Dynamic Reporting with *R*/Sweave and LaTeX

Dr. Uwe Ziegenhagen
Lokomotivstr. 9
50733 Cologne
Germany
ziegenhagen@gmail.com
http://www.uweziegenhagen.de

Abstract

R is a sophisticated statistical programming language available for various platforms. Since its initial development in 1992 it has become the major open source tool for statistical data analysis. For the integration with LaTeX it provides tools which allow the convenient dynamic creation of reports. In this article I will give a very brief introduction to *R* and show how *R* integrates in the LaTeX workflow.

1 Introduction to *R*

The history of *R* dates back to 1969 when John M. Chambers from Bell Labs published the outlines of *S*, a programming language for statistics and data analysis that was first implemented in 1975 for Honeywell computers. Starting in 1992 Ross Ihaka and Robert Gentleman from the university of Auckland in New Zealand took up the concepts underlying *S* to develop *R*, a free implementation of the language. Today *R* is for many statisticans the tool of choice for visualization and data analysis, covering all aspects of modern computer-based statistics. The *R* project team has more than 500 members, more than 1'000 packages are available on CRAN, the Comprehensive *R* Archive Network.

1.1 *R* as a calculator

Since the focus of this paper lies more on the interaction with LaTeX I cannot give a thorough introduction into *R*. Interested readers may want to have a closer look on the bibliography of this article for suitable materials. Nevertheless I would like to point out some of the main features of the language. Listing 1.1 shows some of the operators for basic calculations.

```
1+2
1*2
1/2
1-2
2^2
sqrt(2)
sin(pi) # cos, tan
trunc(-pi) # -3
round(pi)  # 3
```

Listing 1.1: Basic calculations with *R*

The basic objects to store variables in *R* are vectors, matrices and dataframes. Vectors and matrices may contain a single data type only, complex data structures are stored in so-called dataframes, which are in fact lists of objects that may have different data types. Various ways of creating and assigning vectors to variables are shown in Listing 1.2.

```
a <- 1:3 # store vector 1..3 in a
b = 2:4 # store 2..4 in b
c(a,b) # [1] 1 2 3 2 3 4 # cat a & b
# generate sequence
seq(1,2,by=0.1) [1] 1.1 1.2 1.3 ...
# repeat 1..4 twice
rep(1:4,2) # [1] 1 2 3 4 1 2 3 4
```

Listing 1.2: Generating vectors in *R*

The last *R* example in Listing 1.3 shows how a simple linear model can be computed with *R*. The vector of independent variables x just contains the numbers 1 to 10, for the vector of dependent variables y we just multiply the x-vector with a random factor taken from a normal distribution. The linear model is then calculated using the `lm` command which presents the coefficients of the linear model.

```
> x<-1:10
> y=rnorm(10)*x
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     0.1079       1.0697
```

Listing 1.3: A linear model with *R*

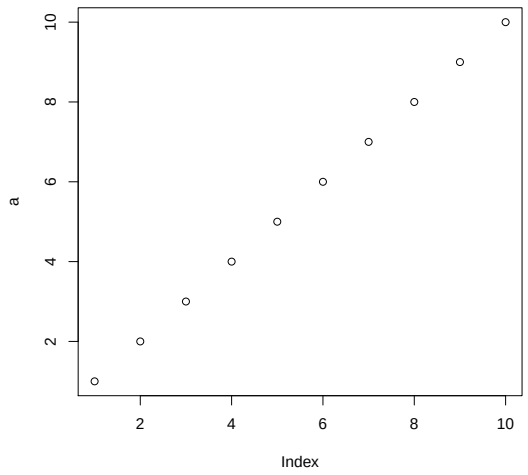

Figure 1: Graphics generated by Listing 1.4

1.2 *R* Graphics

R manages its graphical output (see the basic example code in Listing 1.4 and its output in Figure 1) through *graphics devices* which take the graphics object data and convert them into printable or viewable form. The list of available graphics devices is extensive, there are devices for PDF, Postscript, X11, Java and SVG just to mention a few. Listing 1.5 shows for example how the PDF device can be used to produce high-quality PDF files.

```
a<- c(1:10)
plot(a)
```

Listing 1.4: The most basic *R* plot

```
pdf(file = "c:/punkte.pdf",width = 6,
height = 6, onefile = FALSE,
family = "Helvetica",
title = "R Graphics Output",
fonts = NULL, version = "1.4",
paper = "special")

a<- c(1:10)
plot(a)
# switch back to screen device
dev.off()
```

Listing 1.5: Example code for the PDF device

1.3 The TikZ Graphics Device

Especially interesting for TEXnicians is the TikZ device which generates source code for the respective LATEX package. This device either creates files that can be compiled standalone or just the graphics code to be embedded in a LaTeX document. The advantage of this device – if compared with others – is that the internal fonts of the document are used and mathematical code may be used in captions and labels as well. Listing 1.7 shows an excerpt from the file generated by the code from Listing 1.6.

```
tikz(file = "c:/test2.tex",standAlone=F)
# StandAlone=T
plot(1:10)

dev.off()
```

Listing 1.6: Example code for the TikZ device

```
% Created by tikzDevice
\begin{tikzpicture}[x=1pt,y=1pt]
\draw[color=white,opacity=0] (0,0)
rectangle (505.89,505.89);
\begin{scope}
\path[clip] ( 49.20, 61.20) rectangle (480.69,456.69);
\definecolor[named]{drawColor}{rgb}{0.56,0.96,0.51}
\definecolor[named]{fillColor}{rgb}{0.13,0.09,0.52}
\definecolor[named]{drawColor}{rgb}{0.00,0.00,0.00}
\draw[color=drawColor,line cap=round,line join=round,
fill opacity=0.00,] ( 65.18, 75.85) circle (  2.25);
\draw[color=drawColor,line cap=round,line join=round,
fill opacity=0.00,] (109.57,116.54) circle (  2.25);
\end{scope}
\begin{scope}
```

Listing 1.7: Excerpt from the generated TikZ code

2 Sweave and *R*

2.1 Introduction

In the second part of the article I want to introduce the `Sweave` package, developed by Friedrich Leisch. `Sweave` is part of the standard *R* installation so it requires no additional effort to install.

The package allows to include both *R* and LATEX code in a single file. The *R* code is enclosed in "noweb"-tags, <<>>= for the beginning, @ for the end. Noweb is a free tool implementing Donald Knuth's approach of literate programming, for more details on this topic please see the respective articles in Wikipedia. The noweb-file is then processed within *R* using the command `Sweave("<filename>")`. To extract the *R* code from the file, Sweave also provides a second command, `Stangle`.

Listing 2.1 shows a very basic example, just calculating $1 + 1$.
When we process the file from Listing 2.1 using the `Sweave` command in *R* we receive the LaTeX document shown in Listing 2.2. This document can then be compiled to PDF or DVI shown in Figure 2. As we can see in the document, Sweave requires the LATEX-package of the same name which provides

```
\documentclass{article}
\begin{document}
<<>>=
1+1
@
\end{document}
```

Listing 2.1: Basic Sweave example

commands for the input and output of Sweave code to be in the search path.

```
\documentclass{article}
\usepackage{Sweave}
\begin{document}

\begin{Schunk}
\begin{Sinput}
> 1 + 1
\end{Sinput}
\begin{Soutput}
[1] 2
\end{Soutput}
\end{Schunk}
\end{document}
```

Listing 2.2: LaTeX document generated from Listing 2.1

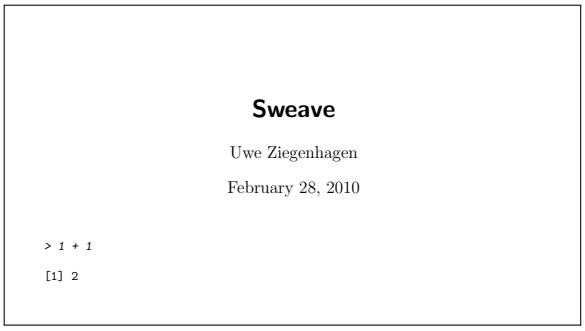

Figure 2: PDF-file generated from Listing 2.1

2.2 Sweave Options

Sweave allows various options to be set within the <<>>= tag, `echo=false` for example suppresses the output of the original *R* source, `results=hide` suppresses the output of results. A combination of both options may not make sense however it can be used to load data in the beginning of the analysis or set default values for variables, etc.

Since there are *R* packages that directly create valid LaTeX-source, Sweave supports `results=tex`, a mode that passes the generated output through to LaTeX without tempering with its content.

If images are created in an *R* code chunk the option `fig=true` needs to be set. The default setting is to create Postscript and PDF versions of each plot, with `pdf=true/false` respectively `eps=true/false` this behavior can be adjusted. Finally, the size of the plot can be set using the `width` and `height` parameter specifying the width and height of each plot in inches. Options may also be set globally by \SweaveOpts<Option>, see the Sweave manual for details.

Sweave also implements the noweb way of reusing code chunks, a certain piece of code can be named with «*name*, opt=...»=, the user may then address these parts with «*name*».

For scalar results which can be for example embedded in the running text, Sweave provides the \Sexpr<*R*-code>. The only requirement for the code is that the return value must either be a string or an object that can be converted to string. We will use this command in the following example, shown in Listing 2.3.

2.3 The Iris Example

Listing 2.3 shows a brief example for a statistical analysis of the well-known iris dataset, consisting of each 50 observations for three different species of iris flowers. In the first *R* code chunk the data is loaded, since this step may not be relevant for the reader we omit both the output and the *R* code.

To print the number of rows and columns for this dataset we use the \Sexpr() command in the LaTeX-text before displaying a small summary of the data. Afterwards we have *R* compute the linear model for two variables and print the results using *R*'s xtable command which provides the output in valid LaTeX syntax. Therefore we prevent Sweave of putting the code into verbatim by specifiying `results=tex`. Finally the last code chunk plots a scatterplot for the variables Petal.length and Sepal.width, please note the `fig=true` statement here, specifying that the result is a picture.

2.4 Dynamic Reports

The final example shows how reports with dynamic data sources can be created easily. Let's suppose we need a report on the USD/EURO exchange rates on a frequent basis. The data can downloaded from the homepage of the European Central Bank where it is provided in XML or CSV format. The retrieval process can be controlled from *R* using the `system()` command calling an external wget (standard on Linux/Unix, Windows users may need to install it.) To extract the data from the zip-file the *R*-internal zip-tool is used and the data set stored in the variable *data*.

```
\documentclass[a4paper]{scrartcl}

\begin{document}

<<echo=false,results=hide>>=
data(iris) # load iris data
@

The data set has \Sexpr{ncol(iris)}
  columns
and \Sexpr{nrow(iris)} rows.

<<echo=false>>=
summary(iris$Petal.Length)
@

<<echo=false,results=tex>>=
xtable(lm(iris$Sepal.Width~iris$Petal.
  Length),
caption="Linear Model of Sepal.Width
and Petal.Length")
@

\centering
\begin{figure}[h]
<<fig=true,echo=false>>=
pch.vec <- c(16,2,3)[iris$Species]
col.vec <- c(16,2,3)[iris$Species]
plot(iris$Sepal.Width,iris$Petal.Length,
col = col.vec,pch=pch.vec)
@
\caption{Plot of iris\$Petal.Length vs.
  iris\$Sepal.Width}
\end{figure}

\end{document}
```

Listing 2.3: Sweave code to generate Figure 3

We use the `\Sexpr()` command to print the dimensions of the dataset in our document before plotting a chart (see Figure 4) with the development of the Euro/Dollar exchange rate.

3 Conclusion

R provides an incredible set of functions for all aspects of data analysis, together with Sweave user can easily generate dynamic reports holding the results and methods leading to them in just one document. With this article I want to encourage everybody to give *R* a try when facing a data analysis challenge. If there are questions or remarks please feel free to contact me.

References

[intro] *R* Core Team *An Introduction to R* `http://cran.r-project.org/doc/manuals/R-intro.pdf`

[Ligges] Uwe Ligges *Programmieren mit R* Springer-Verlag

[crawley] Michael J. Crawley *Statistics – An Introduction using R* Wiley

[MaindonaldBrown] John Maindonald und John Brown *Data Analysis and Graphics Using R* Cambridge

[dalgaard] Peter Dalgaard *Introductory Statistics with R* Springer-Verlag

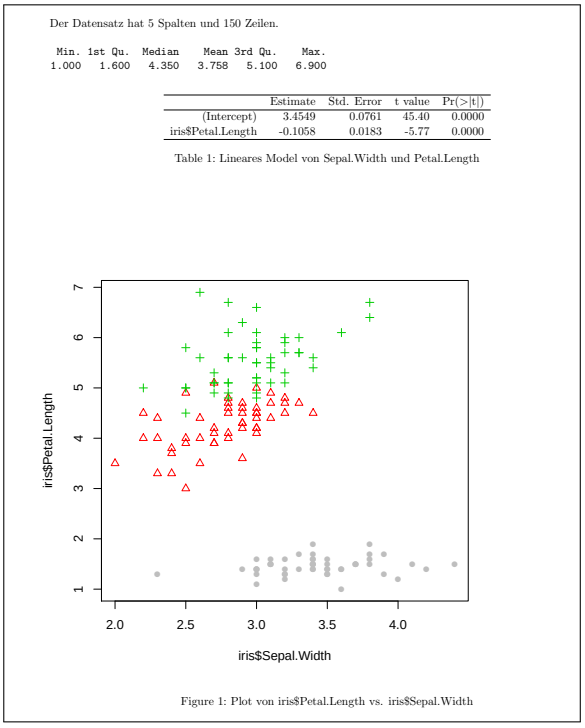

Figure 3: Document generated from Listing 2.3

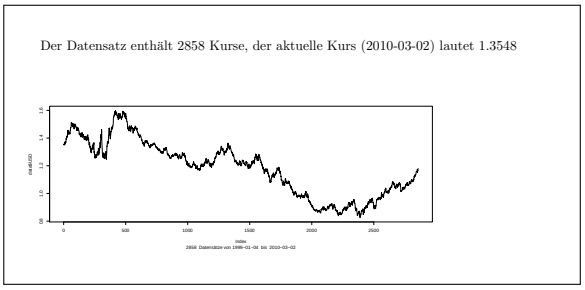

Figure 4: Exchange rate example

```
\documentclass{scrartcl}
\begin{document}

<<echo=f,results=hide>>=
windows(width = 8, height = 4)
system("wget -O d.zip http://www.ecb.int/
  stats/eurofxref/eurofxref-hist.zip")
zip.file.extract(file="eurofxref-hist.csv"
  ,zip="d.zip",unzip="",dir=getwd())
data= read.csv("eurofxref-hist.csv",sep=",
  ",header=TRUE)
@

The data contains \Sexpr{nrow(data)} rates
  , the latest rate (\Sexpr{data$Date[1]})
   was \Sexpr{data$USD[1]}

\centering
\begin{figure}[h]
<<fig=true,echo=false,width=15,height=6>>=
plot(data$USD,t="l", sub=paste(nrow(data),
  " datasets from ",data$Date[nrow(data)],
  " until ",data$Date[1]),asp=)
@
\end{figure}
\end{document}
```

Listing 2.4: Sweave code with dynamic data source