Introduction to Sweave*

Michael Lundholm†

October 15, 2012
Version 1.3a

1 Introduction

Literate programming was introduced by Knuth (1984) as a way to explain for humans what the programmer wants the computer to do, rather than instructing the computer:

> "The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence in style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable mean." (Knuth, 1984, p. 97).

That is, rather than including code comments, the programmer writes a human readable text of integrated code and code documentation.

Statisticians and econometricians have adopted this idea of integrating code and documentation of code in order establish procedures for *reproducible* (econometric) *research*. Gentleman and Temple Lang (2007) define it as

> "...research papers with accompanying software tools that allow the reader to directly reproduce the result and employ the methods that are presented ..." (abstract)

An article with research results based on these procedures does not just contain a general description about the research leading up to the presented results but complete code for an exact reproduction of the results (including graphs and tables).

Several implementations exists for different combinations of statistical and typesetting or word processing engines:

*I am grateful for comments on earlier versions from Mahmood Arai, Fabio Frascati and Klaus Zeuge.

†Department of Economics, Stockholm University, michael.lundholm@ne.su.se.

- `Sweave`: R and LaTeX.[1]
- `odfWeave`: R and Open Document Format (ODF).[2]
- `R2HTML`: R and HTML.[3]
- `SASweave`: SAS and LaTeX.[4]
- StatWeave: R, SAS or Stata and LaTeX or Open Document Format (ODF).[5]

Using literate programming statistical research can easily be reproduced and there are (at least for `Sweave`) several examples of how it and other concepts, important in reproducible research, can be implemented. See for instance Meredith and Racine (2008) and Koenker and Zeileis (2009).

The structure of this note is as follows: In section 2 some basic concepts in literate programming as well as the `noweb` code syntax are introduced. Section 3 introduces `Sweave` and `Stangle`. In section 4 a series of examples are given of how the output from `Sweave` can be controlled. Some examples concerning tables in 5. Section 6 shows a series of examples of cross–references. In section 7 we show how `Sweave` can be used to create files.

2 Background

To implement the principles of literate programming Knuth designed the `WEB` system, which name indicates that programs can be "...regarded as a *web* that has been delicately pieced together from simple materials".[6]

In `WEB` and in literate programming generally, code and the documentation of the code both reside in the same file. Processing this file to achieve a human readable document which describes the program and help to maintain it is called *weaving*. Processing the file to produce a machine–executable program is called *tangling*.

The complexity of the `WEB` system made it, according to Ramsey (1994), hard to explore the idea of the literate programming. Instead he offered a simpler environment called `noweb`, which organised the code in *chunks*; code chunks and documentation (or text) chunks. Code chunks begins with `<<`*chunk label*`>>=` on a single line and documentation chunks with `@` on a

[1] For R see R Development Core Team (2008), `Sweave` see Leisch (2002) and LaTeX see Lamport (1994).

[2] For `odfWeave` see Kuhn (2009) and ODF Organization for the Advancement of Structered Information Standards (OASIS) (2005).

[3] For `R2HTML` see Lecoutre (2003).

[4] For `SASweave` see Lenth and Højsgaard (2007) and SAS SAS Institute Inc (2009).

[5] Under development by Russel V. Lenth. See `http://www.stat.uiowa.edu/~rlenth/StatWeave/`. For Stata see Stata Corp LP (2009).

[6] Knuth (1984, p. 97).

single line. Note that the beginning of the code file by default is a documentation chunk and that each chunk is terminated by the beginning of a new chunk.

In `noweb` noweave corresponds to weave and notangle to tangle. Noweave produces a TEX source file for typeset documentation. Notangle extracts all code chunks to a separate file and ignores documentation chunks. How `noweb` via `noweave` works is illustrated by Example 1:

Example 1 This is the content of the file `example1.nw`:

```
This is the beginning of the code file. It is always
a documentation chunk unless explicitly marked as code
chunk.

We can insert a possible description of the code below
here ...
<<First code chunk>>=
Here come some arbitrary code. Note that the '<<' begins
in column 1.
@

... or in the documentation chunk below.
```

When processed in `noweave`'s default mode a LATEX–file is the result (TEX and HTML code are also possible). Run through LATEX the resulting LATEX code is typeset as follows:

November 11, 2010 `example1.nw` 1

This is the beginning of the code file. It is alwaysa documentation chunk unless explicitly marked as code chunk.

We can insert a possible description of the code below here ...

⟨*First code chunk*⟩≡

```
Here come some arbitrary code. Note that the '<<' begins
in column 1.
```

... or in the documentation chunk below.

■

However, the code chunks are not actually evaluated. If the code is for statistical software, such as (say) R, this is a drawback since authors of documents reporting about statistical results most frequently would prefer to combine text with the output of statistical software. A solution for this in the case of R and LATEX is `Sweave`.[7]

[7] If one (for some reason) is content with just typesetting code without evaluating it,

3 Basic Sweave and Stangle

The basic Sweave syntax is essentially the the same as noweb's.[8] A code chunk has the format <<*chunk label, list of options*>>=. For the use of chunk labels see section 6 on page 17 and regarding options section 4 on page 6.

Example 2 This is the content of the file example2.rnw:

```
\documentclass[11pt, a4paper]{article}
\begin{document}
We start by loading the data set \texttt{AirPassengers}
and present a summary of the data:
<<AirPassengers>>=
data(AirPassengers)
summary(AirPassengers)
@
\end{document}
```

Note that the example file contains the structure of a complete LaTeX–document. ■

In R we run Sweave on the file example2.rnw:

```
> Sweave("example2.rnw")
```

or in batch mode using (say) the Mac OS X or Linux terminal or the Windows command prompt:

```
R CMD Sweave example2.rnw
```

This produces a file called example2.tex.

```
\documentclass[11pt, a4paper]{article}
\usepackage{Sweave}
\begin{document}
We start by loading the data set \texttt{AirPassengers}
and present a summary of the data:
\begin{Schunk}
\begin{Sinput}
> data(AirPassengers)
> summary(AirPassengers)
\end{Sinput}
\begin{Soutput}
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

then a LaTeX package like **fancyvrb**, which is in fact used by Sweave.sty to typeset the code, can be used directly without the use of Sweave.

[8] There is also an alternative LaTeX–style syntax for code chunks as well as an alternative to new syntax alternatives for the user; see Leisch (2008, p. 8).

```
\end{Soutput}
\end{Schunk}
\end{document}
```

Note that `\usepackage{Sweave}` is inserted on line 2. This is always done if not the user has inserted the command in the Sweave–file. After processing it with LaTeX the result is as follows:

We start by loading the data set `AirPassengers` and present a summary of the data:

```
> data(AirPassengers)
> summary(AirPassengers)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

We may note that the file processed by `Sweave` was named with extension `.rnw`. This is the convention for such files. We will refer to such files as *Sweave–files.*[9]

Code and its output is weaved together with the sourrounding documentation chunks. On the other hand `Stangle`,

```
> Stangle("example2.rnw")
```

or in batch mode using (say) Mac OS X or Linux:

```
R CMD Stangle example2.rnw
```

creates the code file `example2.R` the content of which is just R code:

```
### R code from vignette source 'example2.rnw'

###################################################
### code chunk number 1: AirPassengers
###################################################
data(AirPassengers)
summary(AirPassengers)


```

[9] Frequentlu the extension `Rnw` is used, but it is not a good convention since it mixes upper and lower case letter in the file name. This is a potential problem when files are used on different operating systems.

4 Finetuning Sweave

Detailed information about Sweave is found in the user manual Leisch (2008). Here we just state some important options with their default values:

- keep.source = FALSE: If TRUE the original code is copied 'as it is' to the output file.
- echo = TRUE: Includes the R code in the output file.
- results = verbatim: The output is included in an verbatim-like environment. If tex the output is taken to proper LaTeX markup. If hide all output is suppressed.
- quiet = FALSE: If TRUE all messages are suppressed.
- fig = FALSE: If TRUE the code chunk produces graphical output.
- eps = FALSE/pdf = TRUE: Indicates that figures should be produced as Portable Document Format (PDF) but not as Encapsulated Postscript (EPS). This presumes that pdflatex is used to produce PDF–files. If instead dvips and ps2pdf are used the user should set eps = TRUE/pdf = FALSE. Ignored if fig = FALSE.

By selecting options appropriately the author can fine tune the behaviour of Sweave. We illustrate these options by a sequence of examples.

Example 3 This is the main example with all options at their default values (i.e. we do not set any options at all). We estimate an autoregressive model of order 1 using OLS. Please note that in the code the definition of the data is split on lines 3–4 with an intention to have the code within the document margins:

```
1 <<>>=
2 data(AirPassengers)
3 ap.data <- ts.intersect(diff(log(AirPassengers)),
4      lag(diff(log(AirPassengers)), - 1))
5 lm(ap.data[, 1] ~ ap.data[, 2])
6 @
```

Processed by Sweave we get the following code and output:

```
1 \begin{Schunk}
2 \begin{Sinput}
3 > data(AirPassengers)
4 > ap.data <- ts.intersect(diff(log(AirPassengers)),
5 +      lag(diff(log(AirPassengers)), - 1))
6 > lm(ap.data[, 1] ~ ap.data[, 2])
7 \end{Sinput}
8 \begin{Soutput}
```

```
Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])

Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
\end{Soutput}
\end{Schunk}
```

The code lines 3–4 are concatenated into one by `Sweave`. Here at line 4, but with a new line break. The consequence is the the code continues into the margin. Typeset by LaTeX will look as follows:

```
> data(AirPassengers)
> ap.data <- ts.intersect(diff(log(AirPassengers)),
+      lag(diff(log(AirPassengers)), - 1))
> lm(ap.data[, 1] ~ ap.data[, 2])

Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])

Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
```

The concatenation is (of course) inherited in the typeset material. The result for the type set material is that this line will be to long for the printing area at our disposal. ■

Even if we wrote the code carefully, the typeset result did not look nice. We can control the way the typeset code will be displayed by LaTeX by invoking option `keep.source = TRUE`:

Example 4 Option `keep.source = TRUE` invoked:

```
<<keep.source = TRUE>>=
data(AirPassengers)
ap.data <- ts.intersect(diff(log(AirPassengers)),
     lag(diff(log(AirPassengers)), - 1))
lm(ap.data[, 1] ~ ap.data[, 2])
@
```

Processed by `Sweave` we get the following code and output

```
\begin{Schunk}
\begin{Sinput}
> data(AirPassengers)
```

```
> ap.data <- ts.intersect(diff(log(AirPassengers)),
+      lag(diff(log(AirPassengers)), - 1))
> lm(ap.data[, 1] ~ ap.data[, 2])
\end{Sinput}
\begin{Soutput}
Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])

Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
\end{Soutput}
\end{Schunk}
```

which typeset by LaTeX will look as follows:

```
> data(AirPassengers)
> ap.data <- ts.intersect(diff(log(AirPassengers)),
+      lag(diff(log(AirPassengers)), - 1))
> lm(ap.data[, 1] ~ ap.data[, 2])

Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])

Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
```

Now the typeset code corresponds to the way we wrote the code. ■

For some purposes it is desirable to suppress all output. For instance, we want to display the code and we want the code to be evaluated (some resulting objects may be used later) but we do not want to display the results:

Example 5 Option `results=hide` invoked to suppress all results from code:

```
<<keep.source = TRUE, results = hide>>=
data(AirPassengers)
ap.data <- ts.intersect(diff(log(AirPassengers)),
     lag(diff(log(AirPassengers)), - 1))
lm(ap.data[, 1] ~ ap.data[, 2])
@
```

Processed by `Sweave` we get the following code and output

```
\begin{Schunk}
\begin{Sinput}
```

```
> data(AirPassengers)
> ap.data <- ts.intersect(diff(log(AirPassengers)),
+      lag(diff(log(AirPassengers)), - 1))
> lm(ap.data[, 1] ~ ap.data[, 2])
\end{Sinput}
\end{Schunk}
```

which typeset by LaTeX will have the following looks:

```
> data(AirPassengers)
> ap.data <- ts.intersect(diff(log(AirPassengers)),
+      lag(diff(log(AirPassengers)), - 1))
> lm(ap.data[, 1] ~ ap.data[, 2])
```

■

The option `eval = FALSE` will have a similar effect on the typeset material. The difference is that the code is not evaluated and for this reason does not produce any results.

Finally, we can also suppress the code:

Example 6 Option `echo = FALSE` invoked to suppress the code but not the output:

```
<<echo = FALSE>>=
data(AirPassengers)
ap.data <- ts.intersect(diff(log(AirPassengers)),
     lag(diff(log(AirPassengers)), - 1))
lm(ap.data[, 1] ~ ap.data[, 2])
@
```

Processed by `Sweave` we get the following code and output

```
\begin{Schunk}
\begin{Soutput}
Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])

Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
\end{Soutput}
\end{Schunk}
```

which typeset by LaTeX will have the following looks:

```
Call:
lm(formula = ap.data[, 1] ~ ap.data[, 2])
```

```
Coefficients:
 (Intercept)  ap.data[, 2]
    0.007375      0.200815
```

■

The option `echo = FALSE` may not seem to be so useful. However, both code and output can be suppressed by combining the options `echo = FALSE` and `results = hide`. Any messages will also be suppressed by adding the option `quiet = TRUE`. To do this can be useful when the output to be shown from an R session is (say) a graph:

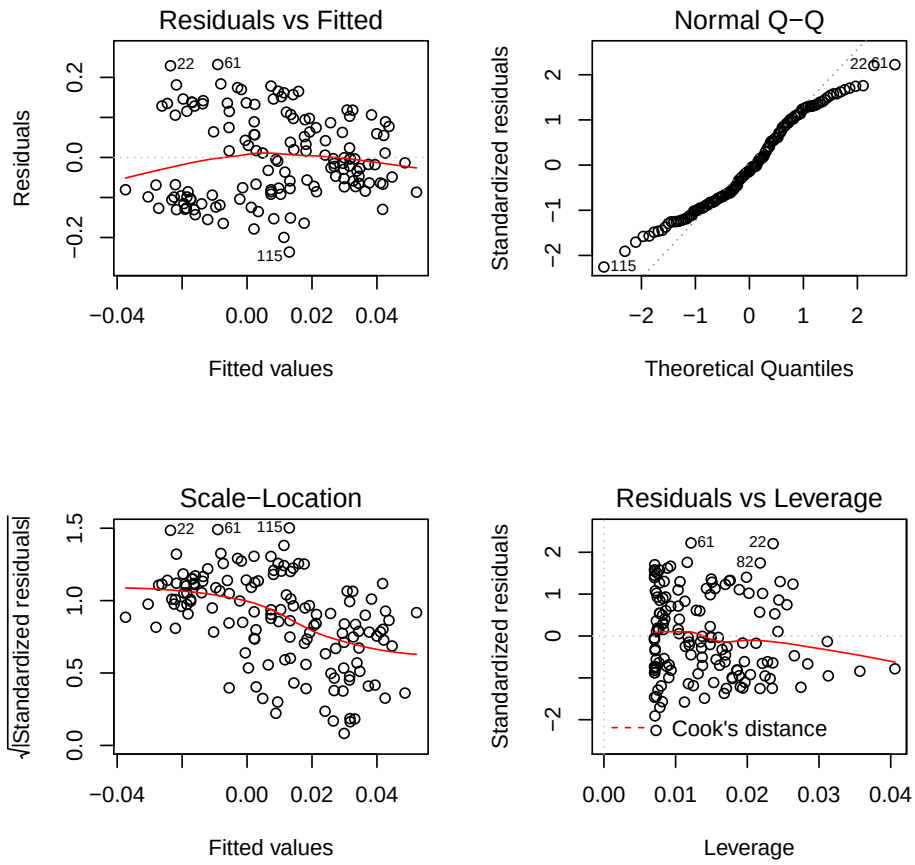
Example 7 Creating a plot of an `lm`-object without displaying anything else:

```
<<echo = FALSE, results = hide, fig = TRUE>>=
data(AirPassengers)
ap.data <- ts.intersect(diff(log(AirPassengers)),
     lag(diff(log(AirPassengers)), - 1))
par(mfrow = c(2, 2))
plot(lm(ap.data[, 1] ~ ap.data[, 2]))
@
```

The `results=hide` is not necessary in this example. However, it is a good precaution in order to prevent the results of commands to be shown in a code shunk in the document. Here, we only want the graph to be shown, but on the other hand, the commands we are using have no results. Processed by `Sweave` we get the following code:

```
\includegraphics{example7-001}
```

This Example 7 is in a file with name `example7.rnw`. The base name is used by `Sweave` to generate a file name for the graphics file. It is the first generated file and is given the base name `example7-001`. Its extension is given by how the options for graphics are set. By default it would be `.pdf` so the complete file name would be `example7-001.pdf`. Typeset by LaTeX we will have the following looks:



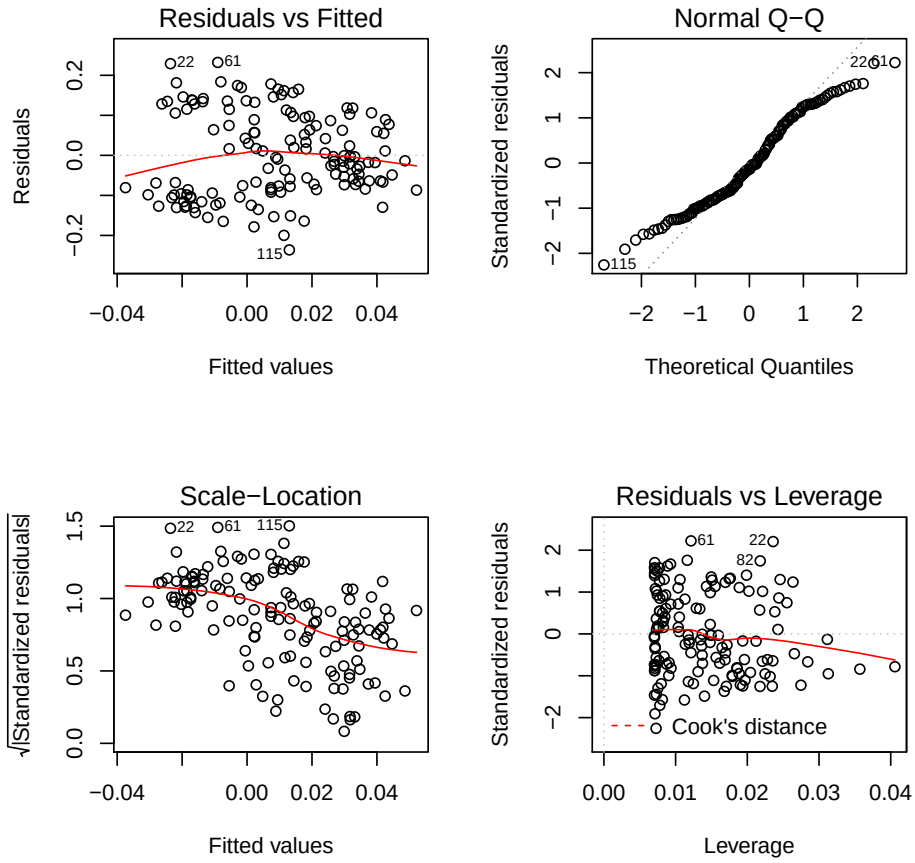
A graph is created and included into LaTeX using `\includegraphics` from the LaTeX–package **graphicx**. ■

In the previous example the graph created by R was inserted in the document at the place where `\includegraphics` was issued; not as float.

Example 8 Creating the same plot but now as a float:

```
\begin{figure}
\caption{Now the plot is a float using
the \code{figure}--environment.\label{fig:ex8}}
<<echo = FALSE, results = hide, fig = TRUE>>=
data(AirPassengers)
ap.data <- ts.intersect(diff(log(AirPassengers)),
     lag(diff(log(AirPassengers)), - 1))
par(mfrow = c(2, 2))
plot(lm(ap.data[, 1] ~ ap.data[, 2]))
@
\end{figure}
```

Figure 1: Now the plot is a float using the `figure`–environment.



Processed by `Sweave` we get the following code:

```
\begin{figure}
\caption{Now the plot is a float using
the \code{figure}--environment.\label{fig:ex8}}
\includegraphics{example8-001}
\end{figure}
```

■

The figure is typeset on page 12 in Figure 1.

5 Writing tables

In the following we will learn how to create regular tables typeset by LaTeX from R objects.First we set up an example and present the result using the standard `summary()` function.

Example 9 Consider the Student-Teacher test score example in (Stock and Watson, 2007, eq. (6.12)) where test scores (`testscr`) are regressed against the student–teacher ratio (`str`) and the percentage of students who are English learners (`el_pct`). The data set (called `Caschool`) is available in package **Ecdat**:

```
<<keep.source = TRUE, echo = FALSE, quiet = TRUE>>=
data(Caschool, package = "Ecdat")
scr.result <- lm(testscr ~ str + elpct, data = Caschool)
summary(scr.result)
@
```

The resulting output from Sweave after LaTeX typesetting is:

```
Call:
lm(formula = testscr ~ str + elpct, data = Caschool)

Residuals:
    Min      1Q  Median      3Q     Max
-48.845 -10.240  -0.308   9.815  43.461

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 686.03225    7.41131  92.566  < 2e-16 ***
str          -1.10130    0.38028  -2.896  0.00398 **
elpct        -0.64978    0.03934 -16.516  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.46 on 417 degrees of freedom
Multiple R-squared: 0.4264,        Adjusted R-squared: 0.4237
F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

■

Example 10 Although `summary()` gives a concise summary of an object of class `lm` one may want to typeset the result using LaTeX capabilities. This can be done exploiting the function `xtable()` in package **xtable**:

```
<<xtable, keep.source = TRUE, echo = FALSE, results = tex>>=
library(xtable)
data(Caschool, package = "Ecdat")
scr.result <- lm(testscr ~ str + elpct, data = Caschool)
xtable(summary(scr.result),
     caption="Results from Student--Teacher Ratio",
     align=c("l", "r", "r", "r", "r"),
     digits = 2)
@
```

Note that `echo = FALSE` here prevents the commands to be shown in a code shunk; we only want the resulting LaTeX–code that generates the table. Note the following options used:

- `caption`: Provides a `\caption{`*text*`}` to the table float.
- `align`: Controls the alignment of columns in the tabular environment inside the table.
- `digits`: A numeric vector providing the accuracy of the numbers printed in the table. If length is one, the vector is replicated if necessary.
- Additionally `label` can be used to insert a `\label{`*label*`}` to the table float.

The LaTeX output from `Sweave` is

```
% latex table generated in R 2.15.1 by xtable 1.7-0 package
% Mon Oct 15 12:18:47 2012
\begin{table}[ht]
\begin{center}
\begin{tabular}{lrrrr}
  \hline
 & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
  \hline
(Intercept) & 686.03 & 7.41 & 92.57 & 0.00 \\
  str & -1.10 & 0.38 & -2.90 & 0.00 \\
  elpct & -0.65 & 0.04 & -16.52 & 0.00 \\
   \hline
\end{tabular}
\caption{Results from Student--Teacher Ratio}
\end{center}
\end{table}
```

and the typeset result:

Note that not all output from summary is present in the table. The following example shows how to add information in the table. ■

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	686.03	7.41	92.57	0.00
str	-1.10	0.38	-2.90	0.00
elpct	-0.65	0.04	-16.52	0.00

Table 1: Results from Student–Teacher Ratio

Example 11 We now exploit that **xtable** equips `print()` with methods for `xtable` (see `?print.xtable` for details):

```
<<xtable, keep.source = TRUE, echo = FALSE, results = tex>>=
library(Ecdat)
library(xtable)
data(Caschool)
scr.result <- lm(testscr ~ str + elpct, data = Caschool)
Footer <- paste(
     "\\hline\n\\multicolumn{5}{l}{Residual standard error:
\n",
     round(summary(scr.result)$sigma, 2), " on ",
     scr.result$df.residual, " degrees of freedom.}\\\\\n",
     "\\multicolumn{5}{l}{Multiple R-squared: \n",
     round(summary(scr.result)$r.squared, 2), ", ",
     "Adjusted R-squared: ",
     round(summary(scr.result)$adj.r.squared, 2), ".}\\\\\n",
     "\\multicolumn{5}{l}{F-statistic:   \n",
     round(summary(scr.result)$fstatistic[1], 0), " on ",
     summary(scr.result)$fstatistic[2], " and ",
     summary(scr.result)$fstatistic[3], "
DF.}\\\\\\hline",sep="")
addtorow <- list()
addtorow$pos <- list()
addtorow$pos[[1]] <- 3
addtorow$command <- Footer
print(xtable(summary(scr.result),
     caption = "Results from Student--Teacher Ratio",
     label = "ta:str_print",
     align = c("l", "r", "r", "r", "r"), digits = 2),
     floating.environment = "table",
     table.placement = "ht",
     caption.placement = "top",
     latex.environments = c("center"),
     tabular.environment = "tabular",
     add.to.row = addtorow,
     hline.after = c( - 1, 0, 0))
@
```

Note the following options used:

- `floating.environment`: Possible options are `table` and `sideways-table`; the latter defined by package **rotating**. Default is `table`.
- `table.placement`: Either `NULL` or contain elements of from `"h", "t", "b", "p", "!", "H"`. Default value is `ht`.
- `caption.placement`: `top` or `bottom`, with the latter as default.
- `latex.environments`: The specified character vector will enclose the tabular environment. Default is `center`.
- `tabular.environment`: `tabular` (default) or `longtable` from **longtable**. If `longtable` is used the set `floating=FALSE`.
- `add.to.row`: A list consisting of two elements; `pos` and `command`. `pos` is a list of positions of rows where commands should be added. `command` is a charcater of the same length as `pos` containing the commands that should be added and the end of the specified rows.
- `hline.after`: A vector of integers between -1 and `nrow(x)`, where `x` is the object of class `xtable` to be printed. Default is `c( - 1, 0, nrow(x))`.

For the option `add.to.row`, note that `\` is an escape character, so that the initial `\` in LaTeX commands has to be doubled inside `paste()`. With `\n` inside `paste()` a line break is pasted. They occur here only to create suitable line breaks in the the LaTeX output from `Sweave`, which is

```
% latex table generated in R 2.15.1 by xtable 1.7-0 package
% Mon Oct 15 12:18:48 2012
\begin{table}[ht]
\begin{center}
\caption{Results from Student--Teacher Ratio}
\label{ta:str_print}
\begin{tabular}{lrrrr}
  \hline
 & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
  \hline
\hline
(Intercept) & 686.03 & 7.41 & 92.57 & 0.00 \\
  str & -1.10 & 0.38 & -2.90 & 0.00 \\
  elpct & -0.65 & 0.04 & -16.52 & 0.00 \\
   \hline
\multicolumn{5}{l}{Residual standard error:

14.46 on 417 degrees of freedom.}\\
\multicolumn{5}{l}{Multiple R-squared:
0.43, Adjusted R-squared: 0.42.}\\
\multicolumn{5}{l}{F-statistic:
155 on 2 and 417
DF.}\\\hline\end{tabular}
```

```
\end{center}
\end{table}
```

The typeset result is in Table 2 on page 17. ■

Table 2: Results from Student–Teacher Ratio

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	686.03	7.41	92.57	0.00
str	-1.10	0.38	-2.90	0.00
elpct	-0.65	0.04	-16.52	0.00
Residual standard error: 14.46 on 417 degrees of freedom.				
Multiple R-squared: 0.43, Adjusted R-squared: 0.42.				
F-statistic: 155 on 2 and 417 DF.				

6 Cross references

`Sweave` works sequentially. This means that once a code chunk is evaluated it can be used by code chunks evaluated later. Some simple examples of reusing material in code chunks:

Example 12 This example just reuses previously defined objects in later code chunks:

```
<<>>=
x <- 1
@
<<>>=
y <- 1; x + y
@
```

This produces the typeset output:

```
> x <- 1

> y <- 1; x + y

[1] 2
```

. ■

Example 13 One can also reuse previous code chunks by explicit reference.

```
<<chunk1>>=
x <- 1
@

<<chunk2>>=
y <- 1
x + y
@

<<chunk3>>=
y <- 2
x + y
@
<<chunk4>>=
<<chunk1>>
<<chunk2>>
@
```

Note that the reference is coded as `<<label>>` without a `=`. The typeset output is:

```
> x <- 1

> y <- 1
> x + y

[1] 2

> y <- 2
> x + y

[1] 3

> x <- 1
> y <- 1
> x + y

[1] 2
```

Note the text `chunk1` etc in the chunk tags corresponds to a label in LaTeX terminology. `<<`*chunk label*`>>` then corresponds to reference in LaTeX. Chunks 1 and 2 defines $x = y = 1$ so $x + y = 2$. Chunk 3 redefines $y = 2$, so that $x + y = 3$. Finally Chunk 4 reuses chunks 1 and 2 so again $x = y = 1$ and $x + y = 2$.

Finally, we note that the references to code chunks 4, 2 and 1 are expanded into R–code. This is can be changed by setting the code chunk option `expand = FALSE`. ■

Example 14 Using `\Sexpr{`*object*`}` one refer to R objects in code chunks from running text. For example, first defining

```
<<>>=
x <- 2
y <- 3
(z <- x + y)
@
Defining $z$ as above we get $z=\Sexpr{z}$.
```

Note, however, that `object` must be defined before `\Sexpr{`*object*`}` is issued. The resulting LaTeX code is

```
\begin{Schunk}
\begin{Sinput}
> x <- 2
> y <- 3
> (z <- x + y)
\end{Sinput}
\begin{Soutput}
[1] 5
\end{Soutput}
\end{Schunk}
Defining $z$ as above we get $z=5$.
```

which typeset with LaTeX become

```
> x <- 2
> y <- 3
> (z <- x + y)

[1] 5
```

Defining z as above we get $z = 5$. ∎

Example 15 In a document one sometimes want to refer to objects which are defined later on in the document. In LaTeX the problem is solved by running LaTeX several times, so that what `\label` writes to the auxiliary file in the first run can be read by `\ref` in the second run. Here we can utilise that files created by R/`Sweave` can be read by LaTeX, although what we refer to in the beginning of `Sweave` file is defined later in the same file:

- The LaTeX command `\input{`*file.tex*`}` includes the content of *file.tex*. But this LaTeX command is neglected when `Sweave` is run.
- We now set up the code chunk that is to write something to *file.tex*:

```
The coefficient for the student--teacher ratio is
$\input{file.tex}$.
<<>>=
library(Ecdat)
data(Caschool)
scr.result <- lm(testscr ~ str + elpct, data = Caschool)
coef(scr.result)[[2]]
write(coef(scr.result)[[2]], file = "file.tex")
@
```

The results in the LaTeX code is

```
The coefficient for the student--teacher ratio is
$\input{file.tex}$.
\begin{Schunk}
\begin{Sinput}
> library(Ecdat)
> data(Caschool)
> scr.result <- lm(testscr ~ str + elpct, data = Caschool)
> coef(scr.result)[[2]]
\end{Sinput}
\begin{Soutput}
[1] -1.101296
\end{Soutput}
\begin{Sinput}
> write(coef(scr.result)[[2]], file = "file.tex")
\end{Sinput}
\end{Schunk}
```

■

7 Distribution of additional files

When the files of a project shall be distributed it can be a good idea to keep as much code and information as possible in a single Sweave–file and let R generate the necessary files.

Obne example is bibliographical references using BibTeX for which different strategies are possible. One alternative is to create the relevant `.bbl` file and include in the Sweave-file at the position of the bibliography. Another alternative, which makes it possible to document the runs with BibTeX, is to include the information of the entire `.bib` in the Sweave file and then export as the following example shows:

Example 16 Suppose we want to create a file `project.bib` which contains the biblipgraphical reference to Knuth (1984). In the Sweave the following code is included to export it to `project.bib`:

```
<<>>=
writeLines("@Article{ Knuth84,
          title   = {Literate programming},
          author  = {Donald E. Knuth},
          journal = {The Computer Journal},
          pages   = {97--111},
          volume  = {27},
          year    = {1984}
}",con="project.bib")
@
```

Note that with more than one entry each should be surrounded by the double quotations works for `Sweave` to properly. The resulting `.bib` file will then be:

```
@Article{ Knuth84,
          title   = {Literate programming},
          author  = {Donald E. Knuth},
          journal = {The Computer Journal},
          pages   = {97--111},
          volume  = {27},
          year    = {1984}
}
```

■

8 The LaTeX package Sweave.sty

Along with the basic functions in R `Sweave` also comes with a LaTeX package called `Sweave.sty`. This package must be loaded in the LaTeX–file created by `Sweave`. The user should therefore insert the code line

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

in the preamble of the `Sweave` script file. If this is neglected, the weaving procedure will insert the code line in the created LaTeX file. The package comes with two options: `nogin` which do not scale graphics (default is 80% of the textwidth) and `noae` which do not load the almost European fonts (which is default). I recommend the use of the `noae` option only.

There is one issue about the LaTeX package `Sweave.sty` that should be noted. *It is not available on CTAN (the Comprehensive TeX Archive Network) but only via the* R *installation.* This may create two types of problems:

1. `\usepackage{Sweave}` presumes that the package is in the search path of the LaTeX software, but whether it is or not depends on where R was installed.

2. On Windows operating systems the search path to the R installation may contain a blank space (i.e., `c:\Program Files\...`, which LaTeXcannot handle in the search path.

The best solution to both of these problems is to install `Sweave.sty` in a directory the search path of which do not include any blank spaces and which is in the search path of LaTeX; see the installation instructions of your LaTeX–distribution. A quick and easy fix is, however, to copy the file `Sweave.sty` to the current directory. The disadvantage is that this has to be repeated whenever the current directory is changed. A more permanent fix is to do the following:

- Mac OS X users with MacTeX. In the terminal:

  ```
  mkdir -p ~/Library/texmf/tex/latex
  cd ~/Library/texmf/tex/latex
  ln -s /Library/Frameworks/R.framework/Resources/share/texmf Sweave.sty
  ```

- Windows TeXLive users: Copy `Sweave.sty` to `.../texlive/texmf-local/tex/latex`

- Windows MikTeX users: Use the "Settings" application to add the directory where `Sweave.sty` as a so called "root" directory.

9 Setting different types of Sweave options

We have above seen how options can be set for each individual code chunk. However, these options can also be set globally. Also, the LaTeX package `Sweave` also has options that affect the typesetting for LaTeX via the **graphicx** LaTeX–package. But it is also possible to achieve this interaction via LaTeX–coding directly so as to affect individual graphs. Finally, `Sweave` uses the LaTeX–package **fancyvrb** to do the typesetting of code and its output and it is possible for the user to change this behaviour also.

9.1 Global options for code chunks

If one wants to deviate from the default setting for the code chunks this can be by done by the command:

```
\SweaveOpts{option1 = value, option2 = value, ...}
```

9.2 Interacting with graphicx

When `Sweave` includes a graph created by a plotting command it utilises the LaTeXgraphicx. Using this package a LaTeX user would use (say)

```
\includegraphics[key/val-list]{file}
```

where *key/val-list* is a comma separated list of keys setting options that are used to control how the individual graph would appear in the resulting output file. The actual extension on file `file`, i.e. `eps` or `pdf`, depends on which graphics driver that is used.

However, since `Sweave` just inserts a simple `\includegraphics{file}` without options, this method of controlling the appearance of graphs is not possible. Instead one can utilise the alternative using the `\setkeys` command; i.e.,

```
\setkeys{Gin}{key/val-list}
\includegraphics{file}
```

where `Gin` is the name used for the **keyval** keys associated with *Graphics inclusion*. In a `Sweave` file this would correspond to

```
\setkeys{Gin}{key/val-list}
<<echo = FALSE, fig = TRUE>>=
plot(x)
@
```

where `x` is the object to be plotted. Since the values given to the keys are valid until they are redefined by a new `\setkey` command, the `\setkeys` commands can be changed prior to any new graph if necessary.

Some important keys are:

`bb` Bounding box set by four dimensions separated by spaces (the lower left and upper right (x, y)–coordinates of the bounding box of the image).

`angle` Rotation in degrees counterclockwise.

`scale` Scale factor $\in [0, 1]$

`clip` Clip image to bound box (Boolean).

`width` Required width.

`height` required height.

`keepaspectratio` Boolean. If `true`, then specifying `width` or `height` will not distort the figure..

Note that Boolean keys are set `key= true/false`.

The default value of the `Gin` key in `Sweave` is

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

This can be altered by changing the `Gin` key using `\setkeys`. However, it also exists a package option to `Sweave` called `gin`, which is set in the usual way; i.e.[10]

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

It has the effect of altering the `Gin` key to

```
\setkeys{Gin}{}
```

For details on the **graphicx** package see Carlisle and The LaTeX3 Project (2005) for details.

9.3 Interacting with fancyvrb

The `Sweave` package utilises the **fancyvrb** package by defining the following verbatim environments (line numbers refer to numbers in the `Sweave.sty` file):

```
24 \DefineVerbatimEnvironment{Sinput}{Verbatim}{fontshape = sl}
25 \DefineVerbatimEnvironment{Soutput}{Verbatim}{}
26 \DefineVerbatimEnvironment{Scode}{Verbatim}{fontshape = sl}
27
28 \newenvironment{Schunk}{}{}
```

These are exactly the environments that are used to typeset the R code and its output; see e.g., Example 2 on page 4. They are defined using the **fancyvrb** package described by Van Zandt (2010).

Example 17 Consider the code from Example 2 but where we want change the appearance of the typeset code; some distances are increased, in– and output distinguished by colours and the font size changed. The change in the behaviour of these verbatim environments is achieved using the `\Recustom VerbatimEnvironment` and `\renewenvironment`; see code lines $3-6$:[11]

```
1 \documentclass[11pt,a4paper]{article}
2 \usepackage[noae]{Sweave}
3 \usepackage{color}
4 \definecolor{Blue}{rgb}{0,0,0.5}
5 \definecolor{Green}{rgb}{0,0.5,0}
```

[10] There is also another option called `ae`, which initiates (among other things) the Almost European Computer Modern fonts. Inspect `Sweave.sty` and `ae.sty` for details.

[11] Some of these changes are taken from Ihaka (2009).

```
\RecustomVerbatimEnvironment{Sinput}{Verbatim}{%
     xleftmargin=2em,%
     fontsize=\footnotesize,%
     fontshape=sl,%
     formatcom=\color{Blue}%
     }
\RecustomVerbatimEnvironment{Soutput}{Verbatim}{%
     xleftmargin=2em,%
     fontsize=\footnotesize,%
     formatcom=\color{Green}%
     }
\RecustomVerbatimEnvironment{Scode}{Verbatim}{xleftmargin=2em}
\renewenvironment{Schunk}{\vspace{\topsep}}{\vspace{\topsep}}
\fvset{listparameters={\setlength{\topsep}{6pt}}}
\begin{document}
We start by loading the data set \texttt{AirPassengers}
and present a summary of the data:
<<AirPassengers>>=
data(AirPassengers)
summary(AirPassengers)
@
\end{document}
```

Note that the **Sweave** package is loaded before the verbatim environments are redefined. The reason is that **Sweave** loads **fancyvrb** and then defines the original verbatim environments `Sinput` and `Soutput`. Once they are defined they can be redfined. An alternative order of doing this may result in error messages from LaTeX. After processing it with `Sweave` and LaTeX the result is as follows:

We start by loading the data set `AirPassengers` and present a summary of the data:

```
> data(AirPassengers)
> summary(AirPassengers)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

■

10 Colophon

This article was written using R version 2.15.1 (2012-06-22) and LaTeX from the TeX Live 2012 distribution on a i386-apple-darwin9.8.0 platform. Code examples are processed and typeset using the implementation of the compendium concept described by Lundholm (2011).

References

Carlisle DP, The LaTeX3 Project (2005). "Packages in the 'graphics' bundle." URL http://tug.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.pdf.

Gentleman R, Temple Lang D (2007). "Statistical Analyses and Reproducible Research." *Journal of Statistical and Graphical Statistics*, **16**, 1–23. URL http://pubs.amstat.org/doi/pdfplus/10.1198/106186007X178663.

Ihaka R (2009). *Customizing Sweave to produce better looking LaTeX output*. URL http://www.stat.auckland.ac.nz/%7eihaka/downloads/sweave-customisation.pdf.

Knuth DE (1984). "Literate programming." *The Computer Journal*, **27**, 97–111.

Koenker R, Zeileis A (2009). "On reproducible econometric research." Accepted for publication in Journal of Applied Econometrics, URL http://statmath.wu-wien.ac.at/~zeileis/papers/Koenker+Zeileis-2008.pdf.

Kuhn M (2009). *The OdfWeave Package*. URL http://cran.r-project.org/web/packages/odfWeave/vignettes/odfWeave.pdf.

Lamport L (1994). *LaTeX. A Document preparion System. User's Guide and reference Manual*. Addison–Wesley, 2 edition.

Lecoutre E (2003). "The R2HTML Package." *R News*, **3**(3), 33–36.

Leisch F (2002). "Sweave: Dynamic generation of statistical reports using literate data analysis." In W Härdle, B Ränz (eds.), "Compstat 2002 - Proceedings in Computational Statistics," pp. 575–580.

Leisch F (2008). *Sweave User Manual*. R version 2.7.1, URL http://www.stat.uni-muenchen.de/~leisch/Sweave/Sweave-manual.pdf.

Lenth RV, Højsgaard S (2007). "SASweave: Literate programming using SAS." *Journal of Statistical Software*, **19**(8).

Lundholm M (2011). "Implementing the Compendium Concept with Sweave and DOCSTRIP." *The R Journal*, **3(2)**, 16–21. URL http://journal.r-project.org/archive/2011-2/RJournal_2011-2_Lundholm.pdf.

Meredith E, Racine JS (2008). "Towards reproducible econometric reserach: The Sweave framework." *Journal of Applied Econometrics*. Published Online: 12 Nov 2008, URL http://dx.doi.org/10.1002/jae.1030.

Organization for the Advancement of Structered Information Standards (OASIS) (2005). *Open document formats of office applications (OpenDocument) v1.0*. URL http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

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

SAS Institute Inc (2009). *SAS/STAT Software*. URL http://www.sas.com/technologies/analytics/statistics/stat/index.html.

Stata Corp LP (2009). *Stata 10*. URL http://www.stata.com.

Stock JH, Watson MW (2007). *Introduction to econometrics*. Pearson International Education, 2 edition. ISBN 0-321-44253-9.

Van Zandt T (2010). "The 'fancyvrb' package. Fancy Verbatims in LaTeX." URL http://mirror.ctan.org/macros/latex/contrib/fancyvrb/fancyvrb.pdf.