# Exercise 2: Text as data

*Philipp Kling*

*16.06.2020*

## Contents



```
library(knitr)

### Global options
options(max.print="75")
opts_chunk$set(echo=FALSE,
                 cache=FALSE,
               prompt=FALSE,
               tidy=TRUE,
               comment=NA,
               message=FALSE,
               warning=FALSE)
opts_knit$set(width=75)
rm(list = ls())
```

## Improvement of a regular expression

In the exercise slides we extracted the first names of a string that was obtained from the UZH website. Remember?

```
text <- "Abou-Chadi Tarik AFL-H-359 +41 44 634 52 03Caramani Daniele AFL-H-344
+41 44 634 40 10Donnay Karsten AFL-H-350 +41 44 634 58 57"
unlist(stringr::str_extract_all(string = text, pattern = "[A-Z]{1}[a-z]+ [A-Z]{1}[a-z]+"))
```

```
[1] "Chadi Tarik"      "Caramani Daniele" "Donnay Karsten"
```

Upon inspection of the result, we see that we only extracted 'Chadi' despite the full name of Tarik being 'Abou-Chadi'. We can try to improve this regular expression. Visit https://regexr.com/ to test the regular expressions in real time.

**Solution:** (one of many)

```
unlist(stringr::str_extract_all(string = text, pattern = "([A-Z]{1}[a-z]+[-]{1})?[A-Z]{1}[a-z]+ [A-Z]{1
```

```
[1] "Abou-Chadi Tarik" "Caramani Daniele" "Donnay Karsten"
```

Explanation: we add the '([A-Z]{1}[a-z]+[-]{1})?' at the beginning. This matches everyhing that starts with one big letter, then at least one small letter and subsequently at least one 'minus' (-). We group this statement overall with the parentheses and then specify that this may or may not occur with the '?'.

## Analyses of Text data with regular expressions

Take the following text which is the body of an article retrieved from the Guardian. It still has some HTML code attached to it. Have a look at the text.

```
load("/home/philipp/Documents/fds-2020-exercise/data/ex2/articlebody.Rda")
# substr(articlebody, start=1, stop=1000)
articlebody
```

```
[1] "<p>Migrants in professional occupations will protest outside Downing Street on Tuesday to raise awa
```

In HTML paragraphs are separated by the opening tag ' ' and the closing tag ' '. Suppose you would like to know of how many paragraphs this article consists. The code to retrieve this information could look like this: everything that is a printable character and comes in between two paragraph tags. Note: "+?" is indicating a "lazy" search. If we would use "+" it would match everything from the first paragraph until the last paragraph. Using "+?" only matches to the first occurrence of an ending tag.

```
load("/home/philipp/Documents/fds-2020-exercise/data/ex2/articlebody.Rda")
parags <- unlist(stringr::str_extract_all(articlebody, pattern = "<p>([:print:])+?</p>"))
length(parags)
```

```
[1] 18
```

Suppose we also want to know how many links are included in the article body and how many of those links also redirect back to another Guardian article. We could, again, visit this page: https://regexr.com/ and copy the article body into the text field and see the results of your Regular expressions. However, there are small differences e.g. word and non-word characters on this page would be [\w\W] and in R you could write [:print:] instead.

Extract every pattern that is included in an "<a>" tag. These are the links.

```
hrefs <- unlist(stringr::str_extract_all(articlebody, pattern = "<a href=\\\"([:print:])+?\\\">"))
length(hrefs)
```

```
[1] 7
```

Use **grepl()** to get an indicator if a pattern is included in a text. Links that include "theguardian.com" are the ones that link to other articles.

```
sum(grepl("theguardian.com", hrefs))
```

```
[1] 2
```