Sweave User Manual

Friedrich Leisch

R Version 2.0.0

Contents

1 Introduction

Sweave provides a flexible framework for mixing text and S code for automatic document generation. A single source file contains both documentation text and S code, which are then *woven* into a final document containing

- the documentation text together with
- the S code and/or
- the output of the code (text, graphs)

by running the S code through an S engine[1] like R[2]. This allows to re-generate a report if the input data change and documents the code to reproduce the analysis in the same file that also contains the report. The S code of the complete analysis is embedded into a LaTeX document[3] using the noweb syntax (Ramsey, 1998). Hence, the full power of LaTeX (for high-quality typesetting) and S (for data analysis) can be used simultaneously. See Leisch (2002) and references therein for more general thoughts on dynamic report generation and pointers to other systems.

Many S users are also LaTeX users, hence no new software or syntax has to be learned. The Emacs text editor (Stallman, 1999) offers a perfect authoring environment for Sweave, especially for people which already use Emacs for writing LaTeX documents and interacting with an S engine. We have chosen to use noweb as basis for the Sweave system because

1. the syntax is extremely simple and hence easy to learn
2. the ESS noweb mode for Emacs already provides a perfect authoring environment(Rossini et al., 2003)

The importance of 2 should not be underestimated: A document format without convenient tools for authors will almost certainly be ignored by prospective users. 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.

Sweave uses a modular concept using different drivers for the actual translations. Obviously different drivers are needed for different text markup languages (LaTeX, HTML, ...). Unfortunately we will also need different drivers for different S engines (R, S-Plus[4]), because we make extensive usage of `eval()`, connections, and the graphics devices, and the various S engines have some differences there. Currently there is only the driver `RWeaveLatex` which combines R and LaTeX.

2 Noweb files

Noweb (Ramsey, 1998) is a simple literate-programming tool which allows to combine program source code and the corresponding documentation into a single file. Different programs allow to extract documentation and/or source code. A noweb file is a simple text file which consists of a sequence of code and documentation segments, these segments are called *chunks*:

Documentation chunks start with a line that has an at sign (`@`) as first character, followed by a space or newline character. The rest of this line is a comment and ignored. Typically documentation chunks will contain text in a markup language like LaTeX.

Code chunks start with `<<`*name*`>>=` at the beginning of a line; again the rest of the line is a comment and ignored.

[1] See Becker et al. (1988) and Chambers (1998) for definitions of the S language, and Venables and Ripley (2000) for details on the term *S engine* and detailed descriptions of differences between various implementations of the S language.

[2] `http://www.R-project.org`

[3] `http://www.ctan.org`

[4] `http://www.insightful.com`

The default for the first chunk is documentation.

In the simplest usage of noweb, the (optional) names of code chunks give the name of source code files, and the tool `notangle` can be used to extract the code chunk from the noweb file. Multiple code chunks can have the same name, the corresponding code chunks are the concatenated when the source code is extracted. Noweb has some additional mechanisms to cross-reference code chunks (the `[[...]]` operator, etc.), Sweave does currently not use or support this features, hence they are not described here.

3 Sweave files

3.1 A simple example

Sweave source files are regular noweb files with some additional syntax that allows some additional control over the final output. Traditional noweb files have the extension `.nw`, which is also fine for Sweave files (and fully supported by the software). Additionally, Sweave currently recognizes files with extensions `.rnw`, `.Rnw`, `.snw` and `.Snw` to directly indicate a noweb file with Sweave extensions. We will use `.Snw` throughout this document.

A minimal Sweave file is shown in Figure 1, which contains two code chunks embedded in a simple LaTeX document. Sweave translates this into the LaTeX document shown in Figures 2 and 3. The first difference between the `example-1.Snw` and `example-1.tex` is that the LaTeX style file `Sweave.sty` is automatically loaded, which provides environments for typesetting S input and output (the LaTeX environments `Sinput` and `Soutput`). Otherwise, the documentation chunks are copied without any modification from `example-1.Snw` to `example-1.tex`.

```
\documentclass[a4paper]{article}

\title{Sweave Example 1}
\author{Friedrich Leisch}

\begin{document}

\maketitle

In this example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:

<<>>=
data(airquality)
library(ctest)
kruskal.test(Ozone ~ Month, data = airquality)
@
which shows that the location parameter of the Ozone
distribution varies significantly from month to month. Finally we
include a boxplot of the data:

\begin{center}
<<fig=TRUE,echo=FALSE>>=
boxplot(Ozone ~ Month, data = airquality)
@
\end{center}

\end{document}
```

Figure 1: A minimal Sweave file: `example-1.Snw`.

The real work of Sweave is done on the code chunks: The first code chunk has no name, hence the default behavior of Sweave is used, which transfers both the S commands and their respective output to the LaTeX file, embedded in `Sinput` and `Soutput` environments, respectively.

The second code chunk shows one of the Sweave extension to the noweb syntax: Code chunk names can be used to pass options to Sweave which control the final output.

- The chunk 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, a `\includegraphics{example-1-002}` statement is inserted into the LaTeX file (details on the choice of filenames for figures follow later in this manual).
- Option `echo=FALSE` indicates that the S input should not be included in the final document (no `Sinput` environment).

```
\documentclass[a4paper]{article}

\title{Sweave Example 1}
\author{Friedrich Leisch}

\usepackage{/usr/lib/R/share/texmf/Sweave}
\begin{document}

\maketitle

In this example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:

\begin{Schunk}
\begin{Sinput}
> data(airquality)
> library(ctest)
> kruskal.test(Ozone ~ Month, data = airquality)
\end{Sinput}
\begin{Soutput}
        Kruskal-Wallis rank sum test

data:  Ozone by Month
Kruskal-Wallis chi-squared = 29.2666, df = 4, p-value = 6.901e-06
\end{Soutput}
\end{Schunk}
which shows that the location parameter of the Ozone
distribution varies significantly from month to month. Finally we
include a boxplot of the data:

\begin{center}
\includegraphics{example-1-002}
\end{center}

\end{document}
```

Figure 2: The output of `Sweave("example-1.Snw")` is the file `example-1.tex`.

3.2 Sweave options

Options control how code chunks and their output (text, figures) are transfered from the `.Snw` file to the `.tex` file. All options have the form `key=value`, where `value` can be a number, string or

Sweave Example 1

Friedrich Leisch

May 30, 2003

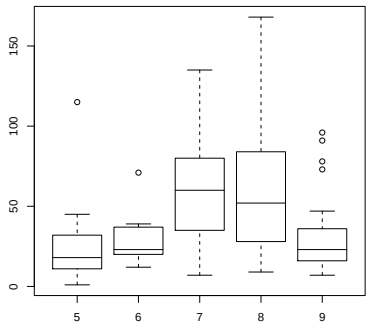
In this example we embed parts of the examples from the `kruskal.test` help page into a LaTeX document:

```
> data(airquality)
> library(ctest)
> kruskal.test(Ozone ~ Month, data = airquality)

        Kruskal-Wallis rank sum test

data:  Ozone by Month
Kruskal-Wallis chi-squared = 29.2666, df = 4, p-value = 6.901e-06
```

which shows that the location parameter of the Ozone distribution varies significantly from month to month. Finally we include a boxplot of the data:



1

Figure 3: The final document is created by running `latex` on `example-1.tex`.

logical value. Several options can be specified at once (seperated by commas), all options must take a value (which must not contain a comma or equal sign). Logical options can take the values `true`, `false`, `t`, `f` and the respective uppercase versions.

In the `.Snw` file options can be specified either

1. inside the angle brackets at the beginning of a code chunk, modifying the behaviour *only for this chunk*, or

2. anywhere in a documentation chunk using the command

   ```
   \SweaveOpts{opt1=value1, opt2=value2, ..., optN=valueN}
   ```

   which modifies the defaults for the rest of the document, i.e., *all code chunks after the statement*. Hence, an `\SweaveOpts` statement in the preamble of the document sets defaults for all code chunks.

Which options are supported depends on the driver in use. All drivers should at least support the following options (all options appear together with their default value, if any):

engine=S: a character string describing which S engines are able to handle the respective code chunks. Possible values are, e.g., `S`, `R`, `S3` or `S4`. Each driver only processes compatible code chunks and ignores the rest.

split=FALSE: a logical value. If `TRUE`, then the output is distributed over several files, if `FALSE` all output is written to a single file. Details depend on the driver.

label: a text label for the code chunk, which is used for filename creation when `split=TRUE`. If the label is of form `label.engine`, then the extension is removed before further usage (e.g., label `hello.S` is reduced to `hello`).

The first (and only the first) option in a code chunk name can be optionally without a name, then it is taken to be a label. I.e., starting a code chunk with

```
<<hello.S, split=FALSE>>
```

is the same as

```
<<split=FALSE, label=hello.S>>
```

but

```
<<split=FALSE, hello.S>>
```

gives a syntax error. Having an unnamed first argument for labels is needed for noweb compatibility. If only `\SweaveOpts` is used for setting options, then Sweave files can be written to be fully compatible with noweb (as only filenames appear in code chunk names).

3.3 Using scalars in text

There is limited support for using the values of S objects in text chunks. Any occurrence of `\Sexpr{`*`expr`*`}` is replaced by the string resulting from coercing the value of the expression `expr` to a character vector; only the first element of this vector is used. E.g., `\Sexpr{sqrt(9)}` will be replaced by the string `'3'` (without any quotes).

The expression is evaluated in the same environment as the code chunks, hence one can access all objects defined in the code chunks which have appeared before the expression and were not ignored. The expression may contain any valid S code, only curly brackets are not allowed. This is not really a limitation, because more complicated computations can be easily done in a hidden code chunk and the result then be used inside a `\Sexpr`.

3.4 Code chunk reuse

Named code chunks can be reused in other code chunks following later in the document. Consider the simple example

```
<<a>>=
x <- 10
@

<<b>>=
x + y
@

<<c>>=
<<a>>
y <- 20
<<b>>
@
```

which is equivalent to defining the last code chunk as

```
<<c>>=
x <- 10
y <- 20
x + y
@
```

The chunk reference operator `<<>>` takes only the name of the chunk as argument, without any additional Sweave options.

3.5 Syntax definition

So far we have only talked about Sweave files using noweb syntax (which is the default). However, Sweave allows the user to redefine the syntax marking documentation and code chunks, using scalars in text or reuse code chunks.

Figure 4 shows the example from Figure 1 using the `SweaveSyntaxLatex` definition. Code chunks are now enclosed in `Scode` environments, code chunk reuse is performed using `\Scoderef{chunkname}`. All other operators are the same as in the nowb-style syntax.

Which syntax is used for a document is determined by the extension of the input file, files with extension `.Rtex` or `.Stex` are assumed to follow the LaTeX-style syntax. Alternatively the syntax can be changed at any point within the document using the commands

```
\SweaveSyntax{SweaveSyntaxLatex}
```

or

```
\SweaveSyntax{SweaveSyntaxNoweb}
```

at the beginning of a line within a documentation chunk. Syntax definitions are simply lists of regular expression for several Sweave commands, see the two default definitions mentioned above for examples (more detailed intructions will follow once the API has stabilized).

4 Tangling and weaving

The user frontends of the Sweave system are the two S functions `Stangle()` and `Sweave()`, both are contained in the base R package `utils`. (`http://www.R-project.org`). `Stangle` can be used to extract only the code chunks from an `.Snw` file and write to one or several files. `Sweave()` runs

```
\documentclass[a4paper]{article}

\title{Sweave Example 1}
\author{Friedrich Leisch}

\begin{document}

\maketitle

In this example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:

\begin{Scode}
data(airquality)
library(ctest)
kruskal.test(Ozone ~ Month, data = airquality)
\end{Scode}
which shows that the location parameter of the Ozone
distribution varies significantly from month to month. Finally we
include a boxplot of the data:

\begin{center}
\begin{Scode}{fig=TRUE,echo=FALSE}
boxplot(Ozone ~ Month, data = airquality)
\end{Scode}
\end{center}

\end{document}
```

Figure 4: An Sweave file using LaTeX syntax: `example-1.Stex`.

the code chunks through an S engine and replaces them with the respective input and/or output. `Stangle` is actually just a wrapper function for Sweave, which uses a tangling instead of a weaving driver by default.

`Sweave` *Automatic Generation of Reports*

Description

`Sweave` provides a flexible framework for mixing text and S code for automatic report generation. The basic idea is to replace the S code with its output, such that the final document only contains the text and the output of the statistical anlysis.

Usage

```
Sweave(file, driver = RweaveLatex(),
       syntax = getOption("SweaveSyntax"), ...)

Stangle(file, driver = Rtangle(),
        syntax = getOption("SweaveSyntax"), ...)
```

Arguments

`file`	Name of Sweave source file.
`driver`	The actual workhorse, see details below.
`syntax`	An object of class `SweaveSyntax` or a character string with its name. The default installation provides `SweaveSyntaxNoweb` and `SweaveSyntaxLatex`.
`...`	Further arguments passed to the driver's setup function.

Details

Automatic generation of reports by mixing word processing markup (like latex) and S code. The S code gets replaced by its output (text or graphs) in the final markup file. This allows to re-generate a report if the input data change and documents the code to reproduce the analysis in the same file that also produces the report.

`Sweave` combines the documentation and code chunks together (or their output) into a single document. `Stangle` extracts only the code from the Sweave file creating a valid S source file (that can be run using `source`). Code inside `\Sexpr{}` statements is ignored by `Stangle`.

`Stangle` is just a frontend to `Sweave` using a simple driver by default, which discards the documentation and concatenates all code chunks the current S engine understands.

Hook Functions

Before each code chunk is evaluated, a number of hook functions can be executed. If `getOption("SweaveHooks")` is set, it is taken to be a collection of hook functions. For each logical option of a code chunk (`echo`, `print`, ...) a hook can be specified, which is executed if and only if the respective option is `TRUE`. Hooks must be named elements of the list returned by `getOption("SweaveHooks")` and be functions taking no arguments. E.g., if option `"SweaveHooks"` is defined as `list(fig = foo)`, and `foo` is a function, then it would be executed before the code in each figure chunk. This is especially useful to set defaults for the graphical parameters in a series of figure chunks.

Note that the user is free to define new Sweave options and associate arbitrary hooks with them. E.g., one could define a hook function for option `clean` that removes all objects in the global environment. Then all code chunks with `clean=TRUE` would start operating on an empty workspace.

Syntax Definition

Sweave allows a very flexible syntax framework for marking documentation and text chunks. The default is a noweb-style syntax, as alternative a latex-style syntax can be used. See the user manual for details.

Author(s)

Friedrich Leisch

References

Friedrich Leisch: Sweave User Manual, 2002
`http://www.ci.tuwien.ac.at/~leisch/Sweave`

Friedrich Leisch: Dynamic generation of statistical reports using literate data analysis. In W. Hrdle and B. Rnz, editors, Compstat 2002 - Proceedings in Computational Statistics, pages 575-580. Physika Verlag, Heidelberg, Germany, 2002. ISBN 3-7908-1517-9.

See Also

`RweaveLatex`, `Rtangle`

Examples

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")

## create a LaTeX file
Sweave(testfile)

## create an S source file from the code chunks
Stangle(testfile)
## which can be simply sourced
source("Sweave-test-1.R")
```

4.1 The `RweaveLatex` driver

This driver transforms `.Snw` files with LaTeX documentation chunks and R code chunks to proper LaTeX files (for typesetting both with standard `latex` or `pdflatex`).

`RweaveLatex` *R/LaTeX Driver for Sweave*

Description

A driver for `Sweave` that translates R code chunks in LaTeX files.

Usage

```
RweaveLatex()

RweaveLatexSetup(file, syntax, output = NULL, quiet = FALSE,
                 debug = FALSE, echo = TRUE, eval  =  TRUE,
                 split = FALSE, stylepath = TRUE,
                 pdf = TRUE, eps = TRUE)
```

Arguments

`file`	Name of Sweave source file.
`syntax`	An object of class `SweaveSyntax`.
`output`	Name of output file, default is to remove extension '.nw', '.Rnw' or '.Snw' and to add extension '.tex'. Any directory names in `file` are also removed such that the output is created in the current working directory.
`quiet`	If `TRUE` all progress messages are suppressed.
`debug`	If `TRUE`, input and output of all code chunks is copied to the console.
`stylepath`	If `TRUE`, a hard path to the file 'Sweave.sty' installed with this package is set, if `FALSE`, only \usepackage{Sweave} is written. The hard path makes the TeX file less portable, but avoids the problem of installing the current version of 'Sweave.sty' to some place in your TeX input path. The argument is ignored if a \usepackage{Sweave} is already present in the Sweave source file.
`echo`	set default for option `echo`, see details below.
`eval`	set default for option `eval`, see details below.
`split`	set default for option `split`, see details below.
`pdf`	set default for option `pdf`, see details below.
`eps`	set default for option `eps`, see details below.

Supported Options

RweaveLatex supports the following options for code chunks (the values in parentheses show the default values):

echo: logical (`TRUE`). Include S code in the output file?

eval: logical (`TRUE`). If `FALSE`, the code chunk is not evaluated, and hence no text or graphical output produced.

results: character string (`verbatim`). If `verbatim`, the output of S commands is included in the verbatim-like Soutput environment. If `tex`, the output is taken to be already proper latex markup and included as is. If `hide` then all output is completely suppressed (but the code executed during the weave).

print: logical (`FALSE`) If `TRUE`, each expression in the code chunk is wrapped into a `print()` statement before evaluation, such that the values of all expressions become visible.

term: logical (`TRUE`). If `TRUE`, visibility of values emulates an interactive R session: values of assignments are not printed, values of single objects are printed. If `FALSE`, output comes only from explicit `print` or `cat` statements.

split: logical (`FALSE`). If `TRUE`, text output is written to separate files for each code chunk.

strip.white: logical (`TRUE`). If `TRUE`, blank lines at the beginning and end of output are removed.

prefix: logical (`TRUE`). If `TRUE` generated filenames of figures and output have a common prefix.

prefix.string: a character string, default is the name of the '.Snw' source file.

include: logical (`TRUE`), indicating whether input statements for text output and includegraphics statements for figures should be auto-generated. Use `include = FALSE` if the output should appear in a different place than the code chunk (by placing the input line manually).

fig: logical (`FALSE`), indicating whether the code chunk produces graphical output. Note that only one figure per code chunk can be processed this way.

eps: logical (`TRUE`), indicating whether EPS figures shall be generated. Ignored if `fig = FALSE`.

pdf: logical (`TRUE`), indicating whether PDF figures shall be generated. Ignored if `fig = FALSE`.

width: numeric (6), width of figures in inch.

height: numeric (6), height of figures in inch.

Author(s)

Friedrich Leisch

References

Friedrich Leisch: Sweave User Manual, 2002
`http://www.ci.tuwien.ac.at/~leisch/Sweave`

See Also

`Sweave`, `Rtangle`

4.1.1 Writing to separate files

If `split` is set to `TRUE`, then all text corresponding to code chunks (the `Sinput` and `Soutput` environments) is written to seperate files. The filenames are of form `prefix.string-label.tex`, if several code chunks have the same label, their outputs are concatenated. If a code chunk has no label, then the number of the chunk is used instead. The same naming scheme applies to figures.

4.1.2 LaTeX style file and figure sizes

The driver automatically inserts a `\usepackage{.../Sweave.sty}` command as last line before the `\begin{document}` statement of the final LaTeX file if no `\usepackage{Sweave}` is found in the Sweave source file. This style file defines the environments `Sinput` and `Soutput` for typesetting code chunks. If you do not want to include the standard style file, e.g., because you have your own definitions for Sinput and Soutput environemts in a different place, simply insert a comment like

```
% \usepackage{Sweave}
```

in the preamble of your latex file, this will prevent automatic insertion of the line.

`Sweave.sty` also sets the default *LaTeX* figure width (which is independent of the size of the generated EPS and PDF files). The current default is

```
\setkeys{Gin}{width=0.8\textwidth}
```

if you want to use another width for the figures that are automatically generated and included by Sweave, simply add a line similar to the one above *after* `\begin{document}`. Note that a new graphics device is opened for each figure chunk (option `fig=TRUE`), hence all graphical parameters of the `par()` command must be set in each single figure chunk and are forgotten after the respective chunk (because the device is closed when leaving the chunk).

Attention: One thing that gets easily confused are the width/height parameters of the R graphics devices and the corresponding arguments to the LaTeX `\includegraphics` command. The Sweave options `width` and `height` are passed to the R graphics devices, and hence affect the default size of the produced EPS and PDF files. They do not affect the size of figures in the document, by default they will always be 80% of the current text width. Use `\setkeys{Gin}` to modify figure sizes or use explicit `\includegraphics` commands in combination with Sweave option `include=FALSE`.

We need example code for that

4.1.3 Prompts and text width

As of R version 1.6.0 the driver gets the prompts used for input lines and continuation lines from R's `options()` settings. To set new prompts use somthing like

```
options(prompt="MyR> ", continue="...")
```

see `help(options)` for details. Similarly the text width is controlled by option `"width"`.

4.2 The `Rtangle` driver

This driver can be used to extract S and R code chunks from a `.Snw` file. Code chunks can either be written to one large file or seperate files (one for each label). The options `split`, `prefix`, `prefix.string` and `engine` have the same defaults and interpretation as for the `RweaveLatex` driver. Use the standard noweb command line tool `notangle` if other chunks than R or S code should be extracted.

`Rtangle` *R Driver for Stangle*

Description

A driver for `Stangle` that extracts R code chunks.

Usage

```
Rtangle()
RtangleSetup(file, syntax, output = NULL, annotate = TRUE,
             split = FALSE, prefix = TRUE, quiet = FALSE)
```

Arguments

`file`	Name of Sweave source file.
`syntax`	An object of class `SweaveSyntax`.
`output`	Name of output file, default is to remove extension '.nw', '.Rnw' or '.Snw' and to add extension '.R'. Any directory names in `file` are also removed such that the output is created in the current working directory.
`annotate`	By default, code chunks are seperated by comment lines specifying the names and numbers of the code chunks. If `FALSE`, only the code chunks without any decorating comments are extracted.

`split`	Split output in single files per code chunk?
`prefix`	If `split = TRUE`, prefix the chunk labels by the basename of the input file to get output file names?
`quiet`	If `TRUE` all progress messages are suppressed.

Author(s)

Friedrich Leisch

References

Friedrich Leisch: Sweave User Manual, 2002
`http://www.ci.tuwien.ac.at/~leisch/Sweave`

See Also

`Sweave`, `RweaveLatex`

Acknowledgements

The author wants to thank Vince Carey, Robert Gentleman, Kurt Hornik, Markus Jäntti, Brian Ripley, Anthony Rossini, and Dietrich Trenkler for providing valuable comments and ideas, testing development versions of the software or fixing bugs.

A Frequently Asked Questions

A.1 Where can I find the manual and other information on Sweave?

The newest version of the Sweave manual can always be found at the Sweave homepage

`http://www.ci.tuwien.ac.at/~leisch/Sweave`

where you also find several example files, and the lisp and shell code snippets of the FAQ. In addition, the homepage has several papers on Sweave like the CompStat paper and the 2-part miniseries from R News (Issues 2/3 and 2/3).

A.2 How can I get Emacs to automatically recognize files in Sweave format?

Include something like the following in your `.emacs` file:

```
(defun Rnw-mode ()
  (require 'ess-noweb)
  (noweb-mode)
  (if (fboundp 'R-mode)
      (setq noweb-default-code-mode 'R-mode)))

(add-to-list 'auto-mode-alist '("\\.Rnw\\'" . Rnw-mode))
(add-to-list 'auto-mode-alist '("\\.Snw\\'" . Rnw-mode))

(setq reftex-file-extensions
      '(("Snw" "Rnw" "nw" "tex" ".tex" ".ltx") ("bib" ".bib")))
(setq TeX-file-extensions
      '("Snw" "Rnw" "nw" "tex" "sty" "cls" "ltx" "texi" "texinfo"))
```

The first block of commands makes emacs to automatically load the ESS version of noweb mode (with R mode the default code mode) for files with extension `.Rnw` and `.Snw`. The last two lines make RefTeX work on noweb files. You can download the code from `http://www.ci.tuwien.ac.at/~leisch/Sweave/sweave-site.el`.

A.3 Can I run Sweave directly from a shell?

E.g., for writing makefiles it can be useful to run Sweave directly from a shell rather than manually start R and then run Sweave. This can easily be done using a simple shell script along the lines of

```
#!/bin/sh
echo "library(tools); Sweave(\"$1\")" | R --no-save --no-restore
```

A.4 Why does LaTeX not find my EPS and PDF graphic files when the filename contains a dot?

Sweave uses the standard LaTeX package `graphicx` to handle graphic files, which automatically uses EPS files for standard LaTeX and PDF files for PDFLaTeX, if the name of the input file has no extension, i.e., contains no dots. Hence, you may run into trouble with graphics handling if the name of your Sweave file contains extra dots: 'foo.Rnw' is OK, while 'foo.bar.Rnw' is not.

A.5 Why does Sweave by default create both EPS and PDF graphic files?

The LaTeX package `graphicx` needs EPS files for plain LaTeX, but PDF files for PDFLaTeX (the latter can also handle PNG and JPEG files). Sweave automatically creates graphics in EPS and PDF format, such that the user can freely run `latex` or `pdflatex` on the final document as needed.

A.6 Why do R lattice graphics not work?

The commands in package `lattice` have different behavior than the standard plot commands in the `base` package: lattice commands return an object of class `"trellis"`, the actual plotting is performed by the `print` method for the class. Encapsulating calls to lattice functions in `print()` statements should do the trick, e.g.:

```
<<fig=TRUE>>=
library(lattice)
print(bwplot(1:10))
@
```

should work. Future versions of Sweave may have more automated means to deal with trellis graphics.

A.7 How can I get Black & White lattice graphics?

What is the most elegant way to specify that strip panels are to have transparent backgrounds and graphs are to be in black and white when lattice is being used with Sweave? I would prefer a global option that stays in effect for multiple plots.

Answer by Deepayan Sarkar: I'd do something like this as part of the initialization:

```
<<...>>
library(lattice)
ltheme <- canonical.theme(color = FALSE)     ## in-built B&W theme
ltheme$strip.background$col <- "transparent" ## change strip bg
lattice.options(default.theme = ltheme)      ## set as default
@
```

A.8 Creating several figures from one figure chunk does not work

Consider that you want to create several graphs in a loop similar to

```
<<fig=TRUE>>
for (i in 1:4) plot(rnorm(100)+i)
@
```

This will currently **not** work, because Sweave allows **only one graph** per figure chunk. The simple reason is that Sweave opens a postscript device before executing the code and closes it afterwards. If you need to plot in a loop, you have to program it along the lines of

```
<<results=tex,echo=FALSE>>=
for(i in 1:4){
    file=paste("myfile", i, ".eps", sep="")
    postscript(file=file, paper="special", width=6, height=6)
    plot(rnorm(100)+i)
    dev.off()
    cat("\\includegraphics{", file, "}\n\n", sep="")
}
@
```

A.9 How can I place all those auto-generated graphics files in a subdirectory rather than the same directory as the Sweave file?

After

```
\SweaveOpts{prefix.string=foo/bar}
```

Sweave will place all figures in subdirectory `foo` and their name will start with `bar` (instead of the name of the Sweave file). The subdirectory `foo` should exist before you run Sweave.

A.10 How can I set default `par()` settings for figure chunks?

Because each EPS and PDF file opens a new device, using `par()` has only an effect if it is used inside a figure chunk. If you want to use the same settings for a series of figures, it is easier to use a hook function than repeating the same `par()` statement in each figure chunk.

The effect of

```
options(SweaveHooks=list(fig=function() par(bg="red", fg="blue")))
```

should be easy to spot. Do not forget to remove the hook at the end of the Sweave file unless you want to use it as a global option for all Sweave files.

A.11 Running `latex` fails on Windows

If you can create the '.tex' file by running `Sweave()` in R, but cannot convert the '.tex' file to '.dvi' or '.pdf', this is most likely caused by a space in the path of your R installation. If the path of your R installation contains any blank characters (like the default `"c:\Program Files\..."` in English versions of Windows), this may cause problems, because programs like `tex` or `latex` cannot handle blanks in paths properly.

Two possible solutions:

1. Install R in a path not containing any blanks.
2. Copy the file 'Sweave.sty' to a directory in your tex path or the directory containing the Sweave file and put a `\usepackage{Sweave}` into the preamble of your Sweave file.

A.12 How can I change the formatting of S input and output chunks?

Sweave uses the `fancyvrb` package for formatting all S code and text output. `fancyvrb` is a very powerful and flexible package that allows fine control for layouting text in verbatim environments. If you want to change the default layout, simply read the `fancyvrb` documentation and modify the definitions of the `Sinput` and `Soutput` environments in 'Sweave.sty', respectively.

A.13 How can I change the line length of S input and output?

Sweave respects the usual way of specifying the desired line length in S, namely `options(width)`. E.g., after `options(width=40)` lines will be formatted to have at most 40 characters (if possible).

A.14 Why are chunks with `eval=FALSE` executed by `R CMD check`?

This is a feature, not a bug. From the *"Writing R Extensions"* manual:

> Package vignettes found in directory `inst/doc` are tested by `R CMD check` by executing *all* R code chunks they contain to ensure consistency between code and documentation. Note that even code chunks with option `eval=FALSE` are tested, 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.

A.15 Can I use Sweave for HTML files?

Package `R2HTML` provides a driver for using Sweave in combination with HTML rather than LaTeX.

A.16 After loading package `R2HTML` Sweave doesn't work properly!

Package `R2HTML` registers an Sweave driver for HTML files, and after that the Syntax for HTML is in the search list before the default syntax.

```
options(SweaveSyntax="SweaveSyntaxNoweb")
```

or calling Sweave like

```
Sweave(..., syntax="SweaveSyntaxNoweb")
```

ensures the default syntax even after loading `R2HTML`.

References

Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The New S Language*. Chapman & Hall, London, UK, 1988.

John M. Chambers. *Programming with data: A guide to the S language*. Springer Verlag, Berlin, Germany, 1998.

Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Hrdle and Bernd Rnz, 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.

R Development Core Team. *R: A language and environment for statistical computing*, 2002. URL `http://www.R-project.org`. version 1.6.1.

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

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).

Richard M. Stallman. *The Emacs Editor*. Free Software Foundation, Boston, MA, USA, 1999. URL `http://www.gnu.org`. version 20.7.

William N. Venables and Brian D. Ripley. *S Programming*. Springer, 2000.