2.6 R code chunks and inline R code

You can insert an R code chunk either using the RStudio toolbar (the `Insert` button) or the keyboard shortcut `Ctrl + Alt + I` (`Cmd + Option + I` on macOS).

There are a lot of things you can do in a code chunk: you can produce text output, tables, or graphics. You have fine control over all these output via chunk options, which can be provided inside the curly braces (between ```` ```{r ```` and `}`). For example, you can choose hide text output via the chunk option `results = 'hide'`, or set the figure height to 4 inches via `fig.height = 4`. Chunk options are separated by commas, e.g.,

````
```{r, chunk-label, results='hide', fig.height=4}
````

The value of a chunk option can be an arbitrary R expression, which makes chunk options extremely flexible. For example, the chunk option `eval` controls whether to evaluate (execute) a code chunk, and you may conditionally evaluate a chunk via a variable defined previously, e.g.,

````
```{r}
# execute code if the date is later than a specified day
do_it = Sys.Date() > '2018-02-14'
```


```{r, eval=do_it}
x = rnorm(100)
```
````

There are a large number of chunk options in **knitr** documented at https://yihui.name/knitr/options. We list a subset of them below:

- `eval`: Whether to evaluate a code chunk.
- `echo`: Whether to echo the source code in the output document (someone may not prefer reading your smart source code but only results).
````

- `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 (like `cat('**Markdown** is cool.\n')`). By default, text output will be wrapped in verbatim elements (typically plain code blocks).

- `collapse` : Whether to merge text output and source code into a single code block in the output. This is mostly cosmetic: `collapse = TRUE` makes the output more compact, since the R source code and its text output are displayed in a single output block. The default `collapse = FALSE` means R expressions and their text output are separated into different blocks.

- `warning` , `message` , and `error` : Whether to show warnings, messages, and errors in the output document. Note that if you set `error = FALSE` , `rmarkdown::render()` will halt on error in a code chunk, and the error will be displayed in the R console. Similarly, when `warning = FALSE` or `message = FALSE` , these messages will be shown in the R console.

- `include` : Whether to include anything from a code chunk in the output document. When `include = FALSE` , this whole code chunk is excluded in the output, but note that it will still be evaluated if `eval = TRUE` . When you are trying to set `echo = FALSE` , `results = 'hide'` , `warning = FALSE` , and `message = FALSE` , chances are you simply mean a single option `include = FALSE` instead of suppressing different types of text output individually.

- `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. However, I want to honestly remind you of the two hard problems in computer science (via Phil Karlton): naming things, and cache invalidation. Caching can be handy but also tricky sometimes.

- `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.
- `child` : You can include a child document in the main document. This option takes a path to an external file.

Chunk options in **knitr** can be surprisingly powerful. For example, you can create animations from a series of plots in a code chunk. I will not explain how here because it requires an external software package, but encourage you to read the documentation carefully to discover the possibilities. You may also read Xie (2015), which is a comprehensive guide to the **knitr** package, but unfortunately biased towards LaTeX users for historical reasons (which was one of the reasons why I wanted to write this R Markdown book).

There is an optional chunk option that does not take any value, which is the chunk label. It should be the first option in the chunk header. Chunk labels are mainly used in filenames of plots and cache. If the label of a chunk is missing, a default one of the form `unnamed-chunk-i` will be generated, where `i` is incremental. I strongly recommend that you only use alphanumeric characters (`a-z` , `A-Z` and `0-9`) and dashes (`-`) in labels, because they are not special characters and will surely work for all output formats. Other characters, spaces and underscores in particular, may cause trouble in certain packages, such as **bookdown**.

If a certain option needs to be frequently set to a value in multiple code chunks, you can consider setting it globally in the first code chunk of your document, e.g.,

````
```{r, setup, include=FALSE}
knitr::opts_chunk$set(fig.width = 8, collapse = TRUE)
```
````

Besides code chunks, you can also insert values of R objects inline in text. For example:

```
```{r}
x = 5  # radius of a circle
```

For a circle with the radius `r x`,
its area is `r pi * x^2`.
```

2.6.1 Figures

By default, figures produced by R code will be placed immediately after the code chunk they were generated from. For example:

```
```{r}
plot(cars, pch = 18)
```
```

You can provide a figure caption using `fig.cap` in the chunk options. If the document output format supports the option `fig_caption: true` (e.g., the output format `rmarkdown::html_document`), the R plots will be placed into figure environments. In the case of PDF output, such figures will be automatically numbered. If you also want to number figures in other formats (such as HTML), please see the **bookdown** package in Chapter 12 (in particular, see Section 12.4.4).

PDF documents are generated through the LaTeX files generated from R Markdown. A highly surprising fact to LaTeX beginners is that figures float by default: even if you generate a plot in a code chunk on the first page, the whole figure environment may float to the next page. This is just how LaTeX works by default. It has a tendency to float figures to the top or bottom of pages. Although it can be annoying and distracting, we recommend that you refrain from playing the "Whac-A-Mole" game in the beginning of your writing, i.e., desparately trying to position figures "correctly" while they seem to be always dodging you. You may wish to fine-tune the positions once the content is complete using the `fig.pos` chunk option (e.g., `fig.pos = 'h'`). See https://www.overleaf.com/learn/latex/Positioning_images_and_tables for possible values of `fig.pos` and more general tips about this behavior in LaTeX. In short, this can be a difficult problem for PDF output.

To place multiple figures side-by-side from the same code chunk, you can use the `fig.show='hold'` option along with the `out.width` option. Figure 2.5 shows an example with two plots, each with a width of `50%`.

```
par(mar = c(4, 4, .2, .1))
plot(cars, pch = 19)
plot(pressure, pch = 17)
```

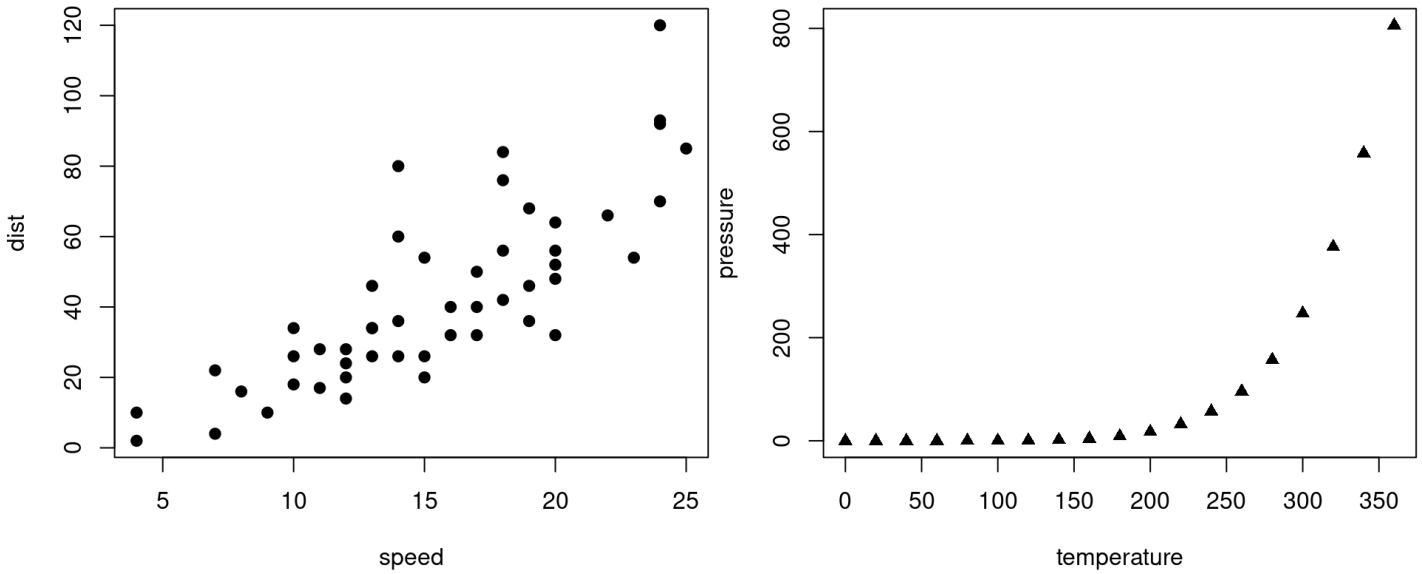

FIGURE 2.5: Two plots side-by-side.

If you want to include a graphic that is not generated from R code, you may use the `knitr::include_graphics()` function, which gives you more control over the attributes of the image than the Markdown syntax of `![alt text or image title](path/to/image)` (e.g., you can specify the image width via `out.width`). Figure 2.6 provides an example of this.

````
```{r, out.width='25%', fig.align='center', fig.cap='...'}
knitr::include_graphics('images/hex-rmarkdown.png')
```
````

FIGURE 2.6: The R Markdown hex logo.

2.6.2 Tables

The easiest way to include tables is by using `knitr::kable()`, which can create tables for HTML, PDF and Word outputs.[3] Table captions can be included by passing `caption` to the function, e.g.,

````
```{r tables-mtcars}
knitr::kable(iris[1:5, ], caption = 'A caption')
```
````

Tables in non-LaTeX output formats will always be placed after the code block. For LaTeX/PDF output formats, tables have the same issue as figures: they may float. If you want to avoid this behavior, you will need to use the LaTeX package longtable, which can break tables across multiple pages. This can be achieved by adding `\usepackage{longtable}` to your LaTeX preamble, and passing `longtable = TRUE` to `kable()`.

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.

We explain in Section 12.3 how the **bookdown** package extends the functionality of **rmarkdown** to allow for figures and tables to be easily cross-referenced within your text.

References

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.name/knitr/.

3. You may also consider the **pander** package. There are several other packages for producing tables, including **xtable**, **Hmisc**, and **stargazer**, but these are generally less compatible with multiple output formats.↩