Exercise 2: Text as Data

Karsten Donnay

15.06.2021

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 did a very simple extractions of the names in a string that we scraped from the IPZ staff directory.

```
text <- "Abou-Chadi Tarik AFL-H-359 +41 44 634 52 03Bornschier Simon AFL-H-307 \t+41 44 634 40 79Caramar
unlist(stringr::str_extract_all(string = text, pattern = "[A-Z]{1}[a-z]+ [A-Z]{1}[a-z]+"))
```

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

The result of our simple extraction is not really correct though because we only extracted 'Chadi' despite the full last name of Tarik being 'Abou-Chadi'. We next try to extend our regular expression to also include hyphenated last names. You can 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" "Bornschier Simon" "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 'hyphen' (-). 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 is still formated as HTML, i.e., there is not only text but also html tags etc. Let's first have a quick look at the text:

```
load("data/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 how many paragraphs this article has. The logic for the code to retrieve this information could look like this: a paragraph is 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 a closing tag, i.e., the first closing tag after the opening tag, which is what we want.

```
load("data/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 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, note that there are small syntax differences e.g. word and non-word characters on this page would be denoted as [\w\W] and in R you could simply write [:print:] instead.

In the first step, we extract all the links, i.e., we search for every pattern that is included in an "<a>" tag:

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

```
[1] 7
```

```
head(hrefs)
```

```
[1] "<a href=\"https://www.facebook.com/events/142212043115323/?ti=as\">"
[2] "<a href=\"https://www.gov.uk/settle-in-the-uk\">"
[3] "<a href=\"https://www.theguardian.com/uk-news/2013/nov/05/migration-target-useless-experts\">"
[4] "<a href=\"http://www.imperiumchambers.co.uk/paul-turner/\">"
[5] "<a href=\"http://www.imperiumchambers.co.uk/\">"
[6] "<a href=\"https://www.jandoerfel.com/\">"
```

In the second step, we use **grepl()** to check whether a given pattern is included in a text. This is the easiest way here to count those links that include "theguardian.com", i.e., those articles that link to other articles in The Guardian.

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

```
[1] 2
```