Introduction to Biostatistical Computing

Literate Programming

Arthur Allignol

arthur.allignol@uni-ulm.de

Table of Contents

Introduction

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.
— D. Donoho

- The concept of reproducible research is based on the idea of *literate programming* such that the logic of the analysis is clearly represented in the final product by combining computer code/programs with ordinary human language
  $\Rightarrow$ Combine analysis code and report

Literate Programming with R

2 main tools

- **Sweave** (2002): a tool that allows to embed the R code for complete data analyses in latex documents. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change.
  - Sweave is part of every R installation
- **knitr**: “The knitr package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package”
  e.g., caching, html and markdown

Data Example

Televisions, Physicians, and Life Expectancy

The data set contains information on

- Life expectancies in 40 countries with population $\geqslant$ 20 millions (1990)
  - For female and male. Average used as the country's overall life expectancy
- Number of people per television set
- Number of people per physician

Data set taken from the *The World Almanac and Book of Facts*, 1993

Data Example

Televisions, Physicians, and Life Expectancy

	Country	life	tv	phys	fem	male
1	Argentina	70.5	4.0	370	74	67
2	Bangladesh	53.5	315.0	6166	53	54
3	Brazil	65.0	4.0	684	68	62
4	Canada	76.5	1.7	449	80	73
5	China	70.0	8.0	643	72	68
6	Colombia	71.0	5.6	1551	74	68

Data Example

Televisions, Physicians, and Life Expectancy

Is the number of people per TV set a good predictor of life expectancy?

Table of Contents

Sweave

- A .tex file with .Rnw extension
- Include R code as 'chunks' or inline

- .Rnw file converted to .tex using Sweave
- The .tex file contains output from R; no raw code

- .tex file converted to .pdf (or .ps, .dvi)

Sweave

.Rnw file

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

\begin{document}

Some R code:
<<>>=
x <- rnorm(1000)
@
Let's display an histogram of {\tt x}
\begin{figure}[!h]
  \centering
<<myhist, fig = TRUE, eps = FALSE>>=
hist(x)
@
\caption{An histogram of {\tt x}}
   \label{fig:hist}
\end{figure}

\end{document}
```

Sweave

.tex file

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

\begin{document}

Some R code:
\begin{Schunk}
\begin{Sinput}
> x <- rnorm(1000)
\end{Sinput}
\end{Schunk}

Let's display an histogram of {\tt x}
\begin{figure}[!h]
  \centering
\begin{Schunk}
\begin{Sinput}
> hist(x)
\end{Sinput}
\end{Schunk}
\includegraphics{simple_example-myhist}
\caption{An histogram of {\tt x}}
   \label{fig:hist}
\end{figure}

\end{document}
```

Sweave

Some R code:

```
> x <- rnorm(1000)
```

Let's display an histogram of `x`

```
> hist(x)
```

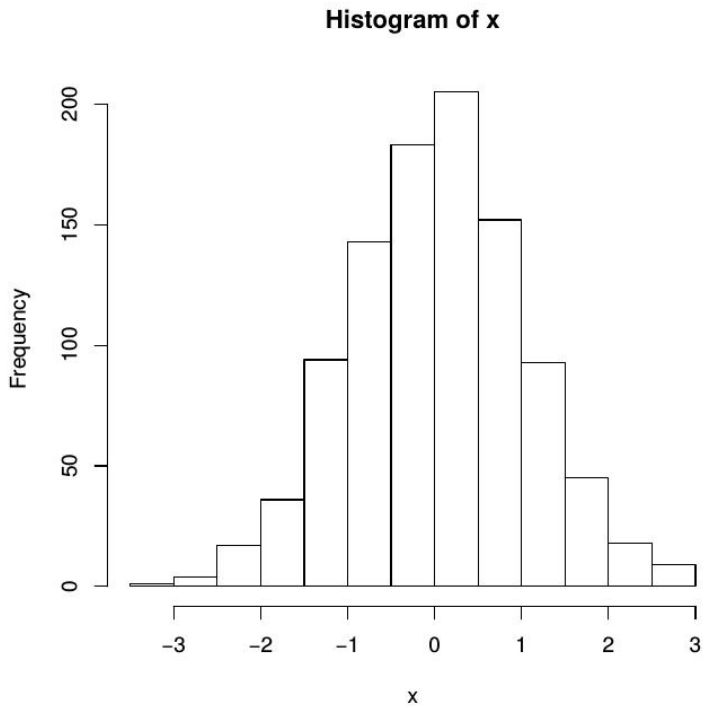

Figure 1: An histogram of `x`

Sweave

Code chunks

R code is entered in the LaTeX document using *code chunks*.

```
<< >>=
@
```

- Text within the code chunk is interpreted as R code
- Arguments for the code chunk are entered within `<<here>>=`

Sweave

Code chunks options

- `eval`: Default is `TRUE`. If set to `FALSE`, code is not evaluated
- `echo`: Include R code in the output file. Default is `TRUE`
- `keep.source`: Default to `FALSE`. If `TRUE`, the original source code is copied to the file when echoing. Otherwise it gets through R's parsing engine
- `results`: Default to `verbatim`, the output of the R code is copied "verbatim" to the file. Other option is `tex` for latex output, e.g., table, or `hide`
- `include`: Default to `TRUE`. If you want to place the figure yourself, add `include=FALSE`

Sweave

Code chunks options

- `fig`: Default to `FALSE`. Indicate whether the code chunk produces a graphical output
- `eps`: Indicates whether a .eps figure should be generated. Default is `TRUE` but ignored if `fig` is `FALSE`
- `pdf`: generate .pdf figure. Default to `TRUE`
- `width` and `height`: width and height of the figures in inches (default is 6)

```
\SweaveOpts{option1 = value1, option2 = value2, ...}
```

in the preamble sets default for all the code chunks in the document

Sweave

Named code chunks

Named code chunks can be reused in other code chunks later in the document

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

```
<<b>>=
x + y
@
```

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

Sweave

Figures

Easy to include figures in your document

```
\begin{figure}
\begin{center}
<<my_fig, fig=TRUE, echo=FALSE, width=10, eps = FALSE>>=
plot(x, y)
@
\caption{A beautiful figure}
\end{center}
\end{figure}
```

The chunk name is used as figure name, i.e., *myfile*-myfig.pdf

Sweave

Figures

Alternative:

```
<<my_fig, fig=TRUE, echo=FALSE, width=10, include = FALSE>>=
plot(x, y)
@
\begin{figure}
\begin{center}
\includegraphics{myfile-myfig.pdf}
\caption{A beautiful figure}
\end{center}
\end{figure}
```

Sweave

Tables

`results` option should be set to `tex`

Necessitates R packages that convert R object into LaTeX

- **xtable**: The most general
- **stargazer**, **texreg**, ..., provides LaTeX representations for several models (e.g., linear models, mixed models)

See `http://cran.r-project.org/web/views/ReproducibleResearch.html` for a relatively exhaustive list

Sweave

xtable

```
<<results=tex, echo=FALSE>>=
library(xtable)
y <- rnorm(100)
x <- rnorm(100)
fit.lm <- lm(y ~ x)
xtable(summary(fit.lm)$coefficients)
@
```

Sweave

xtable

```
<<results=tex, echo=FALSE>>=
library(xtable)
y <- rnorm(100)
x <- rnorm(100)
fit.lm <- lm(y ~ x)
xtable(summary(fit.lm)$coefficients)
@
```

```
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
  \hline
 & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
  \hline
(Intercept) & -0.037 & 0.090 & -0.418 & 0.677 \\
  x & -0.020 & 0.092 & -0.217 & 0.829 \\
   \hline
\end{tabular}
\end{table}
```

Sweave

xtable

Useful options in `xtable()`

- `caption`
- `label`: For cross-reference in LaTeX
- `align`: alignment of the columns
- `digits`: number of digits to display

Useful options in `print.xtable()`

- `floating.environment`: control the floating environment. Default is `table`
- `table.placement`: Control placement of the table in the pdf. Default is `ht`
- `include.rownames` and `include.colnames`

See `?xtable` and `?print.xtable` for all the options

Sweave

Expressions

All objects within a code chunk is in R's global environment each time a document is compiled

This allows the information saved in the global environment to be reproduced in the final document as inline text via *expressions*

```
<<echo = FALSE>>=
x <- rnorm(100)
@

The mean of $x$ is \Sexpr{round(mean(x), 2)}
```

Sweave

Expressions

All objects within a code chunk is in R's global environment each time a document is compiled

This allows the information saved in the global environment to be reproduced in the final document as inline text via *expressions*

```
<<echo = FALSE>>=
x <- rnorm(100)
@

The mean of $x$ is \Sexpr{round(mean(x), 2)}
```

The mean of x is 0.08

Sweave

Run Sweave

In Rstudio

- Set up an option that decides on using Sweave or knitr
- Click on “compile pdf”

From within R

- `Sweave("foo.Rnw")` produces the .tex file
- texi2dvi("foo.tex", pdf = TRUE) for compiling the .tex file within R

In the shell

- `R CMD Sweave foo.Rnw` produces the .tex file
- `pdflatex foo.tex`
- `pdflatex foo.tex`
  produces the pdf (done 2 times to actualise references, etc.)

In emacs (+ESS)

- `M-n s` for Sweave, `M-n P` for pdflatex

Table of Contents

knitr

knitr essentially reimplements and extends Sweave

- Support for additional document types: Markdown, html, reStructuredText
- Improved chunk options
- Code decoration
- Support multiple graphics devices
- Better plot manipulation capabilities
- Cacheing
- Different error handling

knitr

knitr not included in R, so you might have to install it

```
install.packages("knitr")
```

and load it in R

```
library("knitr")
```

To process a report document from within R

```
knit("my_report.Rnw")
```

knitr is well integrated within Rstudio

knitr

Chunk options

For .Rnw files, it works mostly as with Sweave

```
<<some_code, eval=TRUE, results="asis">>=
```

- As in Sweave, chunk options should be written in one line
- Option values must be valid R expressions (unlike in Sweave)

knitr

Chunk Options

- `eval`
- `echo`
- `results` character that takes 3 possible values
  - `'markup'` mark up the results using the output hook, e.g. put results in a special LaTeX environment
  - `'asis'` output as-is, i.e., raw output of R
  - `'hide'`
- `highlight` whether to highlight the source code
- `dev` The graphical device to record plots on, e.g., `'pdf'` for latex, `'png'` for html
- `cache` caching?

Comprehensive list of options `yihui.name/knitr/options`

knitr

Chunk Options

Unlike Sweave, global options for knitr are set in R

```
<<setup, include=FALSE>>=
opts_chunk$set(fig.width=5, fig.height=5)
@
```

This example will constrain the dimensions of all figures to 5×5 inches

Inline code

As in Sweave using `\Sexpr{}`

knitr

Figures

- No need of an extra option in the chunk header to include a figure. In Sweave, `fig=true` was needed
- (at least for latex) no need to wrap the R chunk around `\begin{figure} \end{figure}` if `fig.cap='caption'` is specified
- It is possible to include multiple graphs within one chunk. Figures will be automatically put side-by-side with one caption as indicated by `fig.cap`
- Choice of the graphical device through the `dev` argument
- You can automatically put all automatically generated figures into a separate directory using the `fig.path` chunk option, e.g.,

```
<<fig.path="Graphics", caption="Some text">>=
```

knitr

Figures

Control of the plot size made easier

- `fig.height` and `fig.width` control the size of the plot made by the plotting device. Defaults are 7 inches
- `out.width` and `out.height` control the size of the plot in the output document. For LaTeX, e.g.,

```
out.width="10cm"
out.width=".7\\linewidth"
```

knitr with HTML

html is the the standard markup language used to create web pages
HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

knitr with HTML

Code chunks

```
<!--begin.rcode echo=FALSE, results='asis'
x <- rnorm(100)
end.rcode-->
```

knitr with HTML

Code chunks

```
<!--begin.rcode echo=FALSE, results='asis'
x <- rnorm(100)
end.rcode-->
```

- In my experience, medical doctors don't like pdf
- An HTML file can be opened in Word

knitr with Markdown

Markdown is a plain text formatting syntax[5] designed so that it can optionally be converted to HTML using a tool by the same name

```
Heading
=======

Sub-heading
-----------

Paragraphs are separated
by a blank line.

Text attributes *italic*,
**bold**, `monospace`.

A [link](http://example.com).

Shopping list:

  * apples
  * oranges
  * pears
```

knitr with Markdown

Heading

Sub-heading

Paragraphs are separated by a blank line.

Text attributes *italic*, **bold**, `monospace`.

A link.

Shopping list:

- apples
- oranges
- pears

Numbered list:

1. apples
2. oranges
3. pears

The rain—not the reign—in Spain.

knitr with Markdown

Code chunk

````
```{r, echo=FALSE}
x <- rnorm(100)
```
````