Sweave User Manual

Friedrich Leisch, Vincent Goulet and R Core Team

October 30, 2026

1 Introduction

Sweave provides a flexible framework for mixing text and R code for automatic document generation. A single source file contains both documentation text and R code. The file may then either be *woven* into a final document containing the documentation text together with the R code and its output (text, graphs), or *tangled* to extract the R code to a script.

This literate programming system (Knuth, 1984) allows the re-generation of a report if the input data change while also documenting the code to reproduce the analysis in the same file that contains the report. The R code of the complete analysis is embedded into a LaTeX document[1] using the `noweb` syntax (Ramsey, 2018). Hence, the full power of LaTeX (for high-quality typesetting) and R (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.

Sweave uses a modular concept using different drivers for the actual translations. Obviously different drivers are needed for different text markup languages (LaTeX, HTML, ...). Several packages on CRAN provide support for other word processing systems (see Appendix A).

2 Noweb files

`noweb` (Ramsey, 2018) is a simple literate-programming tool which allows combining program source code and the corresponding documentation into a single file. A `noweb` file is a simple text file which consists of a sequence of code and documentation segments, 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. The chunk continues until a new code or documentation chunk is started: note that Sweave does not interpret the contents of documentation chunks, and so will identify chunk indicators even inside LaTeX verbatim environments.

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

The default for the first chunk is documentation.

In the simplest usage of `noweb`, the options (if present) of the code chunks give the names of source code files, and the tool `notangle` can be used to extract the code chunks from the `noweb` file. Multiple code chunks can have the same name, the corresponding code chunks are then 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 nor support these features, so they are not described here.

[1] https://www.ctan.org

3 Sweave files

Sweave source files are `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 '`.Rnw`' throughout this document.

3.1 A simple example

A minimal Sweave file is shown in Figure 1. The file contains three code chunks embedded in a simple LaTeX document. Running the following expressions creates the LaTeX document shown in Figure 2.

```
> rnwfile <- system.file("Sweave", "example-1.Rnw", package = "utils")
> Sweave(rnwfile)

Writing to file example-1.tex
Processing code chunks with options ...
 1 : echo keep.source term verbatim (example-1.Rnw:12)
 2 : echo keep.source (label = boxp, example-1.Rnw:22)
 3 : keep.source term verbatim pdf  (example-1.Rnw:27)

You can now run (pdf)latex on 'example-1.tex'
```

Compiling this LaTeX document results in the PDF in Figure 3. The latter can also be created directly from within R using

```
> tools::texi2pdf("example-1.tex")
```

The first difference between '`example-1.Rnw`' and '`example-1.tex`' is that the LaTeX style file '`Sweave.sty`' is automatically loaded, which provides environments for typesetting R input and output (the LaTeX environments `Sinput` and `Soutput`). Otherwise, the documentation chunks are copied without any modification from '`example-1.Rnw`' to '`example-1.tex`'.

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 R commands and their respective output to the LaTeX file, embedded in `Sinput` and `Soutput` environments, respectively.

The second and third code chunks show one of the Sweave extensions to the `noweb` syntax: code chunk names can be used to pass options to Sweave which control the final output.

- The option `eval=FALSE` for the second chunk indicates that the code should not be evaluated (yet). The code is reused in the third chunk (details follow in Section 3.5).
- The third chunk is marked as a figure chunk (`fig=TRUE`) such that Sweave creates (by default) a PDF file of the plot created by the commands in the chunk. Furthermore, the statement `\includegraphics{example-1-003}` is inserted into the LaTeX file (details on the choice of file names for figures follow later in this manual).
- Option `echo=FALSE` indicates that the R input should not be included in the final document (no `Sinput` environment).

3.2 Sweave options

Options control how code chunks and their output (text, figures) are transferred from the '`.Rnw`' file to the '`.tex`' file. All options have the form *key=value*, where *value* can be a number, string or logical value. Several options can be specified at once (separated by commas), all options

```
\documentclass[a4paper]{article}

\title{Simple Sweave Example}
\author{Friedrich Leisch, Vincent Goulet and R Core Team}

\begin{document}

\maketitle

In this simple example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:
<<>>=
data(airquality, package="datasets")
library("stats")
kruskal.test(Ozone ~ Month, data = airquality)
@

The location parameter of the Ozone distribution varies
significantly from month to month. We include a boxplot of the
data with the following code:
%% want an eval=FALSE case and referencing a previous chunk:
<<boxp, eval=FALSE>>=
boxplot(Ozone ~ Month, data = airquality)
@
The resulting plot is displayed below.
\begin{center}
<<fig=TRUE, echo=FALSE>>=
library("graphics")
<<boxp>>
@
\end{center}

\end{document}
```

Figure 1: A minimal Sweave file: '`example-1.Rnw`'.

```
\documentclass[a4paper]{article}

\title{Simple Sweave Example}
\author{Friedrich Leisch, Vincent Goulet and R Core Team}

\usepackage{Sweave}
\begin{document}

\maketitle

In this simple example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:
\begin{Schunk}
\begin{Sinput}
> data(airquality, package="datasets")
> library("stats")
> kruskal.test(Ozone ~ Month, data = airquality)
\end{Sinput}
\begin{Soutput}
        Kruskal-Wallis rank sum test

data:  Ozone by Month
Kruskal-Wallis chi-squared = 29.267, df = 4, p-value = 6.901e-06
\end{Soutput}
\end{Schunk}

The location parameter of the Ozone distribution varies
significantly from month to month. We include a boxplot of the
data with the following code:
%% want an eval=FALSE case and referencing a previous chunk:
\begin{Schunk}
\begin{Sinput}
> boxplot(Ozone ~ Month, data = airquality)
\end{Sinput}
\end{Schunk}
The resulting plot is displayed below.
\begin{center}
\includegraphics{example-1-003}
\end{center}

\end{document}
```

Figure 2: Running `Sweave("example-1.Rnw")` produces the file '`example-1.tex`'.

Simple Sweave Example

Friedrich Leisch, Vincent Goulet and R Core Team

October 30, 2026

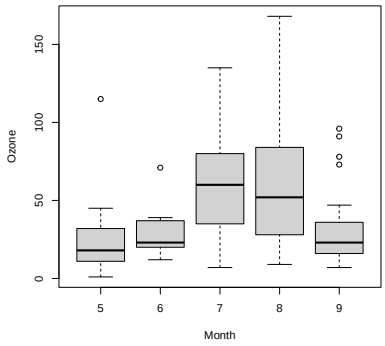
In this simple example we embed parts of the examples from the `kruskal.test` help page into a LaTeX document:

```
> data(airquality, package="datasets")
> library("stats")
> kruskal.test(Ozone ~ Month, data = airquality)

        Kruskal-Wallis rank sum test

data:  Ozone by Month
Kruskal-Wallis chi-squared = 29.267, df = 4, p-value = 6.901e-06
```

The location parameter of the Ozone distribution varies significantly from month to month. We include a boxplot of the data with the following code:

```
> boxplot(Ozone ~ Month, data = airquality)
```

The resulting plot is displayed below.



1

Figure 3: The final document '`example-1.pdf`' created by compiling '`example-1.tex`'.

must take a value (which must not contain a comma or equal sign). Logical options can take the values `TRUE`, `FALSE`, `T`, `F` as well as lower-case and capitalized versions of the first two.

In the '`.Rnw`' 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.

Further, global options can be specified (as a comma-separated list of *key=value* items) in the call to the function `Sweave`, via the '`--options=`' flag of `R CMD Sweave`, and in the environment variable `SWEAVE_OPTIONS`.

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):

`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 file-name creation when `split=TRUE`. It is also used in the comments added above the chunk when the file is processed by `Rtangle` (provided the `annotate` option is true, as it is by default), and as part of the file names for files generated by figure chunks. Because labels can be part of file names, they should contain only alphanumeric characters and `#+-_`. (Including the period `.` can cause confusion with file extensions.)

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. Therefore, starting a code chunk with

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

is the same as

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

but

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

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 file names appear in code chunk names).

Note that `split=TRUE` should *not* be used in package vignettes.

The help pages for `RweaveLatex` and `Rtangle` list the options supported by the default `Sweave` and `Stangle` drivers.

Now for the promised details on the file names corresponding to figures. These are of the form *prefix*`-`*label*`.`*ext*.

- *prefix* can be set by the option `prefix.string`, otherwise is the basename of the output file (which unless specified otherwise is the basename of the input file).
- *label* is the label of the code chunk if it has one, otherwise the number of the code chunk in the range `001` to `999`.

- *ext* is the appropriate extension for the type of graphics, e.g. `pdf`, `eps`, ...

So for the '`example-1.Rnw`' file, the PDF version of the boxplot is '`example-1-003.pdf`', since it is the third code chunk in the document. Note that if `prefix.string` is used it can specify a directory as part of the prefix. For example, using `\SweaveOpts{prefix.string=figures/fig}` will result in the auto-generated figures to be placed in the subdirectory `figures` (the Sweave document should arrange to create this directory before use).

3.3 Using multiple input files

LaTeX files can include others via `\input{}` commands. These can also be used in Sweave files, but the included files will be included by LaTeX and are not processed by Sweave. The equivalent if you want the included files to be processed is the `\SweaveInput{}` command.

Included files should use the same Sweave syntax (see below) and encoding as the main file.

3.4 Using scalars in text

There is limited support for using the values of R 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 which being used. For example, `3` will be replaced by the string `3` in the LaTeX document.

The expression in `\Sexpr` is evaluated in the same environment as the code chunks. Therefore, one can access all objects defined in the code chunks which have appeared before the expression and were evaluated. The expression may contain any valid R code, only braces (`{ }`) 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 `\Sexpr`.)

3.5 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 its argument: no additional Sweave options are allowed. It has to occur at the beginning of a line.

References to unknown chunk names are omitted, with a warning.

3.6 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. It can be created using

```
> SweaveSyntConv(rnwfile, SweaveSyntaxLatex)
```

```
\documentclass[a4paper]{article}

\title{Simple Sweave Example}
\author{Friedrich Leisch, Vincent Goulet and R Core Team}

\begin{document}

\maketitle

In this simple example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX{} document:
\begin{Scode}{}
data(airquality, package="datasets")
library("stats")
kruskal.test(Ozone ~ Month, data = airquality)
\end{Scode}

The location parameter of the Ozone distribution varies
significantly from month to month. We include a boxplot of the
data with the following code:
%% want an eval=FALSE case and referencing a previous chunk:
\begin{Scode}{boxp, eval=FALSE}
boxplot(Ozone ~ Month, data = airquality)
\end{Scode}
The resulting plot is displayed below.
\begin{center}
\begin{Scode}{fig=TRUE, echo=FALSE}
library("graphics")
\Scoderef{boxp}
\end{Scode}
\end{center}

\end{document}
```

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

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 `noweb`-style syntax.

Which syntax is used for a document is determined by the extension of the input file, files with extension[2] '`.Rtex`' or '`.Stex`' are assumed to follow the LaTeX-style syntax. Alternatively

[2]The lowercase versions are also supported for convenience on case-insensitive file systems.

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.

3.7 Encoding

LaTeX documents written in most languages other than English usually require a special input encoding to support accented characters and various symbols beyond those of ASCII. The modern way to deal with this situation is to use the UTF-8 input encoding and compile with LuaLaTeX.

As from R 3.1.0, the UTF-8 encoding is handled preferentially to other encodings. Sweave will leave UTF-8 code in that encoding, and output in that encoding as well. However, it remains required to tell Sweave that a file uses this encoding. This can be done through either the argument `encoding` of the function `Sweave` or the flag '`--encoding`' of `R CMD Sweave`, or by inserting the comment

```
%\SweaveUTF8
```

in the file to process.

If you are still using pdfLaTeX with non-ASCII documents, you should be using the LaTeX package **inputenc**, with a statement such as

```
\usepackage[utf8]{inputenc}
```

in the document's preamble. There is a wide range of input encodings which are supported, at least partially, and the more recent LaTeX package **inputenx** supports more. Other encodings commonly encountered in Sweave documents are `latin1`, `latin9` and `ansinew` (the CP1252 codepage on Windows). Greater coverage[3] of UTF-8 can be obtained by

```
\usepackage[utf8]{inputenx}
\input{ix-utf8enc.dfu}
```

The previous paragraphs covered the documentation sections, but what of the code sections where LaTeX escapes cannot be used? The first piece of advice is to use only ASCII code sections as anything else will reduce the portability of your document. But R code can be included if in the input encoding declared in the preamble.

The next complication is inclusions: the final vignette may well include R output, LaTeX or other Sweave files *via* `\input{}` and `\SweaveInput{}` lines, a bibliography and figures. It is the user's responsibility that the text inclusions are covered by the declared input encoding. LaTeX allows the input encoding to be changed by

```
\inputencoding{something}
```

statements: these may not work well in Sweave processing. Since LaTeX package loading is typically not legal LaTeX code in an included file, it is easiest to declare the encoding to Sweave using the `%\SweaveUTF8` comment.

It is all too easy for BibTeX to pick up UTF-8-encoded references for a Latin-1 document, or *vice versa*.

[3] Including Eastern European, Greek and Hebrew letters.

R output is again to a large extent under the user's control. If a Latin-1 Sweave document is processed by R running a Latin-1 locale or a UTF-8 document is processed in a UTF-8 locale, the only problems which are likely to arise are from handling data in another encoding, but it may be necessary to declare the document's encoding to cover the R output which is generated even if the document is itself ASCII. One common problem is the quotes produced by `sQuote()` and `dQuote()`: these will be in UTF-8 when R is run in a UTF-8 locale, and will be in CP1252 when Sweave is run from `Rgui.exe` on Windows. Two possible solutions are to suppress fancy quotes by `options(useFancyQuotes=FALSE)` or to force UTF-8 by `options(useFancyQuotes="UTF-8")`.

The encoding of figures is not usually an issue as they are either bitmaps or include encoding information: it may be necessary to use the `pdf.encoding` Sweave option to set the `pdf()` device up appropriately.

4 Weaving and tangling

The user front-ends of the Sweave system are the two R functions `Sweave` and `Stangle`, both contained in package **utils**. `Sweave` runs the code chunks through R and replaces them with any combination of their input and output. `Stangle` extracts only the code chunks from an '`.Rnw`' file and writes them out to one or several files. `Stangle` is actually just a wrapper function for `Sweave`, which uses a tangling instead of a weaving driver by default. See

```
> help("Sweave")
```

for more details and arguments of the functions.

4.1 The `RweaveLatex` driver

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

```
> help("RweaveLatex")
```

for details, including the supported options.

4.1.1 Using objects as chunk options

Starting with R 4.6.0, the `RweaveLatex` driver allows the *value* of logical and numerical chunk options (only) to be the name of an object defined in earlier, evaluated code chunks. This is useful to pass computed values to options. The Sweave file in Figure 5 provides two examples of use. Processing the file with

```
> rnwfile <- system.file("Sweave", "example-2.Rnw", package = "utils")
> Sweave(rnwfile)
> tools::texi2pdf("example-2.tex")
```

creates the PDF in Figure 6. Note that the second code chunk was not evaluated, and how the proportions of the plot differ from the otherwise same one in Figure 3.

4.1.2 Unparsable code in code chunks

In traditional Sweave usage, the expressions in code chunks are always parsed on weaving, whether option `eval` is `TRUE` or `FALSE`. Furthermore, code chunks with `eval=FALSE` are automatically and irrevocably commented out on tangling. These conventions may prove limiting in uses of literate programming beyond pure statistical analysis. For example, they make impossible to include in a code chunk, say for didactic purposes, invalid R code that should appear uncommented in the tangled script.

```
\documentclass[a4paper]{article}

\title{Using objects as chunk options}
\author{Vincent Goulet and R Core Team}

\begin{document}

\maketitle

The following code chunk checks if a package \textbf{foo} is
available on the system, and defines two dimensions.
<<>>=
hasfoo <- requireNamespace("foo", quietly = TRUE)
ht <- 5
wd <- (1 + sqrt(5))/2 * ht
@

Sweave will evaluate the following code chunk only if the package
is available (which should normally not be the case).
<<eval=hasfoo>>=
foo::foo(42)
@

We may also redo the boxplot of the \texttt{airquality} dataset
in proportions approximately equal to the golden ratio.
\begin{center}
<<boxplot, echo=FALSE, fig=TRUE, width=wd, height=ht>>=
boxplot(Ozone ~ Month, data = airquality)
@
\end{center}

\end{document}
```

Figure 5: Sweave file '`example-2.Rnw`' with options taking their values from objects in the second and third code chunks.

Using objects as chunk options

Vincent Goulet and R Core Team

October 30, 2026

The following code chunk checks if a package **foo** is available on the system, and defines two dimensions.

```
> hasfoo <- requireNamespace("foo", quietly = TRUE)
> ht <- 5
> wd <- (1 + sqrt(5))/2 * ht
```

Sweave will evaluate the following code chunk only if the package is available (which should normally not be the case).

```
> foo::foo(42)
```

We may also redo the boxplot of the `airquality` dataset in proportions approximately equal to the golden ratio.

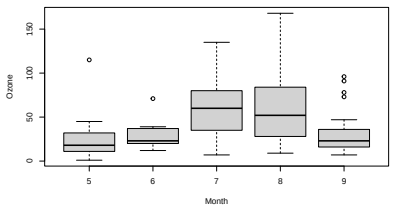


Figure 6: The final document '`example-2.pdf`' created from '`example-2.Rnw`'.

To ease with this sort of scenario, R 4.6.0 introduced options for the standard drivers to completely ignore code chunks on weaving, tangling, or both. With the `RweaveLatex` driver, chunks using the option `ignore.on.weave=TRUE` are not parsed on weaving, but remain written out verbatim on tangling. A shorter alias `weave` for `!ignore.on.weave` is also supported. An option `ignore` also sets at once `ignore.on.weave` and the equivalent option for `Rtangle`.

The Sweave file in Figure 7 provides examples of use of the option `ignore.on.weave` (and of `Rtangle` options that are covered later in Section 4.2).

```
> rnwfile <- system.file("Sweave", "example-3.Rnw", package = "utils")
> Sweave(rnwfile)
> tools::texi2pdf("example-3.tex")
```

Note the "missing" code chunks in the PDF output shown in Figure 8.

4.1.3 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 separate files. The file names 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. You do need to ensure that the file names generated are valid and not so long that there are not regarded as the same by your OS (some file systems only recognize 13 characters).

4.1.4 LaTeX style file

The driver automatically inserts the statement `\usepackage{Sweave}` as the last line before `\begin{document}` in the final LaTeX file if the statement is not found in the Sweave source file. The style file '`Sweave.sty`' defines the environments `Sinput` and `Soutput` for typesetting code chunks. If you do not want to include the standard style file, perhaps because you have your own definitions for `Sinput` and `Soutput` environments in a different place, simply insert a comment like

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

in the LaTeX preamble of to file to prevent automatic insertion of the line.

4.1.5 Figure sizes

The style file '`Sweave.sty`' sets the default LaTeX figure width (which is independent of the size of the generated EPS or PDF files). The current default is 80% of the current text width as set with:

```
\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}`. If you want no default width for figures, load the style manually with the option `nogin` by inserting the statement

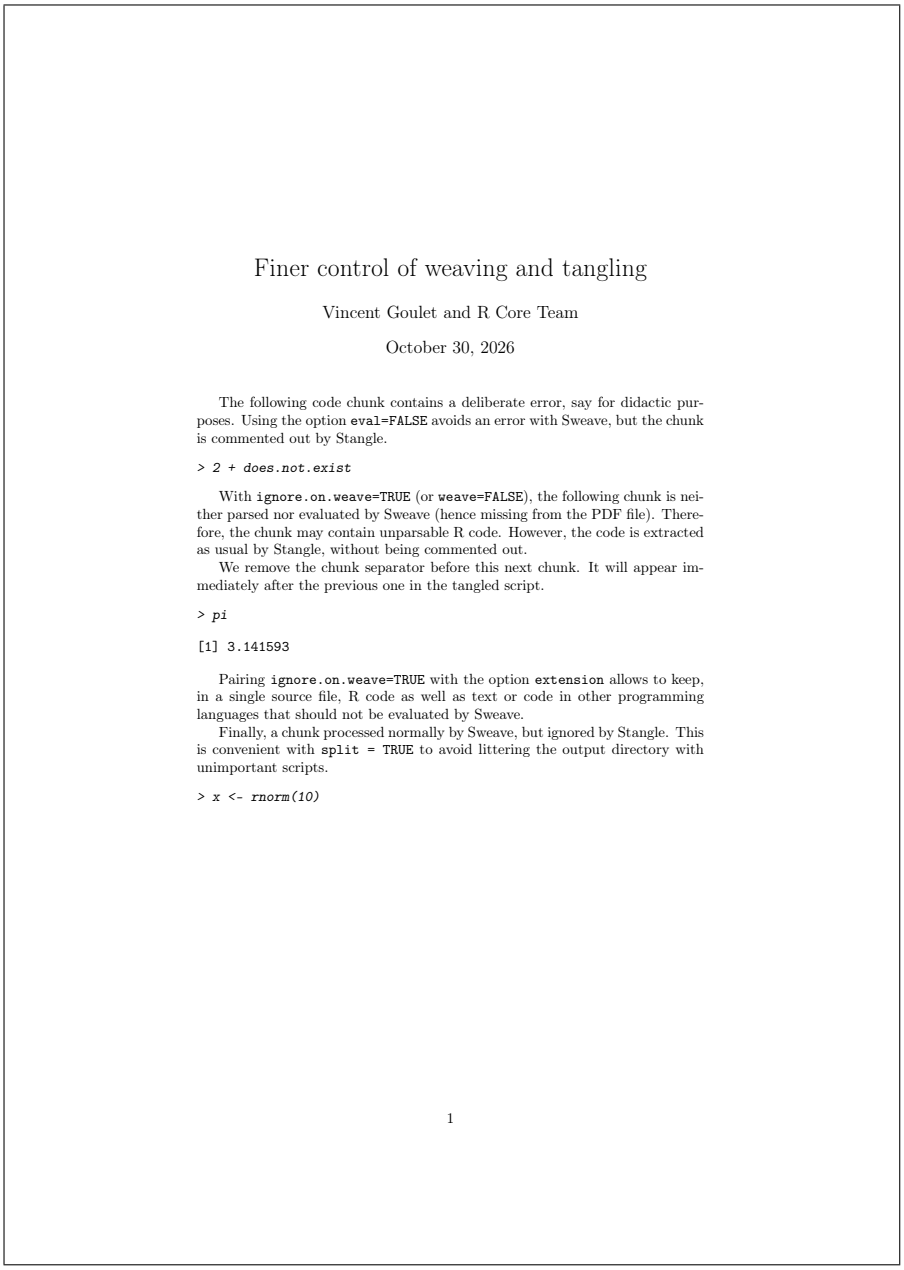
```
\usepackage[nogin]{Sweave}
```

in the preamble.

Note that a new graphics device is opened for each figure chunk (one with 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).

```
\documentclass[a4paper]{article}

\title{Finer control of weaving and tangling}
\author{Vincent Goulet and R Core Team}

\begin{document}

\maketitle

The following code chunk contains a deliberate error, say for
didactic purposes. Using the option \texttt{eval=FALSE} avoids an
error with Sweave, but the chunk is commented out by Stangle.
<<main, eval=FALSE>>=
2 + does.not.exist
@

With \texttt{ignore.on.weave=TRUE} (or \texttt{weave=FALSE}), the
following chunk is neither parsed nor evaluated by Sweave (hence
missing from the PDF file). Therefore, the chunk may contain
unparsable R code. However, the code is extracted as usual by
Stangle, without being commented out.
<<main, ignore.on.weave=TRUE>>=
2 a
@

We remove the chunk separator before this next chunk. It will
appear immediately after the previous one in the tangled script.
<<main, chunk.sep=FALSE>>=
pi
@

Pairing \texttt{ignore.on.weave=TRUE} with the option
\texttt{extension} allows to keep, in a single source file, R
code as well as text or code in other programming languages that
should not be evaluated by Sweave.
<<hello, extension=sh, prefix=FALSE, ignore.on.weave=TRUE>>=
#!/bin/sh
echo "Hello, World!"
exit 0
@
<<README, extension=FALSE, prefix=FALSE, ignore.on.weave=TRUE>>=
Hello, World!
@

Finally, a chunk processed normally by Sweave, but ignored by
Stangle. This is convenient with \texttt{split = TRUE} to avoid
littering the output directory with unimportant scripts.
<<ignore.on.tangle=TRUE>>=
x <- rnorm(10)
@

\end{document}
```

Figure 7: Sweave file '`example-3.Rnw`' with finer control of weaving and tangling.



Finer control of weaving and tangling

Vincent Goulet and R Core Team

October 30, 2026

The following code chunk contains a deliberate error, say for didactic purposes. Using the option `eval=FALSE` avoids an error with Sweave, but the chunk is commented out by Stangle.

```
> 2 + does.not.exist
```

With `ignore.on.weave=TRUE` (or `weave=FALSE`), the following chunk is neither parsed nor evaluated by Sweave (hence missing from the PDF file). Therefore, the chunk may contain unparsable R code. However, the code is extracted as usual by Stangle, without being commented out.

We remove the chunk separator before this next chunk. It will appear immediately after the previous one in the tangled script.

```
> pi
```

```
[1] 3.141593
```

Pairing `ignore.on.weave=TRUE` with the option `extension` allows to keep, in a single source file, R code as well as text or code in other programming languages that should not be evaluated by Sweave.

Finally, a chunk processed normally by Sweave, but ignored by Stangle. This is convenient with `split = TRUE` to avoid littering the output directory with unimportant scripts.

```
> x <- rnorm(10)
```

1

Figure 8: The final document '`example-3.pdf`' created from '`example-3.Rnw`'.

One thing that gets easily confused are the width and height parameters of the R graphics devices and the corresponding arguments to the LaTeX command `\includegraphics`. 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, which is the default mentioned above. If you want to modify the size of included figures, use `\setkeys{Gin}` or explicit `\includegraphics` commands in combination with the Sweave option `include=FALSE`.

4.1.6 Prompts and text width

By default the driver gets the prompts used for input lines and continuation lines from R's `options()` settings. To set new prompts use something like

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

See `help(options)` for details. Similarly the width of the output text is controlled by the R option `"width"`.

4.1.7 Graphics devices

The default graphics device for figure chunks is `pdf()`: the standard options `pdf`, `eps`, `png` and `jpg` allow one or more of these to be selected for a particular chunk or (via `\SweaveOpts`) for the whole document. It can be convenient to select PNG for a particular figure chunk by something like

```
<<figN, fig=TRUE, pdf=FALSE, png=TRUE>>
```

pdfLaTeX and LuaLaTeX should then automatically include the PNG file for that chunk.

You can define your own graphics devices and select one of them by the option `grdevice`. The value of the option is a function device defined in a hidden code chunk. For example, we could make use of the `cairo_pdf` device by using `grdevice=my.Swd` with `my.Swd` defined as:

```
my.Swd <- function(name, width, height, ...)
    grDevices::cairo_pdf(filename = paste(name, "pdf", sep = "."),
                         width = width, height = height)
```

Specialized custom graphics devices may need a customized way to shut them down in place of `graphics.off()`: this can be supplied *via* a companion function `my.Swd.off`, which is called with no arguments.

For a complete example see the file '`src/library/utils/tests/customgraphics.Rnw`' in the R sources.

4.2 The `Rtangle` driver

The driver `Rtangle` can be used to extract R code chunks from a '`.Rnw`' file. Code chunks can either be written to one large file, or to separate files (one for each code chunk). The options `split`, `prefix`, and `prefix.string` have the same defaults and interpretation as for the `RweaveLatex` driver. See

```
> help("Rtangle")
```

for further details.

Note that when using `split=TRUE` with labeled code chunks, the resulting script files sorted alphabetically will most likely not preserve the logical order of the code.

As of R 4.6.0, a number of options give users finer control on the tangling process. First, the option `chunk.sep` allows to specify the separator between code chunks when they are collected

in a single file (two blank lines by default). One may also use `chunk.sep=FALSE` to completely omit the separator (between two chunks when used as a chunk option).

Second, `Rtangle` also gains an option `ignore.on.tangle` to completely omit a code chunk on tangling, irrespective of the value of option `eval` (the option `drop.evalFALSE=TRUE` only omits unevaluated chunks). The driver also supports the shorter alias `tangle` for `!ignore.on.weave`, and the option `ignore` to set both `ignore.on.weave` and `ignore.on.tangle` at once.

Finally, `Rtangle` now provides an option `extension` to specify the extension, without the leading dot, for the file name of a tangled code chunk when splitting is selected. (Using an extension other than `.R` when `split=FALSE` does not appear as a feature worth supporting.) If used as `extension=TRUE`, the default extension (usually `R`) is used. If the option is `FALSE`, no extension is added to the file name.

The option `extension` is specially useful with `ignore.on.weave=TRUE` of `RweaveLatex` to include code or text that the engine would not be able to parse on weaving, yet that needs to be tangled to a file with a specific extension, such as `.sh` or `.txt`.

The Sweave file in Figure 7 takes advantage of the `Rtangle` options above. Tangling the file with

```
> Stangle(rnwfile, split = TRUE, annotate = FALSE, chunk.sep = "\n")

Writing chunks to files ...
1 : example-3-main.R
2 : example-3-main.R
3 : example-3-main.R
4 : hello.sh
5 : README
```

creates the files in Figure 9.

‘example-3-main.R’

```
## 2 + does.not.exist

2 a
pi
```

‘hello.sh’

```
#!/bin/sh
echo "Hello, World!"
exit 0
```

‘README’

```
Hello, World!
```

Figure 9: Files created by tangling ‘example-3.Rnw’.

5 Obtaining the name of the processed file

So far, we presented weaving and tangling as separate, somewhat orthogonal procedures. However, one may devise various schemes where both weaving *and* tangling are needed to create a document. Here are a few examples:

- maintaining documentation and code together even though the code is not directly used to create the text;
- code that needs to be included verbatim in a document, but not necessarily as or where it appears in the sources;
- pieces of code that rely on other pieces being saved as files.

This manual is an example of the third item: the '`.tex`' and PDF files in the figures need to be created from the '`.Rnw`' files before they can be included in this document.

A nice trick to achieve this consists of embedding a tangling procedure *inside* a weaving procedure by calling `Stangle` in a code chunk. Now, this requires the name of the file to process... which is the file being processed. To avoid hard coding the file name in the `Stangle` call, package **utils** provides, since R 4.6.0, a function `SweaveGetSourceName` that returns the name of the file being woven by an `Sweave` process launched from the command line. This allows to start `Stangle` inside `Sweave` with something like:

```
<<echo=FALSE, results=hide>>=
FILE <- SweaveGetSourceName()
Stangle(FILE)
@
```

The function supports weaving processes started by means of either `R CMD Sweave`, `R -e`, or `Rscript -e` (see Appendix A).

6 Adding Sweave Drivers

Adding drivers is relatively simple by modelling them on the existing `RweaveLatex` and `Rtangle` drivers.[4]

An Sweave driver is a function of no arguments which should return a named list of five functions:

`setup(file, syntax, ...)` Set up the driver, e.g. open the output file. Its return value is an object which is passed to the next three functions, and can be updated by the next two. The value should be a list, and contain a component `options`, a named list of the default settings for the options needed by `runcode`.

`runcode(object, chunk, options)` Process a code chunk. Returns a possibly updated `object`. Argument `chunk` is a character vector, and `options` is a options list for this chunk.

`writedoc(object, chunk)` Write out a documentation chunk. Returns a possibly updated `object`.

`finish(object, error)` Finish up, or clean up if `error` is true. Its return value determines that of `Sweave`.

`checkopts(options)` Convert or validate `options` given as a named list of character strings.

Note that the `setup` function should have a `...` argument. It will be passed additional arguments from a `Sweave()` call, but in future `Sweave` may itself set options and pass them on the setup function: one such may be the encoding used for the text to be processed.

[4] If you copy the standard drivers, **do** be careful not to infringe R-core's copyright: the copyright notices in the R sources **must** also be copied.

References

Knuth DE (1984). "Literate Programming." *The Computer Journal*, **27**(2), 97–111. doi: 10.1093/comjnl/27.2.97.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *Compstat 2002 — Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9. doi: 10.1007/978-3-642-57489-4_89.

Ramsey N (2018). "Noweb – A Simple, Extensible Tool for Literate Programming." Noweb home page, Tufts University, USA. URL https://www.cs.tufts.edu/~nr/noweb/.

A Frequently Asked Questions

A.1 How can I get Emacs to automatically recognize files in Sweave format?

ESS[5] automatically recognize files with extension '`.Rnw`' as Sweave files and turn on the correct modes. Please follow the instructions on the ESS homepage on how to install ESS on your computer.

A.2 Can I run Sweave directly from a shell?

You may run Sweave and Stangle directly from a shell, rather than manually starting R and then running calling `Sweave` or `Stangle`, using `R CMD Sweave` and `R CMD Stangle` (these commands take the name of the file to process in argument). For `Sweave` and `Stangle` calls requiring many arguments, it may be simpler to pass the call to R by means of `R -e` or `Rscript -e`.

A.3 Why does LaTeX not find my EPS and PDF graphic files when the file name contains a dot?

Sweave uses the standard LaTeX package **graphicx** to handle graphic files, which automatically chooses the type of file, provided the basename given in `\includegraphics` has no extension. Hence, you may run into trouble with graphics handling if the basename of your Sweave file contains dots: '`foo.Rnw`' is OK, while '`foo.bar.Rnw`' is not.

A.4 Empty figure chunks give LaTeX errors.

When a code chunk with `fig=TRUE` does not call any plotting functions, invalid PDF (or EPS) files may be created. Sweave cannot know if the code in a figure chunk actually plotted something or not, so it will try to include the graphics, which is bound to fail.

A.5 Why do R lattice graphics not work?

In recent versions of Sweave they do if they would when run at the command line: some calls (e.g. those inside loops) need to be explicitly `print()`-ed.

A.6 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
@
```

[5] https://ESS.R-project.org/

A.7 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 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) {
   fname <- paste("myfile", i, ".pdf", sep = "")
   pdf(file = fname, width = 6, height = 6)
   plot(rnorm(100)+i)
   dev.off()
   cat("\\includegraphics{", fname, "}\n\n", sep = "")
}
@
```

A.8 How can I set default `par()` settings for figure chunks?

Because Sweave opens a new device for each graphics file in each figure chunk, 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.

A.9 How can I change the formatting of R input and output chunks?

Sweave uses the **fancyvrb** package for formatting all R code and text output. **fancyvrb** is a very powerful and flexible package that allows fine control for displaying 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.10 How can I change the line length of R input and output?

Sweave respects the desired line length of R, namely `options("width")`. Therefore, after e.g. `options(width = 40)` lines will be formatted to have at most 40 characters (if possible).

A.11 Can I use Sweave for Word files?

Not directly, but package **officer** provides functionality to manipulate Microsoft Word documents from R.

A.12 Can I use Sweave for OpenDocument files?

Not directly. Package **odfWeave** (archived in 2018) provided functions for using Sweave in combination with OpenOffice Writer rather than LaTeX. See the CRAN Task View: Reproducible Research for an up-to-date list of related developments.

A.13 Can I use Sweave for HTML files?

Yes, package **R2HTML** provides a driver for using Sweave in combination with HTML rather than LaTeX.

A.14 After loading package R2HTML Sweave doesn't work properly!

Package **R2HTML** registers an Sweave driver for HTML files using the same file extensions as the default `noweb` syntax, and after that its syntax is on the search list before the default syntax. Using

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

or calling Sweave like

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

ensures the default syntax even after loading package **R2HTML**.