

*UW Biostatistics Working  
Paper Series*  
University of Washington

---

*Year 2003*

*Paper 194*

---

Literate Statistical Practice

Anthony Rossini\*

Friedrich Leisch<sup>†</sup>

\*University of Washington

<sup>†</sup>Institut für Statistik und Wahrscheinlichkeitstheorie, Technische Universität Wien

This working paper site is hosted by The Berkeley Electronic Press (bepress).

<http://www.bepress.com/uwbiostat/paper194>

Copyright ©2003 by the authors.

# Literate Statistical Practice

## **Abstract**

Literate Statistical Practice (LSP, Rossini, 2001) describes an approach for creating self-documenting statistical results. It applies literate programming (Knuth, 1992) and related techniques in a natural fashion to the practice of statistics. In particular, documentation, specification, and descriptions of results are written concurrently with writing and evaluation of statistical programs. We discuss how and where LSP can be integrated into practice and illustrate this with an example derived from an actual statistical consulting project. The approach is simplified through the use of a comprehensive, open source toolset incorporating Noweb, Emacs Speaks Statistics (ESS), Sweave (Ramsey, 1994; Rossini, et al, 2002; Leisch, 2002; Ihaka and Gentleman, 1996). We conclude with an assessment of LSP for the construction of reproducible, auditable, and comprehensible statistical analyses.

# 1 Introduction

Progress in research activities requires understanding statistical analyses. These results, when placed in context, provide evidence to support conclusions drawn from studies and experiments. Deep understanding of the results requires knowledge of the experimental background. This information, which is not often conveyed in detail, includes statistical design, visual and numerical summaries of the data, criteria and results for inferential testing and parameter estimation. We describe an approach which provides this information and allows for auditing and reproduction of the results.

The ability to document programming language code, mathematical activity, and statistical thought is critical for the construction of understandable, explainable, and reproducible data analysis. We will refer to the activities involved in statistical research and data analysis as *statistical practice*. These activities often involve some form of computing, ranging from finding reference, background, and annotation material to data management to programming and actual computation. Our goal is to improve practice, with improvement defined by an increase in the level of comprehension by others of what occurred during the statistical task.

Literate Statistical Practice (LSP) describes the application of Literate Programming to statistical research. This approach stresses the simultaneous documentation of process, workflow, and generation of results, for such activities as data management and statistical analysis. This is accomplished by weaving statistical programming code with both documentation for the code as well as documentation for the results. The resulting document presents the approach for constructing a cleaned, working dataset; describes code used for descriptive, exploratory, and confirmatory analyses; and discusses methods used in the final analysis as well as unsuccessful approaches which have been attempted during the analysis. This document can then be conditionally processed to address the needs and requirements of different targeted audiences (consulting report, internal auditable report, source code for data analysis, dataset codebook, etc).

One example that is naturally enhanced through a literate approach is statistical consulting. Statistical consulting is characterized by a short, usually incomplete presentation by the client of the scientific and statistical issues, a set of questions that the client would like addressed, and finally, the proposal of a simple and valid solution by the consultant. Information and knowledge are divided asymmetrically between consultant and client; thus, a single project report fails to

provide optimal information for either. Two natural styles among the many that can be employed for statistical consulting are a *Journal* and a *Project* style. Using either approach, the final source documents will consist of the original dataset, possible intermediate datasets and analyses, as well as the literate statistical document.

The *Journal* style uses a narrative style to describe information related between the client and consultant, from the consultants perspective, over time. This details responses by visit, so that other than an introduction and summary, each visit is a short section or chapter containing code and results to be discussed (if this is a later visit), notes from the discussion during the visit, and any code, data, or results processed to address questions raised that might not be discussed again.

The *Project* style provides steps at the top-level the activities, and within activity sections documents temporal changes, rather than placing temporal changes as the top-level component. The flow in this style starts by introducing the consulting relationship and describes the scientific issues and design considerations. Each visit can be described in a subsection of this introduction section, with the statistical issues, as understood by the statistical consultant, contained in parallel subsections in a different part of the document. We provide an example of this style in Appendix B, which is an edited version taken from the early stage of a real consulting relationship. An obvious extension to both of these two styles is to allow the client to edit and augment sections of the document as it is being constructed.

Well-documented and comprehensible statistical analyses are critical for research. The rest of this paper focuses on the use of literate programming techniques by statisticians for daily work. We first describe what literacy means in this context. We follow this by a discussion of styles and forms. A coherent tool-set, using Emacs Speaks Statistics (Rossini et al., 2003), Noweb (Ramsey, 1994), and Sweave (Leisch, 2002) for implementing literacy is presented, and we conclude with future developments.

## 2 Literate Statistics

Statistics has been referred to as *programming with data* (Chambers, 1998). Statistical programming is unlike traditional application programming, the results of statistical computations based on different inputs (or datasets) have equal weight to the design, performance, and correctness of

the computer program. More importantly, communication of the results is a critical component of the analysis, as well as communication of the code for audit, reproduction, and comprehension of the details behind the statistical analysis.

## 2.1 One Project, Different Views

Consider the consulting example; a single document is constructed which, at the highest level, describes the current understanding of the presented research problem. Sections of the document describe the consulting session discussions, as well as what solutions were proposed, tried, and finally chosen as appropriate. While the entire document contains everything necessary to reconstruct the process, different components will be useful for different audiences. For example, a statistically naive consultee will prefer summary materials while another statistician who is following up on the project will prefer technical details of the code. A third possible audience, a statistician auditing the work, will prefer a summary and overview, but probably would not require technical programming details nor basic fundamental descriptions of common statistical methods.

One goal for statistical literature is to present, using multiple scales and selective processing, the **relevant** details behind the work. This should not just be coding with comments, but mix code and documentation of both code and results on appropriate inputs so as to produce a number of forms of documents: computer program files, data files and data code books, books or articles of multiple forms. The definition of “relevant” clearly depends on the target audience so conditional processing of the document is important. Since results could be intended for statisticians, researchers, consultants, collaborators, or others, the macro-level details as well as the micro-level details should all be available for the reader. Making this simple is the difficult but necessary task of the tools and user-interface chosen, which will be described later in Section 4.

## 2.2 Modern Electronic Documents

The primary motivation for Literate Statistical Practice arises from current theories for electronic documents. These approaches use markup language to place the text into context for the purposes of presentation, referencing, and possibly dynamic action. Markup languages, such as  $\text{\TeX}$ , RTF, HTML, XML, SGML and their derivatives, are used to include context for content. Usually the context is related to presentation; however, there are other contexts in which the

document can be placed, including indexing, mapping to file systems or other databases, or tools for evaluation.

Modern document processing takes advantage of tree-like representations of documents, and considers the components of these trees to be document objects upon which to process. The Document Object Model (DOM, [W3C, 2000](#)), is one example of a general programming interface to documents. It specifies a basic set of tasks that one might want to perform on a document for display presentation. These tasks can be static (“display this text in **bold**”) or dynamic (“if a mouse point is directly above this word or sentence, bring up a menu at a certain location on the screen”). We consider the initial construction of such an interface focused on documents whose intended purpose is to describe statistical research and results.

Literate statistics discourages the construction of many little unconnected pieces. It focuses on producing a representation of the whole data analysis project as a tree, with leaves for data, code, and documentation. Some leaves like the description of the problem are more or less static, while others are dynamic: all data analysis is represented by the respective code and can be redone whenever needed. Using the consulting example again, some branches of the tree like all the fine technical details will only be of interest to the consultant. A non-technical description of the methodology in another branch of the tree is mainly targeted for the client, while the branch containing the main results is of interest to both.

Markup languages are processed by translators such as  $\text{\TeX}$ , World-Wide-Web (WWW) browsers, Microsoft Word, and others, to produce the desired output such as electronic or paper-based presentation or indices. Literate statistical documents are processed in a similar way, with different processors for different kinds of leaves: data may be read from a relational database management system using SQL, data analysis will be performed by a statistical data analysis environment and text by one of the aforementioned markup translators.

### 3 Literary Styles and Forms

Fictional literature takes many forms such as poetry, short story, or novel, with every piece following a particular style chosen by the author. Likewise, literary statistical analyses have both form and style. While there are a relatively few number of forms that might be used, there are many possible literary styles for each form that can be applied to a particular project. The

chosen style should reflect both the statistical practitioner's ideals as well as the target audiences for the analyses.

Literate programming assumes a minimal level of programming competence on the part of the reader, mostly in terms of basic computer programming workflow. On the other hand, literate statistical practice can hide technical details and present the summaries, in such a way that it is clear how the author can reproduce the exact results. The many realizations of a particular statistical document can target other statisticians, scientists with varying degrees of comfort with statistical analyses, and students who are actively learning the desired material.

There are many other sensible styles for literate statistical practice, and within a literary style, there is much flexibility in terms of presentation. However, common sense should suggest that content such as figures and plots should have the corresponding code-chunk accessible if not actually nearby, and that while data management can be moved to an appendix, the actual code for reconstructing the dataset from a copy of the original is very critical for documenting the analysis process.

Poorly written literature exists. It is important to recognize that the existence of bad literature doesn't mean that all literature is bad. Reasons for bad statistical literature include poor communication, problems with proposed statistical approaches, and failure to describe (or understand) the design and analysis plan appropriately. Written literature is more difficult than note-taking; the onus of organization is placed on the statistician. This creates more work at the beginning of the task but prevents the worst case scenario of being unable to recall or reproduce critical components of the data analysis workflow. It is unfortunately not an uncommon event to forget reasons for why observations are removed from the data set or recoded, or scientific assumptions for choosing statistical procedures!

We present several example literary forms. These forms represent different intentions and target diverse audiences, but the general principle of providing readable examples and documentation remains. Each form can be realized through a number of different styles, the selection of which is based on the particular situation addressed by the document.

### 3.1 Consulting

The *Consultant's Form* was initially presented in section 1. Its primary intent is to describe a short consulting project with the end goal to provide small and simple analyses or design

calculations for the consultation session. Our preferred style, which was presented as the *Project* style, sections the various components of a short-term consulting project as follows:

1. introduction to the problem,
2. planned approach,
3. data management,
4. descriptive statistics,
5. inferential statistics,
6. conclusions and lessons for the analyst,
7. the concluding report for the consulting client.
8. appendices describing issues that arose and should be recorded, including data management and quality control and assurance.

The document can contain pointers and links to the dataset, codebook, and copies of client-supplied background material not incorporated in the introduction. The general tone is that of a scholarly report, and parts of it are intended to be used in writing papers, reports, and grants on the subject.

### 3.2 Institutional Memory

The *Institutional Memory* form documents, within a lab or research group, how statistics and computational mathematics are explicitly computed, as well as how the corresponding generated evidence is assessed. Since the simplest approach to describing mathematics is not the most effective form for computing the same mathematics. This becomes an issue for long-term research programs, especially when a researcher has new colleagues join an ongoing project. One obvious example is the initial training for new students of a professor who are to be working in an area where the groundwork for the approach has been outlined previously as a possible extension.

The requirements placed on styles for this documentation form should highlight both past wisdom and folly in an attempt to learn from history and minimize the number of times mistakes are repeated. The document format will be longer and contain several articles as different



chapters, using a book or report format, to describe individual work by collaborating researchers working on different pieces of the research. A key example of this is the work on Reproducible Research by the Stanford Exploratory Project group in the Stanford Geophysics department (Schwab et al., 1996).

### 3.3 Clinical Trials

Clinical trials are a classic setting where documentation of all project aspects is critical. This includes all decisions, data management, and statistical analyses. In particular, government licensure of medical therapies, prophylactics, and devices requires it.

The common requirements for this form’s literary styles include sections describing the analysis plan, the protocol’s statistics section, and design issues in the concept or final protocol documents. Many different people with completely different background and computing skills are involved in the process, including medical doctors, laboratory scientists, and statisticians. This makes coherent documentation of a research project a challenging task. Without coherent documentation of all steps, precise understanding of results will be impossible.

### 3.4 Instruction

This example form can also be referred to as the *Classroom Form*. It applies to documents such as lecture notes, software manuals or student’s homework. In all cases, the author of the document wants to demonstrate how to perform a certain statistical analysis, while at the end, the reader should understand how to write code which obtains similar results. Electronic versions of examples allow the students to repeat and modify the examples, simplifying the task of evaluating the performance and behavior of complex statistical procedures. By allowing students to return their homework results as modifications of the original literate documents, they learn literate statistical practice on the fly. One positive side effect is that it is much easier for the teacher to verify results.

Software manuals also can employ this approach, as seen in the BioConductor vignettes (available from <http://www.bioconductor.org>). Even for mature statistical methods, synchronizing software and the related documentation is non-trivial. Printed material—especially books—tend to have a much longer lifespan than software, which often changes on a rapid pace. This is even more the case for emerging fields like bioinformatics, (e.g., microarray data analysis): statistical

methodology, corresponding software and documentation are changing permanently. Vignettes are short software manuals where all enclosed code is automatically checked regularly against the current implementation through the results of evaluation.

## 4 Tools

Tools are critical to literate statistical practice, and should be minimally intrusive to the statistician. One of the hardest tasks known is to convince an adequate computer user to switch from a preferred computing environment to a new one. Users are quite hesitant, usually with good reason, to learn new ways. This can be seen in the common *editor wars*, found on nearly every mailing list with a computer-related topic, which are pointless attempts to convince people to switch to a different set of tools.

Integrated Development Environments (IDEs) combine features and tools in a single interface in an attempt to increase programmer productivity and efficiency. The increased speed in commercial software development time over the last decade can be partially attributed to the use of IDEs and similar Rapid Application Development (RAD) tools and methodologies. In the field of statistics, programming is an important skill which can be augmented and enhanced by the right tools and environment. This is especially true with the rise of computational analysis tools such as resampling methods (including bootstrap, jackknife approaches) as well as Markov Chain Monte Carlo (MCMC) sampling (Gibbs, Metropolis–Hastings algorithms).

We now describe one set of open source tools which provides a start for an IDE for literate statistical practice. The advantages of this set include platform independence, features which are focused on statistical computing and presentation. The components, which consist of a file format and processor (Noweb), an enhanced editing environment (ESS), and a report processor (Sweave), are detailed in the next few sections.

### 4.1 File format: Noweb

Noweb (Ramsey, 1994) was probably the first literate programming tool commonly employed for statistical practice (Carey, 2001; Rossini, 2001). This language-independent literate programming tool specifies a file format for mixing code and documentation, along with markers for constructing indices and references. The Noweb program weaves the document into an article

for viewing or tangles the document into source code for further processing using compilers and interpreters.

A Noweb document is divided into code chunks and document chunks. The code chunks are literally pieces of a program, while the documentation chunks consist of documentation using markup in either HTML or  $\text{\LaTeX}$ . Documentation chunks are pertinent to nearby code chunks, while code chunks form named, indexable, and reusable components for computer programs. The primary advantage is that code can be ordered to enhance presentation rather than located near those chunks that it must lie next to in the final source file.

The Noweb processor takes this file and tangles it into source files for compilation or interactive evaluation, or weaves both the computer code and the documentation into a document in some readable form; for example, HTML, PostScript, and PDF are all possible forms for readable output.

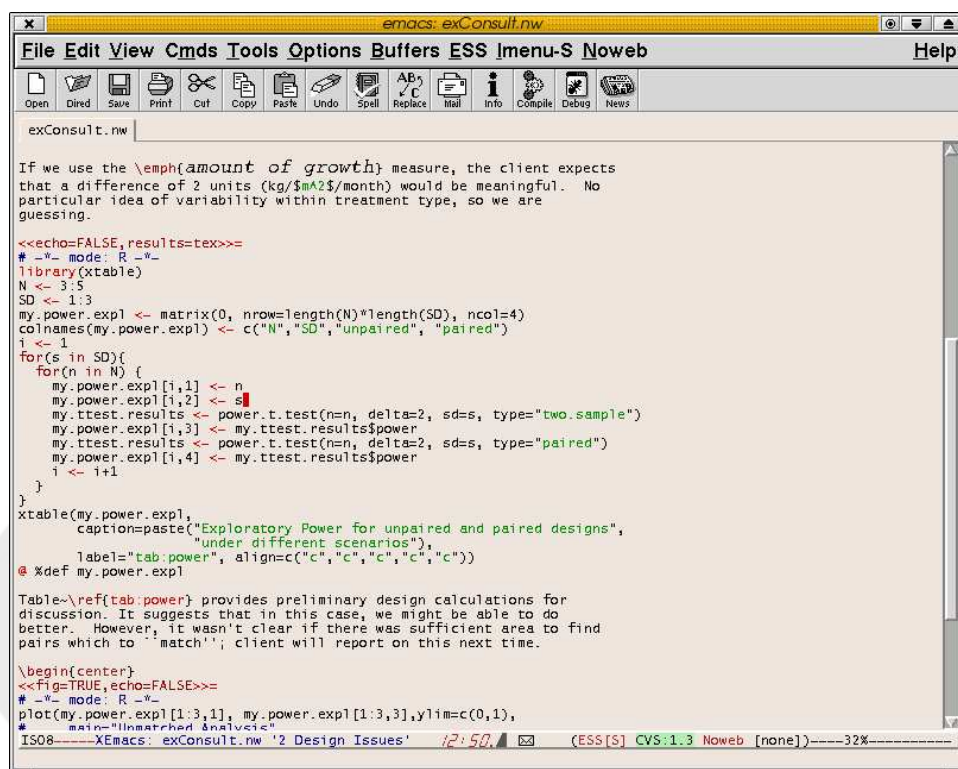


Figure 1: Consulting Example: Noweb File as viewed in ESS

We primarily use the file format in this work, but it is also possible to take advantage of

Noweb's indexing system for navigating the components of the document. We then construct tools that understand the basic components of the Noweb file. Emacs Speaks Statistics (ESS) provides an editing and development environment for constructing and evaluating literate documents. The editor is enhanced to monitor the location of the cursor in the file and use either ESS as the editing mode for statistical coding and processing, or a  $\text{\LaTeX}$  or similarly appropriate mode for documentation editing. Sweave, described more fully in Section 4.3, uses the format and a processing program to generate literate reports, with the current focus on those which use S for statistical analysis.

## 4.2 Interactive: Emacs Speaks Statistics

Emacs Speaks Statistics (ESS, [Rossini et al., 2003](#)) currently supports a number of interactive statistical programs as well as a few interpreted statistical languages, including the S family of languages (recent versions of S, S-PLUS, and R), SAS, STATA and XLISPSTAT (including extensions, ARC and ViSta). There are various levels of support depending on the capabilities of the program and the needs of users. Because ESS builds on the extremely powerful Emacs editing capabilities and the Emacs ability to communicate directly with a running process, ESS provides very powerful and uniform interaction with the statistical programs and languages.

ESS supports LSP through specialized support for Noweb documents. Our work on ESS focused on enhancing the editing interface for Noweb documents. The editor is used to call the appropriate statistical software system, allowing the interface to remain language neutral for both the statistical language as well as the documentation language. A single document allows only one (of many possible) documentation languages for a particular document, but provides support for numerous programming languages. The ESS-Noweb interface extends ESS by

- providing a coding environment enhanced for the production of Noweb documents
- adding options for submitting either a code chunk (including any or all embedded code chunks) to the running statistical process
- adding options to submit a thread of chunks to the running statistical process

The form of documentation can be satisfied through the use of  $\text{\LaTeX}$ , SGML, HTML, or XML markup, which can then be processed into a human readable document, or used for construction of statistical reports using Sweave, as discussed in the next section.

### 4.3 Reports: Sweave

Sweave (Leisch, 2002) is tool for constructing literate reports using the R statistical language (R Development Core Team, 2002). It is an integral part of Literate Statistical Practice, focusing on report generation rather than use as a live document. Like the RevWeb system (Lang and Wolf, 1997–2001) it mixes S and L<sup>A</sup>T<sub>E</sub>X in a sequence of code and documentation chunks, and by default, it also uses the Noweb syntax for separating the chunks.

Sweave performs all tangling and weaving steps using the S language; this allows for fine-grained control over the resulting output. Options can be set either globally to modify the default behavior or separately for each code chunk determine how the output of the code chunks is inserted into the L<sup>A</sup>T<sub>E</sub>X file, that is, one can hide or show S input code lines; include text output verbatim, as L<sup>A</sup>T<sub>E</sub>X code or hide it completely; emulate an interactive R session: values of assignments are not printed, values of single objects are printed; suppress evaluation of single code chunks; write output to separate files for each code chunk; generate EPS and PDF files of graphics which can be re-inserted into the document through the use of `\input{}` or `\includegraphics{}` statements. In addition the user can specify hook functions which are automatically executed before code chunks of certain types, e.g., to set graphical parameters.

These features allow to write L<sup>A</sup>T<sub>E</sub>X documents which contain the S code for all figures and tables instead of static numbers and graphic files. Any editor can be used for writing Sweave documents, although ESS provides an enhanced authoring environment. Together with ESS the user can easily turn the transcript of an R session into a literate document with the full power of L<sup>A</sup>T<sub>E</sub>X for text processing available in a natural way.

T<sub>E</sub>X is not only a powerful markup language for word processing, but can also be used like a programming language. E.g., `\if... \else... \fi` constructs allow for conditional processing of documents. L<sup>A</sup>T<sub>E</sub>X also has an `ifthen` style package which provides higher level conditional processing. These tools allow the document author to reuse components by maintaining the different views of a literate statistical document within a single file and processing it conditionally for each target audience.

## 5 Discussion

Literate Statistical Practice has many benefits for both practitioners and their colleagues. The results are completely contained in a thorough and presentable document of the thought process involved. Second, there is an obvious approach for reproducing the results. Finally, a history of work on the data is provided which can eliminate repeating approaches which became dead ends. While tools for LSP exist, they need further evaluation and development to reduce the burden placed on the statistical practitioner.

We can apply the same tools to methodological statistical research. Literate Programming has been used to document statistical methods simultaneously with implementation (Lumley, 1998). This enhances the value of the research by lowering the difficulty in exactly reproducing analytic approaches. This makes sensitivity and performance evaluations of statistical procedures much easier, as well as forcing the original authors' results to be reproducible. Thus, the results of the statistical literary document form a template for subsequent applications of the described methodology. The only twist from what has been described so far is a possibly small shift in focus towards explaining the methods, including one or all of theory, computation, and implementation, rather than on the data analysis.

We now summarize some of the issues that have been discussed.

### 5.1 Data Analysis

The best or most appropriate analysis of a given data set is rarely known in advance. Hypotheses are often built by exploring the data set through both descriptive and inferential statistical procedures, trying out several possible models, and finally choosing the one that fit the data and questions of interest the best. Thus, the true workflow path consists of a feedback loop of several failed approaches and their results leading to the final model for the data. While the final project report will only contain an optimized (false) workflow path, it is important to also keep notes on unsuccessful approaches in order to avoid repeating mistakes in the future.

To make LSP feasible it must be tightly integrated with the analyst's preferred data analysis tool, such that writing the literate document is not a burden, but a natural part of data analysis. If results need to be copied manually to a report, then every single update of the report needs manual intervention and this will tempt the analyst to do it only once "when the project is

finished”, and keep intermediate results more or less undocumented, if at all.

The literate document should contain a *repeatable* description of the whole data analysis project, and allow for simple extraction of reports and code for:

1. results
2. failed experiments/approaches
3. true workflow path of analysis
4. optimized (false) workflow path

The hardest part to integrate into such a document is exploratory data analysis, especially if interactive visualization methods are used. While numerical results can in almost all cases be adequately described by code in a computer programming language, this will most likely fail for interactive graphics. Even if we consider interactive dynamic documents, replaying a grand tour or a sequence of brushing operations is not necessarily the optimal solution. Storing only the highlights and drawn conclusions may be sufficient, though.

## 5.2 Reporting

All literary styles discussed above involve mixing documentation, numerical and graphical results from data analysis and the code to obtain these results. Under the LSP paradigm there is no single “final report” for a project, but a literate document for which multiple views can be generated.

Creation of a view of the literate document consists of the following steps:

1. determine which parts of the literate document to include (at which granularity)
2. optionally re-run the analysis and create all tables and figures for the view
3. generate the view and render it on the medium of choice (paper printout, web page, ...)

Creation and inclusion of figures is comparably easy, because most statistical software packages can store them in separate files using common graphics formats like JPEG or EPS, which in turn can easily be included by word processing markup languages.

Integration of numerical results and documentation is both difficult and important. For the institutional memory form, publication-quality typesetting is not necessary, but it often can

considerably enhances readability. For the consultant's form, publication-quality typesetting is often be a requirement. While the weaving steps of literate programming produce sequences of code and documentation chunks, LSP goes one step further: we need to be able to include both code *and* the resulting output.

A simple solution is to include textual output of the data analysis directly and without any additional markup. For the instructional form, such as for software manuals, this is the layout of choice since it directly emulates what the user would see on screen. However, more sophisticated views require tight integration of text and analysis through the use of translators from statistical objects like a matrix or a fitted linear model to text markup.

### 5.3 Reproducible Research

Technological advances in the last decades drive statistics beyond mathematics towards experimentally supported information science (Beran, 2001). Of course we want to gain as much theoretical/mathematical understanding as possible, but are increasingly filling gaps in theory by experiments. Even where we have sound theoretical results complicated numerical algorithms are used. And some statistical methods such as the bootstrap and Markov Chain Monte Carlo, which use stochastic simulation approaches, are experimental by nature.

Although we currently have to accompany every theorem in a paper by either proof or proper references, computational results (figures, tables, ...) are mostly presented without any proof or means for readers and reviewers to determine correctness. But computations can be proofed by delivering the code *for all results* along with the paper. Buckheit and Donoho (1995) define what de Leeuw (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.

Literate statistical practice insists that this principle applies throughout the whole project, and not only for a final project report. All essential parts of the workflow path must be reproducible.



## References

- Rudolf Beran. The role of experimental statistics. URL <http://www.stat.ucdavis.edu/~beran>. Department of Statistics, UC Davis, CA, USA, 2001.
- Jonathan B. Buckheit and David L. Donoho. Wavelab and reproducible research. Technical report, Stanford University Statistics Department, 1995.
- Vincent J. Carey. Literate statistical programming: Concepts and tools. *Chance*, 14(3):46–50, 2001.
- John M. Chambers. *Programming with Data; A Guide to the S Language*. Springer-Verlag, New York, 1998.
- Jan de Leeuw. Reproducible research: The bottom line. Technical Report 301, UCLA Statistics Department, 1996.
- Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.
- Donald E. Knuth. *Literate Programming*. Number 27 in CSLI Lecture Notes. Center for the Study of Language and Information, 1992.
- Lorenz Lang and Hans Peter Wolf. *The REVWEB Manual for Splus in Windows*. Uni. Bielefeld, Germany, 1997–2001. <http://www.wiwi.uni-bielefeld.de/StatCompSci/software/revweb/revweb.html>.
- Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physika Verlag, Heidelberg, Germany, 2002. ISBN 3-7908-1517-9.
- Thomas Lumley. Survival analysis in XLisp-Stat: a semiliterate program. *Journal of Statistical Software*, 3(2), 1998.
- R Development Core Team. R 1.6.0, October 2002. URL <http://www.R-project.org/>.
- Norman Ramsey. Literate programming simplified. *IEEE Software*, 11(5):97–105, September 1994.

- A.J. Rossini. 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, 2001. <http://www.ci.tuwien.ac.at/Conferences/DSC.html>, ISSN 1609-395X.
- A.J. Rossini, R.M. Heiberger, R Sparapani, M Maechler, and K Hornik. Emacs speaks statistics. *Journal of Computational and Graphical Statistics*, 2003.
- M. Schwab, M. Karrenbach, and Jon Claerbout. Making scientific computations reproducible. Technical report, Stanford University, Stanford Exploration Project, 1996.
- W3C. Document object model (dom) level 2. W3C Recommendation, World Wide Web Consortium, November 2000.

## A Literate Programming

Literate Programming [Knuth \(1992\)](#) describes an approach for combining code and documentation in a source file which can be woven into a description of the processes, algorithms, and results obtained from the system and tangled into the actual code. The approach was created to provide a sensible means for documenting mathematical computer programs. This takes advantage of modern documentation production to allow for a single source document which can be processed in ways which can produce different results. 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 which can be compiled or evaluated using a language interpreter; and **weaving**, which produces formatted documentation files. The initial work was done in a language-specific form for Pascal (Web), with the intention of documenting the T<sub>E</sub>X program. This was later extended to C (CWeb). Further research took two forms, language-independent work such as Noweb, Nuweb, and Funnelweb, and their supporting and contributed facilities, and language-specific implementations, which provide tighter integration.

The original application behind Literate Programming was to describe the computational and mathematical aspects behind the T<sub>E</sub>X typesetting system. The T<sub>E</sub>XBook is the classic example

of a literate computer program.

Literate Programming has been applied in the past to statistical practice. Probably the earliest work was simply to use Literate Programming for writing statistical software in compiled languages such as C or Fortran. Further extensions include the use of literate programming for the production of codebooks for datasets (V. Carey, Seminar, Fall 1993, Harvard). RevWeb (Lang and Wolf, 1997–2001), a system for reviveable analyses, relates to the work on reproducible research by Schwab et al. (1996) as done in a non-literate fashion by Buckheit and Donoho (1995), and discussed by de Leeuw (1996). However, much of this work was intended for reproduction rather than creation. In 1997, the creation of Emacs modes for Noweb support lead to the combination of ESS with Noweb, to provide a tool for allowing efficient interactive analysis. Mark Lunt refactored this code during 1999 to construct a better Noweb interface and integrate this tightly with ESS, no longer requiring Noweb to run analyses. Current ideas include the development of tools which clean up the user interface.

## B Consulting Example

```
\documentclass{article}
```

```
\title{Consulting Session}
```

```
\author{Consulting Statistician}
```

```
\date{Oct, 2002}
```

```
\begin{document}
```

```
\maketitle
```

```
\section{Introduction}
```

```
\label{SEC:introduction}
```

The initial problem brought up in this session is from Urban Studies, and looks at the design of an experiment to evaluate approaches for noxious weed control. 3 approaches are under consideration: newspaper, black landscaping fabric, and Round-up. The primary issues are:

Collection of Biostatistics  
Research Archive

```

\begin{itemize}
\item types of measurements to take (abundance? presence? amount of
      growth?)
\item blocking by landscape type (creek-side, hill, full-sun flat,
      shade flat).
\item number of comparisons that can be made given resources
      (bias/variance in size of plots, area available for experiment).
\end{itemize}

```

Result of first session: client needs to think harder about better approaches.

```

\section{Design Issues}
\label{SEC:design-issues}

```

If we use the `\emph{amount of growth}` measure, the client expects that a difference of 2 units ( $\text{kg}/\text{m}^2/\text{month}$ ) would be meaningful. No particular idea of variability within treatment type, so we are guessing.

```

<<echo=FALSE,results=tex>>=
# -*- mode: R -*-
library(xtable)
N <- 3:5
SD <- 1:3
my.power.expl <- matrix(0, nrow=length(N)*length(SD), ncol=4)
colnames(my.power.expl) <- c("N","SD","unpaired", "paired")
i <- 1
for(s in SD){
  for(n in N) {
    my.power.expl[i,1] <- n
    my.power.expl[i,2] <- s
    my.ttest.results <- power.t.test(n=n, delta=2, sd=s, type="two.sample")
    my.power.expl[i,3] <- my.ttest.results$power
    my.ttest.results <- power.t.test(n=n, delta=2, sd=s, type="paired")
  }
  i <- i + 1
}

```

```

my.power.expl[i,4] <- my.ttest.results$power
i <- i+1
}
}
xtable(my.power.expl,
       caption=paste("Exploratory Power for unpaired and paired designs",
                     "under different scenarios"),
       label="tab:power", align=c("c","c","c","c","c"))
@ %def my.power.expl

```

Table~\ref{tab:power} provides preliminary design calculations for discussion. It suggests that in this case, we might be able to do better. However, it wasn't clear if there was sufficient area to find pairs which to 'match'; client will report on this next time.

```


\begin{center}
<<fig=TRUE,echo=FALSE>>=
# -*- mode: R -*-
plot(my.power.expl[1:3,1], my.power.expl[1:3,3],ylim=c(0,1),
#     main="Unmatched Analysis",
      ylab="Power",xlab="Sample Size per group", type="b",lty=1)
lines(my.power.expl[4:6,1], my.power.expl[4:6,3], type="b",lty=2)
lines(my.power.expl[7:9,1], my.power.expl[7:9,3], type="b",lty=3)
lines(my.power.expl[1:3,1], my.power.expl[1:3,4], type="b",lty=1, pch=8)
lines(my.power.expl[4:6,1], my.power.expl[4:6,4], type="b",lty=2, pch=8)
lines(my.power.expl[7:9,1], my.power.expl[7:9,4], type="b",lty=3, pch=8)
@ %def plot.my.power.expl
\end{center}

\end{document}

```

Figure 2: Pages

21



## Consulting Session

Consulting Statistician

Oct., 2002

### 1 Introduction

The initial problem brought up in this session is from Urban Studies, and looks at the design of an experiment to evaluate approaches for noxious weed control. 3 approaches are under consideration: newspaper, black landscaping fabric, and Round-up. The primary issues are:

- types of measurements to take (abundance? presence? amount of growth?)
- blocking by landscape type (creek-side, hill, full-sun flat, shade flat).
- number of comparisons that can be made given resources (bias/variance in size of plots, area available for experiment).

Result of first session: client needs to think harder about better approaches.

### 2 Design Issues

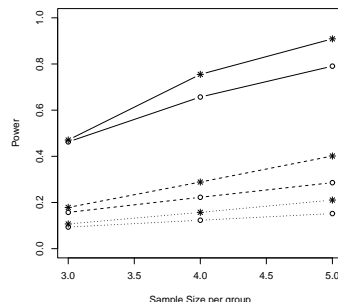
If we use the *amount of growth* measure, the client expects that a difference of 2 units ( $\text{kg}/\text{m}^2/\text{month}$ ) would be meaningful. No particular idea of variability within treatment type, so we are guessing.

Table 1 provides preliminary design calculations for discussion. It suggests that in this case, we might be able to do better. However, it wasn't clear if there was sufficient area to find pairs which to "match"; client will report on this next time.

1

	N	SD	unpaired	paired
1	3.00	1.00	0.46	0.47
2	4.00	1.00	0.66	0.75
3	5.00	1.00	0.79	0.91
4	3.00	2.00	0.16	0.18
5	4.00	2.00	0.22	0.29
6	5.00	2.00	0.29	0.40
7	3.00	3.00	0.09	0.11
8	4.00	3.00	0.12	0.16
9	5.00	3.00	0.15	0.21

Table 1: Exploratory Power for unpaired and paired designs under different scenarios



2