Elements of Rmarkdown

David García Callejas

ELEMENTS OF A Rmd FILE

- CODE CHUNKS
  - show/hide source code: `echo`
  - show/hide output: `results`
  - show/hide warnings: `warning`
  - show/hide messages: `message`
  - show/hide figues: `fig.show="hide"`
  - show/hide all of the above, but running the chunk: `include`
  - gather outputs in a single block: `collapse`
  - run the code of the chunk: `eval`
  - global options: `knitr::opts_chunk\$set(...)\`
- TEXT FORMATTING
  - SECTIONS:
    - NOT NUMBERED: `#` before the name of the section

```
# Section
## Sub-section
### etc
#### etc
```

    - NUMBERED:

    Including `number_sections` in heading:

```
output:
  html_document:
    number_sections: true
  pdf_document:
    number_sections: true
```

    It is possible to add an unnumbered section inside a document with numbered sections:

```
# Unnumbered section {-}
```

  - ITALICS:

    `*text in italics*`, or `_text in italics_`

  - BOLD:

    `**bold text**`, or `__bold text__`

  - UNDERLINE:

    Does not exist as an option in Rmarkdown, but can be done indirectly using LaTeX:

```
$\text{\underline{Underlined text using \LaTeX syntax}}$
```

– FONT SIZE:

  Include in heading:

  ```
  fontsize: 12pt
  ```

– FONT COLOUR:

  We may use LaTeX syntax:

  ```
  $\color{red}{\text{text in red}}$
  ```

  text in red

  If we generate html output, we may also use its syntax:

  ```
  <span style="color: red;">text in red for html output</span>
  ```

– FONT FAMILY:

  This option is not available in the `.Rmd` file. We need to create an auxiliary file that will specify the "style" that the LaTeX translator will use to generate the final output. In the same folder where the `.Rmd` file is located, create a file with `.sty` extension. If we want to use the "Biolinum sans serif" font, this will be the content of the file `my_style.sty`:

  ```
  \usepackage{biolinum}
  \renewcommand{\familydefault}{\sfdefault} % sans serif
  \fontfamily{ppl}\selectfont
  ```

  Note that, besides setting the font name, it tells LaTeX to use the sans serif by default (the line "renewcommand"). If you choose a serif font, this line will not be necessary. The catalogue of available fonts in LaTeX can be found in https://tug.org/FontCatalogue/.

  For these changes to take effect, we need to reference the style file in the heading of our `.Rmd`:

  ```
  ---
  output:
  pdf_document:
      includes:
          in_header: my_style.sty
  ---
  ```

– HYPERLINKS:

  ```
  [text in the document](URL address)
  ```

– NUMBERED LISTS:

  ```
  1. First element
  2. Second element
     1. sub-element

        text in sub-element 1
     2. sub-element 2
  ```

  It is important not to forget the dot after every number, to begin every sub-list at the same point where the first character of the above level is, and to leave an empty line between the level heading and the text.

– UNNUMBERED LISTS:

  ```
  * First element
  * Second element
    + sub-element
  ```

```
    - sub-sub
  + sub-element 2
```

– PARAGRAPHS/LINE BREAKS:

To begin a new paragraph, add to spaces at the end of the previous one, or an empty line:

A sentence that should mark the end of a paragraph. Another paragraph?

A sentence that should mark the end of a paragraph.
By leaving two spaces after the previous sentence, now it works.

The html option `<br>` also adds a line break.

– HORIZONTAL LINES:

Three asterisks or three underscores (separated from the rest of the text by line breaks) draw a horizontal line: `***`/`---`

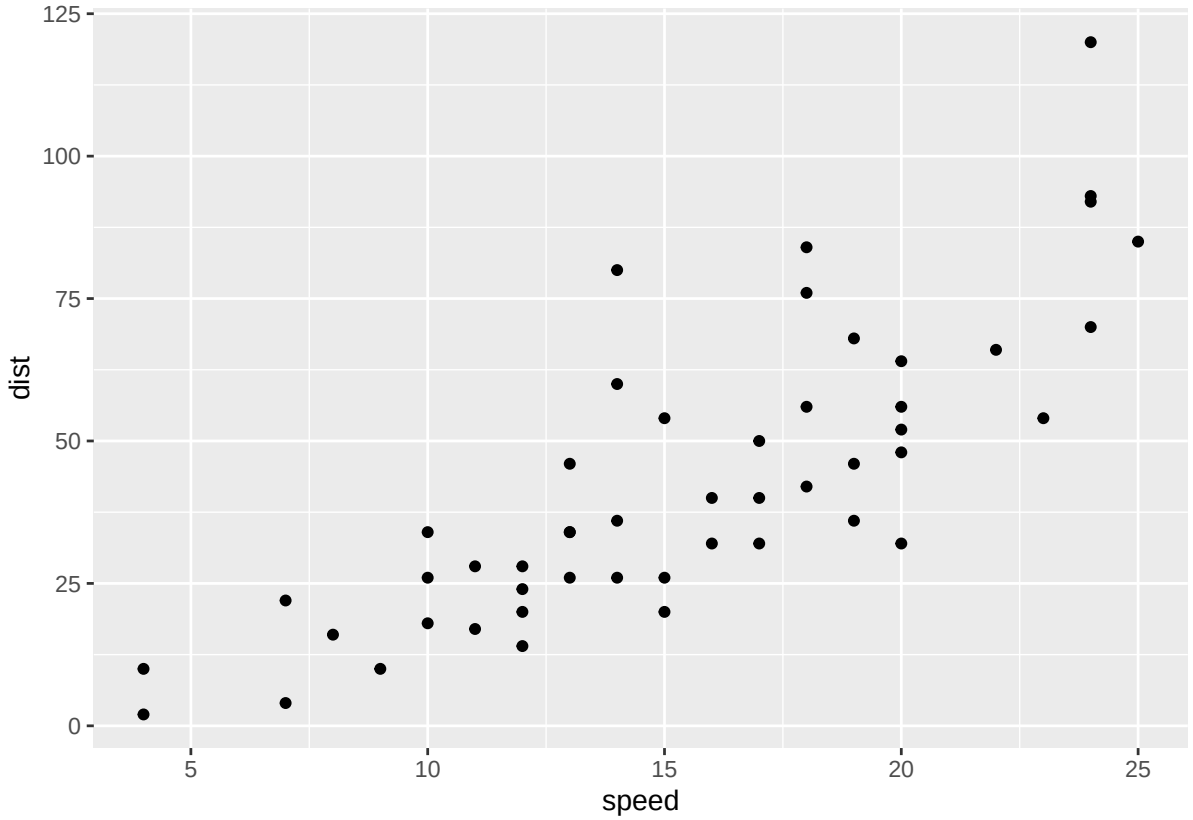
– PAGE BREAKS:

`\newpage`

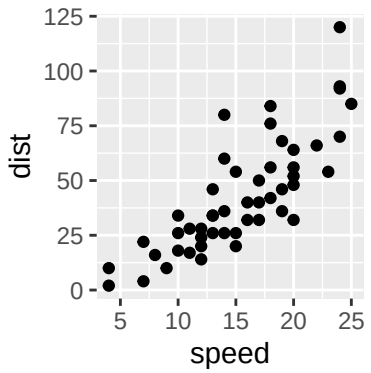
- FIGURES

Rmarkdown allows to include in our documents external figures as well as dynamical ones, generated inside R. For the inside-generated ones, it is enough with create a figure inside a R code chunk.

```
ggplot(cars, aes(speed, dist)) + geom_point()
```

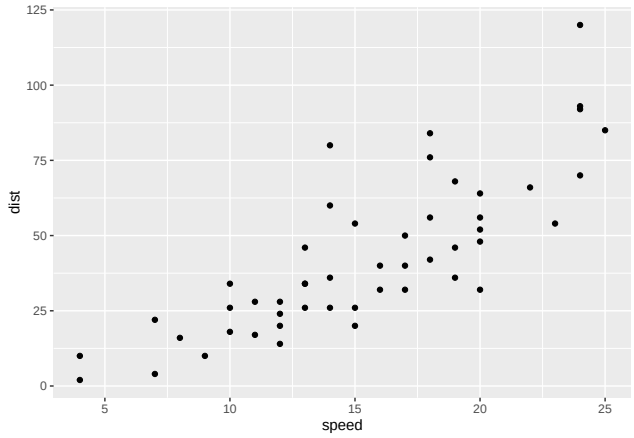


The options `fig.width` and `fig.height`, whose arguments are given in inches, allow to adjust size and resolution of the generated figures. For example, the figure below is generated with the parameters `{r fig.width = 2,fig.height = 2}`:



A more intuitive alternative can be to use the options `out.width` and `out.height`, that take percentage values relative to the box where the image is placed:

```
```{r out.width = "50%"}
ggplot(cars, aes(speed, dist)) + geom_point()
```
```



There are two ways of including external images. The first one is with the syntax `![caption](path_to_image)`. For example, the image below is generated with the syntax `![Iberian lynx](../data/figures/Lynx_pardinus.jpg)`. The image path is always *relative* to the folder where the `.Rmd` is located.

Figure 1: Iberian lynx

This syntax is very simple but does not allow for much flexibility. To modify the size or other characteristics of an image, it is more appropriate to import images inside code chunks, using the function `include_graphics`. In the heading of the code chunk we may specify the align (`fig.align`), size (as we saw above), etc. We may also specify the caption with `fig.cap`. For this last option, we also need to specify in the heading of the `.Rmd` file the following:

```
  pdf_document:
    fig_caption: true
```

This is what the code chunk looks like:

````
```{r lynx, fig.align = 'center', out.width = "50%", fig.cap='Iberian lynx second time'}
    knitr::include_graphics(here::here("data/figures", "Lynx_pardinus.jpg"))
```
````

Figure 2: Iberian lynx second time

- TABLES

  Rmarkdown allows to create tables on the fly:

variable	value
v1	v1 value.
v2	v2 value.

  There is a wealth of options to tweak these tables, but we will not cover them here (you may consult the Pandoc manual). In general, it is much more useful and efficient to automatically generate tables from data in a matrix or dataframe format. For doing so, a standard way is through the `kable` function of the package `knitr`:

```
knitr::kable(mtcars[1:5,])
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

Table 2: basic kable table

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2

Just like we did with figures, we can add a caption to our tables, among other options

```
knitr::kable(mtcars[1:5,],
             caption = "basic kable table",
             digits = 2)
```

Besides `kable`, there are many packages that help create tables with different formats, like the `stargazer` package:

````
```{r results='asis', echo = FALSE}
stargazer::stargazer(mtcars[1:5,],
summary = FALSE,
header = FALSE,
title = "basic stargazer table")
```
````

Table 3: basic stargazer table

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.900	2.620	16.460	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.900	2.875	17.020	0	1	4	4
Datsun 710	22.800	4	108	93	3.850	2.320	18.610	1	1	4	1
Hornet 4 Drive	21.400	6	258	110	3.080	3.215	19.440	1	0	3	1
Hornet Sportabout	18.700	8	360	175	3.150	3.440	17.020	0	0	3	2

It is important to note that `stargazer` generates LaTeX or html code, that will be passed on to the final document. As these tables are generated inside code chunks, the `knitr` engine needs to understand that this output is not the result of any R operation, but rather must be compiled as if it was text. Hence the option `results='asis'` in the chunk definition. This must always be specified when generating `stargazer` tables.

A recurrent problem in LaTeX is that figures, and also tables, often do not appear exactly where we defined them within a document. This is because the LaTeX compiler tries to automatically place these elements where it thinks is more appropriate, often breaking the flow of the document. We can use a workaround to force LaTeX to maintain the original placement. For this we need to include a series of LaTeX commands in our `.Rmd` file. So, first, we need to create a LaTeX file, with extension `.tex`, in which we will write the following commands:

```
\usepackage{float}
\let\origfigure\figure
\let\endorigfigure\endfigure
\renewenvironment{figure}[1][2] {
  \expandafter\origfigure\expandafter[H]
} {
  \endorigfigure
}
```

We will save this file as, for example, `latex_options.tex`, in the same folder as the `.Rmd` file. Then, in the heading of our `.Rmd` file we will tell it to include these commands when compiling the file:

```
output:
pdf_document:
  fig_caption: yes
  includes:
    in_header: latex_options.tex
```

Lastly, regarding tables, it is very common to report directly the results of statistical analyses in tables. We can use, again, the `stargazer` package:

```
example <- lm(Petal.Width ~ Petal.Length,data = iris)

stargazer::stargazer(example,
header = FALSE,
label = "tab:regression1",
type=ifelse(knitr::is_latex_output(),"latex","html"),
title="regression table with stargazer")
```

Table 4: regression table with stargazer

	Dependent variable:
	Petal.Width
Petal.Length	0.416***
	(0.010)
Constant	−0.363***
	(0.040)
Observations	150
R^2	0.927
Adjusted R^2	0.927
Residual Std. Error	0.206 (df = 148)
F Statistic	1,882.452*** (df = 1; 148)
Note:	*p<0.1; **p<0.05; ***p<0.01

- FIGURE AND TABLE CAPTIONS

  By default, captions are created with the expressions `Figure X:` and `Table X:`. You can choose the language of these expressions by adding the option `lang: es` (for spanish) in the heading. Likewise, you can change these expressions by including the following, either directly in the `.Rmd` file or in a referenced LaTeX file (of course, modifying the text inside braces with your own expressions):

```
\def\figurename{MyFigureCaptionText}
\def\tablename{MyTableCaptionText}
```

- MATHEMATICAL EXPRESSIONS

  Rmarkdown can profit from the rich LaTeX mathematical syntax. Math expressions can be added either inline, by encapsulating them between two dollar symbols `$ -expression- $`, (e.g. $\lambda = f - 1$), or as independent expressions, with the syntax:

```
\begin{equation}
  mi-ecuación
\end{equation}
```

For example:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{1}$$

Math syntax in LaTeX is overwhelmingly rich. The example above displays fractions, sums, subscripts, etc, and it is written like this: `\bar{X} = \frac{\sum_{i=1}^n X_i}{n}`. Every element is defined with an inverted bar, the name of the element, and its value between braces. A fraction, for example, is defined as `\frac{numerator}{denominator}`.

Here is a list of all available symbols: https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols; and here, a lengthy document about this syntax: https://en.wikibooks.org/wiki/LaTeX/Mathematics.

- REFERENCES
  - REFERENCING FIGURES, TABLES, EQUATIONS

    For this functionality, we need to add the following in the heading (subsituting 'pdf' with 'html' or 'word' if so we want):

    ```
    output:
      bookdown::pdf_document2: default
    ```

    If we have other options in our heading, such as the ones we saw above, and we want to be able to generate both pdf and html output, a full heading would look like this so far:

    ```
    output:
      bookdown::pdf_document2:
        fig_caption: yes
        includes:
          in_header: latex_options.tex
      bookdown::html_document2:
        fig_caption: yes
        df_print: paged
    ```

    Then, every figure and table needs to be defined inside one independent chunk, with different ids, and needs to have a valid caption. For example, the chunk we defined above for showing an external image has the id 'lynx':

    ````
    ```{r lynx, fig.align = 'center', out.width = "50%",
        fig.cap='Iberian lynx, second time'}
    knitr::include_graphics(here::here("data/figures", "Lynx_pardinus.jpg"))
    ```
    ````

    We can place a dynamic reference to that figure with the command `\@ref(fig:lynx)`. In the processed document it will be a reference to Figure 2. Likewise, we can reference tables (it does not matter if the tables or figures referenced are defined previously or below in the .Rmd file): `\@ref(tab:'id')`. For example, with the 'table2' chunk above: `\@ref(tab:table2)` will create a dynamic reference to table 2. Note that the syntax for defining captions in stargazer (table 4) is slightly different.

    We may also reference equations. For doing so, when writing the equations, we can add an id that will allow us to place a reference for it:

    ```
    \begin{equation}
      \bar{X} = \frac{\sum_{i=1}^n X_i}{n} \label{eq:mean}
    \end{equation}
    ```

    The part `(\#eq:mean)` is the relevant one here. Now, we only need to reference it with `\@ref(eq:mean)`, (here we reference Eq. (1)).

– REFERENCES TO EXTERNAL PUBLICATIONS OR DOCUMENTS

One of the most interesting functionalities of Rmarkdown is the ability to generate dynamic references to external documents and generate a bibliography section in our documents. For it, we need a few things. First, we need to compile a list of the references to include in our document. This list will be contained in an external document with extension `.bib`. This file will be stored in the same folder as the `.Rmd` file. The structure of this file is relatively straightforward: it will hold a separate entry for each element we may want to reference, and each entry is written with the following syntax:

```
@Book{ggplot2,
  author = {Hadley Wickham},
  title = {ggplot2: Elegant Graphics for Data Analysis},
  publisher = {Springer-Verlag New York},
  year = {2016},
  isbn = {978-3-319-24277-4},
  url = {http://ggplot2.org},
}
```

After the first '@' we specify the type of document (Book, in the example). Inside the braces, the first word is the document 'id', the name that we will use to reference it. After this id, a series of fields follow, but these are optional and depend on the type of document.

In most situations, thankfully, it will not be necessary to type these .bib files by hand. Any reference manager worth its name (Zotero, Endnote, etc) is able to export a list of references in this format easily. Furthermore, search engines like Google Scholar can also export any reference in .bib format (by clicking on the quotation mark symbol below the name, and select 'Bibtex').

Either way, once we compiled all our references in a .bib file, e.g. 'references.bib', we only need to link it in the heading of our `.Rmd` file, with the option

```
bibliography: references.bib
```

When this is done, we can cite its elements. In this example I included two references, the first one with id 'Broman2018' and the second one with 'Rodriguez-Sanchez2016'. To cite them it suffices to write `@Broman2018`, or inside square brackets if we want the reference to appear inside parentheses: `[@Broman2018]`. This is how it looks like when I cite Broman and Woo (2018) and (Rodriguez-Sanchez et al. 2016). We can also cite the year (`[-@Broman2018]`), or specific pages (`[@Broman2018 2-4]`). By referencing at least one .bib entry, a reference list is automatically created at the end of our document, such that we can name it as we wish (in this document, I named it 'references', and I show it in a new page after everything else). To move this section somewhere else in our document, we can use the command `<div id="refs"></div>`, and the list of references should appear in that position. Despite that command being html code, it should work fine when compiling on pdf.

It is also possible to modify the style of our references (see examples in https://www.zotero.org/styles). Every style is specified by yet another external file, in this case with extension '.csl'. For example, if we want to use the reference style of the journal 'Nature', we can download the appropriate 'nature.csl' file to our folder, and link it in the heading of our .Rmd:

```
csl: nature.csl
```

– FOOTNOTES

Can be created with this syntax: `text^[footnote]`. For example[1].

- PAGE FORMATTING

[1] this is a footnote

- INDEX

  If we are using `bookdown` (e.g. for captions), we can automatically add a table of contents ('toc') with numbered/unnumbered sections, by playing with these options in the heading:

  ```
  bookdown::pdf_document2:
    number_sections: false
    toc: false
  ```

  This option is only available for pdf outputs, as far as I am aware.

- LINE NUMBERING

  To include line numbers throughout the text, but not in code chunks:

  ```
  header-includes:
    - \usepackage[left]{lineno}
    - \linenumbers
  ```

  To include them in code chunks, but not in the text, we need to include the following in the chunks we want to apply it to:

  ```
  attr.source='.numberLines'
  ```

  For example:

```
  if (TRUE) {
    x <- 1:10
    x + 1
  }
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11
```

This numbering is reset in each code chunk:

```
  y <- seq(1:5)
  x <- sqrt(y)
```

- MARGINS

  Include in the heading options inherited from the 'geometry' class in LaTeX:

  ```
  geometry: margin=1in
  ```

- CODE HIGHLIGHTING:

  Include in the heading the style we want (options are "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", and "haddock", can be consulted in https://www.garric kadenbuie.com/blog/pandoc-syntax-highlighting-examples/):

  ```
  output:
    pdf_document:
      highlight: tango
  ```

- PAGE HEADER/FOOTER

  For simple page headers and footers, (as the one of this document) we can directly include them in the heading of our `.Rmd`:

  ```
  header-includes:
    - \usepackage{fancyhdr}
    - \pagestyle{fancy}
    - \fancyhead[CO,CE]{header text}
    - \fancyfoot[CO,CE]{footer text}
  ```

Another interesting option is to remove page numbers that are generated by default when 'knitting' the document.

```
header-includes:
  - \pagenumbering{gobble}
```

In general, commands inside braces understand LaTeX syntax, so we can include page numbers (`\thepage`) or other elements. If we want something fancier, we can create a .tex file with the header/footer we want, and reference it in our heading:

```
output:
  pdf_document:
    includes:
      before_body: header.tex
      after_body: footer.tex
```

You can check some common elements in this webpage: https://bookdown.org/yihui/rmarkdown-cookbook/latex-header.html. These options are, of course, valid for generating pdf (or word) documents. As in html we do not expect independent pages in our document, it makes no sense to specify a page footer or header.

FULL HEADING

Here I summarise most of what we have been looking at, and show a heading that integrates most of them. Specifically, this is the heading of the `.Rmd` document that generates the pdf you are reading:

```
title: "Elements of Rmarkdown"
author: "David García Callejas"
lang: en
output:
  bookdown::pdf_document2:
    number_sections: false
    fig_caption: yes
    toc: false
    includes:
      in_header: latex_options.tex
  bookdown::html_document2:
    number_sections: false
    fig_caption: yes
    df_print: paged
bibliography: references.bib
header-includes:
- \usepackage{fancyhdr}
- \pagestyle{fancy}
- \fancyhead[CO,CE]{Elements of Rmarkdown}
```

References

Broman, Karl W., and Kara H. Woo. 2018. "Data Organization in Spreadsheets." *The American Statistician* 72 (1): 2–10. https://doi.org/10.1080/00031305.2017.1375989.

Rodriguez-Sanchez, Francisco, Antonio Jesús Pérez-Luque, Ignasi Bartomeus, and Sara Varela. 2016. "Ciencia reproducible: qué, por qué, cómo." *Revista Ecosistemas* 25 (2): 83-92-92. https://doi.org/10.7818/re.2014.25-2.00.