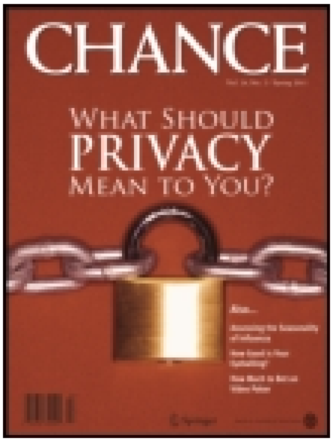


CHANCE
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucha20

On the Edge: Statistics & Computing

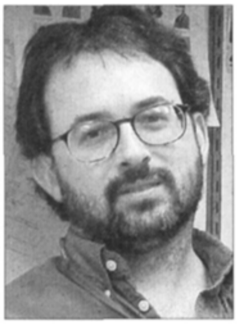
A. J. Rossini [a], Thomas Lumley [a] & Friedrich Leisch
[a] University of Washington, Box 357232, F-600 Health Sciences Building, 1705 N.E. Pacific, Seattle, WA, 98195-7232, USA

Published online: 20 Sep 2012.

To cite this article: A. J. Rossini, Thomas Lumley & Friedrich Leisch (2003) On the Edge: Statistics & Computing, CHANCE, 16:2, 41-45

To link to this article: http://dx.doi.org/10.1080/09332480.2003.10554848

ON THE EDGE: STATISTICS & COMPUTING



A. J. Rossini and Thomas Lumley, Column Editors

Reproducible Statistical Research

Friedrich Leisch and A. J. Rossini

Many recent results in statistical research are based on simulation or experiment-based procedures which have been facilitated by technological advances in computing (Beran 2001). While mathematical theory is still very important, these computational techniques, including Monte-Carlo, Markov Chain Monte-Carlo, and resampling methods, are increasingly used to obtain results which sometimes are more relevant than those based upon low-order approximations to asymptotic theory. These simulation-based techniques can help to fill gaps in understanding theoretical and mathematical procedures as well as provide numerical approximations to computationally infeasible exact solutions. This article will discuss a number of issues common to both statistical research and collaboration that impact the verification, understanding, and subsequent application of novel statistical procedures.

Complicated numerical algorithms must often be used even when we have sound theoretical results. Implementation of these procedures can be just as difficult as the construction of proofs. However, while publication of research papers is based on the verification or proper referencing of proofs for every theorem, there is a tendency to accept seemingly realistic computational results, as presented by figures and tables, without any proof of correctness. Yet, these results are critical for justifying the proposed methods and represent a substantial percentage of the content in many journal articles.

Computations can be proofed. Correctness can be verified by delivering appropriately documented and functional code, along with corresponding inputs and data, *for all results* along with the paper. Buckheit and Donoho (1995) define what deLeeuw (1996) calls *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.

The ability to document programming language code that implements mathematical thought is critical to understandable, explainable, and reproducible research. In particular, scientific progress can be either enhanced or hampered by the ability of others to understanding statistical analysis reports, since these results provide necessary evidence for drawing conclusions from many experiments. Hence, there is a strong need to develop, describe, and evaluate approaches, tools, and practices that support clarity and reproducibility of research.

Reproducibility applies both to the primary results, i.e. the numerical studies and data analyses that are used as supporting evidence for the methodology, as well as to the secondary results, which is the applicability of the methodology to other conditions. It is difficult to determine from a simulation study how results will apply to different underlying data models and under what violations of assumptions the results will break down. Failures can be the result of either the methodology or its implementation, and hence, this sensitivity analysis is important information for those doing applied research and who have to select appropriate methods to use for the analysis of their data sets! Proper interpretation of simulation studies which illustrate methods depends on selection of numerical linear algebra, pseudo-random number generation, and optimization algorithms. It appears to be a strange contradiction that while statisticians worry about the identifiability of important factors in the design of studies and

Column Editors: A. J. Rossini and Thomas Lumley are Assistant Professors of Biostatistics at the University of Washington, Box 357232, F-600 Health Sciences Building, 1705 N.E. Pacific, Seattle, WA 98195-7232; *rossini@u.washington.edu* or *tlumley@u.washington.edu*.

experiments, many research articles fail to provide sufficient information to identify if the numerical evidence for the proposed statistical method is due to the methodology or implementation. This blind faith in our colleagues' programming skills sometimes leaves us a bit perplexed.

We can look to the theory of modern documents, as popularized through the development of the worldwide web, to see what tools and approaches could be used to address these issues. The word "document," as used as a noun and verb, is required for successful research, and clearly, the phrase, "documents that document," characterizes scientific reports! Modern document processing takes advantage of tree-like representations of documents, and considers the components of these trees to be document objects. For example, in the context of HTML, the Document Object Model (DOM) specifies a basic set of tasks that one might want to perform on a document during its display; these tasks can be static ("display this text in **bold**") or dynamic ("if the mouse pointer is directly above this word or sentence, bring up a menu at a certain location on the screen").

Three capabilities provide examples for the scope of research activities: (1) the use of document mark-up for indexing, describing presentation of subsets of a document, and describing the content within various contexts, (2) documents including both static and dynamic nontext components, and (3) intentional document processing allowing for the resulting product to differ, depending on the media used for viewing. All these can be seen in the prototypical multimedia WWW page, that can be conditionally processed depending on the browser used for viewing.

Deep understanding of statistical results requires the knowledge of the experimental background, which includes statistical design, criteria for inferential testing, and parameter estimation. These activities often involve computing, ranging from finding reference and background material to data management to programming and computation. We discuss the requirements for reproducible statistical research, focusing on the verification and evaluation of computational statistical procedures through the reproduction and simple modification of content. This is followed by a brief introduction to modern document systems, and a description of some tools and practices that facilitate communication and reproducibility of results. Finally, we summarize the current state and where the near future might take us.

Requirements

The minimum requirements for providing sufficient material for others to reproduce computational results are:

1. use of a common computing environment,
2. access to the actual program code used to generate results,
3. access to analyzed data and other program inputs,
4. sufficient instructions and details to reproduce the exact results, especially if idiosyncratic approaches are taken.

These are definitely necessary, but clearly not sufficient for others to evaluate the correctness of numerical results. The program code and data are the minimum needed to check results by hand. But unless commonly understood programming languages are used, there is a requirement for those doing verification and evaluation, to approximately replicate the original computing environment on which the research was done.

In addition to the above, one could require further extensions such as using

1. universally available environments and the use of cross-platform open source tools, and
2. literate programming techniques as applied to statistics.

Statistical computing research tools such as R and XLisp-Stat are reasonably platform independent, providing evaluation environments that are close to universally available. Literate programming techniques such as literate statistical practice (Rossini 2001, Rossini and Leisch 2003) and tools such as RevWeb (Land and Wolf 1997–2001) and Sweave (Leisch 2002) can assist those doing the evaluation by providing clearer details of what is going on. Finally, reliable and solid open-source tools are important for opening up the details of the computational process in the event that questionable results occur with different inputs.

There are different notions of costs and barriers to reproducing results. We have stressed the elimination of those that require a direct large monetary investment in equipment and software, but have downplayed the corresponding need for the researcher or his computing support staff to spend time understanding, obtaining, using, and more importantly, trusting, non-commercial software. We are not advocating the elimination of commercial software or support but trying to point out that reliable tools exist that can allow universal dissemination of computational results.

Approaches and Tools: An Example

Two major issues for making statistical computations reproducible are tight integration of code and documentation, and reviving reports when data or analyses change, respectively. We list a few tools that are freely available and somewhat platform independent. Many other tools, including such commercial statistical software packages as Minitab, SAS or SPSS, feature some form of automated report-generating tools. Unfortunately, a full comparison of available technical solutions cannot be provided within the scope of this article.

As a simple example, consider bootstrapping the parameters of a linear regression model for measurements on the heart and body weight of 97 male cats (data from Fisher, 1947). Table 1 shows the result from fitting a linear regression model using body weight as the independent variable and heart weight as response: the regression line seems to pass through the origin (the intercept is not significant) and heart weight increases by approximately 4.3g per kg of body weight. The right part of Table 1 shows bootstrap estimates of bias and standard error of the estimates. Note that the bootstrap standard errors are 15–20% larger than the usual ones.

Even if one had access to the original data, there is insufficient information for reproducing Table 1. This example is taken from Canty (2002), but citing other publications is not always an option, especially if the original publication lacks sufficient details. We need information on:

1. What kind of bootstrap was performed? For linear regression models, resampling cases and resampling residuals are both popular bootstrap methods (among others).

2. Which software was used and how was it used?

For readers familiar with bootstrapping and R it might be sufficient to state that we used R version 1.6.1, the "boot" package, the Marsaglia-Multicarry random number generator with a seed of 123, and case resampling. For complex analyses, the actual programming code will be less clumsy than a description in words; this is similar to the use of formulas for expressing mathematical thought. The code can be an elegant way to describe the analysis, and we now will focus on tools to simplify the task of creating integrated reports which, as a side product, can make Table 1 reproducible.

Running the bootstrap with a different seed does not change results qualitatively, but the actual numbers change for almost all digits except the most significant ones. In the case of a difficult analysis, it might be possible to have qualitatively different results for different seeds.

Literate Programming and Statistical Practice

Literate Programming (Knuth 1992) describes an approach for combining code and documentation in a source file that can be woven into a description of the processes, algorithms, and results obtained from the system and tangled into the actual code. The goal of literate programming is to flexibly extend the notion of "pretty-printed code plus documentation" to allow for maximum documentation efficiency rather than compiler or interpreter efficiency. The two primary steps in constructing the results from the literate document are *tangling*, which produces unformatted code files that can be compiled or evaluated using a language interpreter such as R; and *weaving*, which produces formatted documentation files to be read by humans. Carey (2001) discusses how to utilize these tools for writing statistical software; we now show how the same concepts apply to statistical practice.

Returning to our bootstrap example, the simplest solution for reproducibility of results would be the verbatim inclusion of the code for the analysis in a technical appendix. Noweb (Ramsey 1994) provides tools for literate programming helping exactly with this task: inserting programming code into LaTeX or HTML documents. Some of the advantages of using Noweb rather than plain LaTeX verbatim environments are: support for pretty-printing of code, re-use of code segments in multiple places and automatic extraction of code into separate files.

Table 1—Bootstrapping the Coefficients of a Linear Regression Model for the Cats Data Using R = 999 Replicates

	Linear Model		Bootstrap	
	Estimate	Std.Error	Bias	Std.Error
Intercept	−1.1841	0.9983	0.0306	1.1434
Body weight	4.3127	0.3399	−0.0129	0.4033

A technical appendix containing the code for Table 1 in Noweb format (in its printed form) is given in Figure 1, where we could have placed arbitrary LaTeX documentation between the different code chunks. The real power of using literate programming tools like Noweb lies in flexible processing of files. Running "notangle-RcatsTab1.R" on the Noweb file shown above will extract the file catsTab1.R, which simply contains three of the four code chunks concatenated (because the last four lines define it that way). Sourcing the file catsTab1.R into R will reproduce Table 1; we do not need the code chunk named "view all results" for this. Hence it is omitted from the definition of cats Tab1.R.

For readers fluent in the S language, the code shown above is probably the most concise description of the analysis underlying Table 1. It shows which package was used for bootstrapping (boot), where to obtain the data (they are a standard example from package boot), the kind and seed of the random number generator and what kind of bootstrapping was done (case resampling). Note that none of the above relies on using R or the S language; the code could be a SAS macro.

Authoring Environments

Once all code for an analysis is contained in a report using Noweb or any other file format that allows for automatic extraction and hence evaluation of code, reviewing results is easy, because all figures and tables can be easily reproduced by an independent referee. The burden is on the authors of the documents to actually include the code in a way that gives others a handle for computations with the code. Most good textbooks on computational statistics "suggest" rather strongly to keep the code for each analysis for later reference; hence including it is not too much additional effort and enforces good statistical practice.

Emacs Speaks Statistics (ESS, Rossini et al. 2003) provides a unified interface to SAS, the S family of languages and several other statistical software packages. It also features support for writing Noweb files, such that the user can interactively send code to the statistical software package using the mouse (or a few keystrokes) while writing the file. Noweb files may contain code chunks in differ-

```
⟨setup⟩≡
  library(boot)
  data(catsM, package="boot")
  set.seed(123, kind="Marsaglia-Multicarry")

⟨standard and bootstrapped analysis⟩≡
  cats.lm <- lm(Hwt~Bwt, data=catsM)

  cats.fit <- function(d,i){
      coef(lm(Hwt~Bwt, data=d[i,]))
  }

  cats.case <- boot(catsM, cats.fit, R=999)

⟨view all results⟩≡
  summary(cats.lm)
  print(cats.case)

⟨collect interesting results into table⟩≡
  res <- cbind(summary(cats.lm)$coef[,1:2],
               Bias = colMeans(cats.case$t)-cats.case$t0,
               "Std. Error" = apply(cats.case$t, 2, sd))

⟨catsTab1.R⟩≡
   ⟨setup⟩
   ⟨standard and bootstrapped analysis⟩
   ⟨collect interesting results into table⟩
```

Figure 1. Printed noweb output from data analysis example.

ent languages, e.g., SAS macros and S code. Writing documents that contain reproducible analyses is definitely more a matter of discipline than effort or availability of tools supporting the process.

Revivable Documents

So far we have discussed only utilities that allow for easy extraction and evaluation of code for a finished analysis. Recent tools operating on files in Noweb format allow us to go one step further: including code for figures and tables instead of the actual graphs and numbers. Code for the analysis is embedded into a manuscript, which is then evaluated, and both code and/or the corresponding output go into the final document. The source document contains no numerical or graphical *results*; it contains only the *code* needed to obtain them. Such documents are revivable in the sense that they can be automatically updated whenever data or analysis change.

The RevWeb system (Lang and Wolf 1997–2001) was the first approach for using literate programming techniques to construct revivable statistical documents. It uses Noweb and provides an interface to S-PLUS and R. Recent extensions to RevWeb use R's Tcl/TK interface as a GUI. The main focus of RevWeb is to give the user an interactive handle on the code chunks contained in a document.

Sweave (Leisch 2002) is another approach for generation of dynamic statistical reports. Like RevWeb, it mixes S and LaTeX in a sequence of code and documentation chunks, and by default it also uses the Noweb syntax for separating the chunks. The focus of Sweave is on report generation rather than use as a live document. It uses S itself for all tangling and weaving steps and hence has very fine control over the S output. Options that can be set either globally to modify the default behavior or separately for each code chunk control how the the output of the code chunks is inserted into the LaTeX file. This applies both to textual output as well as graphics.

Returning to our bootstrap example: A Sweave source file for the analysis does not contain any of the numbers shown in Table 1, it contains only the code from the previous section "Literate Programming and Statistical Practice." The actual LaTeX table is created on the fly, e.g., using package xtable (which supports formatting S objects in LaTeX syntax). Again, stand-alone code can be extracted using a tangling step.

Discussion

A minimal fulfillment of the requirements specified is to bundle a report together with data and code in an archive file (zip, tar, ...), or publish data and code on a web page and include the url in the report. Many books and journals use the web to provide online complements that can contain supplemental text and figures, example code, web-based applications and services, and data sets. Reproducing results when data, code and text are delivered in separate files—even if contained in the same archive—usually requires manual intervention.

If reproduction of results is to become common practice and a standard part of the scientific review processes, automation will be critical to reduce the burden on reviewers and assist with timely response. The term *review* definitely applies to the formal peer review process of a journal, but can also apply to colleagues at the same department or for communicating results between the authors of a single document. Who has not asked herself or himself at least once: "How did I do that?" when looking at poorly documented results from an analysis done in the past?

Respecting Claerbout's principle *over long periods of time* is a major challenge of the future for all computational sciences, including statistics. For shorter periods of time, technical solutions exist that help in making results from computational statistics reproducible. The proper choice of tools can assist with the generation of reproducible results, but what we lack most importantly are well-documented and open standards to flexibly automate this process.

References

Beran, R. (2001), "The Role of Experimental Statistics." URL *www.stat.ucdavis.edu/~beran*. Department of Statistics, UC Davis, CA, USA.

Buckheit, J.B. and Donoho, D. L. (1995), "Wavelab and Reproducible Research." Technical report, Stanford University Statistics Department.

Canty, A. J. "Resampling methods in r: The boot package." *R News*, 2(3): 2–7, December 2002. URL CRAN. *R-project.org/doc/Rnews*.

Carey, V. J. (2001), "Literate Statistical Programming: Concepts and Tools." *Chance*, 14(3):46–50.

de Leeuw, J. (1996), "Reproducible Research: The Bottom Line." Technical Report 301, UCLA Statistics Department.

Fisher, R.A. (1947), "The Analysis of Covariance Method for the Relation Between a Part and the Whole." *Biometrics*, 3:65–68.

Knuth, D.E. (1992), "Literate Programming," CSLI Lecture Notes, 27, Center for the Study of Language and Information.

Lang, L. and Wolf, H.P. (1997–2001), *The REVWEB Manual for Splus in Windows*. Uni.Bielefeld, Germany. URL *www.wiwi.uni-bielefeld.de/StatCompSci/software/revweb/revweb.html*.

Leisch, F. (2002), Sweave: "Dynamic Generation of Statistical Reports Using Literate Data Analysis." In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002—Proceedings in Computational Statistics*, 575–580.

Ramsey, N. (1994), Literate programming simplified. *IEEE Software*, 11(5): 97–105, September.

Rossini, A.J. (2001), "Literate statistical practice." In Kurt Hornik and Friedrich Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*. Technische Universität Wien, Vienna, Austria. URL *www.ci.tuwien.ac.at/Conferences/DSC.html*.

Rossini, A.J., Heiberger, R. M., Sparapani, R., Maechler, M., and Hornik, K., "Emacs Speaks Statistics." *Journal of Computational and Graphical Statistics*. (In Press).

Rossini, A.J and F. Leisch. (2003), "Literate Statistical Practice." Technical report #194, University of Washington Biostatistics. URL *www.bepress.com/uwbiostat/paper194/*