Chapter 1

R markdown

One of the interesting features around R is the ability to combine data analysis and writing-up of results into one single document. In practice this means mixing R code (that will be executed) with visual content such as text or images in a file. One way to achieve this is with R Markdown. R Markdown files are text-based and end in in ".Rmd".

When a R markdown file is processed, the contained R code is executed and the results are written to a document, a process called "renderign" (See 1). Rmarkdown documents can be rendered into html, pdf, or word formats.

This has several potential advantages that tie nicely into good accepted good research practices:

- Connection of result reporting and analysis: The explicit connection strengthens truthful reporting of results, reproducibility of research, and transparency.
- Reduced effort: Results do not need to be updated in the report seperatly
- Avalability: Sharing the raw unprocessed file shares both the content and the contained code (ideally the underlying data is shared additionally as well)

R Studio offers an easy way to create R Mardkwon files. To create a basic R Markdown file in R Studio, go to File->New File->R Markdown.

An R Markdown file starts with the YAML header that denotes the title and the output format for the document. Three underscore characters on one line indicate the beginning and the end of the YAML header, which here defines a title and the output format:

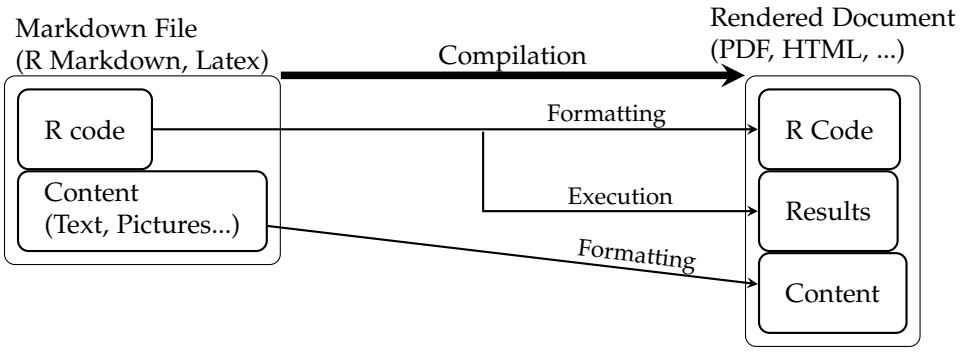

Figure 1.1: Schmeatic of embedding R in Markdown documents

```
---
  title: "Example document"
  output: "html_document"
---
```

After the YAML header follows the actual content. R markdown files contain two main components in addition to the YAML header: Text to be displayed in the document and R code chunks.

1.1 Text formatting

Unlike in visual text processors like Microsoft Word, the R Markdown files do not look like the final document during the editing. The R Markdown files themselves are simple text files. In addition to the actual content, they contain instructions how the content should be formatted in the final document. To get this final document, R markdown files need to be processed, or "compiled". The R package that does this processing is called *knitr*. In R studio, an R markdown document can be compiled by clicking on *Preview* in the toolbar. Alternatively, another output format such as PDF can be selected by clicking on the small downwards arrow to the right of *Preview*.

Text written in an R Markdown file will be put as is in the final document, subject to adjusting line breaks or similar.

R Markdown also provides basic text formatting capabilities. Some special characters combined with the text to be formatted instruct the document compiler to apply the respective formatting. For example, two spaces followed by a linebreak will produce a new paragraph. `*text*` puts the text in italic, `**text**` in bold.

Headings can be inserted by starting a line with `#` and sub-headings with two or more `##`.

R markdown has many more formatting commands to provide all basic formatting capabilities such as insertion of images, citations, lists, and more. See the R Markdown cheatsheet for a list. The R Markdown cheatsheet can also be found in the RStudio Help menu.

1.2 Code chunks and plots

R code chunks contain R code that is executed when the document is compiled. R code chunks start with ```` ```{r} ```` and end with ```` ``` ````

Consider this example:

````
```{r Scatterplot1, fig.align='right', fig.cap='Scatterplot 1'}
plot(cars)
```
````

In the curly `{}` brackets, after "r", follows the name of the chank (here Scatterplot1). Then follow one several options. After the curly `{}` brackets follows the R code. In the example a simple scatterplot of the in R per default available cars data.frame. More complex ggplot2 plots or similar can be included in R markdown documents in the same fashion.

For a list of useful code chunk options check the R markdown cheatsheet. Comprehensive lists can be found on the internet, for example at `https://yihui.org/knitr/options/`. Other important chunk options include `echo=FALSE` to disable printing the R code in the document, `results='hide'` to only run the code but not print the results, and `eval=FALSE` to not actually run the R code (the code itself is still printed, depending on the *echo* option)

There is also the option to include R code in R markdown documents in a shorter fashion, using the inlines syntax. The inline syntax starts with `` `r ``, followed by the R code, and ends with a single `` ` ``. For example, `r Sys.time()` includes the output of *Sys.time()* in the text, without paragraph formatting and code output as with code chunks:

Current date and time: `r Sys.time()` → Current date and time: 2021-06-06 19:41:17

1.3 Tables

Putting tables in R markdown documents is can be done using the *kable* package:

````
```{r table1, results='asis'}
knitr::kable(head(cars))
```
````

The chunk option `results='asis'` is necessary because the kable function returns html that can be directly 'asis' integrated in a html document, and does not need to be processed further by knitr. The format that kable returns is determined automatically, in our case html if no other output format is selected manually.

Note that in the example the kable function is called using the double colon syntax, this way the knitr package does not have to be loaded.