(R)Markdown

Wojciech Hardy; Michał Paliński

3/21/2021

# Cheatsheets are handy as always

[This one for example](https://raw.githubusercontent.com/rstudio/cheatsheets/master/rmarkdown-2.0.pdf)

[Or this one](https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf)

[This one for example](https://raw.githubusercontent.com/rstudio/cheatsheets/master/rmarkdown-2.0.pdf)

[Or this one](https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf)

# Basic formatting

Some basic text formatting includes \*Italics\* or \_Italics\_ (*Italics*) and \*\*Bold\*\* or \_\_Bold\_\_ (**Bold**) text.

`Space between lines to break a paragraph

Like so`

Space between lines to break a paragraph

Like so

Or end a line with a double space to break without starting a new paragraph

Or end a line with a double space to break  
without starting a new paragraph

Superscripts can be done like so: R^2^ R2

Subscripts can be done like so: H~2~O H2O

# Headers

# Header 1

# Header 1

## Header 2

## Header 2

### Header 3

### Header 3

#### Header 4

#### Header 4

##### Header 5

##### Header 5

# Lists

## Ordered

1. Item 1

2. Item 2

2. Item 3 # Note the error in numbering

1. Item 1
2. Item 2
3. Item 3 # It's fine here though

## Unordered

\* Item

\* Another item

* Item
* Another item

## Subitems

1. Item 1
   * Item 2
   * Item 3

# Tables

|  |  |  |
| --- | --- | --- |
| Day | Hour | Group |
| Wednesday | 9:45 | 1 |
| Thursday | 16:45 | 2 |
| Thursday | 18:30 | 3 |

# Quotes

> Hmmm

Hmmm

- Geralt of Rivia

# Using Html

You can also just use html to write stuff within the markdown document. Here’s something copied directly from YAML Wikipedia page source code:

YAML

Filename extensions

.yaml, .yml

Internet media type

Not registered

Initial release

11 May 2001; 19 years ago (2001-05-11)

Latest release

1.2 (Third Edition)(1 October 2009; 11 years ago (2009-10-01))

Type of format

Data interchange

Open format?

Yes

Website

yaml.org

# Equations

You can insert equations with the same syntax as in LaTeX. E.g. within a sentence $ \sum (x + 1) $ or as standalone with double $$ at start and finish

# Using R in RMarkdown

The main thing, however, is the ability to integrate R (or other languages):

```{r}  
cat(“This is a code chunk”)  
```

cat("This is a code chunk")

## This is a code chunk

summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00

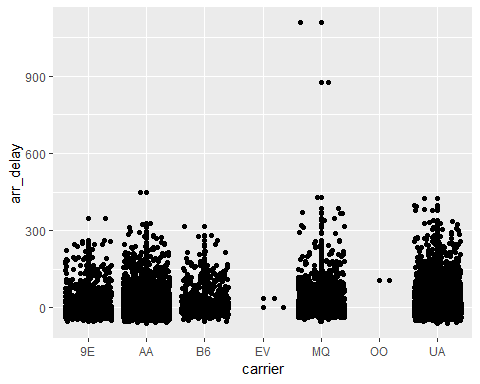
plot(pressure)



You can also use Python within the RMarkdown document like so:

import pandas  
flights = pandas.read\_csv("Data/flights.csv")  
flights = flights[flights['dest'] == "ORD"]  
flights = flights[['carrier', 'dep\_delay', 'arr\_delay']]  
flights = flights.dropna()

library(ggplot2)  
ggplot(py$flights, aes(carrier, arr\_delay)) + geom\_point() + geom\_jitter()



cat("Example from: https://rstudio.github.io/reticulate/articles/r\_markdown.html")

## Example from: https://rstudio.github.io/reticulate/articles/r\_markdown.html

And a nice table alternative -> kable from the knitr package.

knitr::kable(head(mtcars[, 1:4]), caption = "A kable table, ver 1")

A kable table, ver 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | mpg | cyl | disp | hp |
| Mazda RX4 | 21.0 | 6 | 160 | 110 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 |
| Datsun 710 | 22.8 | 4 | 108 | 93 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 |
| Valiant | 18.1 | 6 | 225 | 105 |

knitr::kable(head(mtcars[, 1:4]), "html", caption = "A kable table, ver 2")

A kable table, ver 2

mpg

cyl

disp

hp

Mazda RX4

21.0

6

160

110

Mazda RX4 Wag

21.0

6

160

110

Datsun 710

22.8

4

108

93

Hornet 4 Drive

21.4

6

258

110

Hornet Sportabout

18.7

8

360

175

Valiant

18.1

6

225

105

# Code chunks customisation

([Partially taken from here](https://bookdown.org/yihui/rmarkdown-cookbook/hide-one.html))

## Chunk naming (for reference)

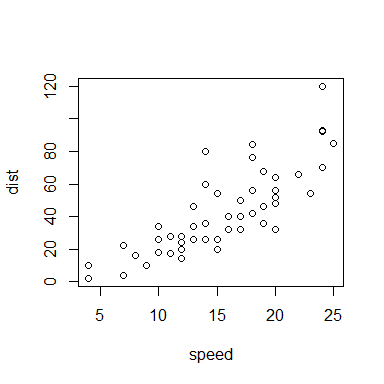
"You just put the name after the language declaration"

## [1] "You just put the name after the language declaration"

## Using variables as parameters

typical\_width <- 4  
typical\_height <- 4

plot(cars)



## Using variables within the text 'r variable'

We have previously set a typical width to 4 and the typical height to 4.

## Conditional execution with eval

(is.weekend comes from the chrono package)

cat("It's the weekend! :)")

cat("It's not the weekend! :(")

## It's not the weekend! :(

## Handling errors

543 + "clearly a text and not a number"

## Error in 543 + "clearly a text and not a number": non-numeric argument to binary operator

By default RMarkdown stops after encountering an error. We can tell it to continue.

## Caching results that take long to compute

Sys.sleep(10)  
a <- 4  
a

## [1] 4

The chunk gets reevaluated if anything changes within the chunk. **Make sure you know what you’re doing when caching**.

We can use cache.extra = to specify additional conditions for cache invalidation (i.e. to repeat the calculations), e.g.:

* file.mtime(‘filename’) # Modification time of the file changed
* tools::md5sum(‘filename’) # Content of the file changed
* getRversion() # R version changed
* etc.

Other stuff:

* cache.comments – if you don’t want to recalculate after changing a comment.
* cache.lazy – loading with lazyload() instead of load() (see [Lazy loading](https://en.wikipedia.org/wiki/Lazy_loading))
* cache.path – to specify where to save cached stuff
* cache.vars – cache specified objects
* dependson – reevaluate conditional on a change in a different chunk (or chunks)
* autodep – knitr will try to find the between-chunk dependencies on its own

## Keeping your report nice and clear

### Hide the source code with echo=FALSE

## [1] 2

### Hide messages (e.g. when loading stuff) with message=FALSE

message("You will not see the message.")

### Hide warnings with warning=FALSE

1:2 + 1:3

## [1] 2 4 4

### Hide plots with fig.show='hide'

plot(cars)

### Hide everything from the chunk with include=FALSE

### Hide the results with results='hide'

a\*typical\_height\*typical\_width

### Generate Markdown content with R code with results='asis'

for (i in 1:10) {  
 cat("- Item", i, "\n")  
}

* Item 1
* Item 2
* Item 3
* Item 4
* Item 5
* Item 6
* Item 7
* Item 8
* Item 9
* Item 10

### Cluster the results with results='hold'

Standard:

x <- 5  
y <- 6  
x + y

## [1] 11

y - x

## [1] 1

x \* y

## [1] 30

With results='hold':

x <- 5  
y <- 6  
x + y  
y - x  
x \* y

## [1] 11  
## [1] 1  
## [1] 30

### Compress the output with collapse=TRUE

Without

1 + 1

## [1] 2

1:10

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

With

1 + 1  
## [1] 2  
1:10  
## [1] 1 2 3 4 5 6 7 8 9 10

# EXERCISE 1

Pick a TV show that had its premieres on TV and thus has some viewership numbers reported on Wikipedia. E.g. [Suits](https://en.wikipedia.org/wiki/List_of_Suits_episodes) (see table just above the References)

Then create a short report (you can copy the content from Wikipedia or other pages for this task) that contains, for example (do a commit after each step!):

1. A brief description of the show (use *italics* for names).
2. A photo with the logo or a shot from the show itself.
3. A summary of some basic statistics (e.g. on viewership or ratings).
4. A graph of the viewership over time.
5. A graph of the episode-to-episode (or season-to-season) changes in viewership.
6. A short description of the observed changes that includes inline references to numbers (e.g. the viewership decreased by insert\_calculated\_number between seasons 3 and 5).
7. Make sure your report looks nice -> this time we’re mostly interested in the output and not necessarily the codes used to achieve it.
8. knitr your report and save it in the “RMarkdown folder” of your repo.
9. Commit the changes and push them to Github.

### Some other chunk options

E.g.:

* highlight – syntax coloring
* tidy– clean and format the code based on, e.g. formatR or styler, with specific options defined by tidy.opts

For chunks with figures, e.g.:

* fig.align – alignment
* fig.ext – image format
* dev – graphical device for the plot generation
* dev.args– arguments to be passed to device (e.g. for image customisation)
* dpi – DPI

You can, e.g., combine the above to produce images in DPI resolution and format appropriate for a publication.