# Exercises - Class 5: Data manipulation

Koen Plevoets

24-10-2023

For the exercises on *Data wrangling in R* you are going to work with the file `baskervilles_TAG.txt` which can be found on the Github repository `https://github.com/kplevoet/texts/tree/main/doyle`. This file contains the Sherlock Holmes novel *The Hound of the Baskervilles* (written by Arthur Conan Doyle) in the following structure:

- `doc_id`: A unique identifier of the text (hence, data sets for different texts could be combined)
- `token`: The actual token as it appears in the text
- `lemma`: The "dictionary form" of the token (i.e. the word without its inflectional variants)
- `upos`: The "universal part-of-speech tag" or word category of the token. The categories are based on the CoNNL-U standard in Natural Language Processing.

### Goal of the exercises: measure computation times

Each of the following exercises will ask you to compute essentially the same result in three different ways: base R, **tidyverse** or `data.table`s. The idea is that you compare the performance of the three computations by *timing* them. You can use whatever means of measuring computation time but the solutions will apply the function `system.time()`.

### 1.

Read the file `baskervilles_TAG.txt` from the Github repository `https://github.com/kplevoet/texts/tree/main/doyle`. Note that the URL on which Github *shows* a file is not the URL from which you can *download* it. Downloading files from Github always involves the use of `https://raw.githubusercontent.com/` and the actual URL can be found by clicking on the `Raw` button or the `View raw` link.

Store this dataset in three objects:

- A data frame, using one of R's core functions
- A tibble, using one of **tidyverse**'s functions
- A data table, using the `fread()` function of the **data.table** package

After timing each function you should also compare the byte sizes of the three resulting objects (using `object.size()`).

### 2.

The `upos` column indicates punctuation sign by the label `PUNCT`. Create subsets by removing these observations, using:

- the `subset()` function on the data frame

- a **tidyverse** function on the tibble (either as part of a pipe or not)
- an index on the data table

### 3.

Create new columns `token_low` and `lemma_low` by converting the (respective) columns `token` and `lemma` to lowercase. Do this using:

- the function `within()` on the data frame
- a **tidyverse** function on the tibble (either as part of a pipe or not)
- the proper operators in the data table

### 4.

Compute the frequencies of the values in the column `lemma_low` and store them in an object with column names `lemma_low` and `Freq`. Do this using:

- the functions `with()` and `table()` on the data frame
- a **tidyverse** function or two on the tibble (either as part of a pipe or not)
- the proper operators in the data table

### 5.

Sort the objects of the previous exercise on the frequencies from largest to smallest. Do this using:

- the function `with()` on the data frame
- a **tidyverse** function or two on the tibble (either as part of a pipe or not)
- the proper syntax in the data table

### 6.

One of the basic distinctions in language studies is between content words and function words. The following data frame relates this distinction to a word's `upos`:

```
upos_type <- data.frame(upos = c("NOUN", "PROPN", "ADJ", "VERB", "ADV", "NUM",
                                 "DET", "PRON", "ADP", "CCONJ", "SCONJ", "PART",
                                 "AUX", "INTJ"),
                        type = c(rep("Content", 6), rep("Function", 8)))
upos_type
```

```
##     upos     type
## 1   NOUN  Content
## 2  PROPN  Content
## 3    ADJ  Content
## 4   VERB  Content
## 5    ADV  Content
## 6    NUM  Content
## 7    DET Function
## 8   PRON Function
## 9    ADP Function
## 10 CCONJ Function
## 11 SCONJ Function
## 12  PART Function
## 13   AUX Function
## 14  INTJ Function
```

Merge/Join this data frame (which you can first convert to a tibble or data table) to the objects of exercise 3 using:

- the function `merge()` on the data frame
- one of the `join()` functions from **tidyverse** on the tibble
- the proper syntax in the data table

Make sure that you retain all the rows of the frequency objects.

### 7.

Add a column `Perc` to the frequency objects of exercise 5 containing the percentages of the counts in the `Freq` column (i.e. the proportion times 100). Reshape these objects into long format whereby the values in the `Freq` and `Perc` columns appear underneath each other for each `lemma_low`. Do this using:

- the `reshape()` function on the data frame
- the proper `pivot_` function from **tidyverse** on the tibble
- either the function `dcast()` or `melt()` on the data table

In this last exercise you should only time the reshaping functions.