Automated Reporting with RMarkdown

L. Egidi - DEAMS, University of Trieste, Italy

June 2020

Contents

> "I thoroughly disapprove of duels. If a man should challenge me, I would take him kindly and forgivingly by the hand and lead him to a quiet place and kill him."
>
> — Mark Twain

Before starting

Install and load in your R consolle the rmarkdown package:

```
install.packages("rmarkdown")
library(rmarkdown)
```

Why Rmarkdown?

A statistical analysis often requires a suitable documentation output consisting of plots, tables and formulas. In order to create such a *final* document, people usually pay much attention to copy and paste their results, conceptually splitting the time for the elaboration from the time of the analysis. However, we can work more automatically. For instance, once we modify our data/models file, we could need to consequently and automatically modify our analysis output; in fact, modifying the data/models and *then* modify the analysis file *by hand* has many drawbacks, such as:

- the error probability increases;
- the working time increases;

- there is not an automatic check between the last version of the elaborations and the last version of the output.

The objective of this chapter is to give a quick overview about the redaction of technical reports with R; in such a way, the user might automatize his work and handle the input data along with the output data through the same routine. R Markdown supports dozens of static and dynamic output formats, including:

- HTML
- PDF
- MS Word
- Beamer
- HTML5 slides
- Tufte-style handouts
- books
- dashboards
- shiny applications
- scientific articles
- websites
- more

For a comprehensive and definitive guide about R Markdown, check this website:

https://bookdown.org/yihui/rmarkdown/

For other references, see Xie, Allaire, and Grolemund (2018) and Baumer and Udwin (2015).

Preamble

The preamble is the first step of your markdown document, where you specify the format type, the font size, the authors, the date, etc. Every preamble is delimited from the symbols — and contains the following main commands:

- **title**: presentation title
- **author**: author(s)
- **date**: date
- **output**: presentation type
  - pdf_document
  - word_document
  - html_document
  - beamer_presentation
- **fontsize**: font size
- **subtitle**: presentation subtitle
- **style**, **bibliography**, **theme**,...

To set up your first preamble:

- open an empty R file;
- copy and paste the preamble above;
- save the file in .Rmd format;
- compile.

Indentation

Indentation in RMarkdown is a substantial issue: after **output** above, the user must go to a new line, indent with the tab command of the keyboard and write **pdf_document**. In order to include a table of contents, the user needs to go to the new line, indent *twice* before writing **toc**: yes. Uncorrect indentation may cause compilation errors.

Sections and subsections

It is very easy to divide the document in sections and subsections in R Markdown: all you need is using the symbol # for the sections, and ## for the subsections. The sections and the subsection will be automatucally created at the beginning of your document.

Compilation and visualization

In the RMarkdown language, *compiling* means specifying the desired type of document and executing the command for producing it. All you need is to click on the **Knit** button in RStudio and waiting for some seconds before visualizing the Word document.

Embedding R code

Write down a log-likelihood function for some data

Assume to code a log-likelihood function for some Gaussian data by considering the parameter μ and with σ^2 fixed:

$$y_i \sim \mathcal{N}(\mu_i, \sigma^2),\ i = 1, \ldots, N.$$

Then the log-likelihood is:

$$\log L(\mu; y) = \sum_{i=1}^{N} \log \mathcal{N}(\mu_i, \sigma^2).$$

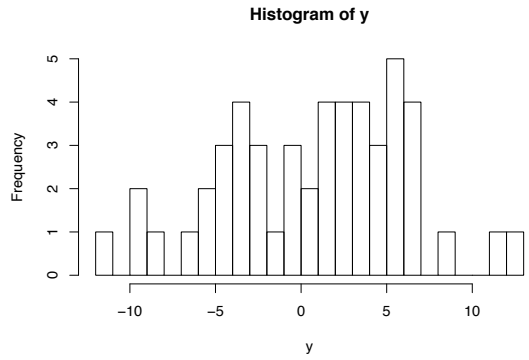
We can include our first R code, in R Markdown term a **chunk**: each chunk is delimited by the simbols “‘ and must start with the syntax {r chunk_name}:

```r
# generate some fake data
y <- rnorm(50, mean = 2, sd =5)
# code the log-likelihood
loglik <- function(data, param){
-sum(dnorm(x = data, mean = param, sd =5, log = TRUE))
}
```

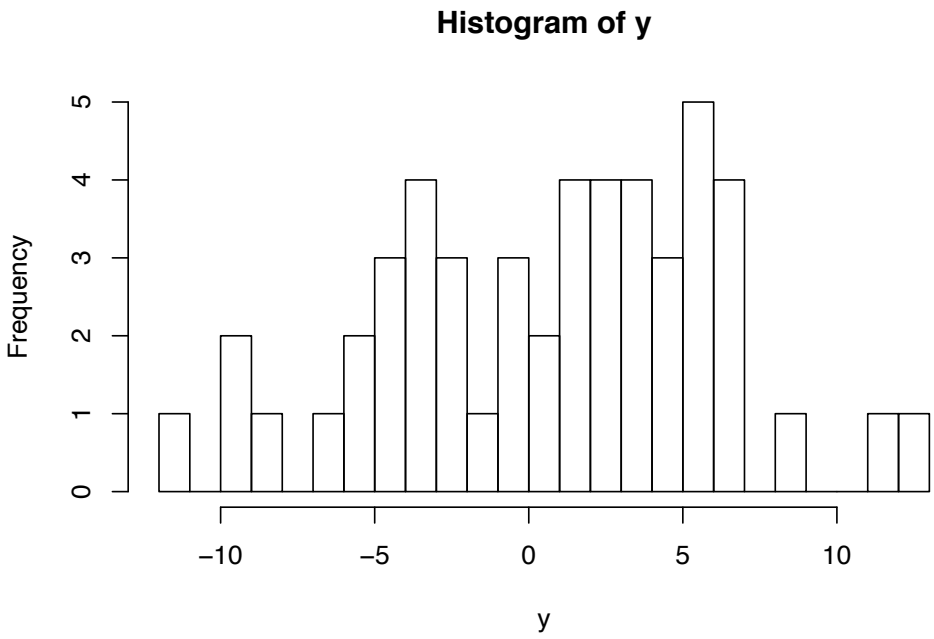
Plotting the results

Suppose now that you want to produce some plots. You can modify the size of your images with the `out.width` option in the following way:

```r
hist(y, breaks=30)
```

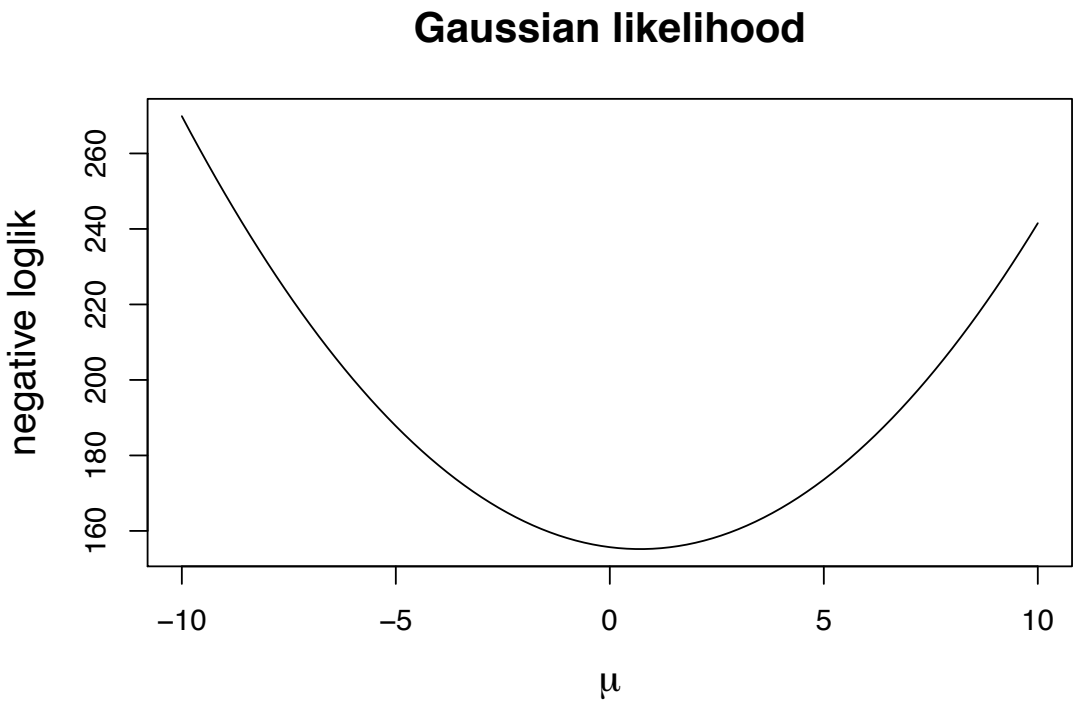



```r
hist(y, breaks=30)
```



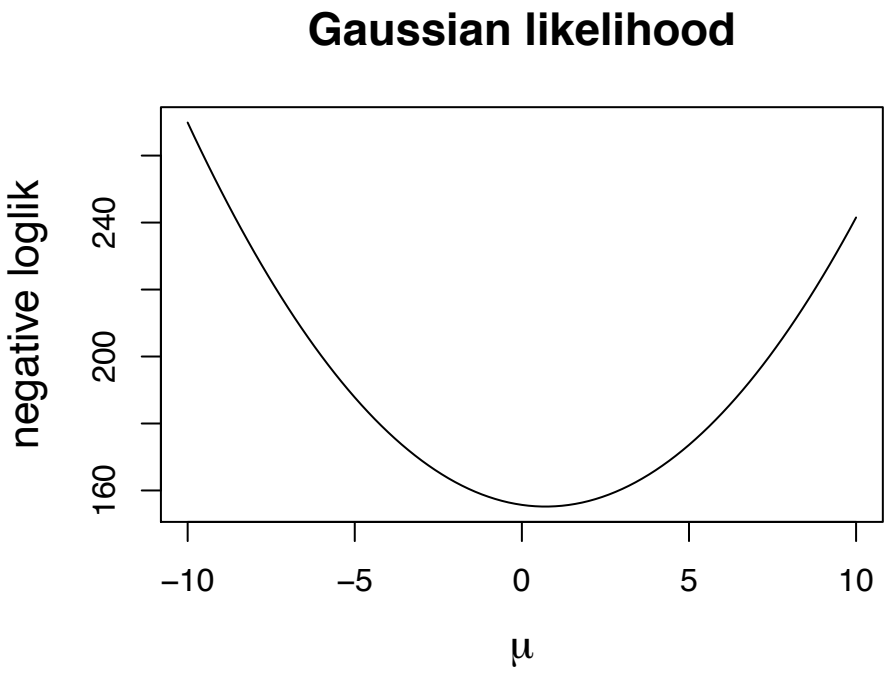
Let's now plot the log-likelihood function

```
# Vectorize command
loglik <- Vectorize(loglik, 'param')
# plot
curve(loglik(data=y, param = x), xlim = c(-10,10),
      xlab = expression(mu),
      ylab = "negative loglik", cex.lab = 1.3,
      main = "Gaussian likelihood", cex.main =1.4 )
```



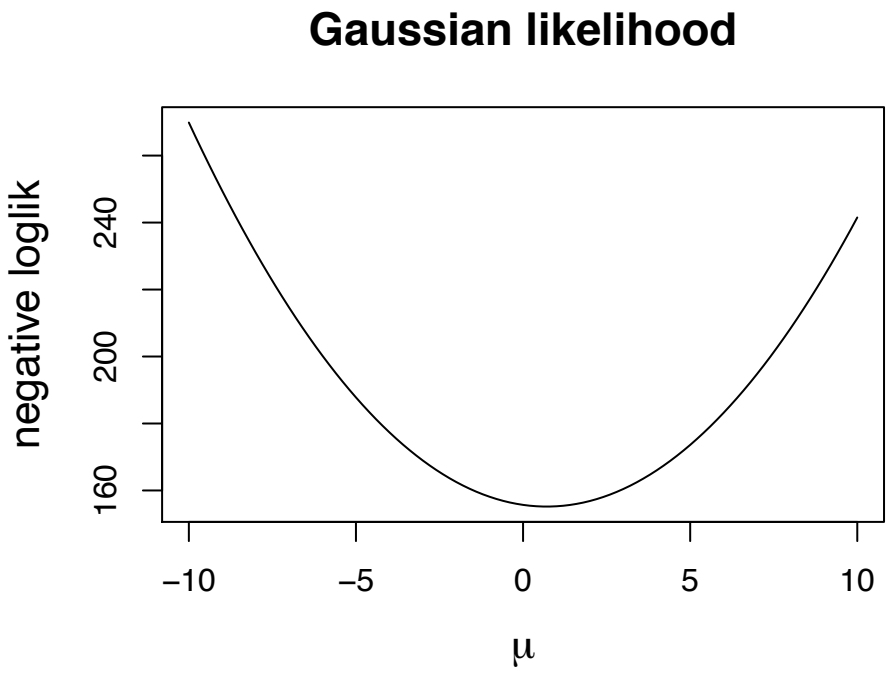
You can also modify the height and the width of your plot as follows, through the commands `fig.height` and `fig.width`:

```
# Vectorize command
loglik <- Vectorize(loglik, 'param')
# plot
curve(loglik(data=y, param = x), xlim = c(-10,10),
      xlab = expression(mu),
      ylab = "negative loglik", cex.lab = 1.3,
      main = "Gaussian likelihood", cex.main =1.4 )
```



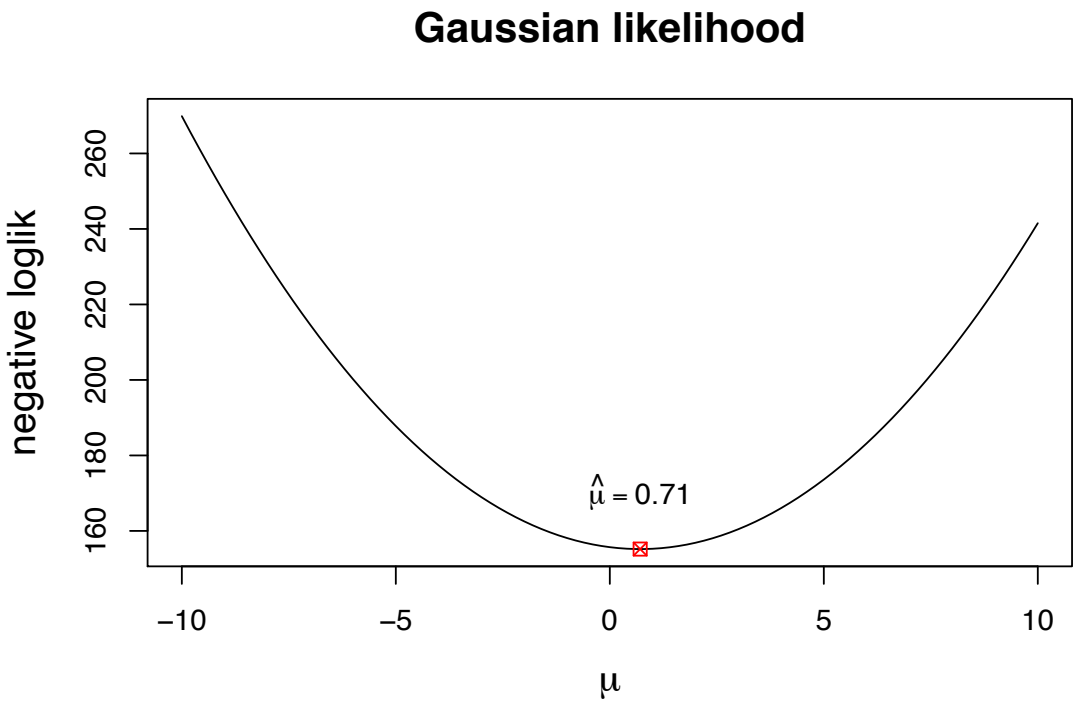
Finally, you can also align the plot with the argument `fig.align='center'`

```
# Vectorize command
loglik <- Vectorize(loglik, 'param')
# plot
curve(loglik(data=y, param = x), xlim = c(-10,10),
      xlab = expression(mu),
      ylab = "negative loglik", cex.lab = 1.3,
      main = "Gaussian likelihood", cex.main =1.4 )
```



We can find the Maximum Likelihood Estimate with numerical methods and then adding it in the plot above:

```
mle <- optimize(f = loglik, interval = c(-10, 10),
                data = y)
curve(loglik(data=y, param = x), xlim = c(-10,10),
      xlab = expression(mu),
      ylab = "negative loglik", cex.lab = 1.3,
      main = "Gaussian likelihood", cex.main =1.4 )
points(mle$minimum,  loglik(mle$minimum, data =y),
       col ="red", pch =7)
text(mle$minimum, loglik(mle$minimum, data =y)+ 15,
    bquote(hat(mu) == .(round(mle$minimum,2))))
```



To summarise, the main graphical chunck options are listed here:

- `fig.width` and `fig.height`: the (graphical device) size of R plots in inches. R plots in code chunks are first recorded via a graphical device in knitr, and then written out to files. You can also specify the two options together in a single chunk option `fig.dim`, e.g., `fig.dim = c(6, 4)` means `fig.width = 6` and `fig.height = 4`.
- `out.width` and `out.height`: the output size of R plots in the output document. These options may scale images. You can use percentages, e.g., `out.width = '80%'` means 80% of the page width.
- `fig.align`: the alignment of plots. It can be `'left'`, `'center'`,

or `'right'`.

- `dev`: the graphical device to record R plots. Typically it is `'pdf'` for LaTeX output, and `'png'` for HTML output, but you can certainly use other devices, such as `'svg'` or `'jpeg'`.
- `fig.cap`: the figure caption.

Other chunk options

There are many chunck opttions for your R code. You may want to hide the code, or to not evaluate it. For instance, something like:

```
a <- 2
b <- 3
a+ b
```

There are some basic options through the following arguments, which may be also combined:

- `echo`: can be set to FALSE if you want to evaluate your code, but not display it. Default is TRUE.
- `eval`: if set to FALSE, the code is shown but not evaluated.
- `warning`, `message`, and `error` : if set to FALSE, any eventual warning/message/error from your R code is suppressed.
- `results`: when set to `'hide'`, text output will be hidden; when set to `'asis'`, text output is written "as-is", e.g., you can write out raw Markdown text from R code.
- `cache`: whether to enable caching. If caching is enabled, the same code chunk will not be evaluated the next time the document is compiled (if the code chunk was not modified), which can save you time.

Tables

The easiest way to include tables is by using `knitr::kable()`, which can create tables for HTML, PDF and Word outputs. Table captions can be included by passing `caption` to the function, e.g.,

```
knitr::kable(iris[1:5, ], caption = 'Details from Iris dataset')
```

Table 1: Details from Iris dataset

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

If you are looking for more advanced control of the styling of tables, you are recommended to use the `kableExtra` package, which provides functions to customize the appearance of PDF and HTML tables. Formatting tables can be a very complicated task, especially when certain cells span more than one column or row. It is even more complicated when you have to consider different output formats. For example, it is difficult to make a complex table work for both PDF and HTML output. We know it is disappointing, but sometimes you may have to consider alternative ways of presenting data, such as using graphics.

Global document options

The user may customize his document in many ways: figure dimensions, font size,
aligning, etc. So far, the user may rely upon some global default options; in what follows, some other advanced options will be mentioned.

Mathematics

Many times we could need to include some formulas in the document. Including formulas in a R Markdown document comes from the Latex syntax. Each *in-line* formula is delimited by the `$` operator, whereas each centered formula is delimited by `$$`. For instance, writing `$z^2=x^2+y^2$` yields $z^2 = x^2 + y^2$, whereas writing `$$z^2=x^2+y^2$$` yields

$$z^2 = x^2 + y^2$$

.

Change format

You can easily change the format of your R Markdown presentation by adding the string, for instance, `html_document`, above the string `pdf_document`, and then clicking the `Knit` button.

It is worth to say that many times you can encounter errors when changing format; or, some text can be not well displayed. This happens because each format has its own features, related to tables, text font and other issues.

Alternatively, you can directly specify your format by accessing the window-menu close to `Knit` and selecting your favorite choice.

Theme

Valid HTML themes include: default, cerulean, journal, flatly, darkly, readable, spacelab, united, cosmo, lumen, paper, sandstone, simplex, and yeti. Pass `null` for no theme (in this case you can use the `css` parameter to add your own styles).

References

Baumer, Benjamin, and Dana Udwin. 2015. "R Markdown." *Wiley Interdisciplinary Reviews: Computational Statistics* 7 (3): 167–77.

Xie, Yihui, Joseph J Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. CRC Press.