Roger Koenker and Achim Zeileis

Reproducible Econometric Research. A Critical Review of the State of the Art.

Paper

Reproducible Econometric Research
(A Critical Review of the State of the Art)

Roger Koenker, Achim Zeileis

Department of Statistics and Mathematics
Wirtschaftsuniversität Wien

Research Report Series

Report 60
November 2007

http://statmath.wu-wien.ac.at/

Reproducible Econometric Research
(A Critical Review of the State of the Art)

Roger Koenker[a] Achim Zeileis[b*]

[a] *Department of Economics, University of Illinois at Urbana-Champaign, United States of America*

[b] *Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Austria*

SUMMARY

Recent software developments are reviewed from the vantage point of reproducible econometric research. We argue that the emergence of new tools, particularly in the open-source community, have greatly eased the burden of documenting and archiving both empirical and simulation work in econometrics. Some of these tools are highlighted in the discussion of three small replication exercises.

Keywords: reproducibility, replication, software, literate programming, literate econometric practice.

1. INTRODUCTION

The renowned dispute between Gauss and Legendre over priority for the invention of the method of least squares might have been resolved by Stigler (1981). A calculation of the earth's ellipticity reported by Gauss in 1799 alluded to the use of *meine Methode*; had Stigler been able to show that Gauss's estimate was consistent with the least squares solution using the four observations available to Gauss, his claim that he had been using the method since 1795 would have been strongly vindicated. Unfortunately, no such computation could be reproduced leaving the dispute in that limbo all too familiar in the computational sciences. Stigler, deferring to Gauss's legendary reputation as a calculator, concludes that Gauss may have estimated a more sophisticated model than the simple first order approximation employed by prior researchers, but we may never have a conclusive resolution to this puzzle.

The question that we would like to address in this review is this: 200 hundred years after Gauss, can we do better? What can be done to improve our ability to reproduce computational results in econometrics and related fields? Our main contention is that recent software developments, notably in the open-source community, make it much easier to achieve and distribute reproducible research.

What do we mean by reproducible research? Buckheit and Donoho (1995) have defined what de Leeuw (2001) has called *Claerbout's Principle*:

> An article about computational science in a scientific publication is *not* the scholarship itself, it is merely *advertising* of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

See Schwab, Karrenbach and Claerbout (2000) for an elaboration of this point of view.

*Correspondence to: Achim Zeileis, Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Augasse 2–6, A–1090 Wien, Austria. E-mail: Achim.Zeileis@wu-wien.ac.at

The transition of econometrics from a handicraft industry (Wilson, 1973, Goldberger, 2004) to the modern sweatshop of globally interconnected computers has been a boon to productivity and innovation, but sometimes seems to be a curse. Who among us expected to be in the "software development" business? And yet many of us find ourselves precisely in this position, and those who are not, probably should be. As we will argue below, software development is no longer something that should be left to specialized commercial developers, but instead should be an integral part of the artisanal research process. Effective communication of research depends crucially on documentation and distribution of related software and data.

Some journals, such as the *Journal of Applied Econometrics* (JAE, `http://JAE.Wiley.com/`), support authors in this task by providing data archives (MacKinnon, 2007). However, a more integrated approach encompassing data, software, empirical analysis, and documentation is often desirable to facilitate replication of results. Section 2 reviews software tools that assist authors in the attempt to make their research easily reproducible for themselves and for others. In Section 3, the usage of some of these tools in practice is illustrated in three small replication exercises. Section 4 provides some conclusions and discusses challenges for the future. Computational details, necessary for the replication of this paper, are summarized in Section 5.

2. SOFTWARE TOOLS

In this section, we review some recent software developments that facilitate a more reproducible approach to econometric research. The tools discussed encompass the most common components of econometric practice: from data handling, over data analysis in some programming environment, to preparing a document describing the results of the analysis. Additionally, we provide information on literate programming tools and techniques that enable an integrated approach to these three steps, and on software for version control of all components.

2.1 Version Control

Econometric research on a given project is often carried out over an extended period, by several authors, on several computers, so it is hardly surprising that we often have difficulty reconstructing exactly what was done. Files proliferate with inconsistent naming conventions, get overwritten or deleted or are ultimately archived in a jumble with a sigh of relief when papers are finally accepted for publication. Sometimes, as is the case with JAE papers, a part of the archive is submitted to a central repository, but more often the archive resides peacefully on one or more authors' computers until the next disk crash or system upgrade.

Such difficulties have long been recognized in major software development efforts, but it is only relatively recently that practical version control systems have emerged that are well adapted to econometric research. Many version control systems have grown out of the open-source software community where rigorous archive cataloging of development efforts involving many participants is absolutely essential. We will briefly describe one such system, **Subversion** (SVN, `http://Subversion.Tigris.org/`) that we have used for this project, see Pilato, Collins-Sussman and Fitzpatrick (2004) for more detailed information. SVN is used for many open-source projects which involve large numbers of developers authorized to make changes in base code and documentation. However, even at the other extreme, the individual author toiling in isolation, SVN brings many conveniences and the peace of mind in knowing that project file structures and their history are safely archived. There are many similar systems, notably SVN's widely used predecessor **CVS** (Cederqvist et al., 2006), designed for software development, but SVN is particularly convenient for managing the combination of text, data and software found in most econometrics projects. Cross-platform compatibility is also an important consideration in many projects; SVN's basic command-line interface will feel comfortable to most Linux/Unix users, but the graphical clients **TortoiseSVN** for Windows (`http://TortoiseSVN.Tigris.org/`) embedded into the Windows **Explorer** or **svnX** for Mac (`http://www.Apple.com/downloads/macosx/development_tools/svnx.html`) will prove convenient for others.

The first stage of any SVN project involves the creation of a central repository for the project. Once the repository is created, any of the authorized collaborators can "checkout" the current version of the project files, update the local copies of the files, and make changes, additions or deletions to the files. At any point, the modified file structure can be "committed" thereby creating a new version of the project. When changes are committed they are recorded in the repository as incremental modifications so that all prior versions are still fully accessible (even if a file was deleted from the current revision). In rare cases that more than one author has modified the same passage of the same file, the system prompts authors to reconcile the changes before accepting the commitment. A complete historical record of the project is available at any point in the development process for inspection, search and possible restoration. Since modifications are stored as increments, file "diffs" in Unix jargon, storage requirements for the repository are generally far less severe than for prior improvised archiving strategies. Version numbering is fully automatic, so one consistent set of naming conventions for project files can be adhered to, thereby avoiding the typical mélange of ad hoc files resulting from impromptu version control strategies.

If an SVN server is available, setting up an SVN project/repository does not pose many technical difficulties: The server administrator just needs create a new repository (essentially executing a one-line `svnadmin create ...` command) and set a few access-control options. After that all users of this SVN just have to install the client of their choice (e.g., **TortoiseSVN**, **svnX**, see above), check out the repository and can already start to add files (see Pilato et al., 2004, Chapter 2). Only if no SVN server is available, the setup requires some more technical work. It is not un-similar to setting up a Web server but can be a lot easier than that depending on the network protocol chosen for communication (see Pilato et al., 2004, Chapter 6). As one SVN server can easily host many SVN projects/repositories, the individual researcher can often avoid setting up a server alone, especially if the university/departments already provides such a server, typically along with other Web services.[1]

2.2 Data Technologies and Data Archiving

In the beginning data was flat, representable as matrices, and easily storable in text files.[2] As data has become more plentiful it has also become more structurally complex. Relational database management systems (DBMSs) are still quite exotic in econometrics, but seem likely to become more commonplace. Stable, well-documented databases with proper version control are critical to the reproducibility goal. In many cases, data archives for individual projects can be most efficiently represented by software describing selection rules from publicly available data sources, like the Panel Study on Income Dynamics (PSID) and the US Census Bureau. Such data sources often provide their contents via the World Wide Web (WWW), either using one of the data technologies above or new standards such as XML or PHP that emerged with the WWW. However, as we stress in Section 3.1 regarding the Penn World Tables, reproducibility of such data retrieval strategies relies heavily on future access to current versions of the database.

If the data to be stored is not too complex and/or large, a flat plain text file is almost ideal for reproducibility (and hence also used for most publications in the JAE data archive): it is portable across platforms, many software packages can read and write text files and they can be efficiently archived, e.g., in an SVN repository.

For larger data sets, it may be useful or even necessary to store the data in a DBMS. Many candidates are available here, but most use some form of the SQL (structured query language) developed at IBM in the 1970s. From a reproducibility standpoint it is important to maintain cross-platform compatability and minimize licensing costs. Given these objectives, the open-source candidates **PostgreSQL** (`http://www.PostgreSQL.org/`) and **MySQL** (`http://MySQL.com/`) are the obvious candidates.

[1] For software-related projects, there are also various Web sites that host free SVNs, e.g., `http://SourceForge.net/`, `http://Code.Google.com/`, and `http://R-Forge.R-project.org/` among others.

[2] Easily stored need not imply easily retrieved, as anyone with with a drawer full of floppy disks or a collection of magnetic tapes is undoubtedly aware.

In addition to the data technologies above, other new standards have emerged, especially for sharing data across the WWW. These include XML (extensible markup language, Bray, Paoli, Sperberg-McQueen, Maler and Yergeau, 2006): a text-based format, well-suited for storing both data and meta-information, which is therefore used as the basis for many other data and text formats. A key feature of XML is that it is an open standard, maintained by an international consortium. In contrast to many proprietary formats which are often altered periodically[3], data formats based on such a standard have a higher likelihood of remaining accessible in the future.

Although not a data technology in the narrower sense, an important method of data retrieval (especially in replication tasks) is optical character recognition (OCR). Increasingly, previously published data is available electronically (e.g., in JPEG or PDF format) and can be extracted automatically using OCR software. Adopting such an approach, it is crucial to have a well-documented protocol describing the path from the original sources to analyzed data. A simple example of this sort is available in the data archive for Koenker (2006), available from http://www.econ.uiuc.edu/~roger/research/frechet/. In this case, the final version of the data is made available in three distinct formats: first, as scanned JPEG images of the original tables as published in 1873, second as text files after optical character recognition in the same format as the original tables, and third as a documented R data frame stored as an R binary data file. Code to transform the raw data into final form for analysis was conveniently incorporated into the documentation of the dataset.

Further discussion of data technologies for statistical/scientific applications may be found in Murrell (2009); a preliminary version of the book is available along with further material on the author's Web page at http://www.stat.auckland.ac.nz/~paul/ItDT/.

2.3 Programming Environments

Econometrics has always been a Tower of Babel with many languages, or programming environments, competing for attention in the never-ending struggle to communicate effectively with machines. Over the course of our disparate careers we have had opportunities to explore many of these, including, in approximate chronological order: TSP, MIDAS, SAS, Statlib, GLIM, S, LIMDEP, SHAZAM, GAUSS, S-PLUS, Lisp-Stat, SPSS, Minitab, Stata, Ox, MATLAB, Mathematica, and R. Clearly, these vary considerably in their level of "programmability" and degree of specialization. Lower-level languages such as C or Fortran also play an important role. A historical survey of the early development of econometric software may be found in Renfro (2004), see also Ooms and Doornik (2006). There is also an extensive literature on the individual merits and comparative performance of various forms of statistical software; see Baiocchi (2007) for further references, and a more proscriptive review of reproducibility issues. Here, we will try to restrict our attention quite narrowly to properties of programming environments that might facilitate reproducible research.

Language Stucture Software reviews of programming environments often stress speed of execution, rather than accuracy, breadth of coverage, or other less tangible attributes. This is unfortunate since the exertions of the machine are rarely a matter concern, or sympathy. Of more direct relevance in our view is ease of writing software (for avoiding errors) and reading software (for replication of results).[4] The programming language should support the user in turning theory into software that reflects how we think about the underlying method conceptually. Several language features are helpful in obtaining this goal: functional languages, object orientation, and modular programs with re-usable components. Environments for statistical computing are gradually moving away from the "canned soup model" toward functional languages that permit fresh ingredients to be assembled in a more flexible manner. Formulae specification for linear and

[3] There is an obvious incentive for commercial software developers to do so in to effort to bind existing users to systems that have exclusive capabilities. SAS and Microsoft have used these strategies particularly effectively.

[4] For those who believe that computer programming should be, like prayer, a private language between the individual supplicant and the *deus ex machina*, we recommend Kernighan and Plauger (1974) who emphasize the need to write programs that other *people* are capable of reading.

nonlinear models, and unified interfaces for general linear models constitutes two developments that have brought software and theoretical specification of models closer together. Combining such an approach with object orientation allows one to encapsulate complex structures (such as a fitted regression models) and define typical tasks for them (such as inference or visualization). Programs or analyses written in such a way are more intelligible and hence easier to reproduce. Furthermore, it is highly desirable to have a single environment within which to conduct empirical analyses and simulation studies. Re-using the same functionality across different tasks assures better consistency and avoids duplication of programming effort. But environments designed for the convenience of empirical analysis may not provide good simulation environments, and vice-versa. Also, to assure reproducibility and re-usability by other authors, the structural features of a language should facilitate (and not suppress) the ability to build on innovations of prior authors. Exploiting common structure in problems often leads to general software solutions that facilitate critical comparison of various methods as illustrated for structural change testing in Zeileis (2006a). The modern trend toward functional languages and object-orientation is visible in several of the currently successful econometrics environments. MATLAB (The MathWorks, Inc., 2007, http://www.MathWorks.com/), Ox (Doornik, 2006, http://www.doornik.com/ox/) and R (R Development Core Team, 2007, http://www.R-project.org/) are notable in this respect.

Open Sources Languages are acquired by mimicry so it is extremely valuable to have access to a large body of text written by those proficient in the language. This is one of the prime advantages of open-source software projects. When we first embark on a new econometric project, we have at our fingertips an extensive body of existing code, some of which will be directly useful as building blocks—provided that the language is structured to facilitate such modularity—and some code will offer only stylistic hints. In either case, adapting prior code when available from reliable sources is almost always preferable to reinventing the wheel—and, as argued above, increases the readability and intelligibility of the resulting programs. Most languages now offer some form of forum for sharing code across different projects, and thus provide access to an existing body of tested code. An exemplary approach is the Comprehensive R Archive Network (CRAN, http://CRAN.R-project.org/) that hosts more than 1,200 software packages (containing code, data, and documentation) which are checked on a daily basis on several platforms. In addition to such open platforms for user-contributed code, there are also a few journals publishing peer-reviewed software, e.g., *The Stata Journal* (http://www.Stata-Journal.com/) and the *Journal of Statistical Software* (http://www.JStatSoft.org/). Furthermore, even commercial software producers usually provide some functionality written in the language itself and therefore open to inspection and emulation for the individual user. MATLAB and Stata (StataCorp., 2007, http://www.Stata.com/) are notable in this respect. However, even in these favorable circumstances one may eventually wish to dig below the surface only to discover that crucial elements of the story are accessible only in the binary form readable only by machines. At this point the computations become a black box visible only to the select few, and scientific trust becomes a leap of faith. Source code is itself the ultimate form of documentation for computational science, when it is well written it can serve as an inspiration for subsequent work, when it is flawed, but accessible, then it can be much more easily subjected to necessary critical scrutiny.[5] With computational methods, gradual refinement is seriously impeded unless source code is open.

Language Interfaces To combine the convenience of high-level programming environments with the efficiency of compiled languages, it is desirable to either (byte-)compile program code directly or to be able to link code written in lower-level languages like Fortran or C and their dialects. Most modern languages employed in econometrics—including MATLAB, Ox, R, and Stata—offer some means for ordinary users to accomplish this sort of sorcery. Doing so in ways that are platform independent is a considerable challenge. From a reproducibility perspective, it should be assured that even code interfacing lower level languages behaves consistently across

[5] This is the essential strength of scientific discourse captured Piet Hein's grook: "The road to wisdom? Well it's plain and simple to express: Err and err and err again, but less, and less and less."

platforms with differing hardware and operating systems. This problem is particularly acute in simulation settings where it is often desirable to be able to reproduce sequences pseudo-random numbers across machines and operating systems. R is notable for assuring reproducibility of random number generator results across platforms and versions of the software.

Environmental Stability Since hardware and software environments for econometric research are constantly evolving, a major challenge for reproducibility involves proper documentation of the environment used to conduct the computational component of the project. Ideally, in our view, authors and journals should be expected to provide a section similar to Section 5 ("Computational Details") of the present paper describing in some detail the software and hardware environment employed. Even when these environments are completely specified, it is likely to be difficult several years later to restore a prior version of the environment. In this respect, again, the open-source community is exemplary, since prior versions are typically archived and easily downloadable. In R, for example, all prior versions of the base system and its contributed packages are readily available from CRAN. For commercial software prior versions are sometimes difficult to obtain due to licensing and distribution constraints; Zeileis and Kleiber (2005) describes an exercise in "forensic econometrics" exploring the evolution of the evaluation of the Gaussian distribution function in successive versions of GAUSS (Aptech Systems, Inc., 2006, http://www.Aptech.com/). Such investigations are obviously handicapped by the unavailability of older versions and lack of adequate documentation of algorithms and their evolution.

2.4 Document Preparation Systems

TEX and LATEX have become the *lingua franca* for the composition of mathematical text in general and econometric text in particular. TEX (http://www.TUG.org/) developed by Knuth (1984) beginning in the late 1970s constitutes an exceptional case study in software development and the effectiveness of the gradual refinement process of open-source projects. LATEX (http://www.LaTeX-project.org/), originally written by Lamport (1994), still constitutes an ongoing development effort to build a higher level markup language on the super-structure provided by TEX. Nevertheless, LATEX is also a remarkably stable environment and serves as a convenient and portable format across many platforms for a wide variety of documents.

Although these systems are clearly state of the art in econometric document preparation, casual users of document preparation systems often prefer to avoid the somewhat steeper learning curve of LATEX in favor of *WYSIWYG* (what you see is what you get) text processors such as Microsoft's **Word** (http://www.Office2007.com/) or **OpenOffice.org** (http://www.OpenOffice.org/). To avoid a general discussion about the relative merits of LATEX over WYSIWYG processors, we focus on the perspective of reproducibility: Especially **Word**'s proprietary binary document format is problematic in this respect as its documents are less stable across platforms or versions of **Word**. With the advent of the open-source suite **OpenOffice.org** the situation improved considerably as it not only introduced an XML-based open document file format (ODF) for WYSIWYG documents but also supports export to a number of older proprietary file formats.[6]

For Web-based publishing, documents are typically written in HTML (hypertext markup language, http://www.w3.org/, see Raggett, Hors and Jacobs, 1999) or prepared in PDF (portable document format) which can be easily created from both LATEX or **OpenOffice.org**. HTML is usually preferred for multiple linked documents and PDF—although it also provides hyperlinking—for single documents.

Thus, in addition to the inevitable LATEX, there are other open systems and standards—fulfilling different purposes—such as ODF and HTML that are rather stable across platforms and versions. All of them have in common that they are based on a markup language which is particularly useful for assuring reproducibility as they can also contain markup for computer programs and

[6] Whereas ODF is an ISO standard (ISO 26300) Microsoft has not yet succeeded in establishing its own XML-based standard, see http://www.iso.org/iso/pressrelease.htm?refid=Ref1070.

data analysis (visible or invisible) along with the rest of the document. This is discussed in more detail in the following section.

2.5 Literate Programming

The idea of merging text, documentation and various forms of computer code arose from the structured programming discussions of the 1970s and was championed by Knuth (1992) following his initial development of TeX. Literate programming, as this movement has come to be called, encourages a more readable programming style by making the code itself an integral part of the documentation. The basic operations on documents containing both code chunks and documentation chunks are known as *tangling* and *weaving*: the former strips off the documentation chunks and extracts only the code chunks while the latter weaves the code chunks into the documentation, typically by adding appropriate markup for display of the code. Following Knuth (1992), the initial literate programming systems were **WEB** and **CWEB** for combining Pascal and C code, respectively, with TeX documentation. In order to provide a leaner and more flexible literate programming system, Ramsey (1994) developed **noweb**, a set of open-source tools for combining code in arbitrary languages and LaTeX documentation. In particular, it provides the commands `notangle` and `noweave` and can be used out of the box with the programming languages/environments discussed in Section 2.3.

Literate programming is an important first step in supporting reproducibility in econometric practice but one would like to carry the idea a step further and include not only the code but also its results dynamically in a document (Leisch and Rossini, 2003). This idea of *literate econometric practice* seems especially well-suited to statistical computing applications where models, data, algorithms and interpretation all coalesce to produce scientific documents. Directly linking text with computational procedures reduces the likelihood of inconsistencies and facilitates reproducing the same type of analysis with an extentended/modified data set or different parameter settings.[7] Mathematica's concept of "notebooks" is an approach to documents of this type where the displayed document can be easily altered by changing the associated Mathematica code. Another implementation, more closely modelled after the literate programming ideas discussed above, is the `Sweave` system of Leisch (2002) for R. Re-using the markup commands of **noweb**, it allows to create "revivable" documents containing a tightly coupled bundle of code and documentation. To illustrate this, and to try to practice what we preach, the archived version of this paper includes a source file '`RER.Rnw`'[8] that includes all of the text as well as all of the code used to generate the statistical analyses presented in the next section. The input file is structured into code chunks written in R and text chunks written in LaTeX and it can be processed in R with the command `Sweave("RER.Rnw")`. This extracts the code chunks and executes them in R, produces the associated output (either in numerical or graphical form) and weaves them into a LaTeX file (plus graphics files in PDF). This resulting LaTeX file can then be processed to PDF by pdfLaTeX. The choice of PDF is specific to our document, by default `Sweave` generates both PDF and EPS graphics so that it can be processed by various flavors of LaTeX. Furthermore, `Sweave` can be extended to other documentation formats: **R2HTML** (Lecoutre, 2003) provides an `Sweave` driver for weaving R code and HTML documentation and **odfWeave** (Kuhn, 2006) supports mixing R code with ODF documentation. The ideas of the `Sweave` system are also not restricted to R: Lenth and Højsgaard (2007) provide infrastructure for dynamic documents mixing SAS code with LaTeX documentation. As far as we are aware, none of the other environments used in econometrics provide similar facilities, but it would be relatively easy to rectify this.

[7] Proximity is, of course, no guarantee that text and code will agree; we have all read and probably written commented code for which the comments contradicted the unintended consequences of the code. Indeed, for the original author such comments may serve as a way to obscure these unintended consequences, allowing him to read the comments, not the code. See, e.g. the comments by Doob in Snell (1997, pp. 308–309) on the virtues of random proof reading.

[8] The suffix '`.Rnw`' stands for R and **noweb**.

3. REPLICATION CASE STUDIES

In this section, we describe briefly three replication exercises chosen to illustrate several aspects of the reproducibility problems discussed above. The software tools used are **Subversion** for version control, flat text files for data storage, R for programming and empirical analysis, and LaTeX for document preparation. As mentioned above, a literate data analysis approach is adpoted based on the `Sweave` tools. The SVN archive, containing the full sources of the document, can be checked out anonymously from `svn://svn-statmath.wu-wien.ac.at/repro/`. For convenience of non-SVN users, there is also an online archive available at `http://www.econ.uiuc.edu/~roger/research/repro/`.

3.1 An Empirical Example: Cross-Country Growth Regression

Durlauf and Johnson (1995, henceforth DJ) investigate cross-country growth behavior based on an extended Solow model suggested in Mankiw, Romer and Weil (1992, henceforth MRW). The variables in the growth regression model are real GDP per member of working-age population, Y/L (separately for 1960 and 1985); fraction of real GDP devoted to investment, I/Y (annual average 1960–1985); growth rate of working-age population, n (annual average 1960–1985); fraction of working-age population enrolled in secondary school, $SCHOOL$ (annual average 1960–1985); and the adult literacy rate, LR in 1960. Data for these variables (except LR) for 121 countries is printed in the appendix of MRW and provided in electronic form in the JAE data archive along with DJ (who added LR).

The unconstrained extended Solow model suggested by MRW in their Table V and given by DJ in their Equation 7 regresses the GDP growth $\log(Y/L)_{1985} - \log(Y/L)_{1960}$ on initial GDP in logs $\log(Y/L)_{1960}$ as well as $\log(I/Y)$, $\log(n + g + \delta)$ (assuming $g + \delta = 0.05$), and $\log(SCHOOL)$. DJ first fit the model by least squares for all 98 non-oil-producing countries (reproducing the model fit of MRW) and subsequently to various subsets of countries—with subset selection based on initial output $(Y/L)_{1960}$ and/or literacy LR_{1960}—finding multiple regimes rather than a single stable set of coefficients.

Here, we aim to reproduce the unconstrained regression results given by DJ in their Table II and V given the sample splits described in the paper. Although this seems to be a rather modest task, it turned out to be surprisingly difficult, illustrating some typical pitfalls. We first read the data, provided in file '`data.dj`' in the JAE data archive, into R, coding missing values as described in the accompanying documentation and subsequently selecting the non-oil-producing countries.

```
R> dj <- read.table("data.dj", header = TRUE, na.strings = c("-999.0", "-999.00"))
R> dj <- subset(dj, NONOIL == 1)
```

The relevant columns in the resulting data set `dj` are `GDP85`, `GDP60`, `IONY`, `POPGRO`, `SCHOOL`, and `LIT60`. The last four are described as ratios/fractions, but it was not clear from the documentation that they were given in percent. Of course, this is quickly revealed by a look at the actual data; and MRW probably used this scaling because it is easier to read when printed. However, it remains unclear which scaling of the variables was used for model fitting. After first attempting to use the variables as printed, it turned out that MRW had scaled all variables back to the unit interval. Thus, the model employed by MRW and given in the first column of Table II in DJ can be written in R as the formula

```
R> mrw_model <- I(log(GDP85) - log(GDP60)) ~ log(GDP60) + log(IONY/100) +
+     log(POPGRO/100 + 0.05) + log(SCHOOL/100)
```

where the response `I(log(GDP85) - log(GDP60))` is explained by (`~`) the four scaled regressors.[9] This can then be fitted with R's linear model function `lm()` to the `dj` data set, producing the following regression summary:

[9] Wrapping the response into `I()` is necessary for using the `-` symbol in its arithmetic meaning (rather than as a model formula operator).

```
R> dj_mrw <- lm(mrw_model, data = dj)
R> summary(dj_mrw)

Call:
lm(formula = mrw_model, data = dj)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.9104 -0.1760  0.0179  0.1844  0.9385 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)              3.0215     0.8275    3.65  0.00043 ***
log(GDP60)              -0.2884     0.0616   -4.68  9.6e-06 ***
log(IONY/100)            0.5237     0.0869    6.03  3.3e-08 ***
log(POPGRO/100 + 0.05)  -0.5057     0.2886   -1.75  0.08306 .  
log(SCHOOL/100)          0.2311     0.0595    3.89  0.00019 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.327 on 93 degrees of freedom
Multiple R-Squared: 0.485,        Adjusted R-squared: 0.463 
F-statistic: 21.9 on 4 and 93 DF,  p-value: 8.99e-13 
```

reproducing coefficient estimates, standard errors, adjusted $\bar{R}^2$, and the residual standard error σ_ε provided in the first column of Table II in DJ (identical to the first column in Table V of MRW) with only small deviations for some of the entries. Another way in R to extract the same table of coefficients (along with standard errors and Wald tests) is `coeftest(dj_mrw)` (from package **lmtest**, see Zeileis and Hothorn, 2002) which will be used below.

In the next step we want to fit the two other columns of DJ's Table II, pertaining to the same model fitted to two different subsets: DJ employ a low-output/low-literacy sample of 42 countries with $(Y/L)_{1960} < 1950$ and $LR_{1960} < 54\%$ and the corresponding high-output/high-literacy sample, also consisting of 42 countries. However, when selecting these sub-samples and computing their size we obtain

```
R> c(nrow(subset(dj, GDP60 <  1950 & LIT60 <  54)),
+    nrow(subset(dj, GDP60 >= 1950 & LIT60 >= 54)))

[1] 43 39
```

showing that either the subset definitions or the sample sizes are misstated in DJ's Table II. Some brute-force search (via all-subsets regression) coupled with some educated guessing, yielded revised cut-offs for model fitting of 1800 and 50%, respectively, yielding the consistent sample sizes, and also model fits as we will see below.

```
R> c(nrow(subset(dj, GDP60 <  1800 & LIT60 <  50)),
+    nrow(subset(dj, GDP60 >= 1800 & LIT60 >= 50)))

[1] 42 42
```

Fitting the same model to these two subsets yields DJ's slope coefficients, but the wrong intercepts. After further combinatorial search using different logarithm bases and variable scalings it turned out that DJ appear to have employed the model:

```
R> dj_model <- I(log(GDP85) - log(GDP60)) ~ log(GDP60) + log(IONY) +
+    log(POPGRO/100 + 0.05) + log(SCHOOL)
```

i.e., only scaling POPGRO to the original unit interval but keeping IONY and SCHOOL in percent. Thus, the intercepts in their Table II are not comparable between the first and the other two columns, while the coefficients of interest are unaffected by the scaling. Using the model formula dj_model, we can then go on and fit the model to both sub-samples:

```
R> dj_sub1 <- lm(dj_model, data = dj, subset = GDP60 <  1800 & LIT60 <  50)
R> dj_sub2 <- lm(dj_model, data = dj, subset = GDP60 >= 1800 & LIT60 >= 50)
```

and perform the Wald test for all coefficients, via,

```
R> coeftest(dj_sub1, vcov = sandwich)

t test of coefficients:

                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.4001     1.8460    0.76   0.4530
log(GDP60)              -0.4438     0.1566   -2.83   0.0074 **
log(IONY)                0.3097     0.1141    2.71   0.0100 *
log(POPGRO/100 + 0.05)  -0.3794     0.4676   -0.81   0.4224
log(SCHOOL)              0.2088     0.0941    2.22   0.0327 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

which reproduces the results from the second column in their Table II (with only minor deviations). Note that heteroskedasticity-consistent sandwich standard errors are used (provided by package **sandwich** in R, see Zeileis, 2004, 2006b) whereas conventional standard errors are used for the first MRW column. Analogously, the results for dj_sub2 can be obtained via

```
R> coeftest(dj_sub2, vcov = sandwich)
```

for which the output is suppressed here but given in condensed form in our Table I (generated using Sweave() with the models fitted above).

Thus, we have reproduced DJ's Table II but only after some effort: cut-offs for subset selection are different than those displayed in the table, variable scaling is different leading to different intercepts, and the standard errors used vary across columns (although DJ state briefly in a footnote that they would use sandwich standard errors). None of this changes their results qualitatively, all conclusions drawn from the analysis of DJ are supported. Nevertheless, it would be desirable to avoid such uncertainty both from the author's and reader's point of view. Organizing both code and documentation in a single revivable document as we suggest in the previous section will assist in this but can, of course, not prevent human error. It will, however, be much easier for the authors and readers—provided the code is made publicly available—to find the sources of such confusions and untie the knots.

As an epilogue to this example, we would like to speculate about what might have caused this confusion. As stated above, MRW consistently scaled all fraction/ratio variables in their models and used conventional standard errors so that we could easily reproduce all of their results (up to small deviations) in their Table I–VI. Possibly, DJ shared this experience and therefore simply copied the MRW column from the original paper, overlooking that they used a different scaling and different standard errors for the models fitted to sub-samples. The cut-offs for high/low output/literacy probably changed slightly over the work on the manuscript while the table header remained the same. However, there seems to be something more to it because in their Table V DJ present results for the model fitted to four different sub-samples (determined by recursive

Table I: Replication of cross-country growth regressions, corresponding to models `dj_mrw`, `dj_sub1`, `dj_sub2` with conventional standard errors for the first column and sandwich standard errors for the other two.

	MRW	$(Y/L)_{1960} < 1800$ $LR_{1960} < 50\%$	$(Y/L)_{1960} \geq 1800$ $LR_{1960} \geq 50\%$
Constant	3.022	1.400	0.450
	(0.827)	(1.846)	(0.723)
$\log(Y/L)_{1960}$	−0.288	−0.444	−0.435
	(0.062)	(0.157)	(0.085)
$\log(I/L)$	0.524	0.310	0.689
	(0.087)	(0.114)	(0.170)
$\log(n+g+\delta)$	−0.506	−0.379	−0.545
	(0.289)	(0.468)	(0.283)
$\log(SCHOOL)$	0.231	0.209	0.114
	(0.059)	(0.094)	(0.164)
$\bar{R}^2$	0.46	0.27	0.48
σ_ε	0.33	0.34	0.30

partitioning) for which coefficients are easily reproduced but conventional standard errors are used in sub-sample 1 and 4 and sandwich standard errors in sub-sample 2 and 3. Interestingly, there is another follow-up study published in JAE by Masanjala and Papageorgiou (2004) who use the same data set. They used all variables without any scaling—as we did in our first attempt—and we could exactly reproduce all their results on this data set. Detailed and commented R code for reproducing the regression results from all three studies is made available with this manuscript. Finally, Masanjala and Papageorgiou (2004) went on to produce an updated version using data up to 1995 which they constructed from the Penn World Table (PWT, Heston, Summers and Aten, 2002), using version 6.0 rather than 4.0 (used by MRW). This pointed us to another question in replication studies: Is it possible not only to reproduce the analysis given the data, but also the data itself? We tried this for the PWT 6.0 data but it did not seem to be very straightforward. Hence, we conclude this example with the recommendation that also some information (or preferably code) should be made available that documents the data pre-processing, especially if it is derived from a standard data source such as PWT.

3.2 Another Empirical Example: Wage-Equation Meta-Model

Our second case study in replication reports our experience trying to reproduce the empirical results in one of our own papers. Koenker (1988) describes a meta-analysis of published wage equations intended to explore how parametric dimension of econometric models depends upon sample size. Since this paper predated the JAE archive policy, we have had to rely on personal archives, but a complete record of the replication exercise will be submitted to the JAE archive, and—if accepted—would become the earliest entry in that archive.

The data for the study consists of 733 wage equations from 156 published papers in mainstream journals and essay colllections. In addition to citation information and a topic indicator, the sample size and parametric dimension of the model was recorded for each equation. Data was originally entered by hand on 3 by 5 index cards and then recorded as an ASCII data file. Fortunately, this data file and a small SED/AWK script to preprocess the data was preserved in easily accessible form.[10] For convenience, we have also written a brief R script replacing the original script and yielding a standard comma-separated file '`rk.csv`'.

[10] For a few agonizing hours it seemed that the only extant version of the data resided on a 3M DC 600A backup tape cartridge, and prospects for reading this tape looked rather bleak.

The models used for the analysis are standard count data models. Observations on each wage equation are weighted by the reciprocal of the number of equations, m, appearing in the paper, so the effective unit of analysis is really the paper not the equation. The first, and simplest, version of the model, appears in the paper as,

$$\log\lambda \quad = \quad \underset{(0.149)}{1.336} \; + \; \underset{(0.017)}{0.235} \log z.$$

where λ is Poisson rate parameter, the expected number of parameters y in the equation, and z is the sample size. The parameter of interest is the elasticity of parsimony, or *parsity*, the log derivitive of λ with respect to z. In this case, parsity is constant and equal to about $1/4$ implying that parametric dimension increases roughly like the fourth root of the sample size. The paper dutifully reports that the computations were conducted using release 3 of the GLIM system (Baker and Nelder, 1978). Our first attempt to replicate these results in R invoked the incantation:

```
R> rk <- read.csv("rk.csv")
R> names(rk)[2:4] <- c("m", "y", "z")
R> rk1 <- glm(y ~ log(z), weights = 1/m, family = quasipoisson, data = rk)
R> summary(rk1)

Call:
glm(formula = y ~ log(z), family = quasipoisson, data = rk, weights = 1/m)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-8.059  -1.046  -0.631  -0.215  21.588

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3362     0.1739    7.68    5e-14 ***
log(z)        0.2353     0.0199   11.82   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 6.417)

    Null deviance: 4328.2  on 732  degrees of freedom
Residual deviance: 3459.8  on 731  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

The reader can imagine our relief seeing that the estimated coefficients agreed perfectly with the published results, turning to dismay as we observed that the standard errors differed by a factor of 0.859. Covariance matrices for the parameters in quasi-Poisson models require an estimated dispersion parameter typically some generalization of a residual sum of squares scaled by the residual degrees of freedom. In the text of the paper, this was claimed to be the scaled Pearson statistic $\hat\sigma^2_P = (n-p)^{-1}\sum w\cdot(y-\hat y)^2/\hat y$, following the advice offered by McCullagh and Nelder (1983, pp. 172–173). Unfortunately, however, the authors of the GLIM system, in their wisdom, chose to use the corresponding *deviance* statistic instead of the Pearson statistic, that is they scaled the variance-covariance matrix of the estimated coefficients by $\hat\sigma^2_D = (n-p)^{-1}2\sum w\cdot(y\log(y/\hat y)-(y-\hat y))$. R adopts the Pearson-based estimate as its default and hence the dispersion of 6.42 reported in the summary above does not conform with the estimate of 4.73 reported in the original paper. It is relatively simple to recalculate the R results using the deviance-based estimate:

```
R> dispersion <- function(object, type = "deviance")
+    sum(residuals(object, type = type)^2)/df.residual(object)
R> summary(rk1, dispersion = dispersion(rk1))

Call:
glm(formula = y ~ log(z), family = quasipoisson, data = rk, weights = 1/m)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-8.059  -1.046  -0.631  -0.215  21.588

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.3362     0.1493    8.95   <2e-16 ***
log(z)        0.2353     0.0171   13.76   <2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

(Dispersion parameter for quasipoisson family taken to be 4.733)

    Null deviance: 4328.2  on 732  degrees of freedom
Residual deviance: 3459.8  on 731  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

And we see that the standard errors and the reported dispersion estimate, now agree with the published results. The lesson of this “conflict of defaults” is that it is dangerous to make assumptions about what software is doing; more careful reading of the GLIM manual would have revealed that deviance residuals were being used and this error could have been avoided.

In an effort to explore the sensitivity of this estimate to the specification of the functional form of the model, several other models were reported in Koenker (1988). Table II summarizes our attempt to replicate these results. For each of the four Poisson models we report, in addition to coefficients and standard errors, both the Pearson and deviance dispersion estimates and an estimate of the parsity parameter π evaluated at $z = 1000$ which is roughly the geometric mean of the observations on sample size. The quadratic-in-logs model was very sensitive to a few outlying z observations and two versions of the estimates were reported, one for the full sample as Equation 2 and one for a reduced sample excluding points with $z > 250,000$, as Equation 3. Again, replication revealed an error: the text of the paper reports that this cut-off as $500,000$, but the yellowing page in the project file folder and the GLIM output clearly reveal that $250,000$ was used. This is a clear case in which a more literate style of programming might have prevented the error by enforcing consistency between the text and the econometric estimation. Our use of Sweave is one way to accomplish this.

```
R> rk2 <- glm(y ~ log(z) + I(log(z)^2), weights = 1/m, family = quasipoisson,
+    data = rk)
R> rk3 <- glm(y ~ log(z) + I(log(z)^2), weights = 1/m, family = quasipoisson,
+    data = rk, subset = z <= 250000)
```

The results reported in Table II for Equations 1 through 4 agree with the published results to the precision reported, except for the coefficient on $\log z$ in Equation 2 which appears to be an old-fashioned typo[11], again something that could be more easily avoided by better integration of text and data analysis.

[11] The parsity estimate reported in the paper is consistent with our replicated value rather than the published value.

Table II: Replication of wage-equation meta-models, corresponding to `rk1`–`rk4` and `rk_nb` with deviance-based dispersion estimates for the quasi-Poisson models and sandwich standard errors for the negative binomial model.

	Quasi-Poisson				Neg-Bin
	1	2	3	4	
Constant	1.336	−0.439	1.737	−0.777	−0.677
	(0.149)	(0.512)	(0.517)	(0.315)	(0.383)
$\log z$	0.235	0.663	0.058		
	(0.017)	(0.118)	(0.129)		
$(\log z)^2$		−0.024	0.015		
		(0.007)	(0.008)		
$\log\log z$				1.947	1.898
				(0.148)	(0.209)
σ^2_D	4.73	4.64	4.41	4.67	
σ^2_P	6.42	6.26	5.69	6.33	
$\pi(1000)$	0.24	0.32	0.27	0.28	

Two final models employed $\log\log z$ as the only covariate; first in the quasi-Poisson specification and then using the negative binomial likelihood. The former, given in Equation 4, can again be reproduced as before

```
R> rk4 <- glm(y ~ log(log(z)), weights = 1/m, family = quasipoisson, data = rk)
```

using the same specification of dispersion as above. The negative binomial results in Koenker (1988) were computed, not in GLIM, but with general code written by Richard Spady for maximum likelihood estimation and linked to S. The same model can now be estimated in R using the `glm.nb()` function from the **MASS** package (Venables and Ripley, 2002). However, due to an inconsistency in the weight handling of `glm()` and `glm.nb()` (the latter assumes weights are case weights) it was necessary to weight with `max(m)/m` instead of `1/m`:

```
R> rk_nb <- glm.nb(y ~ log(log(z)), data = rk, weights = max(m)/m)
```

For this model, sandwich standard errors were reported based on Spady's estimation of the Hessian and outer-product matrix of the gradient. Our attempt to replicate these results employed the R **sandwich** package as for the growth regressions in Section 3.1.[12]

The results are again shown in condensed form in our Table II. Comparing these results with the published estimates we find somewhat larger discrepancies than for the quasi-Poisson regressions, but the results are remarkably consistent. We would conjecture that this degree of agreement would be rare in most instances where independent software was used to do non-linear maximum likelihood estimation.

3.3 A Simulation Example: Power of Fluctuation Tests

Another component of many econometrics papers that is often difficult to reproduce involves simulation studies. For these, replication is usually even harder than for empirical analyses because there is typically neither code nor data but only a textual description of the simulation setup in the paper. In such descriptions, it happens rather easily that not all tiny details of either the data-generating process (DGP) or the analysis methods are spelled out. Making readable code available

[12] Note that using `summary()` here will give misleading results due to the different weight handling. We have reported this to the package authors and hope that the inconsistency can soon be resolved.

along with the manuscript would alleviate this task because code is often less clumsy than words, especially for the small details. One could ask the question why readers would want to re-run the code given that it is not unusual that it takes days or even weeks to obtain the results. The answer is that, even without re-running the whole simulation, code for the DGP would be useful to re-use similar designs in follow-up publications, the code for the analysis methods would give insight into how exactly the methods under study were applied to the data (e.g., which "flavor" of a particular test was used or which measures of performance, etc.).

To illustrate how useful building blocks for carrying out simulation studies could be made available, we reproduce a simulation of structural change tests of Ploberger and Krämer (1992). They compared their CUSUM test based on OLS residuals with the standard CUSUM test based on recursive residuals, showing that neither have power against orthogonal shifts. Here, we follow their simulation setup, but to make it a little more satisfying methodologically (and cheaper to compute) we also compare the OLS-based CUSUM test to the Nyblom-Hansen test (Nyblom, 1989, Hansen, 1992) which is consistent for orthogonal changes.

The simulation setup considered by Ploberger and Krämer (1992) is quite simple, and very clearly described. They consider a linear regression model with a constant and an alternating regressor $x_t = (1, (-1)^t)$ $(t = 1, \dots, T)$, independent standard normal errors, and a single shift in the regressor coefficients from β to $\beta + \Delta$ at time $T^* = z^* T$. In the simulation, they vary the intensity b, the timing z^* and the angle ψ of the shift as given by $\Delta = b/\sqrt{T}(\cos\psi, \sin\psi)$, corresponding to their Equation 35. They give sequences of values for all three parameters and simulate power for a sample size of $T = 120$ from $N = 1,000$ runs at 5% significance level. This is a rather good description of the DGP, only two very small pieces of information are missing: β was left unspecified (presumably because the tests are invariant to it) and it is not completely clear whether the observation T^* (if it is an integer) belongs to the first or second regime. Given the design of their previous simulation in their Equations 33 and 34, it probably should belong to the second regime. However, we will place it in the first regime so that $z^* = 0.5$ corresponds to two segments of equal size.

To turn their simulation design into modular and easily re-usable code, we split it into three functions that capture the most important conceptual steps: (1) the DGP that simulates a data set for a given scenario, (2) a function that evaluates the tests on this DGP by power simulations, (3) a wrapper function that runs a loop over all scenarios of interest using the previous two functions. For step (1), we define a function `dgp()` in R.

```
dgp <- function(nobs = 100, angle = 0, intensity = 10, timing = 0.5,
  coef = c(0, 0), sd = 1)
{
  coef <- rep(coef, length.out = 2)
  delta <- intensity/sqrt(nobs) * c(cos(angle * pi/180), sin(angle * pi/180))
  err <- rnorm(nobs, sd = sd)
  x <- rep(c(-1, 1), length.out = nobs)
  y <- ifelse(seq(along = x)/nobs <= timing,
         coef[1] + coef[2] * x + err,
         (coef[1] + delta[1]) + (coef[2] + delta[2]) * x + err)
  return(data.frame(y = y, x = x))
}
```

This is a concise code description of the DGP described verbally above which can now be employed to apply the test procedures of interest to data frames resulting from `dgp()`. Based on this, we can define the function `testpower()` for (2) that takes the number of replications and the size of the test as its main parameters. Furthermore, by using the `...` notation in R all arguments used previously for `dgp()` can simply be passed on. We provide functionality for evaluating the tests of interest, the OLS-based CUSUM test and the Nyblom-Hansen test, as well as the recursive CUSUM test considered by Ploberger and Krämer (1992). All tests are available in the **strucchange** package (Zeileis, Leisch, Hornik and Kleiber, 2002, Zeileis, 2006a).

```
testpower <- function(nrep = 100, size = 0.05,
  test = c("Rec-CUSUM", "OLS-CUSUM", "Nyblom-Hansen"), ...)
{
  pval <- matrix(rep(NA, length(test) * nrep), ncol = length(test))
  colnames(pval) <- test
  for(i in 1:nrep) {
    dat <- dgp(...)
    compute_pval <- function(test) {
      test <- match.arg(test, c("Rec-CUSUM", "OLS-CUSUM", "Nyblom-Hansen"))
      switch(test,
      "Rec-CUSUM" = sctest(efp(y ~ x, data = dat, type = "Rec-CUSUM"))$p.value,
      "OLS-CUSUM" = sctest(efp(y ~ x, data = dat, type = "OLS-CUSUM"))$p.value,
      "Nyblom-Hansen" = sctest(gefp(y ~ x, data = dat, fit = lm),
        functional = meanL2BB)$p.value)
    }
    pval[i,] <- sapply(test, compute_pval)
  }
  return(colMeans(pval < size))
}
```

The function simply runs a loop of length `nrep`, computes p-values for the tests specified and finally computes the empirical power at level `size`.

In step (3), a wrapper function `simulation()` is defined that evaluates `testpower()` for all combinations of the simulation parameters, by default setting them to $b = 0, 2.5, 5, 7.5, 10$, $z^* = 0.25, 0.5$, $\psi = 0, 45, 90$ for $T = 100$ and $N = 100$ using only the OLS-based CUSUM test and the Nyblom-Hansen test. This is a coarser grid as compared to Ploberger and Krämer (1992) with fewer replications which are sufficient for illustration here. The function `simulation()` first expands the grid of all parameter combinations, then calls `testpower()` for each of them (which in turn calls `dgp()`) and then re-arranges the data in a data frame.

```
simulation <- function(intensity = seq(from = 0, to = 10, by = 2.5),
  timing = c(0.25, 0.5), angle = c(0, 45, 90),
  test = c("OLS-CUSUM", "Nyblom-Hansen"), ...)
{
  prs <- expand.grid(intensity = intensity, timing = timing, angle = angle)
  nprs <- nrow(prs)
  ntest <- length(test)

  pow <- matrix(rep(NA, ntest * nprs), ncol = ntest)
  for(i in 1:nprs) pow[i,] <- testpower(test = test, intensity =
    prs$intensity[i], timing = prs$timing[i], angle = prs$angle[i], ...)

  rval <- data.frame()
  rval <- for(i in 1:ntest) rval <- rbind(rval, prs)
  rval$test <- gl(ntest, nprs, labels = test)
  rval$power <- as.vector(pow)
  rval$timing <- factor(rval$timing)
  rval$angle <- factor(rval$angle)
  return(rval)
}
```

Modular tools are extremely valuable when setting out to reproduce portions of a simulation study. They convey clearly what steps were carried out, the DGP could be re-used for evaluating other structural change tests, or the tests could be re-used on new DGPs. With these tools available, the main simulation amounts to loading the **strucchange** package, setting a random seed and executing `simulation()`.[13]

```
R> library("strucchange")
R> set.seed(1090)
R> sc_sim <- simulation()
```

With a small simulation study as this one, we would generally only store the code and the outcome object `sc_sim`. For more complex simulations, it might make sense to store some intermediate results as well, e.g., the simulated data sets. Given the outcome `sc_sim`, it would then be easy for readers to replicate the table of simulated power curves, e.g. via

```
R> tab <- xtabs(power ~ intensity + test + angle + timing, data = sc_sim)
R> ftable(tab, row.vars = c("angle", "timing", "test"), col.vars = "intensity")
```

```
                           intensity    0  2.5    5  7.5   10
angle timing test
0     0.25   OLS-CUSUM               0.02 0.09 0.43 0.66 0.91
             Nyblom-Hansen           0.02 0.07 0.34 0.61 0.82
      0.5    OLS-CUSUM               0.01 0.20 0.58 0.86 0.98
             Nyblom-Hansen           0.05 0.18 0.58 0.84 0.96
45    0.25   OLS-CUSUM               0.03 0.02 0.17 0.37 0.47
             Nyblom-Hansen           0.05 0.13 0.31 0.57 0.72
      0.5    OLS-CUSUM               0.04 0.11 0.18 0.51 0.83
             Nyblom-Hansen           0.04 0.18 0.47 0.82 0.99
90    0.25   OLS-CUSUM               0.01 0.02 0.01 0.05 0.01
             Nyblom-Hansen           0.01 0.10 0.27 0.57 0.84
      0.5    OLS-CUSUM               0.02 0.08 0.05 0.03 0.00
             Nyblom-Hansen           0.06 0.20 0.53 0.84 0.99
```

which produces a flat table `ftable()` from the cross-tabulation `xtabs()`. The resulting table compares the power curves (over columns corresponding to intensity) for the two tests (nested last in the rows) given angle and timing. Instead of inspecting this table, the differences between the two tests can be brought out even more clearly by displaying the table in a trellis (or lattice) type layout, using the same nesting structure as above. In R, we use the **lattice** package (Sarkar, 2002) for producing Figure 1.

```
R> library("lattice")
R> xyplot(power ~ intensity | angle + timing, groups = ~ test,
+    data = sc_sim, type = "b")
```

This shows rather clearly that only for shifts in the intercept (i.e., angle 0), the OLS-based performs very slightly better than the Nyblom-Hansen test, but dramatically loses power with increasing angle, being completely insensitive to orthogonal changes in slope only (i.e., angle 90). The Nyblom-Hansen test, however, performs similar for all angles. Tests in the middle of the sampling period are easier to detect for both tests.[14] All results are roughly consistent with Table II(b) in Ploberger and Krämer (1992) although we have reduced many of the parameters for the sake of obtaining an almost interactive example. For obtaining a table with exactly the same setting as in the original study:

[13] The parameters have been chosen such that you can interactively run the code, possibly while grabbing a coffee (or another beverage of your choice) and the simulation results would be ready when you return.

[14] These findings are not simply artefacts of using $N = 100$ replications. The results from a larger, $N = 10,000$, experiment that we conducted reveal them even more clearly and additionally shows that the OLS-based CUSUM test is somewhat conservative while the Nyblom-Hansen test is not.

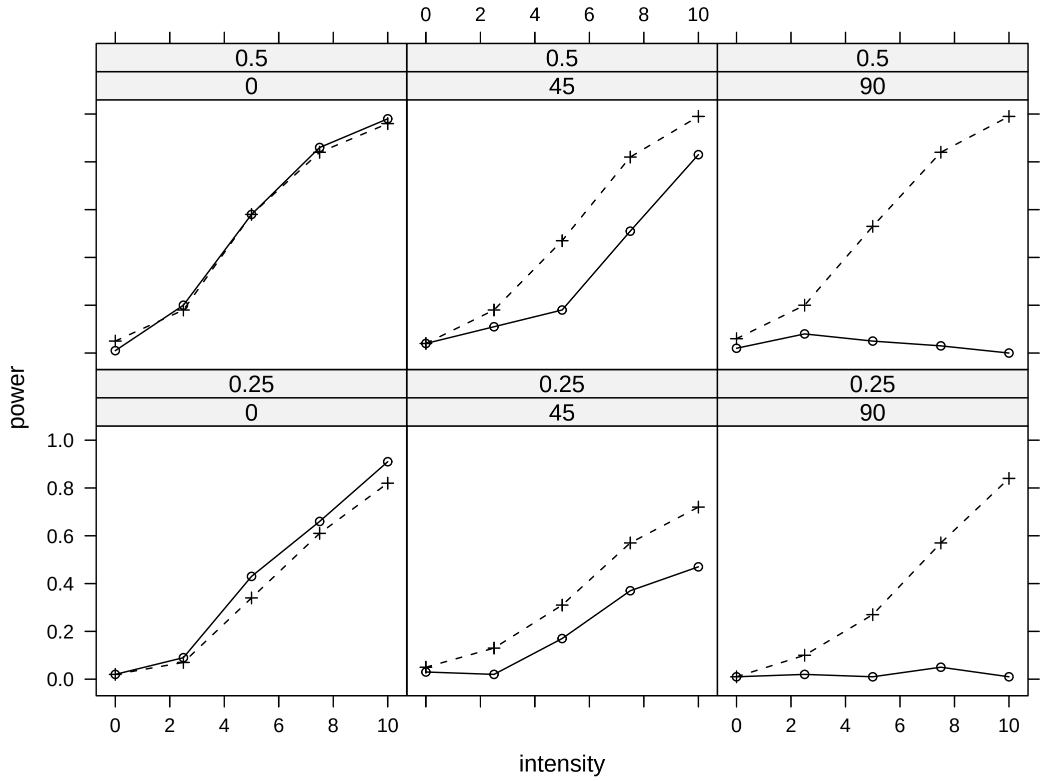

Figure 1: Simulated size and power for OLS-based CUSUM test (solid) and Nyblom-Hansen test (dashed)

```
R> set.seed(1090)
R> pk_sim <- simulation(nobs = 120, nrep = 100,
+    intensity = seq(4.8, 12, by = 2.4), timing = seq(0.1, 0.9, by = 0.2),
+    angle = seq(0, 90, by = 18), test = c("Rec-CUSUM", "OLS-CUSUM"))
R> ftable(xtabs(power ~ intensity + test + angle + timing, data = pk_sim),
+    row.vars = c("test", "timing", "intensity"), col.vars = "angle")
```

This code yields results that are not exactly identical (because we cannot obtain exactly the same random numbers, and `nrep = 1000` still allows for considerable variation), but they are clearly recognizable, leading to qualitatively equivalent conclusions.

4. CHALLENGES AND CONCLUSIONS

From an economic perspective, the real challenge of reproducible econometric research lies in restructuring incentives to encourage better archiving and distribution of the gory details of computationally oriented research. Technical progress in software and computer networking have dramatically lowered the cost of reproducibility, but without stronger incentives from journals and research funding agencies, further progress can be expected to be slow. The JAE is exemplary in this respect since it has strongly encouraged data/software archiving as well as replication studies. It would be excellent if other journals followed this lead. Authors ultimately need to be convinced that it is in their interest to provide detailed protocols for the computational aspects of their work. This may require a sea change in attitudes about acknowledgment and citation practices.

Further technical progress can be expected in all of the realms we have reviewed: more convenient archiving and version control, better tools for literate programming, improved algorithms and user interfaces for statistical computing are all under active development. More rapid diffusion of this new technology is what is really needed.

Web-based "electronic appendices" are increasingly common and this too is a welcome development. However, further pressure by the journals, a better understanding of the corresponding required quality standards by the scientific community, as well as simplified automatic access would be very valuable. We are still far away from the Claerbout Principle, but good models do exist. WaveLab of Buckheit and Donoho (1995) is a relatively early example, the concept of a *data compendium* suggested by Gentleman and Temple Lang (2007) is another. Within economics there has been some discussion and evaluation of the archival policies of the *American Economic Review* and *Journal of Money, Credit and Banking* (see McCullough and Vinod, 2003, McCullough, McGeary and Harrison, 2006), but much more is needed.

5. COMPUTATIONAL DETAILS

Our results were obtained using R 2.6.0—with the packages **lmtest** 0.9–21, **sandwich** 2.0–2, **MASS** 7.2–37, **strucchange** 1.3–2, and **lattice** 0.17–1—and were identical on various platforms including PCs running Debian GNU/Linux (with a 2.6.18 kernel) and Mac OS X, version 10.4.10. The full sources for this document (including data, R code and LaTeX sources) are available from http://www.econ.uiuc.edu/~roger/research/repro/. Hence, readers can do as we do, not as we say, and fully reproduce our analyses.

REFERENCES

Aptech Systems, Inc. 2006. *GAUSS Mathematical and Statistical System 8.0.* Aptech Systems, Inc., Black Diamond, Washington.
URL http://www.Aptech.com/

Baiocchi G. 2007. Reproducible research in computational economics: Guidelines, integrated approaches, and open source software. *Computational Economics* **30**: 19–40.

Baker RJ, Nelder JA. 1978. *The GLIM System, Release 3.* Numerical Algorithms Group, Oxford.

Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F. 2006. *Extensible Markup Language (XML) 1.0.* World Wide Web Consortium, fourth edition.
URL http://www.w3.org/TR/xml/

Buckheit J, Donoho DL. 1995. *Wavelets and Statistics*, chapter Wavelab and Reproducible Research. New York: Springer-Verlag.

Cederqvist et al P. 2006. *Version Management with* ***CVS***. Bristol: Network Theory Limited. Full book available online at http://ximbiot.com/cvs/manual/.

de Leeuw J. 2001. Reproducible research: The bottom line. Technical Report 2001031101, Department of Statistics Papers, University of California, Los Angeles.
URL http://repositories.cdlib.org/uclastat/papers/2001031101

Doornik JA. 2006. *Ox: An Object-Oriented Matrix Language.* London: Timberlake Consultants Press.

Durlauf SN, Johnson PA. 1995. Multiple regimes and cross-country growth behaviour. *Journal of Applied Econometrics* **10**: 365–384.

Gentleman R, Temple Lang D. 2007. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics* **16**: 1–23.

Goldberger AS. 2004. Econometric computing by hand. *Journal of Economic and Social Measurement* **29**: 115–117.

Hansen BE. 1992. Testing for parameter instability in linear models. *Journal of Policy Modeling* **14**: 517–533.

Heston A, Summers R, Aten B. 2002. Penn World Table version 6.1. URL http://pwt.econ.upenn.edu/. Center for International Comparisons at the University of Pennsylvania (CICUP), October 2002.

Kernighan B, Plauger P. 1974. *The Elements of Programming Style*. New York, NY: McGraw-Hill.

Knuth DE. 1984. *The TEXbook*, volume A of *Computers and Typesetting*. Reading, Massachusetts: Addison-Wesley.

Knuth DE. 1992. *Literate Programming*, volume 27 of *CSLI Lecture Notes*. Stanford, California: Center for the Study of Language and Information.

Koenker R. 1988. Asymptotic theory and econometric practice. *Journal of Applied Econometrics* **3**: 139–147.

Koenker R. 2006. The median is the message: Toward the Fréchet median. *Journal de la Société Française de Statistique* **147**: 61–64.

Kuhn M. 2006. `Sweave` and the open document format – The **odfWeave** package. *R News* **6**: 2–8.
URL http://CRAN.R-project.org/doc/Rnews/

Lamport L. 1994. *LATEX: A Document Preparation System*. Reading, Massachusetts: Addison-Wesley, 2nd edition.

Lecoutre E. 2003. The **R2HTML** package. *R News* **3**: 33–36.
URL http://CRAN.R-project.org/doc/Rnews/

Leisch F. 2002. Dynamic generation of statistical reports using literate data analysis. In Härdle W, Rönz B (eds.) *COMPSTAT 2002 – Proceedings in Computational Statistics*. Heidelberg: Physica Verlag, 575–580.

Leisch F, Rossini AJ. 2003. Reproducible statistical research. *Chance* **16**: 46–50.

Lenth RV, Højsgaard S. 2007. **SASweave**: Literate programming using SAS. *Journal of Statistical Software* **19**: 1–20.
URL http://www.jstatsoft.org/v19/i08/

MacKinnon JG. 2007. Journal of Applied Econometrics data archive. URL http://econ.queensu.ca/jae/.

Mankiw NG, Romer D, Weil DN. 1992. A contribution to the empirics of economic growth. *The Quarterly Journal of Economics* **107**: 407–437.

Masanjala WH, Papageorgiou C. 2004. The Solow model with CES technology: Nonlinearities and parameter heterogeneity. *Journal of Applied Econometrics* **19**: 171–201.

McCullagh P, Nelder JA. 1983. *Generalized Linear Models*. London: Chapman & Hall.

McCullough BD, McGeary KA, Harrison T. 2006. Lessons from the JMCB archive. *Journal of Money, Credit and Banking* **38**: 1093–1107.

McCullough BD, Vinod HD. 2003. Verifying the solution from a nonlinear solver: A case study. *American Economic Review* **93**: 873–892.

Murrell P. 2009. *Introduction to Data Technologies*. Boca Raton, Florida: Chapman & Hall/CRC. Forthcoming.

Nyblom J. 1989. Testing for the constancy of parameters over time. *Journal of the American Statistical Association* **84**: 223–230.

Ooms M, Doornik JA. 2006. Econometric software development: Past, present and future. *Statistica Neerlandica* **60**: 206–224.

Pilato CM, Collins-Sussman B, Fitzpatrick BW. 2004. *Version Control with **Subversion***. O'Reilly. Full book available online at http://svnbook.red-bean.com/.

Ploberger W, Krämer W. 1992. The CUSUM test with OLS residuals. *Econometrica* **60**: 271–285.

Raggett D, Hors AL, Jacobs I. 1999. *HTML 4.01 Specification*. World Wide Web Consortium. URL http://www.w3.org/TR/html401/

Ramsey N. 1994. Literate programming simplified. *IEEE Software* **11**: 97–105.

R Development Core Team. 2007. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL http://www.R-project.org/

Renfro CG. 2004. Econometric software: The first fifty years in perspective. *Journal of Economic and Social Measurement* **29**: 9–107.

Sarkar D. 2002. **lattice**: An implementation of trellis graphics in R. *R News* **2**: 19–23.
URL http://CRAN.R-project.org/doc/Rnews/

Schwab M, Karrenbach M, Claerbout J. 2000. Making scientific computations reproducible. *Computing in Science and Engineering* **2**: 61–67.

Snell JL. 1997. A conversation with Joe Doob. *Statistical Science* **12**: 301–311.

StataCorp. 2007. *Stata Statistical Software: Release 10*. StataCorp LP, College Station, Texas.
URL http://www.Stata.com/

Stigler SM. 1981. Gauss and the invention of least squares. *The Annals of Statistics* **9**: 465–474.

The MathWorks, Inc. 2007. *MATLAB – The Language of Technical Computing, Version 7.5*. The MathWorks, Inc., Natick, Massachusetts.
URL http://www.MathWorks.com/products/matlab/

Venables WN, Ripley BD. 2002. *Modern Applied Statistics with S*. New York: Springer-Verlag, 4th edition.

Wilson JH. 1973. Statistics and decision-making in government – Bradshaw revisited. *Journal of the Royal Statistics Society A* **136**: 1–20.

Zeileis A. 2004. Econometric computing with HC and HAC covariance matrix estimators. *Journal of Statistical Software* **11**: 1–17.
URL http://www.jstatsoft.org/v11/i10/

Zeileis A. 2006a. Implementing a class of structural change tests: An econometric computing approach. *Computational Statistics & Data Analysis* **50**: 2987–3008.

Zeileis A. 2006b. Object-oriented computation of sandwich estimators. *Journal of Statistical Software* **16**: 1–16.
URL http://www.jstatsoft.org/v16/i09/

Zeileis A, Hothorn T. 2002. Diagnostic checking in regression relationships. *R News* **2**: 7–10. URL http://CRAN.R-project.org/doc/Rnews/

Zeileis A, Kleiber C. 2005. Validating multiple structural change models – A case study. *Journal of Applied Econometrics* **20**: 685–690.

Zeileis A, Leisch F, Hornik K, Kleiber C. 2002. **strucchange**: An R package for testing for structural change in linear regression models. *Journal of Statistical Software* **7**: 1–38. URL http://www.jstatsoft.org/v07/i02/