Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)
March 20–22, Vienna, Austria ISSN 1609-395X
Kurt Hornik, Friedrich Leisch & Achim Zeileis (eds.)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

Sweave and Beyond: Computations on Text Documents

Friedrich Leisch

Abstract

Sweave is a tool that allows to embed R code in LaTeX documents. The code can be evaluated and the resulting console output, figures and tables are automatically inserted into the final document. In this paper we first give an introduction into the Sweave file format, and then demonstrate how to use these files as R package user guides known as package vignettes. Finally we give an outlook on the design of the next generation of Sweave, which uses S4 classes and methods and will allow for much more complex computations on text documents.

1 Introduction

Typical computational work of a statistician includes import/export/storage of data, analysis of the data and writing reports on the results of the analysis. If standard analysis methods are not sufficient, or steps of the analysis can be automated, then "data analysis" may include writing some code on the side. In fact, the S system has been designed for exactly this paradigm of data analysis, *"turning users into programmers"* (Chambers, 1998).

1.1 Literate statistical practice

The code for an analysis (and be it "only" the command history of ones favorite statistical software package) is probably the most precise description of the analysis itself, because it allows the analysis to be reproduced. Nevertheless it is common practice to keep code and manuscripts in separate files (the same is true for data sets), although data and code could be seen as the "proof" for the results (Buckheit and Donoho, 1995). After several modifications of one of the files involved things tend to get out of sync, and in many cases it becomes unclear which version of the data set and saved code fragments *exactly* correspond to a report describing the final results.

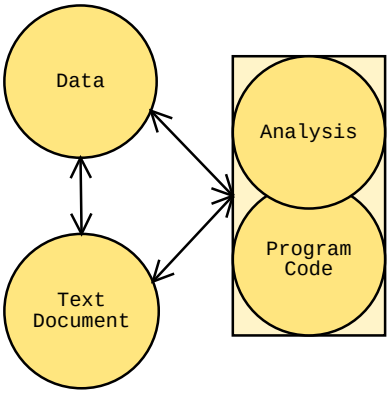

Figure 1: Relations between data storage, analysis, programming and manuscripts.

Hence, data, code and text domains should be integrated to make it comfortable for the analyst to deal with this objects simultaneously in a convenient way. The traditional way of writing a report as part of a statistical data analysis project uses two separate steps: First, the data are analyzed using one's favorite statistical software package, and afterwards the results of the analysis (numbers, graphs, ...) are used as the basis for a written report. In larger projects the two steps may be repeated alternately, but the basic procedure remains the same. Many statistical software packages try to support this process by generating pre-formatted tables and graphics that can easily be integrated into a final report using copy-and-paste from the data analysis system to the word processor. The basic paradigm is to write the report around the results of the analysis.

Another approach for integration of data analysis and document writing is to embed the analysis itself into the document, which reverses the traditional paradigm. Over the last decade a number of systems have been developed that integrate analysis and documentation and allow for *literate statistical practice* (Rossini, 2001; Rossini and Leisch, 2003).

This paradigm is probably most popular for creation of dynamic web pages and offers completely new possibilities for teaching statistics and delivering statistical methodology over the Internet. E.g., the ExploRe system (Härdle, Klinke, and Müller, 1999) provides means to embed statistical quantlets in web pages or electronic books to create interactive documents with direct access to a statistical data analysis package. Another example for a dynamic statistical analysis on a web page is given in Temple Lang (2001), by embedding R into Netscape as a plugin. Report rendering is performed using XML and XSL.

Sweave (Leisch, 2002) combines ideas from both worlds described above using literate programming tools. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change, while using standard tools for both data analysis and word processing.

1.2 R package documentation

The main vehicle for documenting R packages are help files, as accessed by the `help()` command. The source for the help files is in R documentation format (`*.Rd` files). These files contain code mainly in two sections: usage and examples. All

examples in the R help files are by default required to be executable such that the user can copy & paste the code to a running R process using

- the mouse,
- keyboard shortcuts if running R inside Emacs with ESS (*Emacs speaks statistics*, Rossini, Heiberger, Sparapani, Mächler, and Hornik, 2003), or
- the `example()` function directly in R.

Examples should be flagged as non-executable only if there are good reasons, e.g., because they require user interactivity like `identify()` and hence cannot be executed in batch mode.

The tools for package quality control available through the `R CMD check`[1] command test if all examples are executable. In addition, the code in the usage section is compared with the actual implementation to check for inconsistencies or missing documentation.

The .Rd format was designed for reference documentation on single R objects (functions, classes, data sets, ...), it is not intended for demonstrating the interaction of multiple functions in a package. For this task we have developed the concept of *package vignettes*, short to medium-sized documents explaining parts or all of the functionality of a package in a more informal tone than the strict format of reference help pages.

This paper is organized as follows: Section 2 gives a short introduction to the Sweave file format, the focus is on explaining the basic principles rather than all bells and whistles. Section 3 gives an overview of tools in base R and extension packages for reading and interacting with Sweave files, especially if they are package vignettes, and Section 4 shows how to write vignettes and integrate them in an R package. Finally, Section 5 gives a preview of some features of the next version of Sweave, a new implementation using S4 classes and methods.

2 Sweave files: A small example

Sweave source files are regular noweb files (Ramsey, 1998) with some additional syntax for fine control over the final output. Noweb is a simple literate programming tool which allows to combine program source code and the corresponding documentation into a single file. These consist of a sequence of code and documentation segments, called *chunks*. Different command line programs are used to extract the code (*"tangle"*) or typeset documentation together with the code (*"weave"*).

A small Sweave file is shown in Figure 2, which contains four code chunks embedded in a simple LaTeX document. '`<<...>>=`' at the beginning of a line marks the start of a code chunk, while a '`@`' at the beginning of a line marks the start of a documentation chunk. Sweave translates this into a regular LaTeX document, which in turn can be compiled by `latex` to Figure 3.

2.1 The code chunks

The main work of Sweave is done on the code chunks. All code chunks are evaluated by R in the order they appear in the document[2]. Within the double angle brackets

[1] `R CMD xxx` is `Rcmd xxx` in the Windows version of R.

[2] There are ways to suppress evaluation or re-use chunks, which is beyond the scope of this article.

```
\documentclass[a4paper]{article}

\begin{document}

<<echo=false,results=hide>>=
library(lattice)
library(xtable)
data(cats, package="MASS")
@

\section*{The Cats Data}

Consider the \texttt{cats} regression example from
Venables \& Ripley (1997). The data frame contains
measurements of heart and body weight
of \Sexpr{nrow(cats)} cats (\Sexpr{sum(cats$Sex=="F")}
female, \Sexpr{sum(cats$Sex=="M")} male).

A linear regression model of heart weight by sex and
gender can be fitted in R using the command
<<>>=
lm1 = lm(Hwt~Bwt*Sex, data=cats)
lm1
@
Tests for significance of the coefficients are shown in
Table~\ref{tab:coef}, a scatter plot including the
regression lines is shown in Figure~\ref{fig:cats}.

\SweaveOpts{echo=false}

<<results=tex>>=
xtable(lm1,
       caption="Linear regression model for cats data.",
       label="tab:coef")
@

\begin{figure}
  \centering
<<fig=true,width=12,height=6>>=
lset(col.whitebg())
print(xyplot(Hwt~Bwt|Sex, data=cats, type=c("p", "r")))
@
  \caption{The cats data from package MASS.}
  \label{fig:cats}
\end{figure}

\end{document}
```

Figure 2: A minimal Sweave file: `example.Snw`.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.9813	1.8428	1.62	0.1080
Bwt	2.6364	0.7759	3.40	0.0009
SexM	−4.1654	2.0618	−2.02	0.0453
Bwt:SexM	1.6763	0.8373	2.00	0.0472

Table 1: Linear regression model for cats data.

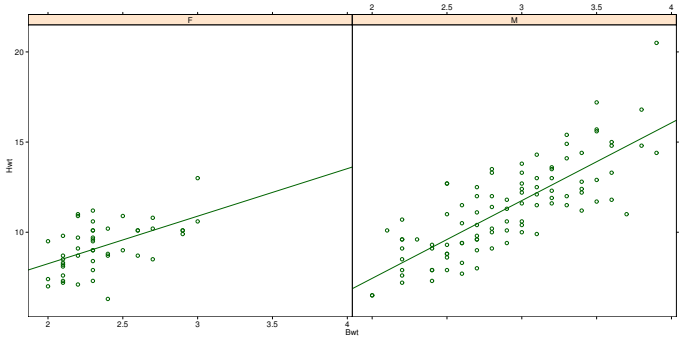

Figure 1: The cats data from package MASS.

The Cats Data

Consider the `cats` regression example from Venables & Ripley (1997). The data frame contains measurements of heart and body weight of 144 cats (47 female, 97 male).

A linear regression model of heart weight by sex and gender can be fitted in R using the command

```
> lm1 = lm(Hwt ~ Bwt * Sex, data = cats)
> lm1

Call:
lm(formula = Hwt ~ Bwt * Sex, data = cats)

Coefficients:
(Intercept)          Bwt          SexM      Bwt:SexM
      2.981        2.636       -4.165        1.676
```

Tests for significance of the coefficients are shown in Table 1, a scatter plot including the regression lines is shown in Figure 1.

Figure 3: The final document is created by running `latex` on the intermediate file example.tex created by `Sweave("example.Snw")`.

we can specify options that control how the code and the corresponding output are rendered in the final document. The first code chunk (lines 5–8 in Figure 2) declares that neither the R code (`echo=false`) nor output (`results=hide`) shall be included. The purpose of this chunk is to initialize R by loading packages and data, we want to hide these technical details from the reader.

Let us skip the text in lines 10–20 for the moment and go directly to the next code chunk in lines 21–23. It uses the default settings for all options (nothing is specified within the double angle brackets): both input and output are shown to the user (see Figure 3), the chunk is rendered such that it emulates the R console when the code is typed at the prompt. All input and output are automatically encapsulated in verbatim-like environments.

The next code chunk can be found at lines 31–34. It uses the package **xtable** to pretty-print the coefficient matrix of the linear regression model. By specifying `results=tex` we tell Sweave that the output of this code chunk is regular TEX code and hence needs no protection by a verbatim environment.

The last code chunk in lines 39–41 is marked as a figure chunk (`fig=true`) such that Sweave creates EPS and PDF files corresponding to the plot created by the commands in the chunk. Furthermore, an `\includegraphics{}` statement is inserted into the LATEX file. Options `width` and `height` are passed to R's graphics devices and determine the size of the figure in the EPS and PDF files.

In line 29 we use `\SweaveOpts{echo=false}` to modify the default for option `echo` to the value of `false` for all code chunks following, hence the code for the last two chunks is not shown in Figure 3. It has exactly the same effect as if we had included `echo=false` within the double angle brackets of the two chunks.

2.2 Using S objects in text

Let us now return to the text paragraph in lines 13–17. It contains three `\Sexpr{}` statements. Sweave replaces them by the value of the corresponding S expression, which should be a simple character string (or something that can be coerced to a string by `as.character()`). In the example we use it to avoid hard-coding the size of the data set. If the number of observations changes we do not need to change anything in our Sweave file, we simply re-run `Sweave()` and `latex` and the report is up-to-date.

3 Computations on Sweave files

3.1 Tangle & weave

Sweave is contained in the standard R package **tools** (R version 1.5.0 or higher). The Sweave file example.Snw can be woven into a regular LATEX file using the R commands

```
> library(tools)
> Sweave("example.Snw")
Writing to file example.tex
Processing code chunks ...
 1 : term hide
 2 : echo term verbatim
 3 : term tex
```

```
 4 : term verbatim eps pdf
You can now run LaTeX on example.tex
```

Sweave shows a status line per code chunk indicating which options are active. The companion command

```
R> Stangle("example.Snw")
Writing to file example.R
```

can be used to extract the code of all chunks into an R source file. If Sweave is used for literate statistical practice, then these two commands will probably be sufficient for most purposes such as keeping research reproducible (Leisch and Rossini, 2003).

3.2 Reading vignettes

Books on using S for data analysis like Venables and Ripley (2002) typically contain a mixture of documentation, code and output. Short documents using a similar style of writing are ideally suited to explain the functionality of a package to new users. The directory inst/doc of an R source package may contain package documentation in arbitrary format, we recommend PDF files due to their platform independence. For Sweave files additional functionality is available, both from the R command line and using a graphical user interface.

We call a user guide in in inst/doc a vignette only if it is a document where the user can extract the R code and interact with it. Currently Sweave is the only format for such documents that is supported by R, there may be others in the future. In short: every vignette is a user guide, but not every user guide is a vignette.

3.2.1 Command line interface

Starting with R version 1.8.0 there is support in base R for listing and viewing package vignettes. The `vignette()` function works similar to `data()` and `demo()`. If no argument is given, a list of all vignettes in all installed packages is returned:

```
R> vignette()

Vignettes in package 'AnnBuilder':

AnnBuilder            AnnBuilder Basic (source, pdf)
HowTo                 AnnBuilder HowTo (source, pdf)

Vignettes in package 'Biobase':

Biobase               Biobase Primer (source, pdf)
Bioconductor          Howto Bioconductor (source, pdf)
HowTo                 HowTo HowTo (source, pdf)
esApply               esApply Introduction (source, pdf)

...

Vignettes in package 'strucchange':
```

```
strucchange-intro     strucchange: An R Package for
                      Testing for Structural Change in
                      Linear Regression Models
                      (source, pdf)
...
```

After the title of each vignette the list in parenthesis shows which formats are available; in the example given all vignettes are available both in source and PDF format. To view the `strucchange-intro` vignette (Zeileis, Leisch, Hornik, and Kleiber, 2002), all one has to do is to issue

```
R> vignette("strucchange-intro")
```

and the PDF file is opened on the screen. If the source file for a vignette is available, one can easily get a handle on the code the vignette contains, although we have not fully automated the procedure yet. First we get the full path to the vignette directory

```
R> vigdir = system.file("doc", package="strucchange")
```

and then we have a look which files it contains

```
R> list.files(vigdir)
[1] "00Index.dcf"
[2] "strucchange-intro.R"
[3] "strucchange-intro.Rnw"
[4] "strucchange-intro.pdf"
[5] "strucchange-intro.tex"
```

File strucchange-intro.Rnw is the original Sweave file, strucchange-intro.R has the extracted R code for all code chunks and could now be executed using `source()` or opened in an editor. If the .R file is not available, we can create it in the current working directory by

```
R> library("tools")
R> vig = listFilesWithType(vigdir, "vignette")
R> Stangle(vig[1])
Writing to file strucchange-intro.R
```

where `listFilesWithType()` returns the full path to all files in `vigdir` that have type `"vignette"`, i.e., an extension marking them as Sweave files.

3.2.2 Graphical user interface

The simplest way to access vignettes is probably through the HTML help system. After `help.start()`, an index of all vignettes is linked into the beginning of a packages' table of contents (if the respective package contains vignettes). Additionally there is a link to the directory containing the vignettes, e.g., to look at source files using a browser.

A more advanced interface to package vignettes is available in the Bioconductor package **tkWidgets**, available from http://www.bioconductor.org. Function `vExplorer()` lists all available vignettes in a nice point & click menu, after selecting the **strucchange** vignette the upper left window shown in Figure 4 is opened. The

PDF version of the vignette can be opened by clicking on the "View PDF" button. Each code chunk of the vignette has a button on the left side of the window, clicking on the button shows the code in the "R Source Code" text field. The code can be executed and the resulting output is shown in the "Results of Execution" area.

The most powerful feature of this kind of interface is that the S code in the source code field can be modified by the user, e.g., to try variations of the prefabricated examples. To modify the example, one simply edits the code in the "R Source Code" area and presses the "Execute Code" button again.

Dynamic statistical documents and their user interfaces are an open research area, see also Buttrey, Nolan, and Lang (2001) and Sawitzki (2002) for other approaches.

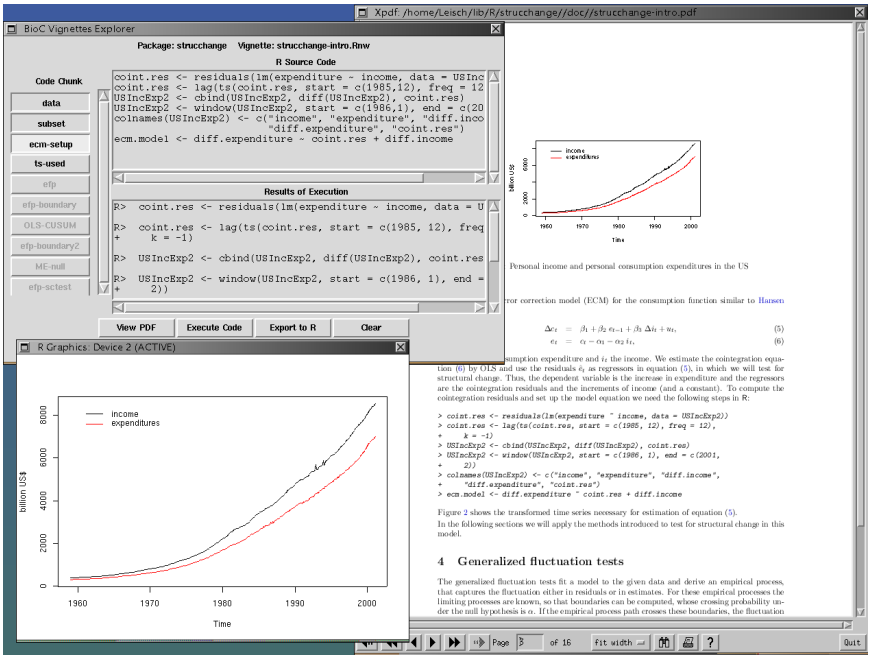

Figure 4: Screenshot of `vExplorer()` showing the vignette from package **strucchange**: main controls for code chunk execution (upper left), currently active R graphics window (lower left) and a PDF viewer (right).

4 Writing Sweave files and package vignettes

The Emacs text editor offers a perfect authoring environment for Sweave, especially for people who already use Emacs for writing LaTeX documents and interacting with R. ESS allows to connect an Sweave file to a running R process while writing the document. Code chunks can be sent to R and evaluated using simple keyboard shortcuts or popup menus. Syntax highlighting, automatic indentation and keyboard shortcuts depend on the location of the pointer: in documentation chunks Emacs behaves as if editing a standard LaTeX file, when the pointer moves to a code chunk the mode switches automatically to S programming.

However, it is not necessary to use Emacs, Sweave is a standalone system, the noweb source files for Sweave can be written using any text editor. Even the noweb syntax in not a necessity, because Sweave is highly configurable. Currently there are two syntaxes available, the noweb syntax described above and a LaTeX-based syntax. In LaTeX syntax the first code chunk of the example looks like

```
\begin{Scode}{echo=false,results=hide}
  library(lattice)
  library(xtable)
  data(cats, package="MASS")
\end{Scode}
```

Using a syntax different from noweb is a necessity when complete R packages are written using literate programming with noweb as proposed by Carey (2001). Using an XML syntax (see Section 5.2 below) will probably play an important role in computations on Sweave-type statistical documents by programs different than R, e.g., to manage vignette repositories.

Once the Sweave file is written, it is almost trivial to include it in an **R** package and make it available to users as a package vignette. Consider that file foo.Rnw shall be used as vignette for package **foo**. First one needs to add some meta-information to the file along the lines of

```
% \VignetteIndexEntry{An R Package for ...}
% \VignetteDepends{foo, bar, ...}
% \VignetteKeyword{kwd1}
% \VignetteKeyword{kwd2}
```

All of these should be in LaTeX comments (after a '`%`' sign) as we have not defined them as proper LaTeX commands. The index entry is used for the listings of `vignette()` or `vExplorer()`, it typically is the same as the title of the document (or an abbreviated version thereof). Note that it is directly used in text and HTML files and hence should not contain any TeX markup. The dependency information is analogous to the `Depends` field of a package DESCRIPTION file and lists packages needed to execute the code in the vignette. The list of `\VignetteXXX` meta-information specifications will probably get longer in the near future, especially for versioning etc.

Once this is done all one has to do is create a subdirectory inst/doc in the package source tree and copy foo.Rnw to it. The rest is taken care of by the **R** package management system, e.g.

- `R CMD check` will extract the code from the vignette and test if it is executable.
- `R CMD build` will run `Sweave()` and `pdflatex` on the vignette to create the PDF version.
- The package installation mechanism creates an index of all vignettes in the package and links it into the HTML help system.

Note that even code chunks with option `eval=FALSE` are tested by `R CMD check`, if you want code in a vignette that should not be tested, move it to a normal LaTeX verbatim environment. The reason for this policy is that users should be able to rely on code examples to be executable as seen in the vignette.

By including the PDF version in the package sources it is not necessary that the vignettes can be compiled at install time, i.e., the package author can use private LaTeX extensions or bibtex files. Only the **R** code inside the vignettes is part of the checking procedure, typesetting manuals is not part of package quality control.

For more details see also the "Writing R Extensions" manual which features a section on package vignettes.

In general it is assumed that package *authors* run `R CMD build` on their machine (and may safely assume that only they do that). `R CMD check` on the other hand should be runnable by *everybody*, e.g., CRAN runs a check on all 250+ packages (as of the time of this writing) on a daily basis, the results are available at http://cran.r-project.org/src/contrib/checkSummary.html. Bioconductor has opted for a stricter policy such that even building packages (including running `latex` on vignettes) should be reproducible on every machine which has the necessary tools installed.

5 The next generation: S4weave

The next generation of Sweave is currently in development. Because the current implementation does not scale up to all ideas we want to realize, we have started to rewrite the complete code base. Sweave started as a small utility script for personal use by the author, more and more features were added to either to satisfy personal needs or user requests as the software became increasingly popular.

The new (unreleased) version is based on

- S4 classes for complete Sweave files including all documentation and code chunks, syntax definition, ...
- Computations on document objects rather than serial processing of chunks.

This turns complete Sweave files into first class S objects following one of the basic principles of S software design (Chambers, 1998).

```
R> x = read.Sweave("example-3.Snw")
R> x

Call:
read.Sweave(file = "example-3.Snw")

File: example-3.Snw
Syntax: noweb

Class and number of chunks:
SweaveCodeChunk  SweaveDocChunk
              4               5
```

Extraction of code and documentation chunks is trivial, as they are slots of the respective S4 objects:

```
R> x@chunks[1:2]
[[1]]
   \documentclass[a4paper]{article}
```

```
   \begin{document}


[[2]]
   # eval term hide
   library(lattice)
   library(xtable)
   data(cats, package="MASS")
```

Both code and documentation chunks have corresponding classes:

```
R> class(x@chunks[[1]])
[1] "SweaveDocChunk"
R> class(x@chunks[[2]])
[1] "SweaveCodeChunk"
```

and all computations are methods for those classes.

5.1 Chunk dependencies

One of the goals is to construct a directed graph of chunk dependencies, which in turn allows for conditional processing of chunks. Consider an Sweave file with the following series of code chunks:

```
<<a>>=
...
<<b>>=
...
<<c, depends=a>>=
...
<<d, depends=c>>=
...
<<e, depends=c>>=
...
<<f, depends=d+e>>=
...
```

A graphical representation of the dependencies can be seen in Figure 5. By default a chunk depends on every previous chunk, such that, e.g., chunk `"b"` depends on `"a"`. However, chunk `"c"` declares that it only depends on `"a"`. Hence, `"c"` has to be re-evaluated only if `"a"` or `"c"` itself change, but not if `"b"` changes. In general, each node has to be re-evaluated only if any *ancestral* node in the graph or the node itself changes. Otherwise cached results from previous computations can be re-used. Especially for Sweave files with time-consuming computations and sparse dependency graphs this should speed up processing considerably. The idea is of course based on the `make` program, which also does computations only when necessary (determined by time stamps of files).

5.2 Using XML

For document exchange with other dynamic document systems there is an XML DTD for Sweave files. Sweave files in XML syntax are not parsed by the Sweave

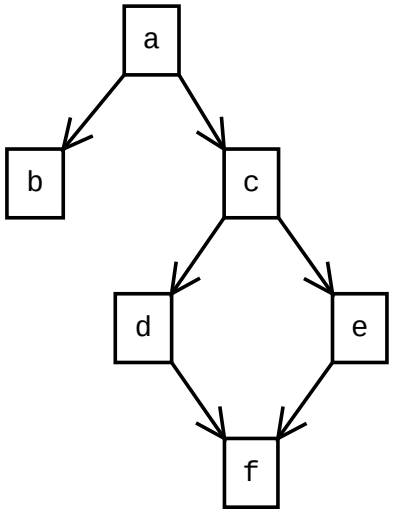

Figure 5: Graph of chunk dependencies.

parser, but by `xmlTreeParse()` from package **XML**: this allows *validation* for syntactical correctness. The resulting object is completely equal to those read from "standard" Sweave files, all methods can be used. `SweaveSyntConv()` allows fully automatic conversion between various file formats. By default the syntax of a file is preserved, e.g.,

```
R> writeSweaveFile(x)
\documentclass[a4paper]{article}

\begin{document}

<<echo=false,results=hide>>=
library(lattice)
library(xtable)
data(cats, package="MASS")
@
...
```

but it can easily be changed:

```
R> writeSweaveFile(x, syntax=SweaveSyntaxXML)
<?xml version="1.0"?>
<!DOCTYPE Sweave SYSTEM "Sweave.dtd">
<Sweave>
<doc><![CDATA[
\documentclass[a4paper]{article}

\begin{document}

]]></doc>
<code options="echo=false,results=hide"><![CDATA[
library(lattice)
```

```
library(xtable)
data(cats, package="MASS")
]]></code>
...
```

6 Summary

Integrated statistical documents are receiving a lot of research interest lately. As the methodology used in computational statistics is getting more sophisticated all the time, the citation of textual descriptions (without an implementation) of algorithms is not necessarily sufficient to reproduce numerical or graphical results. Extensible, well-defined and open standards are needed in order to combine data, analysis and text into packaged entities for exchange between researchers as well as for persistent storage.

Sweave offers a simple yet powerful solution for people familiar with R and LaTeX as almost no additional commands or syntax have to be learned. This way we hope too attract more people to become authors of integrated statistical documents to see what features are actually required and used. As Sweave files can now be converted to S4 objects and regular XML files, more complex computations on text documents, especially for interactive user interfaces, are possible.

Acknowledgements

`vignette()` and most of `R CMD check` was written by Kurt Hornik. `vExplorer()` and its helper functions were written by Jeff Gentry and Jianhua Zhang as part of the Bioconductor project. I want to thank them and Robert Gentleman for helpful ideas and discussions. Parts of this paper have appeared as a mini-series on Sweave in R News (Volumes 2/3 and 3/2, respectively).

References

Jonathan Buckheit and David Donoho. WaveLab and reproducible research. URL http://www-stat.stanford.edu/~donoho/. Statistics Department, Stanford University, CA, USA, 1995.

Samuel E. Buttrey, Deborah Nolan, and Duncan Temple Lang. An environment for creating interactive statistical documents. In Edward J. Wegman, Amy Braverman, Arnold Goodman, and Padhraic Smyth, editors, *Computing Science and Statistics*, volume 33. Interface Foundation of North America, Fairfax Station, VA, USA, 2001.

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, Berlin, Germany, 1998.

Wolfgang Härdle, Sigbert Klinke, and Marlene Müller. *XploRe Learning Guide*. Springer Verlag, 1999.

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. URL http://www.ci.tuwien.ac.at/~leisch/Sweave. ISBN 3-7908-1517-9.

Friedrich Leisch and Anthony J. Rossini. Reproducible statistical research. *Chance*, 16(2):46–50, 2003.

Norman Ramsey. *Noweb man page*. University of Virginia, USA, 1998. URL http://www.cs.virginia.edu/~nr/noweb. version 2.9a.

Anthony Rossini. Literate statistical analysis. In Kurt Hornik and Friedrich Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria*, 2001. URL http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/. ISSN 1609-395X.

Anthony J. Rossini, Richard M. Heiberger, Rodney Sparapani, Martin Mächler, and Kurt Hornik. Emacs speaks statistics: A multi-platform, multi-package development environment for statistical analysis. *Journal of Computational and Graphical Statistics*, 2003. (Accepted for publication).

Anthony J. Rossini and Friedrich Leisch. Literate statistical practice. UW Biostatistics Working Paper Series 194, University of Washington, WA, USA, 2003. URL http://www.bepress.com/uwbiostat/paper194.

Günther Sawitzki. Keeping statistics alive in documents. *Computational Statistics*, 17:65–88, 2002.

Duncan Temple Lang. Embedding S in other languages and environments. In Kurt Hornik and Friedrich Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria*, 2001. URL http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/. ISSN 1609-395X.

William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S. Fourth Edition*. Springer, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4/. ISBN 0-387-95457-0.

Achim Zeileis, Friedrich Leisch, Kurt Hornik, and Christian Kleiber. `strucchange`: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38, 2002. URL http://www.jstatsoft.org/v07/i02/.

Affiliation

Friedrich Leisch
Institut für Statistik und Wahrscheinlichkeitstheorie
Technische Universität Wien
Wiedner Hauptstraße 8-10/1071
1040 Wien, Austria
E-mail: Friedrich.Leisch@ci.tuwien.ac.at