On Reproducible Econometric Research

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, WU 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 two small replication exercises.

Keywords: reproducibility, replication, software, literate programming.

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 **?**. 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.

The question that we would like to address here is this: 200 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? **?** have defined what **?** 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." We view this as a desirable objective for econometric research. See **?** for further elaboration of this viewpoint.

The transition of econometrics from a handicraft industry (**??**) 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 econometric 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 (**?**). However, a more integrated approach

*Correspondence to: Achim Zeileis, Department of Statistics and Mathematics, WU Wirtschaftsuniversität Wien, Augasse 2–6, 1090 Wien, Austria. Tel: +43/1/31336-5053. Fax: +43/1/31336-774. E-mail: Achim.Zeileis@R-project.org

This is a preprint of an article published in *Journal of Applied Econometrics*, **24**(5), 833–847.
 doi:10.1002/jae.1083

encompassing data, software, empirical analysis, and documentation is often desirable to facilitate replication of results.

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 software development efforts, but it is only relatively recently that practical version control systems have emerged that are well adapted to econometric research. Many systems have grown out of the open-source software community where rigorous archive cataloging of development efforts involving many participants is absolutely essential. We briefly describe one such system, **Subversion** (SVN, http://Subversion.Tigris.org/, see **?**) that we have used for this project. Cross-platform compatibility is 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** (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 is very easy. Only if no SVN server is available, the setup requires some technical work. It is not dissimilar to setting up a Web server but can be a lot easier (see **?**, Chapter 6). Also, the individual researcher can often avoid setting up a server alone, especially if the university/department already provides such a server, typically along with other Web services. Some Web pages also host free SVN repositories, esepecially for software projects, e.g., http://SourceForge.net/ or http://Code.Google.com/, and http://R-Forge.R-project.org/ among others.

2.2 Data Technologies and Data Archiving

In the beginning data was flat, representable as matrices, and easily storable in text files. 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. 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 often ideal for reproducibility (and hence 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). Two popular open-source candidates, available across many platforms, are **PostgreSQL** (`http://www.PostgreSQL.org/`) and **MySQL** (`http://MySQL.com/`).

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, **?**): 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. Data formats based on open standards have a higher likelihood of remaining accessible in the future. Further discussion of data technologies for statistical/scientific applications may be found in **?**.

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 **?**, see also **?**. There is also an extensive literature on the individual merits and comparative performance of various forms of statistical software; see **?** 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 Structure 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). 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 **?** who emphasize the need to write programs that other *people* are capable of reading. 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 nonlinear models, and unified interfaces for general linear models constitute 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 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. 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 **?**. The modern trend toward functional languages and object-orientation is visible in several of the currently successful econometrics environments. MATLAB (http://www.MathWorks.com/), Ox (http://www.doornik.com/ox/) and R (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 sort of forum for sharing code, 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 about 1,700 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 several 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 (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 a 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. 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 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.

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. For commercial software prior versions are sometimes difficult to obtain due to licensing and distribution constraints. **?** describe an exercise in "forensic econometrics" exploring the evolution of the evaluation of the Gaussian distribution function in successive versions of GAUSS. Such investigations are obviously handicapped by the unavailability of older versions and lack of adequate documentation of algorithms and their evolution.

User Interfaces In the paragraphs above, we have focused on reproducibility using source code as typically used in programs with a command line interface (CLI). In practice, however, many analyses are carried out using a point-and-click approach based on a graphical user interface (GUI). Although easier to learn and often easier to apply, this procedure poses a challenge to reproducibility because it is much harder to reconstruct the "click history" for a certain analysis. A useful compromise are log files offered by several GUI-based systems that contain the source code associated with the point-and-click analysis. Thus, an author who wants to assure reproducibility, probably cannot avoid the contact with source code alltogether, but the transition to reading/writing source code is often made easier in this way.

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 **?** 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 **?**, still constitutes an ongoing development effort to build a higher level markup language on the foundation 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.

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 **?** 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 **?**, 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, **?** developed **noweb**, a set of open-source tools for combining code in arbitrary languages and LATEX documentation.

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 (**?**). 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. 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. Mathematica's concept of "notebooks" is an approach to documents of this type in which 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 **?** for R. Re-using the markup commands of **noweb**, it allows authors 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 '`JAE-RER.Rnw`' 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("JAE-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. This resulting LATEX file can then be processed to PDF by pdfLATEX. The choice of pdfLATEX is specific to our document. Many other flavours of `Sweave` are available, including LATEX, HTML (**R2HTML**, **?**), and ODF (**odfWeave**, **?**). There are also extensions to mixing SAS code with LATEX documentation (**?**).

3. REPLICATION CASE STUDIES

In this section, we describe briefly two 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 adopted 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/`. An extended version (**?**) with many more details is also available from the same locations.

3.1 An Empirical Example: Cross-Country Growth Regression

?, henceforth DJ investigate cross-country growth behavior based on an extended Solow model suggested in **?**, 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 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 GDP growth $\log(Y/L)_{1985} - \log(Y/L)_{1960}$ on initial GDP $\log(Y/L)_{1960}$ 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 results of MRW) and 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 Tables 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 them to the unit interval. Thus, the model employed by MRW and DJ (Table II, first column) 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)
```

This model can now be fit with R's linear model function `lm()` to the `dj` data set, producing the following regression summary:

```
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

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, $\bar{R}^2$, 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.

Table I: Replication of cross-country growth regressions, corresponding to models `dj_mrw`, `dj_sub1`, `dj_sub2` with dependent variable $\log(Y/L)_{1985} - \log(Y/L)_{1960}$. Conventional standard errors are used in the first column and sandwich standard errors in 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

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 sample sizes of 43 and 39, respectively, 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 consistent sample sizes, and also consistent model fits as we will see below.

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 `coeftest(dj_sub1, vcov = sandwich)`, 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 **??**) whereas conventional standard errors are used for the first MRW column. Results for `dj_sub2` can be obtained analogously. All R output is provided 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.

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. **?** describes a meta-analysis of published wage equations intended to explore how parametric dimension of econometric models depends upon sample size. 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.

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

$$\log \lambda \quad = \quad \underset{(0.149)}{1.336} \; + \; \underset{(0.017)}{0.235} \log z. \tag{1}$$

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 derivative 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 (**?**). 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> coeftest(rk1)

z test of coefficients:

            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.3362     0.1739    7.68  1.5e-14
log(z)        0.2353     0.0199   11.82  < 2e-16
```

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 **?**, 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. Recalculating 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))
```

reproduces the published results; see the first column of Table II for condensed output. 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 the parsity estimate to the specification of the functional form of the model, several other models were reported in **?**. 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 was $500,000$, but a yellowing page in the project file folder and the printed GLIM output in the same folder 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 (because the published parsity estimate is consistent with our replicated coefficient rather than the published coefficient). Mistakes like this could again be more easily avoided by better integration of text and data analysis.

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

```
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 **?** 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 (**?**).

```
R> rk_nb <- glm.nb(y ~ log(log(z)), data = rk, weights = 1/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. The results are again shown 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.

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

Table II: Replication of wage-equation meta-models, corresponding to `rk1`–`rk4` and `rk_nb` with dependent variable y. Deviance-based dispersion estimates are used 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	

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 **?** is a relatively early example, the concept of a *data compendium* suggested by **?** 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 **??**), but much more is needed.

5. COMPUTATIONAL DETAILS

Our results were obtained using R 2.14.0—with the packages **lmtest** 0.9-30, **sandwich** 2.2-9, and **MASS** 7.3-16—and were identical on various platforms including Debian GNU/Linux (with a 2.6.26 kernel) and Mac OS X, version 10.4.10. TEX Live 2007 was used for the original typesetting. 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.