly by the reader. The limitations of traditional printed paper as a support for computational results, are self-evident. Dynamic, interactive electronic documents can be implemented either through a software component approach or through rich interfaced programming environments.[11]

A software component approach integrates regular text documents with spreadsheets, pictures, digital videos, digital audio, and other features. Examples include dynamic web pages and word processing software. For instance, HTML allows the inclusion of Applets and other plug-ins to provide dynamic, interactive, facilities. Another popular example among economists of such an integrated environment is Scientific WorkPlace, which is a word processing system, designed especially for the preparation of technical documents, based on LaTeX with support for computer algebra systems.

Another approach makes use of rich interfaced programming environment. As an illustration, Gentleman (2004) developed the concept of *compendium*, a dynamic document, containing text, code, and software, capable of displaying and recreating computations on demand. The prototype used by Gentleman (2004) to exemplify the concept is based almost entirely on GNUR. This approach could provide reproducible calculations and a formal way for integrating economic computations directly into documents. The general approach is based on the concept *literate programming* originally proposed by Knuth (1983, 1984, 1992). These electronic documents are also interactive and extensible in the sense that they allow the reader to modify the processing options, input new data, or insert additional algorithms and visualizations. There are other more ad hoc suggestions on how to make research reproducible based on commercial

[9] The homepage of the group is located at `http://sep.stanford.edu/`.

[10] For more details see Baiocchi (2003, 2004).

[11] Although this distinction can be blurred in practice.

software applications. Buckheit and Donoho (1995) developed a reproducible research environment in the form of a Matlab library. Koenker (1996) proposed an approach to computational experimentation in econometrics based on the commercial application S-plus. Varian (1992, 1996) collects examples of dynamic electronic documents of computational economics implemented in the Mathematica programming environment.

Distributed computing, a generalization of the component approach, could allow a seamless sharing of data, code, and computational environments. Although the idea is not fully developed yet, it has a considerable appeal for the purpose of reproducibility of computational results.

4 Open source software in economics

Economists typically require a variety of data processing, communication, and other software tools. Software applications used by economists for research and teaching purposes include many proprietary econometric and statistical applications (e.g., PcGive, SAS, SPSS, EViews, TSP), symbolic processing applications (e.g., Maple and MATHEMATICA), and various simulation and optimization packages (e.g., GAMS). Commercial software applications cannot be freely copied and distributed making replication of computational-based results arduous for researchers without expensive computational resources at their disposal.

As more and more complex computational methods are being developed and used in economics, there is a growing concern about relying on commercial proprietary/closed software for academic research. Computational results that largely depend on a black box, such as Mathematica, whose details on algorithms implemented are trade secrets held by a commercial company, though replicable under the same conditions, are much harder to validate. Open source does not suffer from these liabilities, as, in principle, everything about an open source program is open to scrutiny. The closed nature of most commercial software affects also reproducibility. By keeping their software source code hidden, commercial software vendors, make it impractical to modify a software application they develop, and demand fees for its use and improvement. Users have to rely on the company to implement new features and fix bugs. There is a wide consensus among practitioner econometricians that commercial software producers are slow to respond to request of extensions and bug fixes by users. In his review of GAUSS, Vinod (2000) pointed out the failure of GAUSS to fix numerical accuracy problems that have been exposed by specialized literature. Problems discovered by Knüsel (1995) were still present, and more were found. Some still persist at the moment of writing as we will see in subsequent sections. Some of the consequences of software inaccuracies are presented in McCullough and Vinod (1999a).

Open source software, whose source code is made freely available to the public, enabling anyone to copy, modify and redistribute the source, is naturally conducive to reproducibility and verification. Table 1 presents the current

Table 1 Legal status of software applications useful to Economists reviewed by the JAE

Software useful for economists reviewed by the JAE				
Free/open source		Proprietary/closed		
Public domain	Netlib repository[a] (www.netlib.org)	Freeware[b]	BACC	**14**(6), 677–689
			EasyReg	**13** (2), 203–207
Open source			Scilab	**16**(4), 553–559
	GCC (GNU C++)		ViSta	**17**(4), 405–414
	11 (2), 199–202	Commercial	GAUSS	**15**(2), 211–220
	CYGWIN Tools		EViews	**15**(1), 107–110
	15 (3), 331–341		LIMDEP	**14** (2), 191–202
	GNU/Linux		MATLAB	**12** (6), 735–744
	14 (4), 443–452		MicroFit	**13** (1), 77–89
	GNU Octave		Ox [c]	**12** (1), 77–89
	15 (2000) (5), 531–542		PcGive	**13** (4), 411–420
	GNU R		RATS	**12** (2), 181–190
	14, (3), 319–329		Shazam	**14** (2), 191-102
	GRETL		SORITEC	**17** (1), 85–90
	18, (1), 105–110		S-plus	**12** (1), 77–89
	Perl		Stata	**16** (5), 637–646
	18, (3), 371–378		TSP	**12** (4), 445–453
	mikTEX/teTEX		XploRe	**13** (6), 673–679
	16, (1), 81–92		LISREL	**19** (1), 135–141
			Maple	**10**(3), 329–337

[a] Web site directing to mostly public domain Fortran and Java code for matrix computations including solving linear equations, eigenvalue problems and linear least-squares problems
[b] Software that can be downloaded free of charge
[c] A less "user friendly" non-Windows version is available with no charge for academic uses

legal status of software useful to economists reviewed by the JAE. The table also report the volume, issue, and page numbers were the review appears. An important distinction to keep in mind is the one between open source and freeware/shareware software. With open source software, the source code is bundled with the software and is free for everyone to inspect and acquire, with freeware/shareware the software is "free" to be distributed, but the source code is withheld from the public. Open source is made available under a variety of license types. The GNU General Public License (GPL), the GNU Lesser General Public License (LGPL), the Mozilla Public License (MPL), the BSD License, the Apache Software License, the MIT License, the Artistic License, and the Perl license are among the best known. For a an explanation of these different Open Source license flavors, consult St. Laurent (2004). The table clearly shows how most free software deemed useful for economists falls under the Open Source GPL agreement and includes a completely functional UNIX operating system (GNU Linux), programming languages and developing tools (GCC, and CYGWIN), powerful typesetting system (MiKTEX/TeTEX), and a high-level, cross-platform programming language with network and object-oriented programming support (Perl). Software applications that can compete with commercial applications traditionally used in Economics, include a high-level language

for matrix and numerical computations (Octave), which is comparable in terms of functionality to specialized applications such as GAUSS and MATLAB, a sophisticated programming environment for statistical computing and graphics based on the S programming language (GNU R), with functionality analogous to applications such as SAS, SPSS, STATA, or S-plus, and a complete econometric package (GRETL), still under development but already with features that makes it comparable to applications such as PcGive, EViews, and MicroFit.

GNU (sometimes pronounced "guh-NEW") is an acronym for "GNU's Not Unix". It is the name of a project by the Free Software Foundation (FSF) whose purpose is to promote the free exchange of software. The GNU project was started in order to develop a complete Unix-compatible operating system as well as an extensive set of software tools, all to be made freely available to the general public. The project has grown to include programs that were developed by many other people for their own purposes, which shared the same underlying philosophy of software freedom. For more details on the organization, see Stallman (1985). GNU's success as a catalyst in the production of free software is mostly attributable to the introduction of a form of software licensing, known as the GNU General Public License, or GPL, which encourages the free distribution of software.[12] In the next few paragraphs we briefly review some of the most successful OS project useful to economists.

GNU/Linux is a Unix-like computer operating system combined with libraries and tools from other GNU projects. Linux distributions incorporate large number of software applications with the core system. It was originally developed by Linus Torvalds for Intel microprocessors in 1991 but has since then considerably expanded to support a variety of computer architectures. A review of GNU/Linux from an economist's point of view can be found in MacKinnon (1999).

R,[13] an open-source programming environment for data analysis and graphics, has in only a decade grown to become a de-facto standard for statistical analysis against which many popular commercial programs may be measured. R's source code was initially written by Ross Ihaka and Robert Gentleman (see Ihaka and Gentleman, 1996) at the Department of Statistics of the University of Auckland in Auckland, New Zealand. Since the mid 1990s there has been a core group (the "R Core Team") who can modify the R source code archive. R provides cutting-edge statistical and visualization methods. For an introduction on how R can be used in Econometrics see, e.g., Racine and Hyndman (2002).

[12] The crucial difference between GNU software and software placed in the public domain, without copyright, is that the GNU GPL makes sure that anyone who redistributes the software, with or without changes, must pass along the freedom to make further copies and changes.

[13] R is available from the WWW's Comprehensive R Archive Network (CRAN) located at `http://cran.r-project.org/`, where source code, additional libraries, documentation, and links to binaries distributions of R are available for various platforms, including Win32, Mac, and Unix/Linux.

GRETL, an acronym for *GNU Regression, Econometrics and Time-series Library*,[14] is a cross-platform software package for econometric analysis, written in the C programming language. GRETL is the first complete econometric software package to be released under the GNU software license. The software consists of a shared library, a command-line client program, and a graphical client program. It comes with many sample data files from Greene (2000) and Ramanathan (2002), which are immediately accessible from the menu. It supports several least-squares-based statistical estimators (including two-stage least squares and panel data methods), time series models (including the Cochrane-Orcutt procedure and VARs), and some maximum likelihood methods (logit and probit). It also has built-in commands for several econometric tests (including the Chow, Hausman, and Dickey-Fuller tests). It calls gnuplot to generate graphs and is capable of generating output in LaTeX format. GRETL has been written by Allin Cottrell based on ESL (Econometrics Software Library) code written by Ramu Ramanathan of the University of California, San Diego. It can be obtained from the world wide web at `http://gretl.sourceforge.net/`, where the source package and binary distributions running on GNU/Linux and Microsoft Windows in the form of a self-extracting executable can be downloaded. Particularly noteworthy is the fact that the program is also distributed on CDs that accompany two popular econometrics textbooks, Ramanathan (2002) and Wooldridge (2002). These books use GRETL extensively for their applied examples making GRETL a useful tool for practicing and teaching econometrics. An example of how GRETL can be used to analyze economic data can be found in Baiocchi and Distaso (2003).

GNU Octave is a high-level matrix-based language, primarily intended for numerical computations, available for different platforms at the following URL: `http://www.octave.org/`, that is, mostly compatible with MATLAB. It provides a convenient command line interface for solving common numerical linear algebra problems, including the roots of non-linear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential equations. It may also be used as a batch-oriented language. It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages. Octave was originally written by James B. Rawlings of the University of Wisconsin-Madison and John G. Ekerdt of the University of Texas. Octave is free software distributed under the terms of the GNU GPL as published by the FSF. For a survey on how Octave can be used in economics see Eddelbuettel (2000).

GNU Emacs, a program written by Richard Stallman of the FSF, can serve as an integrated environment in which to run applications useful to economists. There is an Emacs package called ESS, an acronym for *Emacs Speaks Statistics*,

[14] There is also an obvious reference to the classic fairy tale "Hansel and Gretel," in which Gretel is the mature and resourceful girl whose ingenuity saves her sibling's life from an evil witch who, after kidnapping them by means of gingerbread and candies, intends to fatten and eventually eat him.

(see, Rossini et al., 2004) which provides a standard interface between statistical and econometric programs and statistical processes. It is intended to provide assistance for interactive statistical and econometrics programming and data analysis. Languages supported include: S dialects (S-Plus, and R), LispStat dialects (XLisp-Stat, ViSta), SAS, Stata, and SPSS dialect (SPSS, PSPP).

A complete computing environment that includes all the above mentioned applications and many more is Quantian Eddelbuettel (2003). Quantian is a Linux-based system that is a directly bootable and self-configuring from a single cdrom/dvdrom. Quantian comprises Knoppix (Knopper, 2003) from which it takes its base system software, along with automatic hardware detection and configuration, and scientific software such as the above mentioned applications and many more including, general purpose computer algebra systems such as Axiom, Maxima, PARI/GP, etc., numerical matrix oriented applications such as Scilab, Numeric Python, Euler, and PDL, optimization software such as lp-solve, GNU Scientific Library, programmable editors such as GNU Emacs with support for econometric and statistical applications including Stata, SAS, S-PLUS, and R, and so on.

One of the features that make many open source projects so successful is their modular nature. The functionality of modular application can be easily extended to cover more specialized areas of application. Modular application make it particularly easy to create, install, update, and access the optional code and data, with accompanying documentation, within the main application. Functions, data, and documentation provided by extra modules are easily made available to the user without the need of any application-specific knowledge typically with just one statement (`\usepackage{...}`, `library(...)`, `use ...`). This allows code written to satisfy the need of a particular researcher to be easily reused and modified by others. For instance, modules (in Perl), libraries (in R), macro packages (LaTeX) useful to economists are continuously added. Modules are made available in the main Web site were the software is distributed. So called *package managers* (for instance the MiKTeX Package Manager and the Perl Package Manager) allow the installation or update on demand of additional packages. Other applications, such as R, allow installation and updating to occur making appropriate selections from the main menu bar.

In the next section, we review some of the main advantages and disadvantages of open source software.

5 Advantages and disadvantages of open source software

Commercial software vendors rely on the law of contracts and intellectual property to protect their softwares source code from being used by researchers for other purposes. Typically, the software application is "purchased" through a licensee/licensor contract. Because the source code is kept secret, software is usually delivered to licensees in object code or executable form, i.e., in machine-only readable form. It is possible to identify several, actual and potential, advantages of using OS software for researchers, students, and academic institutions.

Typically, open source software can be obtained at the cost of the media (CDs or diskettes) or network bandwidth (for distribution via the world wide web). Commercial packages used by economist can be quite expensive, especially if upgrading and licensing occurs frequently. Cost considerations can discourage the adoption of a commercial package by institutions from developing countries, and also by resource constrained universities in more developed countries. Moreover, newer versions that add new features can make previous versions rapidly obsolete (it is of small consolation if, after a long wait, you manage to obtain code that "Requires version *x.x* or greater, and library *y*"). Analogous problems arise when modification or extended functionality is required. Asking for features to be included is a long and tedious process. Some well known software producer are slow to respond, even in fixing serious bugs identified and reported on specialized journals. "Toolboxes," "modules," "packages," etc., can be extremely expensive, sometimes more than the core software itself. GNU's copy-left license guarantees the freedom to improve the program, and release the improvements to the public, so that the whole scientific community can benefit.

Uncertainties about the future development of a software application can also prevent its adoption. Under the GNU license, the software (and the option for support and development) will also be available if the software producer no longer exists.

Open source software is reputed to have a high degree of reliability. Serious errors have been found in some econometric and statistical packages, (see, e.g., Knüsel, 1995;McCullough, 1998, 1999; McCullough and Vinod, 1999b). Vendors of proprietary software rarely describe the algorithms used to implement econometric and statistical procedures, nor provide information about their reliability. This is a serious omission that makes the use of "black box" packages less attractive for academic research. Algorithm used, their implementation benefit form being open source. To ensure the highest standard of quality and degree of confidence in the results obtained, software should be subject to peer review as any other aspect of research and based on openly published and freely available algorithms and source code.

Open source software can promote efficiency and learning. Applied econometricians will sometimes have the necessity to engage in the process of crafting their own programs. Free software allows to obtain the source code and study it. The writing of code or the adapting of existing own to one's needs is thus facilitated. Free Software should avoids "re-inventing the wheel." The GNU license guarantees the freedom to redistribute copies, modified or not, so that the whole scientific community can benefit.

Disadvantages of opens source software include abandoned code and code "forking." For an example of software used in statistics and econometrics that has de facto been abandoned see the discussion on Xlisp-Stat in de Leeuw (2005). This is a problem that affects commercial software as well. With open source the problem is mitigated by the fact that the GNU license guarantees that the software will still be available and, at least in principle modifiable, even when the main project ceases to exist. Forking, as expected, is a problem that

seems to occur more frequently with open source. Forking of a project occurs when a developer takes code from a project and develops it independently of the original project. An example of forking is the Gnu-Emacs/XEmacs split. Forking is generally considered harmful in terms of wasted resources, but it can also create some beneficial competition as the EGCS (Experimental/Enhanced GNU Compiler System) which was a fork from GCC (GNU Compiler Collection) which was eventually reincorporated in the official GCC project. For a description of how GCC and other Unix-like software tools are used in economics see Racine (2000).

6 Reproducible computations using open source software

Economists' increasing dependence on computers has resulted in research that has become increasingly difficult to replicate. Implementation details of computations in Economics are left out of traditional printed publication. Researchers and students find it hard to reproduce published result, making progress in Economics hard to establish and validate. Moreover, researchers are finding it difficult to reproduce their own computational results only a few months after a project has been abandoned (see, e.g., Dewald et al., 1986). In this section, we want to highlight the potential role of open source software in organizing computational-based research and in mediating researcher's interaction with each other, Ph.D. students, and journal editors, by streamlining operations such as replication, mediating the actions and interactions of researchers, and supporting students' participation in the research process that might otherwise be too difficult to manage.

For instance, an empirical application in Economics consist in essence of computation on data. For an econometrics project to be reproducible, besides data and code, a prerequisite is that adequate documentation on the dataset used, its sources, the processing it has been subjected to, as well as documentation on software and programming code used for its analysis, should be made available.

Many of these activities can be automated using scripting languages as seen in Sect. 3. Modern scripting languages are among the main achievement of the open source movement (consider the success of Perl, Phyton, and PHP, to name just a few). They are innovative software of which proprietary equivalents are practically non-existent. The success of the Web would not have been possible without their development. For further examples of successful open source applications see Lerner and Triole (2002). In the next paragraphs, we show how these languages can encourage innovation in the way computational results can be implemented in a way that makes them easily reproducible. Perl will be used for this demonstration.

Perl, an acronym for *Practical Extraction and Report Language*,[15] is a cross-platform, high-level, open source programming language, originally designed

[15] Although many Perl programmer like to think it stands for "Pathologically Eclectic Rubbish Lister."

and implemented by Larry Wall. Perl can be downloaded from the WWW's Comprehensive Perl Archive Network (CPAN) located at `http://www.cpan.org/`, where the source package, modules, documentation, and links to binaries distributions of Perl are available for various platforms, including Win32, Mac, and Unix/Linux.[16] Perl is free, and is released under either the GNU GPL license or the less restrictive Artistic License.[17]

The Perl language originally incorporated and extended some of the best features of the C programming language, and from software tools such as sed, awk, grep, and the Unix shell. Perl has considerably evolved since its beginning and now features a full range of network and object-oriented capabilities. Perl's process, file, and text manipulation facilities make it particularly well-suited for a wide range of tasks that economists face regularly such as data collection and processing, including data transformation, validation and cleaning, and the merging of several data files. It can also be useful in simplifying other more labor intensive and sophisticated tasks such as the conversion of econometric code from one language to another (say GAUSS to Matlab and vice versa), regression output processing and typesetting, preparation of bibliography files (converting ISI/BIDS to bib files), publishing of math-intensive documents on Web pages (converting LaTeX to HTML), the distribution of computing and storage tasks across several nodes of a computer cluster, etc.

Perl's extensive network programming support, allows to collect in a short script, not only details about data processing and the code needed for the analysis, but also, for instance, the commands that retrieve the data itself and the software used for its analysis. A complete computational project with all the data and commands needed for its reproduction can be compactly preserved and effectively communicated through a Perl script. The following example illustrates this point. The example requires only R and LaTeX to be reproduced.

Consider estimating the demand for clean air model using the well-known Boston housing data (see Belsey, Kuh and Welsch, 1980, for example). Note that in the dataset available at STATLIB each record spans over two contiguous lines.

```
0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30
396.90  4.98  24.00
0.02731 0.00  7.070 0 0.4690 6.4210 78.90 4.9671 2 242.0 17.80
396.90  9.14  21.60
...
```

[16] We used ActivePerl release 5.8.0.806, which is based on Perl Version 5.8, the standard Win32 release available at the time of writing the present paper. The code has also being tested on platforms running the Solaris and the Linux operating system. The hardware used in this Chapter was a Dual Intel Pentium IV (Prestonia) Xeon Processors 3.06 GHz with HT Technology with 4 GB of RAM running on Microsoft Windows XP/2002 Professional (Win32 × 86) 5.01.2600 (Service Pack 2).

[17] For details on the Perl license consult the GNU Project's home page at `http://www.gnu.org/`.

The script downloads the Boston data from STATLIB, cleans and prepares the dataset, uses R[18] to estimate the coefficients of the model, saves the results in a file, and finally prints the results in LaTeX format.

```
# downloads Boston dataset from STATLIB
use LWP::Simple;
getstore( "http://lib.stat.cmu.edu/datasets/boston","boston.raw" );

# corrects for record spanning two lines
open( IN,  "boston.raw" );
open( OUT, ">boston.asc" );
do { $line = <IN> } until $. == 22
  or eof;    # Skips the first 22 lines of header
while ( $line = <IN> ) {
    chomp $line;
    $line .= <IN>;    # joins two lines
    print OUT $line;
}
close(OUT);

# sends data to R for regression analysis
open( RPROG, "| c:/rw1081/bin/Rterm.exe --no-restore --no-save" );
select(RPROG);
print << 'CODE';
bost<-read.table("boston.asc",header=F)
names(bost)<- c( "CRIM",   "ZN",     "INDUS",
                 "CHAS",   "NOX",    "RM",
                 "AGE",    "DIS",    "RAD",
                 "TAX",    "PTRAT",  "B",
                 "LSTAT",  "MEDV")
attach(bost)
boston.fit <- lm(log(MEDV) ~ CRIM + ZN + INDUS +
   CHAS + I(NOX^2) + I(RM^2) + AGE + log(DIS) +
   log(RAD) + TAX + PTRAT + B + log(LSTAT))
sum <- summary(boston.fit)$coe[,1:2]
write.table(sum,"boston.out",quote = FALSE)
q()
CODE
close(RPROG);

# creates LaTeX table with regression results
open( TABLE, "boston.out" );
open( TEX,   ">table.tex" );
$prec  = 3;    # sets number of decimals
$width = 9;    # sets the width of the field
do { <TABLE> } until $. == 1 or eof;    # Skips the first  line of
header
while (<TABLE>) {
    chomp;
    @line = split;
    printf TEX "%11s & %${width}.${prec}g & %${width}.${prec}g\\\\
    \n", $line[0],
      $line[1], $line[2];
}
```

[18] We used R release 2.0.0, the standard Win32 release available at the time of writing the present paper.

Table 2 OLS estimates of the demand for clean air model

Variable	Coefficient estimate	Standard error
(Intercept)	4.56	0.154
CRIM	−0.0119	0.00124
ZN	$8.02e-005$	0.000506
INDUS	0.00024	0.00236
CHAS	0.0914	0.0332
I(NOX^2)	−0.638	0.113
I(RM^2)	0.00633	0.00131
AGE	$9.07e-005$	0.000526
log(DIS)	−0.191	0.0334
log(RAD)	0.0957	0.0191
TAX	−0.00042	0.000123
PTRATIO	−0.0311	0.00501
B	0.000364	0.000103
log(LSTAT)	−0.371	0.025

```
close(TABLE);
close(TEX);
```

The TEX file can be included in a document using the command `\input{table}`. The result, after compilation in LATEX is the typeset Table 2.

These results are easily reproducible, as they can be generated in a matter of seconds and do not require proprietary data or licensed software. For more information on Perl and examples on how it can be used in economics to facilitate the reproduction of computational results, see Baiocchi (2003, 2004).

7 Conclusions

Journals that specialize in computational economics and econometrics do not require authors to deposit data and code that can reproduce the computational results of their papers. This implies that in even in principle most computational results cannot be verified. In this paper, we proposed some basic standards to improve the production and reporting of computational results in economics for the purpose of accuracy and reproducibility. In particular, we made recommendations on four aspects of the process: computational practice, published reporting, supporting documentation, and visualization. Also, we reflected on current developments in the practice of computing and visualization, such as integrated dynamic electronic documents, distributed computing systems, open source software, and their potential usefulness in making computational and empirical research in economics more easily reproducible. In this paper, we have also illustrated the advantages and disadvantages of using open source software applications in Economics. We found that the open source software development model can ensure that many software applications produced under this model, because of their availability, quality, reliability, and innovations, can benefit reproducibility of computational results in economics.

References

Amman, H., Kendrick, D., & Rust, J. (Eds.). (1996). *Handbook of computational economics* (Vol. 1). Amsterdam, The Netherlands: Elsevier North-Holland.

Anderson, R. G., Greene, W. H., McCullough, B., & Vinod, H. D. (2005). The role of data and program code archives in the future of economic research. Working Paper No. 2005-014B, FRB of St. Louis.

Baiocchi, G. (2003). Managing econometric projects using Perl. *Journal of Applied Econometrics, 18*(3), 371–378.

Baiocchi, G. (2004). Using Perl for statistics: Data processing and statistical computing. *Journal of Statistical Software, 11*(1), 1–81.

Baiocchi, G., & Distaso, W. (2003). GRETL: Econometric software for the GNU generation. *Journal of Applied Econometrics, 18*(1), 105–110.

Belsey, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics.* New York: Wiley.

Buckheit, J. B., & Donoho, D. L. (1995). *Wavelets and statistics*, chapter Wavelab and Reproducible Research (pp. 55–81). Berlin, New York: Springer.

Cleveland, W. S. (1994). *The elements of graphing data*. Summit, New Jersey: Hobart Press

Cleveland, W. S. (1993). *Visualizing data*. Summit, New Jersey: Hobart Press.

de Leeuw, J. (2005). On abandoning XLISP-STAT. *Journal of Statistical Software, 13*(7), 1–81.

Dewald, W. G., Thursby, J. G., & Anderson, R. G. (1986). Replication in empirical economics: The journal of money, credit and banking project. *American Economic Review, 76*(4), 587–603.

Eddelbuettel, D. (2000). Econometrics with Octave. *Journal of Applied Econometrics, 15*(5), 531–542.

Eddelbuettel, D. (2003). Quantian: A *scientific computing environment*. In *Proceedings of the 3rd international workshop on distributed statistical computing (DSC 2003) March 20–22, Vienna, Austria,* Vienna, Austria. Technische Universitt Wien.

Gentle, J. E. (2003). *Random number generation and Monte Carlo methods* (2nd ed.). New York: Springer.

Gentleman, R. (2004). Some perspectives on statistical computing. Technical report, Department of Biostatistics, Harvard School of Public Health, Boston, Massachusetts, USA.

Gentleman, R. (2005). Reproducible research: A bioinformatics case study. *Statistical Applications in Genetics and Molecular Biology, 4*(1), Article 2. Available at: http://www.bepress.com/sagmb/vol4/iss1/art2.

Gentleman, R., & Lang, D. T. (2004).Statistical analyses and reproducible research. Bioconductor Project Working Papers. Working Paper 2

Greene, W. (2000). *Econometric analysis* (4th ed.). New York: Prentice Hall.

Hoaglin, D. C., & Andrews, D. F. (1975). The reporting of computation-based results in statistics. *The American Statistician, 29*(3), 122–126.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*, 299–314.

Judd, K., & Tesfatsion, L. (Eds.). (2006). *Handbook of computational economics: Agent-based computational economics* (Vol. 2). Amsterdam, The Netherlands: Elsevier North-Holland.

Kernighan, B. W., & Pike, R. (1999). *The practice of programming.* Reading, MA: Addison-Wesley.

Kernighan, B. W., & Plauger, P. J. (1978). *The elements of programming style* (2nd ed.). New York, NY: McGraw Hill.

Knopper, K. (2003). Knoppix. Available at *http://www.knopper.net/knoppix/index-en.html*.

Knüsel, L. (1995). On the accuracy of statistical distributions in GAUSS. *Computational Statistics and Data Analysis, 20*, 699–702.

Knuth, D. E. (1983). Literate programming. Technical report STAN-CS-83-981, Stanford University, Department of Computer Science.

Knuth, D. E. (1984). Literate programming. *The Computer Journal, 27*(2), 97–111.

Knuth, D. E. (1992). *Literate programming.* CSLI Lecture Notes Number 27. Stanford, CA, USA: Stanford University Center for the Study of Language and Information.

Koenker, R. (1996). Reproducible econometric research. Technical report, Department of Econometrics, University of Illinois, Urbana-Champaign, IL.

Koenker, R. (2006). Reproducibility in econometrics research. Technical report, Department of Econometrics, University of Illinois, Urbana-Champaign, IL. http://www.econ.uiuc.edu/~roger/repro.html.
Leisch, F. (2002). Dynamic generation of statistical reports using literate data analysis. In W. Härdle (Ed.), *Proceedings in computational statistics* (pp. 575–580). Heidelberg, Germany: Physika Verlag.
Leontief, W. W. (1966). Input-output economics. In W. W. Leontief (Ed.), *Input-output economics* chapter 2 (pp. 13–29). New York: Oxford University Press.
Lerner, J., & Triole, J. (2002). The simple economics of open source. *Journal of Industrial Economics, 52*, 197–234.
MacKinnon, J. G. (1999). The Linux operating system: Debian GNU/Linux. *Journal of Applied Econometrics, 14*(4), 443–452.
McCullough, B. (1998). Assessing the reliability of statistical software: Part I. *The American Statistician, 52*, 358–366.
McCullough, B. (1999). Assessing the reliability of statistical software: Part II. *The American Statistician, 53*(1), 149–159.
McCullough, B., & Vinod, H. (1999a). The numerical reliability of econometric software. *Journal of Economic Literature, 37*(2), 633–665.
McCullough, B., & Vinod, H. (1999b). The numerical reliability of econometric software. *Journal of Economic Literature,* XXXVII*,* 633–665.
McCullough, B. D., McGeary, K. A., & Harrison, T. D. (2006). Lessons from the JMCB Archive. *Journal of Money, Credit, and Banking, 38*(4), 1093–1107.
McCullough, B. D., & Renfro, C. G. (1999). Benchmarks and software standards: A case study of GARCH procedures. *Journal of Economic and Social Measurement, 25*(2), 59–71.
Racine, J. (2000). The cygwin tools: a GNU toolkit for windows. *Journal of Applied Econometrics, 15*(3), 331–341.
Racine, J., & Hyndman, R. (2002). Using R to teach econometrics. *Journal of Applied Econometrics, 17*(2), 175–189.
Ramanathan, R. (2002). *Introductory econometrics with applications* (5th ed.). Orlando, Florida: Harcourt College Publishers.
Rossini, A. J., Heiberger, R. M., Sparapani, R., Mächler, M., & Hornik, K.(2004). Emacs speaks statistics: A multiplatform, multi-package development environment for statistical analysis. *Journal of Computational and Graphical Statistics, 13*(1), 247–261.
Sawitzki, G. (1999). Software components and document integration for statistical computing. In *Proceedings ISI Helsinki 1999 (52nd session)* Bulletin of the International Statistical Institute Tome LVIII, Book 2, pp. 117–120.
Sawitzki, G. (2005). Keeping statistics alive in documents. *Computational Statistics, 17*(1), 65–88.
Schwab, M., Karrenbach, M., & Claerbout, J. (2000). Making scientific computations reproducible. *Computing in Science and Engineering, 2*(6), 61–67.
St. Laurent, A. M. (2004). *Understanding open source and free software licensing: A straightforward guide to the complex world of licensing*. Sebastopol, CA, USA: O'Reilly & Associates.
Stallman, R. (1985). The GNU manifesto. *Dr. Dobb's Journal of Software Tools, 10*(3), 30–35.
Stokes, H. (2004). On the advantage of using two or more econometric software systems to solve the same problem. *Journal of Economic and Social Measurement, 29*, 307–320.
Tufte, E. R. (2001). The visual display of quantitative information (2nd ed.). Cheshire, Connecticut: Graphics Press.
Ueberhuber, C. W. (1997). *Numerical computation: Methods, software, and analysis* (Vol. 1). Berlin Heidelberg, Germany: Springer.
Varian, H. R. (Ed.). (1992). *Economic and financial modeling with mathematica*. New York: TELOS/Springer.
Varian, H. R. (Ed.). (1996). *Computational economics: Economic and financial analysis with mathematica*. New York: TELOS/Springer.
Vinod, H. D. (2000). Review of GAUSS for windows, including its numerical accuracy. *Journal of Applied Econometrics, 14*(2), 211–220.

Vinod, H. D. (2001). Care and feeding of reproducible econometrics. *Journal of Econometrics, 100*(1), 87–88.
Wooldridge, J. (2002). *Introductory econometrics: A modern approach* (2nd ed.). Mason, OH: Thomson, South-Western.