

KateRMarkdown

Francesco Maria Sabatini

10/03/2021

Contents

1	Why are we here?	2
2	Introducing RMarkdown	3
2.1	Why RMarkdown	3
2.2	Briefest history on RMarkdown ever	3
2.3	What's Markdown	4
2.4	What's Latex	4
2.5	What's Pandoc	4
3	Can we look at cats now?	4
3.1	What am I seeing?	5
3.2	What happens when I <code>knit</code> my notebook?	6
3.3	How to create a beautiful RMarkdown document?	6
4	Slightly more advanced stuff	13
4.1	Long computing times	13
4.2	Alternative ways of knitting	13
4.3	PDFs and paper templates	14
5	Resources	14
6	Your Turn!	15
7	<code>sessionInfo()</code>	16

1 Why are we here?



Figure 1: A Non Reproducible workflow - '<https://www.youtube.com/watch?v=s3JldKoA0zw&t=5s>'

2 Introducing RMarkdown

That's why we are playing around with **RMarkdown** today.

Clearly, there's no best way of doing so than throwing in a bunch of *Cats*.



2.1 Why RMarkdown

RMarkdown is the easiest way to create interactive documents integrating text, code and output from your code. It is fairly versatile, has a shallow learning curve and, as it often happens in R, there is a bunch of people continuously expanding its functionalities and possibilities.

For instance, you can:

- Compile a single R Markdown document to a report in different formats, such as PDF, HTML, or Word.
- Create notebooks in which you can directly run code chunks interactively.
- Make slides for presentations (HTML5, LaTeX Beamer, or PowerPoint).
- Produce dashboards with flexible, interactive, and attractive layouts.
- Build interactive applications based on Shiny.
- Write journal articles.
- Author books of multiple chapters.
- Generate websites and blogs.

Yes, you guessed that. This very document has been generated using **RMarkdown**.

Not all that glitters is gold, though. Expect headaches when setting up your (especially Windows) machine to work if interested in knitting pdfs or more advanced stuff.

2.2 Briefest history on RMarkdown ever

Source: <https://bookdown.org/yihui/rmarkdown/>

The document format “R Markdown” was first introduced in the knitr package (Xie 2015, 2020c) in early 2012. The idea was to embed code chunks (of R or other languages) in Markdown documents.

However, the original version of Markdown invented by John Gruber was often found overly simple and not suitable to write highly technical documents. For example, there was no syntax for tables, footnotes, math expressions, or citations. Fortunately, John MacFarlane created a wonderful package named Pandoc (<http://pandoc.org>) to convert Markdown documents (and many other types of documents) to a large variety of output formats. More importantly, the Markdown syntax was significantly enriched. Now we can write more types of elements with Markdown while still enjoying its simplicity.

In a nutshell, R Markdown stands on the shoulders of knitr and Pandoc. The former executes the computer code embedded in Markdown, and converts R Markdown to Markdown. The latter renders Markdown to the output format you want (such as PDF, HTML, Word, and so on).

The rmarkdown package (Allaire, Xie, McPherson, et al. 2020) was first created in early 2014.

2.3 What's Markdown

[Markdown](#) is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Created by John Gruber in 2004, Markdown is now one of the world's most popular markup languages.

It's the language used to create README's and Project descriptions in GitHub

2.4 What's Latex

[Latex](#) which is pronounced «Lah-tech» or «Lay-tech» (to rhyme with «blech» or «Bertolt Brecht»), is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.

LaTeX is not a word processor! Instead, LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content.

You need LaTeX only for knitting to pdf documents. If you're happy with html notebooks, there's no need of installing it.

2.5 What's Pandoc

If you need to convert files from one markup format into another, [pandoc](#) is your swiss-army knife.

The good news is that your RStudio IDE has already a Pandoc installation embedded! Maybe it's not the latest, but it will work just fine most of the times.

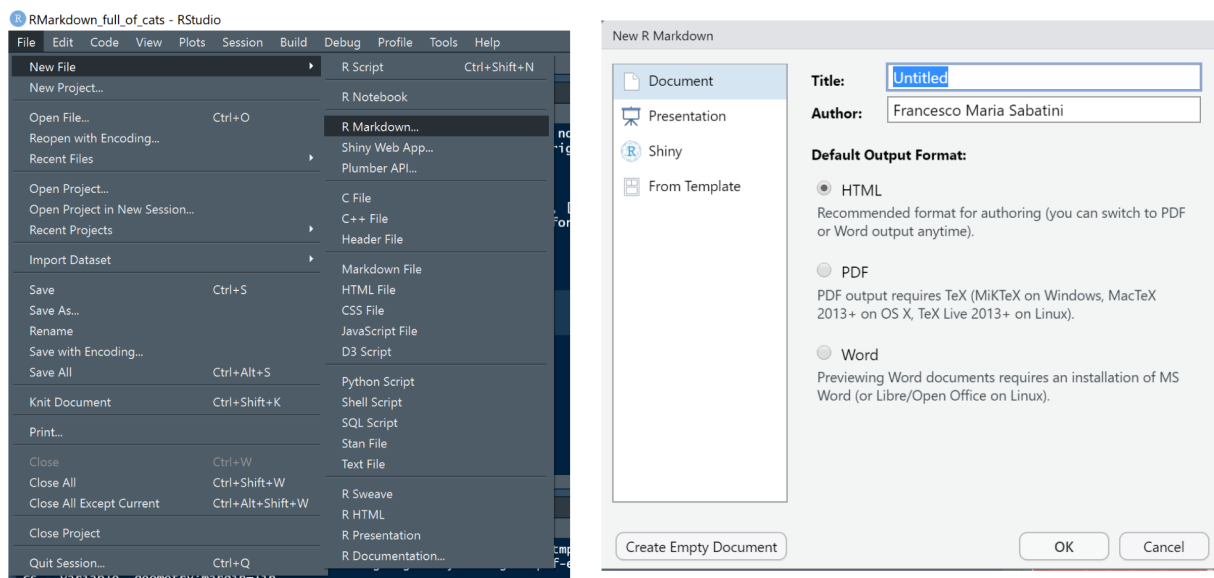
3 Can we look at cats now?

Almost. Fire up your RStudio and install **RMarkdown** first.

```
install.packages("RMarkdown")
```

Time to start your first **RMarkdown** document.

File -> New File -> R Markdown



We'll stick to html today. Knitting documents to pdf is undoubtedly cooler, but requires installing Latex, which might be tricky, depending on the machine one is using.

Congratulations - You've just created your first RMarkdown document.

3.1 What am I seeing?

You should be seeing something like this:



```
1 ---
2 title: "KateR"
3 author: "Francesco Maria Sebastiani"
4 date: "05/03/2021"
5 output: html_document
6 ---
7
8 {r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10
11
12 ## R Markdown
13
```

The first part is called the metadata.

The metadata is written between the pair of three dashes — The syntax for the metadata is YAML (YAML Ain't Markup Language, <https://en.wikipedia.org/wiki/YAML>), so sometimes it is also called the YAML metadata or the YAML frontmatter. Before it bites you hard, we want to warn you in advance that indentation matters in YAML, so do not forget to indent the sub-fields of a top field properly.

(Source - <https://bookdown.org/yihui/rmarkdown/basics.html>).

It sounds intimidating. However, you won't have to do much with your metadata most of the time, besides copy pasting it from a template. Phewww!

In the rest of the script you can distinguish between text and *chunks* of code. Note how chunks of code are enclosed by three **backtick** signs.

```
---
```

If you can't find the backtick sign in your keyboard, try with the ASCII code: **Alt+96**.

Look at the two buttons highlighted by the arrows.

- *Insert* — allows you to add a new chunk of code.
- *Knit* — renders your document to the preferred file type.

3.2 What happens when I knit my notebook?

Knitting your RMarkdown script means rendering it to your chosen output (html in this case). There is quite a lot of machinery (=dark magic) happening behind the scenes. Fortunately, for most applications we don't have to understand how these happen. Just enjoy the result.

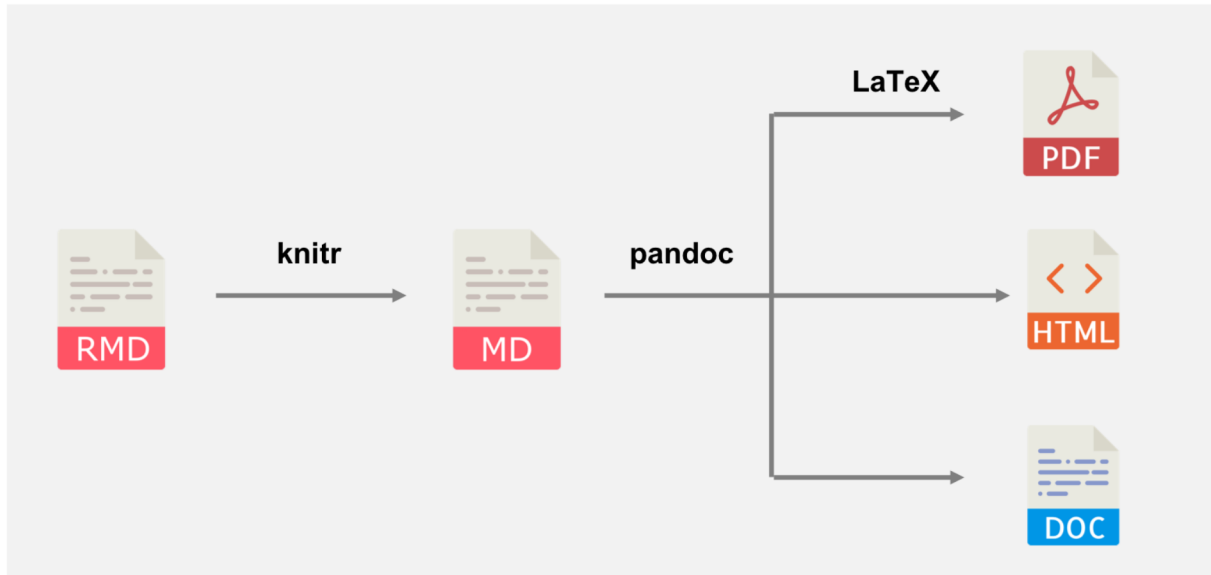


Figure 2: Workflow - Source: '<https://bookdown.org/yihui/rmarkdown-cookbook>'

3.3 How to create a beautiful RMarkdown document?

3.3.1 Playing around with Text

Let's start with the easiest. How to add text (=narrative), to my RMarkdown. Easy-peasy. Just type!

You just need to know a couple of things:

Going to new line -> Need to add *two whitespaces* at the end of a block of text.

Leaving an empty line -> Two ways: 1. `
`, 2. `\newline` (followed by an empty line).

Titles and Headers:

`# Header` - Header 1

`## Header` - Header 2

`### Header` - Header 3

Basic formatting. You can use some basic markdown formatting to make your text:

Italic: `*Felis catus*` -> *Felis catus*

Italic: `_Felis catus_` -> *Felis catus*

Bold: `**Felis catus**` -> **Felis catus**

Bold: `__Felis catus__` -> **Felis catus**
Both: `***Felis catus***` -> ***Felis catus***
Both: `____Felis catus____` -> ***Felis catus***

Adding Lynx:

[Eurasian Lynx - Wikipedia] (https://en.wikipedia.org/wiki/Eurasian_lynx)

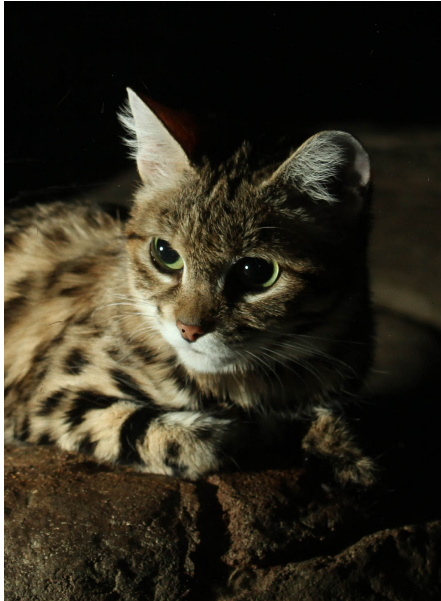
Renders as:

[Eurasian Lynx - Wikipedia](https://en.wikipedia.org/wiki/Eurasian_lynx)

Embed cat from url/file:

```
<center> ![Black Footed Cat] (https://en.wikipedia.org/wiki/Black-footed\_cat#/media/File:Black\_Footed\_Cat)  
</center>
```

Renders as:



Create numbered lists of cats (don't forget to leave an empty line before starting the list). E.g.,
My favourite wildcats:

1. Andean Cat (**Leopardus Jacobita**)
2. Rusty Spotted Cat (**Prionailurus Rubiginosus**)
3. Chinese Mountain Cat (**Felis Bieti**)
4. Kodkod (**Leopardus Guigna**)

Renders as:

1. Andean Cat (*Leopardus Jacobita*)
2. Rusty Spotted Cat (*Prionailurus Rubiginosus*)
3. Chinese Mountain Cat (*Felis Bieti*)
4. Kodkod (*Leopardus Guigna*)

Create unordered lists of cats:

Where small cats live:

- * Small Cats of South America

```

+ Andean Cat
+ Geoffroy's Cat
+ Jaguarundi
+ ...
* Small Cats of SE Asia
+ Leopard Cat
+ Marbled Cat
+ Fishing Cat
+ ...

```

Renders as:

- Small Cats of South America
 - Andean Cat
 - Geoffroy's Cat
 - Jaguarundi
 - ...
- Small Cats of SE Asia
 - Leopard Cat
 - Marbled Cat
 - Fishing Cat
 - ...

Do Yourself a Favour - and download the catsheet (=cheatsheet)

Catsheet - <https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>

3.3.2 Playing around with Code

To insert a chunk of code, just enclose it between three backticks, followed by {r}:

```
```{r}
```

```
4+4
```

```
```
```

It will render as:

```
4+4
```

```
## [1] 8
```

Let's make some practice. We need to import some cat-related data, first.

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3      v purrr  0.3.4
```

```
## v tibble  3.0.6      v dplyr  1.0.4
```

```
## v tidyr   1.1.2      v stringr 1.4.0
```

```
## v readr   1.4.0      v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

```
## Warning: package 'tibble' was built under R version 4.0.3
```

```
## Warning: package 'tidyr' was built under R version 4.0.3
```

```
## Warning: package 'readr' was built under R version 4.0.3
```



```
## Warning: package 'dplyr' was built under R version 4.0.3
## Warning: package 'forcats' was built under R version 4.0.3
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()

big.cats <- read.table("data/Wikipedia_LargestCats.txt", header = T, sep="\t")
big.cats
```

```
##      Rank      Common.name Scientific.name Weight.range.kg
## 1      1          Tiger    Panthera tigris      90-300
## 2      2          Lion     Panthera leo      160-270
## 3      3          Jaguar    Panthera onca      56-120
## 4      4          Cougar    Puma concolor     53-100
## 5      5          Leopard    Panthera pardus    17-90
## 6      6          Cheetah   Acinonyx jubatus    20-60
## 7      7      Snow leopard    Panthera uncia    22-55
## 8      8      Eurasian lynx      Lynx lynx     15-45
## 9      9 Sunda clouded leopard Neofelis diardi    12-26
## 10     10     Clouded leopard Neofelis nebulosa    11.5-23
##      Maximum.weight.kg Maximum.length.m Native.range.by.continent
## 1              388.78              4.17              Asia
## 2              375.00              3.64      Asia, Africa, Europe
## 3              160.00              2.60  North and South America
## 4              125.20              2.80  North and South America
## 5              96.50              2.75      Asia, Africa, Europe
## 6              72.00              2.10      Africa, Asia, Europe
## 7              75.00              2.50              Asia
## 8              38.00              1.50              Asia, Europe
## 9              27.00              1.30              Asia
## 10             23.00              1.08              Asia
```

This data.frame contains the weight range, and the maximum observed weights and lengths of the ten largest wildcats. (Source: [Wikipedia](#)).

When knitting this code, we are getting a bunch of messages related to the tidyverse package. Not so nice in a report. You can deactivate them by opening your chunk with `{r, warning=F, message=F}`.

Let's do some real R code. In the chunk below, we split the weight range field into min and max using some fancy `dplyr` code.

```
big.cats <- big.cats %>%
  separate(Weight.range.kg, into=c("Weight.min", "Weight.max"), sep = "-", remove = T) %>%
  mutate(Weight.min=as.numeric(Weight.min),
         Weight.max=as.numeric(Weight.max)) %>%
  mutate(Common.name=factor(Common.name, levels=big.cats$Common.name))
```

Maybe in your report you want to automatically include a value which you calculate in your r script. This can be done with some *inline code*. This can be done enclosing some code by backticks and specifying the code is in r. For instance the code:

The cat having the highest weight is ``r big.cats[1,'Common.name']``. It weights up to ``r big.cats[1,'Maximum.weight.kg']``.

Renders as:

The cat having the highest weight is Tiger.

It weights up to 388.78.

A couple of useful arguments when you start your chunk:

`{r echo=F}` — For running your code in the background, without showing the code itself.

`{r eval=F}` — Opposite. For showing your code, but without actually running it.

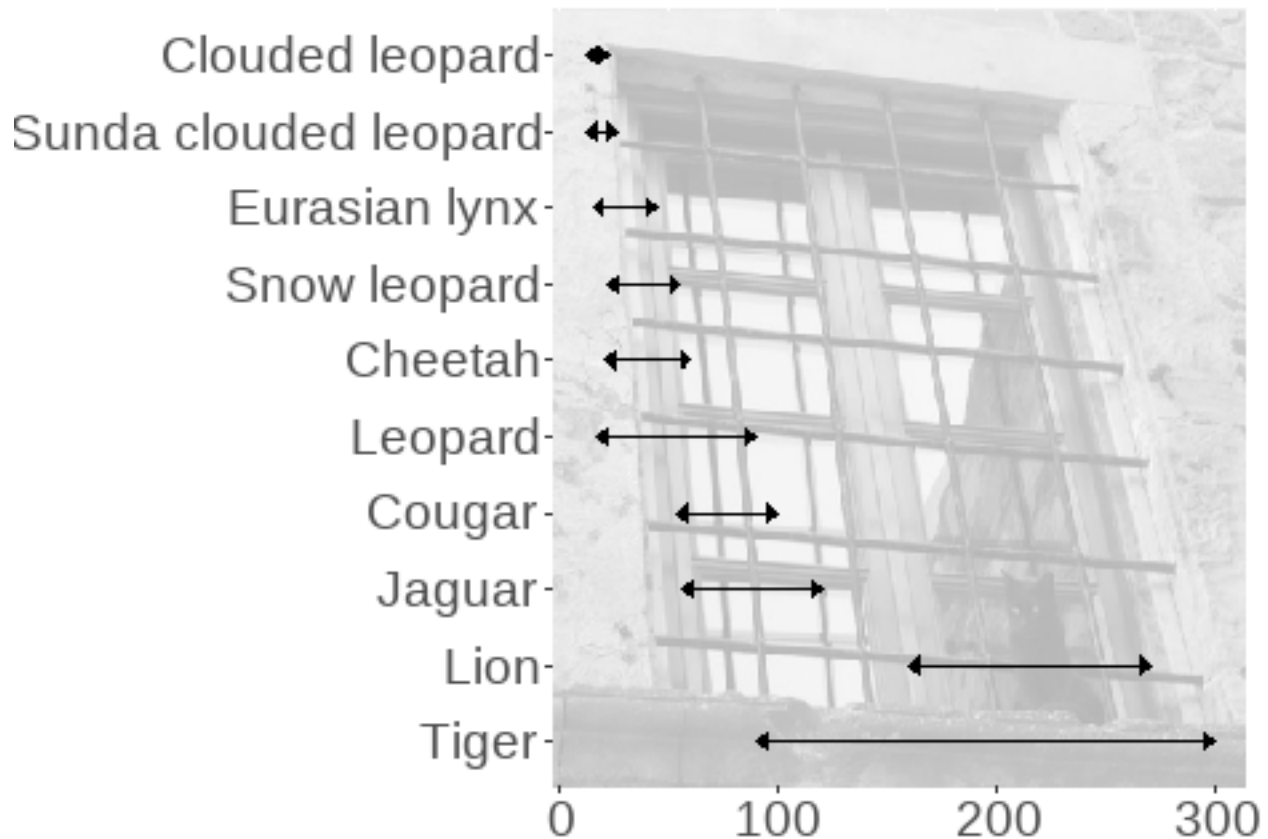
3.3.3 Playing around with Pictures

THE THING about RMarkdown is that it allows embedding graphs directly to your document. Input changed? Just re-knit and you'll have all your graphs updated. It's as easy as simply running a chunk of code.

Let's make a graph to see the differences in cat size more easily.

```
# create spider plot
#remotes::install_github("hilaryparker/cats") #if needed
library(cats)

ggcats <- ggplot(data=big.cats) +
  cats::add_cat() + ## add a random cat image on the background of the graph, if you fancy
  geom_segment(aes(y=Common.name, yend=Common.name, x=Weight.min, xend=Weight.max),
               arrow = arrow(length = unit(5, "points"),
                             ends="both", type = "closed", angle = 40)) +
  ylab(NULL) +
  xlab(NULL) +
  theme(axis.text = element_text(size=14))
ggcats
```



Some useful figure related arguments here:

`{r fig.height=3}` - picture height in inches

`{r fig.width=3}` - picture width in inches

`{r fig.cap="add caption here"}` - Add a caption

`{r fig.align="center"}` - Horizontal alignment of your graph

`{r dpi=150}` - Change resolution of output image (mostly relevant for pdf)

You can also combine them altogether in a single line. For instance, if I rerun the chunk above specifying:

`{r, fig.height=3, fig.width=4, fig.cap="Weight (kg) of the 10 largest wild cats", fig.align="center", dpi=150, echo=F}`

Note how the code isn't visible anymore, having set `echo=F`. I swear it's there, though.

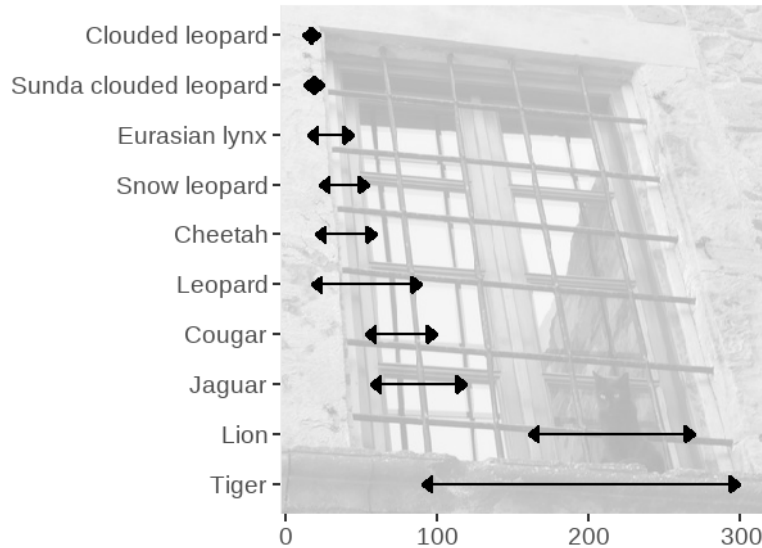


Figure 3: Weight (kg) of the 10 largest wild cats

3.3.4 Playing around with Tables

If you just print an R object on the console as we did above, **RMarkdown**, will show it in the rendered document as well. It won't look that good, though.

The entry level way of rendering tables is the `knitr::kable` function.

```
knitr::kable(head(big.cats[1:4,1:5]), caption="The largest cats!")
```

Table 1: The largest cats!

| Rank | Common.name | Scientific.name | Weight.min | Weight.max |
|------|-------------|-----------------|------------|------------|
| 1 | Tiger | Panthera tigris | 90 | 300 |
| 2 | Lion | Panthera leo | 160 | 270 |
| 3 | Jaguar | Panthera onca | 56 | 120 |
| 4 | Cougar | Puma concolor | 53 | 100 |

An even nicer way is using the `kableExtra` package. For instance, when rendering to html, `kableExtra` allows the creation of responsive tables. This is extremely useful for tables larger than a A4 format.

```
library(kableExtra)
knitr::kable(big.cats, caption="The largest cats!") %>%
  kable_styling(
    bootstrap_options = c("striped", "hover", "condensed", "responsive"),
    position = "center")
```

I won't look good that good on a pdf, but on an html report it's just as good as it can possibly be.

4 Slightly more advanced stuff

4.1 Long computing times

Whenever you knit your RMarkdown report, R will re-run all the code contained in your .Rmd script. It's therefore not the best idea to include a chunk of code taking 3 hours to run. If you do so, even just correcting a typo in the text will require you to wait three hours before the corrected version of your report is rendered!

There are a couple of workarounds, though.

- 1) add the argument `{r cache=T}` to your chunk:

Adding this argument to the slowest chunks of code will save the intermediate results of these chunks in a dedicated folder. RMarkdown will *only* rerun these chunks when changed. Otherwise, it will skip the chunk, and get directly the cached results.

- 2) `{r eval=F}` + save and load

Sometimes, it might be more convenient to run slow chunks of code in a dedicated, interactive session (maybe even on a different machine), save the results in a .RData file, and reload this saved data in your report. For transparency, you might still show the code you used to produce these intermediate outputs, but setting `eval=F`, you'll tell RMarkdown not to run this code.

Something like this.

Chunk 1 with slow code I ran elsewhere:

```
```{r eval=F}

output <- function(input) # A real SLOW chunk of code

save(output, filename='output.slow.RData')
```

...

Chunk 2 reimporting the output of chunk 1:

```
```{r}

load(filename='output.slow.RData')
```

...

4.2 Alternative ways of knitting

Clicking on the knit button is convenient. The keyboard shortcut `Ctrl`Alt`k` is even more.

Sometimes, you might want to knit multiple documents (Yes, you might loop across several .Rmd scripts, and knit hundreds of reports in parallel). To do so, you might want to knit an .Rmd file from console as:

```
knitr::knit("KateRMarkdown.Rmd")
```



4.3 PDFs and paper templates

I know. You you can't wait to produce your fantastic pdf reports, and write your next paper directly in R. Good news - is possible - There are many templates out there, and you just have to fetch them. See for instance: <https://t.co/uJBqWER5h6?amp=1>

Yes, there are templates which will also format bibliographic reference as requested by different journals

Bad news - you'll need to setup your machine first, and it might be tricky sometimes. You find some guidance at: <https://bookdown.org/yihui/rmarkdown-cookbook/install-latex.html>.

5 Resources

Many good resources out there. I only cite two:

- 1) Bookdown - <https://bookdown.org/yihui/rmarkdown/>
- 2) RMarkdown Cookbook - <https://bookdown.org/yihui/rmarkdown-cookbook/>

6 Your Turn!

Now it's up to you to create a beautiful **RMarkdown** report full of cats. The more \ the cuter the cats, the better.

Pick up a cute, wild species of cat and:

- 1) Set up a new **RMarkdown** project
- 2) Report a fun fact about your cat species as text in your report
- 3) Embed a (copyright free) picture of that cat from the internet
- 4) Download some data on that cat from [gbif](#) using the **rgbif** package (help code below)
- 5) Show the data in a table
- 6) Create one graph based on that data (for instance a bar chart of the species occurrences across countries or sampling years, or a map of the coordinates). Any graphs work
- 7) Knit your project to html
- 8) Upload your html report at: <https://portal.idiv.de/nextcloud/index.php/s/ZK7XWMoAdXFbffQ>. I will publish the nicest reports in the [KateRMarkdown](#) GitHub project.

Help code to download data from [gbif](#)

```
library(tidyverse)
library(rgbif)
myspecies <- "Caracal caracal" ## example
get.speciesKey <- function(x){name_backbone(x)$speciesKey} #get GBIF species key
key <- get.speciesKey(myspecies)
# extract the first n occurrences from rgbif
get.occurrences <- function(x, n=100){occ_search(taxonKey=x, return="data",
                                                  limit=n, hasCoordinate = T)}

# clean data
dat <- lapply(key, get.occurrences, n=100)[[1]]
dat <- dat$data %>%
  dplyr::select(species, year:day, country, stateProvince,
                decimalLongitude, decimalLatitude)#, everything())
```

7 sessionInfo()

```
sessionInfo()
```

```
## R version 4.0.1 (2020-06-06)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] kableExtra_1.3.4  cats_0.1      forcats_0.5.1  stringr_1.4.0
## [5] dplyr_1.0.4       purrr_0.3.4   readr_1.4.0    tidyr_1.1.2
## [9] tibble_3.0.6      ggplot2_3.3.3  tidyverse_1.3.0 hexSticker_0.4.9
## [13] cowplot_1.1.1
##
## loaded via a namespace (and not attached):
## [1] httr_1.4.2          viridisLite_0.3.0  ggimage_0.2.8
## [4] jsonlite_1.7.2      showtext_0.9-2     modelr_0.1.8
## [7] assertthat_0.2.1    BiocManager_1.30.10 rvcheck_0.1.8
## [10] highr_0.8           showtextdb_3.0     cellranger_1.1.0
## [13] yaml_2.2.1          pillar_1.5.0       backports_1.2.1
## [16] lattice_0.20-41     glue_1.4.2         digest_0.6.27
## [19] rvest_0.3.6         colorspace_2.0-0   htmltools_0.5.1.1
## [22] pkgconfig_2.0.3     broom_0.7.5        haven_2.3.1
## [25] magick_2.6.0        sysfonts_0.8.3     webshot_0.5.2
## [28] scales_1.1.1        svglite_2.0.0      ggplotify_0.0.5
## [31] jpeg_0.1-8.1        generics_0.1.0     farver_2.0.3
## [34] ellipsis_0.3.1      withr_2.4.1        hexbin_1.28.2
## [37] cli_2.3.0           magrittr_2.0.1     crayon_1.4.1
## [40] readxl_1.3.1        evaluate_0.14      fs_1.5.0
## [43] fansi_0.4.2         xml2_1.3.2         tools_4.0.1
## [46] hms_1.0.0           lifecycle_1.0.0    munsell_0.5.0
## [49] reprex_1.0.0        compiler_4.0.1     systemfonts_1.0.1
## [52] gridGraphics_0.5-1  rlang_0.4.10       grid_4.0.1
## [55] rstudioapi_0.13     labeling_0.4.2     rmarkdown_2.7
## [58] gtable_0.3.0        DBI_1.1.1          curl_4.3
## [61] R6_2.5.0            lubridate_1.7.9.2  knitr_1.31
## [64] utf8_1.1.4          stringi_1.5.3      Rcpp_1.0.6
## [67] vctrs_0.3.6         dbplyr_2.1.0       tidyselect_1.1.0
## [70] xfun_0.21
```