

Desktop Publishing with Sweave

Andrew Ellis

July 28, 2010

1 What is Sweave

Sweave is a tool that allows you to embed the R code in \LaTeX documents. This means that you can include your code for a complete data analysis inside a report. You can then adapt your report dynamically if your data or analysis change.

A single source file contains both documentation text and R code, which are woven into a final document containing the documentation text together with R code and the output of the R code.

This report contains no prefabricated graphs; instead, everything is created on-the-fly. This leads to truly ‘reproducible’ research.

2 Literate programming

The term ‘literate programming’ was first coined by Donald Knuth to describe the following ideas:

- programs need description
- descriptions should be literate, not merely comments in source code
- the code should work

3 Noweb

Noweb is a simple literate-programming tool which allows you 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 @ 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 L^AT_EX.

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

4 Sweave options

Options control how code and output are transferred to the tex file. All options have the form `key=value`. They can either be given in each chunk individually, or globally. For example, to prevent R from parsing the formatting the output, use the following option: `<name,keep.source=TRUE>=`.

5 How does Sweave work?

First, the Sweave source file is run in R. This will result in a .tex file of the same name, with all the R code chunks now transformed into L^AT_EXcode.

5.1 How to produce PDF output

To produce a PDF report from this Sweave file, you go through the following steps:

1. First, call Sweave inside R with the following command: `Sweave("sweave-demo.Rnw")`¹. This will produce a file called `sweave-demo.tex` in the same directory.
2. Second, run `pdflatex sweave-demo.tex`².
3. Open your PDF in a PDF reader.

¹You can also call Sweave from the command line: `R CMD Sweave sweave-demo.Rnw`

²This can be done in MikTeX or automated using a Makefile

5.2 Example: R code chunk

The R command `x <- rnorm(10)` is transformed by Sweave into the following:

```
\begin{Schunk}
\begin{Sinput}
> x <- rnorm(10)
> x
\end{Sinput}
\begin{Soutput}
x
[1]  0.97554630 -1.30483845 -0.44469677  0.97040423 -1.98348367  0.15567820
[7] -2.39671814 -0.33302121  0.03206459 -0.62483179
\end{Soutput}
\end{Schunk}
```

5.3 Verbatim environments

The environments `Sinput`, `Soutput` and `Schunk` have to be defined in the `LaTeXpreamble`. You can either use the standard `Sweave.sty` supplied with R, or you can create your own. In this example, I have created my own customized verbatim environments:

```
\DefineVerbatimEnvironment{Sinput}{Verbatim} {
  fontsize=\footnotesize,fontshape=sl,formatcom=\color{SinColor}}

\DefineVerbatimEnvironment{Soutput}{Verbatim}{
  fontsize=\footnotesize,formatcom=\color{SoutColor}}
\fvset{listparameters={\setlength{\topsep}{0pt}}}
```

`Schunk` is defined as a standard environment, slightly indented using the ‘quote’ environment:

```
\newenvironment{Schunk} {
  \begin{quote}
}
{
  \end{quote}
}
```

6 Useful L^AT_EX packages

The **upquote** package forces `verbatim`, `verbatim*`, `verb`, and `verb*` to leave ‘ and ’ in their original state. This is very important for R code.

The **fancyvrb** package can be used to create custom environments for R input and output; above I defined `Rinput` to be coloured red and use an italic mono-spaced font, whereas R output is coloured dark blue, and is not slanted. Other possibilities include:

```
\DefineVerbatimEnvironment{Sinput}{Verbatim} {
  commentchar=@,
  frame=lines, label=\textrm{\bf R code}, numbers=left,
  framesep=10pt, fontshape=sl, commandchars=\\\{\}\}

\DefineVerbatimEnvironment{Soutput}{Verbatim}{}
```

For further details, please consult the `fancyvrb` manual.

7 An example

The following example shows you how to include R code in your document, and how to include figures. The actual R code is a function to calculate to the price of a call option using the Black-Scholes formula:

```
> callOption <- function(S, K, sigma, Time, r) {
  d1 <- (log(S/K) + (r + sigma^2/2) * Time)/(sigma *
    sqrt(Time))
  d2 <- d1 - sigma * sqrt(Time)
  Call <- S * pnorm(d1) - K * exp(-r * Time) *
    pnorm(d2)
  param <- list(S = S, K = K, sigma = sigma,
    Time = Time, r = r)
  ans <- list(price = Call, parameters = param)
  class(ans) <- "option"
  ans
}
```

Next, we define a `print` method to display the output nicely:

```

> ## print method for 'option' class:
> print.option <- function(x, ...) {
  cat("Price:\n ", x$price, "\n")
  cat("Parameters:\n")
  print(unlist(x$parameters))
}
> call <- callOption(S = 60, K = 65,
  sigma = 0.30, Time = 0.25,
  r = 0.08)
> print(call)

```

```

Price:
  2.1334
Parameters:
   S   K sigma Time   r
60.00 65.00 0.30 0.25 0.08

```

Finally, we create a plot method:

```

> ## plot method for the 'option' class
> plot.option <- function(C, col = "steelblue", ...) {
  callPrice <- C$price
  stockPrice <- C$param$S
  plot(stockPrice, callPrice, type = "l",
    xlab = "Stock Price", ylab = "Call Price",
    main = "Call option vs. stock price", col = col, ...)
  grid()
}

```

Now, let us include the figure in our report. Note the use of the option `fig=TRUE` to tell Sweave that we want to display the figure. If we wanted to show the figure without the R code that generated it, we would use the `echo=FALSE` option.

```
> ## Usage
> S <- seq(35, 95, by = 1)
> Time <- 0.25
> call <- callOption(S, K = 65, sigma = 0.30, Time=Time, r = 0.08)
> plot(call)
```

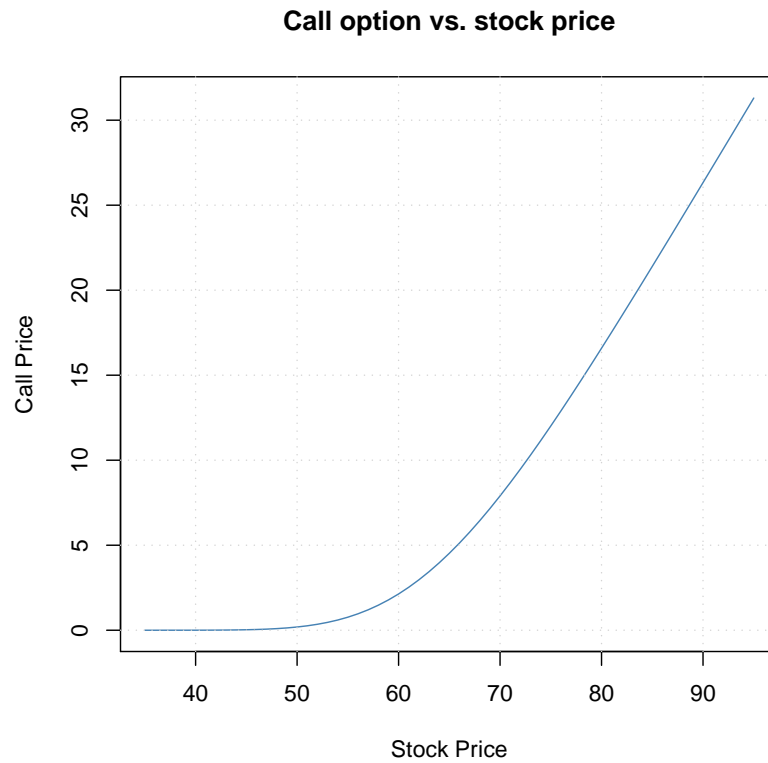


Figure 1: This is a plot