BAN432 fall 2019 - First assignment

Goal of this assignment

This assignment is designed so that you learn the nuts and bolts of `R` as used in this course. In the last lectures, we touched on almost all functions you have to use to solve this assignment. If you are unsure how to use a function, either use `R`'s own documentation (type `?function.name()`) or `www.stackoverflow.com`. We encourage you to code this assignment yourselves, and do not use purpose made solutions or packages as provided on the internet. All tasks in this assignment focus on regular expressions.

Please submit your assignment even though you were not able to find a solution to all tasks.

Hint: Before starting to code, try to separate the task into smaller pieces.

Learning outcomes

In Task 1 and 2 you will scrape data from the internet and extract information. Usually a text file scraped form the internet contains a lot of noise that would disturb the analysis. The ability to isolate relevant information in text files is an essential skill in textual data analysis.

Task 3 is another exercise on regular expressions. In addition, this task is an exercise on how to write a function and how to test logical conditions. Those are important skills for the more advanced analyses later on in the course.

Formalities

This assignment will be handed out on 3rd of September, 2019 at 16:00 and has to be submitted no later than the 10th of September, 2019 at 14:00. Please comment your code shortly so that the grader can reconstruct your thinking. You do not need to explain the used functions.

Please work together in groups of four and submit the assignment via Canvas. There is a page on Canvas where groups can register the names of the members.

Task 1: Web scraping I

Download the Wikipedia page about the Enron scandal using the base-R function `readLines()`. The sections "References" and "Further reading" at the bottom of the page contain a short list of literature that was mentioned in the article or that might be interesting for further reading. Write a regular expression that matches all lines in those two sections ("References" and "Further reading") that contain links to external sources where the documents can be found. Generate an output that lists these external links. Discard the references that do not contain external links to a book or an article.

Hint for the regular expression: look at the source code of the Wiki article in your browser (Firefox/MS Edge: CTRL+U, Safari: Cmd+Opt+U). What do the lines you want to extract have in common?

Your result should look like this:

```
## [1] "https://archive.org/details/smartestguysin00mcle"
## [2] "https://www.webcitation.org/5tZ0yCA9i?url=http://www.ruf.rice.edu/~bala/files/dharan-bufkins_en
## [3] "https://archive.org/details/pipedreamsgreede00bryc_0"
```

```
## [4] "https://archive.org/details/conspiracyoffool00kurt"
## [5] "https://archive.org/details/whatwentwrongate00pete"
## [6] "https://archive.org/details/powerfailure00mimi"
```

```r
# Solution:
# Download the html code
source.code <- readLines("https://en.wikipedia.org/wiki/Enron_scandal")

# Some references do not include any links to the articles. Find the ones that
# do (i.e. the ones that point to an "external text". This is one suggestion for
# a regex that matches those lines. Others work as well.
idx <- grep(".*?<li><cite class=\"citation book\".*?class=\"external text\" href=\"(.+?)\">.+", source.code)

# Remove all html code before and after the actual link. Here we can use
# "backreferencing".
gsub(".*?class=\"external text\" href=\"(.+?)\">.+", "\\1", source.code[idx])
```

Task 2: Web scraping II

For this task you use the whole Wikipedia article on the Enron scandal. Generate a vector that contains all words in the whole Enron article that link to other Wiki articles (the "blue" words). Do not consider words that link to other places in the Enron article, such as the phrase "Rise of Enron". For your solution, use the base-R functions `gregexpr` and `regmatches`.

Bonus: For the output, report how often a word (or phrase) occurs as a link in the document. Output the 25 most frequently linked words (phrases). The function `table()` helps you generating a frequency list:

```
                                      . Freq
1                                  ISBN   15
2                                  SSRN    7
3                   The Washington Post    6
4                                Forbes    5
5                             USA Today    5
6                                Dynegy    4
7    Enron: The Smartest Guys in the Room    4
8                           Kenneth Lay    4
9                     Magnolia Pictures    4
10                        Andrew Fastow    3
11                      Arthur Andersen    3
12                      Associated Press    3
13                         CNNMoney.com    3
14                Connecticut Law Review    3
15                Enron Energy Services    3
16                              Houston    3
17                    Houston Chronicle    3
18                  Houston Natural Gas    3
19                     Jeffrey Skilling    3
20                      McLean, Bethany    3
21                        Merrill Lynch    3
22                  The New York Times    3
23                                Time    3
24 U.S. Securities and Exchange Commission    3
25                             WorldCom    3
```

```r
# Solution:
# Comment: Depending on when you solved this task, your output might differ
# slightly. The Wiki article was edited and a lot of hidden links were placed in the
# html-code with the effect that you get many instances of the word "help".

# Option 1 (with dplyr and pipe operator)
require(dplyr)

m <- gregexpr("<a href=\"/wiki/.+?\" title=\".+?\">.+?</a>", source.code)
regmatches(source.code, m) %>%
  unlist() -> idx.wiki

result <- gsub("(<.*?>|<\\/. ?>)", "", idx.wiki) %>%
  table() %>%
  sort(decreasing = T)
result[1:25]

# Option 2 (traditional way)
m <- gregexpr("<a href=\"/wiki/.+?\" title=\".+?\">.+?</a>", source.code)
r <- regmatches(source.code, m)
idx.wiki <- unlist(r)

result <- gsub("(<.*?>|<\\/. ?>)", "", idx.wiki) # remove html-code
result.t <- table(result) # generate a frequency list of words (phrases)
result.s <- sort(result.r, decreasing = T) # sort, starting with the most frequent item
result.s[1:25]
```

Task 3: Password security

For this task, please only use base-R functions.

Write a function that reads the passwords in the provided file `passwords.txt` using `readLines()` and checks their security level based on the number of requirements that are met.

The requirements are:

- all passwords have to fulfill these two mandatory requirements:
  - at least 8 characters long
  - does not contain any space characters
- in addition, all passwords have to fulfill at least two of the following requirements:
  - contains at least one upper case letter
  - contains at least one lower case letter
  - contains at least one digit
  - contains at least one punctuation character

After reading the passwords, have your code to report their security level:

- level 0: the two mandatory requirements are not fulfilled
- level 1: the two mandatory requirements are fulfilled and in addition two of the optional requirements are fulfilled
- level 2: the two mandatory requirements are fulfilled and in addition three of the optional requirements are fulfilled

- level 3: the two mandatory requirements are fulfilled and in addition four of the optional requirements are fulfilled

Your output should look like in this sample:

```
 [1] "L58jkdjP!: security level 3"    "P@ssw0rd: security level 3"
 [3] "!QAZ2wsx: security level 3"     "1qaz!QAZ: security level 3"
 [5] "Password123: security level 2"  "55BGates: security level 2"
 [7] "Aa123456: security level 2"     "Qwerty123: security level 2"
 [9] "letmein1: security level 1"     "password123: security level 1"
[11] "Password: security level 1"     "1q2w3e4r5t6y: security level 1"
[13] "superman: security level 0"     "123123: security level 0"
[15] "iloveyou: security level 0"     "star wars: security level 0"
```

```r
# Solution:
# Comment: The list contains two passwords ("superman" and "iloveyou") that fulfill
# the mandatory requirements (>=8 characters and no space). At the same time they
# do not fulfill the requirements for security level 1 (2 additional requirements). So
# these passwords fall between two categories. The approach chosen in the sample
# solution is to keep them on level 0. But one could argue for having them on level 1
# as well.

pw <- readLines("passwords.txt")

check.pw <- function(pw){
  # Set level to -1. Two fulfilled requirements will give level 1 etc.
  level <- -1

  # Check the optional requirements
  if(grepl("[[:upper:]]", pw)){
    level <- level + 1
  }

  if(grepl("[[:lower:]]", pw)){
    level <- level + 1
  }

  if(grepl("[[:digit:]]", pw)){
    level <- level + 1
  }

  if(grepl("[[:punct:]]", pw)){
    level <- level + 1
  }

  # Check the mandatory requirements
  if(grepl("[[:space:]]", pw) || nchar(pw) <= 7){
    level <- 0
  }

  output <- paste0(pw, ": security level ", level)
  return(output)
}

# Apply the function we created to the vector that contains the passwords
# from passwords.txt
security.labels <- lapply(pw, check.pw)
unlist(security.labels)
```