The `Rmarkup()` function in the `doBy` package

Søren Højsgaard

January 29, 2012

Contents

1 Introduction

The `Rmarkup()` function in the **doBy** package provides facilities for making reproducible statistical analyses. `Rmarkup()` translates an R–script (a file with R commands and text comments) into an HTML document. This HTML document contains the text (possibly with some markup) and the R–code along with the results from executing the R–code (i.e. tables, graphics etc). Section 2 shows an example of what `Rmarkup()` does.

Reproducible analysis for R combined with LaTeX is facilitated by the Sweave system, [**?**]. The **odfweave** package for R, [**?**] provides similar facilities for using R in conjunction with the text format OpenDocumentText (as used in the OpenOffice program suite). `Rmarkup()` is patterned after `Sweave`, see also `http://www.stat.uni-muenchen.de/~leisch/Sweave/Sweave-manual.pdf`. In particular, specification of R–code follows the `noweb` syntax also employed by Sweave. `Rmarkup()` allows some markup facilities for the text. These are inspired by `txt2tags` markups (see `http://txt2tags.org/`). (Details are provided in Section 3). `Rmarkup()` is implemented by using the `RweaveHTML` driver in the **R2HTML**, [**?**].

2 Introductory example

A small example is shown in Figure 1. This R–script contains R code and text comments (in the lines starting with `##`).

This script is processed with

```
library(doBy)
Rmarkup("Ex1Puro.R")
```

and the result is an HTML document which is shown in Figures 2, 3 and 4.

3 Markup of R-script

All text lines start with one or more hashes (`#`) because these lines are to be regarded as comments by R.

- A line starting with one or two hashes is regarded as a text which is transferred (possibly after some additional processing; see below) to the resulting HTML document.
- A line starting with three hashes is not transferred to the HTML document. (This is useful e.g. for TODOs).

3.1 Headings and text beautifiers

- Headings at different font sizes are produced with (there can be 6 levels of headings):

  `= Title level 1 =`, `== Title level 2 ==`, `=== Title level 3 ===`
- The time of creation of the HTML document is produced by

  `%%date`.
- Beautifiers: **boldface**, *italics*, underline, `monospace` : These are produced with:

  `**`boldface`**`, `//`italics`//` `__`underline`__`, `&&`monospace`&&`
- The beautifiers can be combined in any way, e.g.

  `**__`some text`__**`.
- The text beautifiers can be used in the headings.

3.2 Markups must appear on one line

- All text markups must appear on one line. For example, the following will produce a heading in a large font,

  ```
  ## = HERE COMES A TITLE =
  ```

  whereas a heading in a large font will not be produced if one writes e.g.

  ```
  ## =
  ##   HERE COMES A TITLE
  ## =
  ```

3.3 R code

- A chunk of R code that does not produce graphics: The beginning and end of a chunk of R code can be marked with `##@@` and `##@`. For example:

```
##@@
x <- 1:100
sum(x)
##@
```

  or by using the noweb syntax

```
##<<>>=
x <- 1:100
sum(x)
##@
```

- A chunk of R code that does produce graphics: The beginning and end of a chunk of R code can be marked with `##@@fig` and `##@`. For example:

```
##@@fig
x <- 1:100
plot(x)
##@
```

  or by using the noweb syntax

```
##<<fig=T>>=
x <- 1:100
plot(x)
##@
```

- Notice: Various options may be speficied between `<<` and `>>=`; see the example.

4 More advanced use of `Rmarkup()`

It is possible to customize the HTML file by using a css–file. One may also control the filename for the report. For example

```
Rmarkup("Ex1Puro.R",cssfile="R2HTML.css", postfix="withCSS")
```

This creates the file `Ex1Puro-REPORT2.html` which (by default) is located in the working directory of R. The location of the resulting HTML file can be changed by specifying the `destdir`. The optional `css`–file must reside in this directory.

5 Final remarks

A very natural question is why one would possibly use `Rmarkup()` when much more elaborate systems (e.g. Sweave and odfWeave) are available:

- A major advantage of `Rmarkup()` is that the input to the function is a plain R script file (a text file).
- A design goal of `Rmarkup()` has been that no additional software must be installed.
- There are facilities for marking up text, but these are few and easy to remember. However one can always embed standard HTML code in the script file for more advanced markups (for example for creating lists etc.)

`Rmarkup()` is based processing the source file line–by–line (using `gsub()`) and therefore all text markups must not be split over several lines. The workhorse of `Rmarkup()` is the `Sweave()` function using the `RweaveHTML` driver of the `R2HTML` package.

It is sometimes convenient to convert the HTML report to a pdf file and this can be done in many ways. A simple way (at least on Windows platforms) is to use the `CutePDF` virtual printer. Another option is to open the HTML report in `OpenOffice` and from there export the document in pdf format.

```
##    == The Puromycin data  ==
##    === Søren Højsgaard  ===
##    %%date

## === The __Puromycin__ data ===
## The first lines of data are:

##@@
head(Puromycin,3)
nr = nrow(Puromycin)
## @
## There are \Sexpr{nr} rows in the dataframe.
## (Notice that we may refer to R expression in the text).

## Transformation almost gives __linearity__
##@@fig
par(mfrow=c(1,2))
plot(rate~conc,        data=Puromycin, col=as.numeric(state))
plot(1/rate~I(1/conc), data=Puromycin, col=as.numeric(state))
## @

## Fit a model to **transformed** data
## <<>>=
m1 ← lm(1/rate~state + I(1/conc) + state*I(1/conc), data=Puromycin)
summary(m1)
## @

## Model diagnostics
## <<fig=T,HTMLheight=400,HTMLwidth=600>>=
par(mfrow=c(2,2))
plot(m1)
## @

### TODO: Maybe more could be done...
```

Figure 1: An R–script file with a few markups of text.

The Puromycin data

Søren Højsgaard

2012-01-29 23:14:37 CET

The Puromycin data

The first lines of data are:

> head(Puromycin, 3)

	conc	rate	state
1	0.02	76	treated
2	0.02	47	treated
3	0.06	97	treated

> nr = nrow(Puromycin)

There are 23 rows in the dataframe. (Notice that we may refer to R expression in the text).
Transformation almost gives linearity

> par(mfrow = c(1, 2))

> plot(rate ~ conc, data = Puromycin, col = as.numeric(state))

> plot(1/rate ~ I(1/conc), data = Puromycin, col = as.numeric(state))

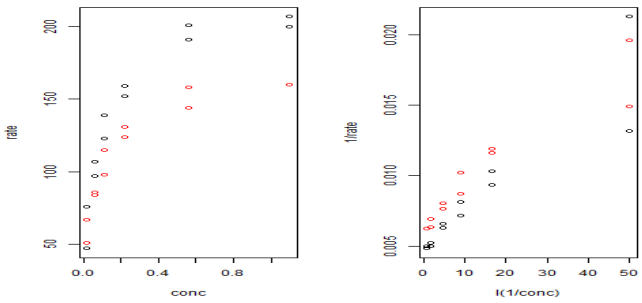


Figure 2: The resulting HTML document produced by `Rmarkup()`.

Fit a model to transformed data

> m1

> summary(m1)

- Call: lm(formula = 1/rate ~ state + I(1/conc) + state * I(1/conc), Call: data = Puromycin)
- Residuals

Min	1Q	Median	3Q	Max
-0.0043	-0.0005	-0.0002	0.0009	0.0038

- Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	5.11e-03	6.27e-04	8.14	1.3e-07	***
stateuntreated	1.86e-03	9.20e-04	2.03	0.057	.
I(1/conc)	2.47e-04	2.86e-05	8.64	5.2e-08	***
stateuntreated:I(1/conc)	-3.22e-05	4.10e-05	-0.79	0.442	

--- Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

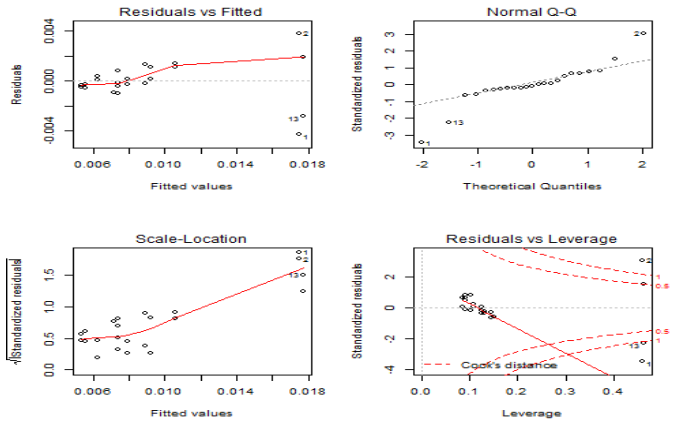
- Residuals standard error: 0.002 on 19 degrees of freedom
- Multiple R-Squared:**0.876**
- Adjusted R-Squared:**0.856**
- F-statistics: **44.614** on 3 and 19 DF. P-value:**0**.

Model diagnostics

> par(mfrow = c(2, 2))

> plot(m1)

Figure 3: The resulting HTML document produced by `Rmarkup()`.



The Puromycin data 3

Figure 4: The resulting HTML document produced by `Rmarkup()`.