Exercises with *Migrant stories* dataset.................................................................................................................. 2
  Exercise 1.1 – Loading the data set.......................................................................................................................... 2
  Exercise 1.2 – Create a unigram frequency dictionary of women who left Afghanistan......................................... 2
  Exercise 1.3 – Create a bigram frequency dictionary of women who left Afghanistan......................................... 2
  Exercise 1.4 – Visualize both frequency dictionaries using barplots................................................................. 3
Exercises with *Andersen* dataset.......................................................................................................................... 5
  Exercise 2.1 – Loading the data set.......................................................................................................................... 5
  Exercise 2.2 – Process "Aunty" tale using UDPipe.................................................................................................... 6
  Exercise 2.3 – Create a lemma frequency dictionary of “Aunty”......................................................................... 7
  Exercise 2.4 – Compare the lengths of the language versions of the books.......................................................... 7
Exercises with *Migrant stories* dataset

**Exercise 1.1 – Loading the data set**

In RStudio create a blank R script

- Move to Files&Plots desktop
- In Files manager
  - move to the home directory
  - create a new directory New folder > 3
  - move to the directory 3 and create a new directory New folder > migrants
  - move to the directory 3/migrants
  - use More in the menu to run Set as working directory
  - use New Blank File in the menu to create a blank R Script and name it migrants.t.R.

Then the script is open in the Code editor window (upper-left window) and you can add the commands listed below to the script.

We suppose using these packages

```r
library(tidyverse)
library(tidytext)
library(gridExtra)
```

Load the *Migrant stories* dataset into your R environment.

```r
dataset <- read_tsv("dataset <- read_tsv("../../1/migrants/migrants.tsv")
names(dataset)  # attributes
```

**Exercise 1.2 – Create a unigram frequency dictionary of women who left Afghanistan**

We will prioritize words that appear in the given stories more than 5 times to ensure a barplot is clear.

```r
uni <- dataset %>%
  filter(gender == "female") %>%  # females only
  filter(country_or == "Afghanistan") %>%  # Afghanistan as an origin country
  unnest_tokens(word, story) %>%  # tokenize stories, i.e. extract unigrams
  anti_join(stop_words) %>%  # exclude stop words
  count(word, sort=TRUE) %>%  # count word frequencies
  filter(n > 5) %>%  # filter out the words used more than 5 times
  mutate(word = reorder(word, n))  # sort by n
```

**Exercise 1.3 – Create a bigram frequency dictionary of women who left Afghanistan**

For extracting bigrams we use the `unnest_tokens` function, which we used for tokenization. The difference is that the tokens will not be individual words but n-grams (`token = "ngrams"`), specifically bigrams (`n=2`).

```r
bi.1 <- dataset %>%
  filter(gender == "female") %>%  # females only
```
Bigrams of stop words are unimportant for understanding the content of the story. Therefore, we filter them out. We will prioritize bigrams that appear in the given stories more than once to ensure a barplot is clear.

```r
bi <- bi.1 %>%
  separate(bigram, into = c("w1", "w2"), sep = " ") %>% # split bigram into w1 and w2
  filter(!w1 %in% stop_words$word, !w2 %in% stop_words$word) %>% # filter out bigrams with stop words
  unite(bigram, c(w1, w2), sep = " ") %>% # join w1 and w2 into bigram
  count(bigram, sort = TRUE) %>% # count bigram frequencies
  filter(n > 1) %>% # filter out the bigrams used more than once
  mutate(bigram = reorder(bigram, n)) %>% # sort by n
```

**Exercise 1.4 – Visualize both frequency dictionaries using barplots**

Draw barplots showing unigram and bigram frequencies and make them side by side (see `grid.arrange` from `gridExtra` package).

```r
plot.uni <- ggplot(uni, aes(word, n)) + # create a chart showing unigram frequencies
  geom_col() +
  xlab("Unigram") + ylab("Frequency 5+") + coord_flip() +
  ggtitle("1-gram freq dictionary of women who left Afghanistan")

plot.bi <- ggplot(bi, aes(bigram, n)) + # create a chart showing bigram frequencies
  geom_col() +
  xlab("Bigram") + ylab("Frequency 2+") + coord_flip() +
  ggtitle("2-gram freq dictionary of women who left Afghanistan")

grid.arrange(plot.uni, plot.bi, ncol = 2) # draw the charts side by side
```
Exercises with *Andersen* dataset

**Exercise 2.1 – Loading the data set**

In RStudio create a blank R script
- Move to Files & Plots desktop
- In Files manager
  - move to the directory 3 and create a new directory New folder > *andersen*
  - move to the directory 3/andersen
  - use More in the menu to run Set as working directory
  - use New Blank File in the menu to create a blank R Script and name it *andersen.t.R*. Then the script is open in the Code editor window (upper-left window) and you can add the commands listed below to the script.

We suppose using these packages

```r
library(tidyverse)
library(tidytext)
```

Fairy tales written by Hans Christian Andersen are available in the *hcandersen* package.

```r
library(hcandersen)
```

It is a multilingual dataset. The books are formatted to be convenient for text analysis. The UTF-8 plain texts were sourced from [http://www.andersenstories.com/](http://www.andersenstories.com/) and are divided into elements of up to about 80 characters each (see the text attribute below).

For each fairy tale, we want to determine the languages in which it is available. *hca_fairytales* is a table with three attributes: text, book and language. We will focus on book and text only. We will remove duplicate rows and create a new binary attribute present, which will be set to 1 for each book-language pair. Then we will transform this table so that the number of rows corresponds to the number of books in the dataset, and the number of columns will be six: book (title), Danish, English, French, German and Spanish. The values in the cells will be either 0 or 1, depending on whether the respective book is available in the given language, ensuring that each row corresponds to a single book.

```r
book.version <- hca_fairytales() %>%
  select(book, language) %>%
  unique() %>%
  mutate(present = 1) %>%
  spread(language, present, fill = 0)

book.version

sum(book.version$Danish)
sum(book.version$English)
```
Exercise 2.2 – Process "Aunty" tale using UDPipe

Read the tale in English.

dataset <- hcandersen_en %>%
  filter(book == "Aunty")

UDPipe is a tool that provides tokenization, parts of speech tagging, lemmatization and syntactic parsing of raw text. It exists as a third-party R CRAN package udpipe.

library(udpipe)

Create a data frame because the udpipe function operates on a dataframe with columns doc_id and text. Therefore we are naming the attributes accordingly.

book.text <- data.frame(doc_id = dataset$book, text = dataset$text)

Read the English UDPipe model

ud_model <- udpipe_download_model(language = "english")
ud_model <- udpipe_load_model(ud_model$file_model)

Run UDPipe on book.text

book.text.ud <- udpipe(book.text, ud_model)
names(book.text.ud)

Exercise 2.3 – Create a lemma frequency dictionary of “Aunty”

```r
docs.text.ud %>%
  filter(!(lemma %in% stop_words$word)) %>%
  count(lemma, sort=TRUE) %>%
  head(n=10)  # display 10 most frequent lemmas
```

Exercise 2.4 – Compare the lengths of the language versions of the books

We will focus on books that are available in the dataset in all five languages. First, we extract a list of such books. Subsequently, we process each of them using UDPipe and count the number of tokens.

```r
book.lang.all <- book.version %>%
  filter (Danish == 1 & English == 1 & French == 1 & German == 1 & Spanish == 1) %>%
  pull(book)

Extract the books and process them using UDPipe: we have a list of English book names (book.lang.all), but the hcandersen_[da|es|fr|de] collections contain book names in their respective languages. Therefore, we will use metadata (EK) to match English names with names in other languages (see left_join below). The udpipe function operates on a dataframe with columns doc_id and text. so in each collection, we are renaming the attributes accordingly (see set_names below).

# English
en.text.ud <- hcandersen_en %>%
  filter (book %in% book.lang.all) %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)

# Danish
ud_model <- udpipe_download_model(language = "danish")
ud_model <- udpipe_load_model(ud_model$file_model)
da.text.ud <- hcandersen_da %>%
  left_join(EK, by = c("book" = "name_da")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en) %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)

# French
ud_model <- udpipe_download_model(language = "french")
ud_model <- udpipe_load_model(ud_model$file_model)
fr.text.ud <- hcandersen_fr %>%
  left_join(EK, by = c("book" = "name_fr")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en) %>%
  set_names(c("text","doc_id")) %>%
  udpipe(ud_model)
```

7
# Spanish
ud_model <- udpipe_download_model(language = "spanish")
ud_model <- udpipe_load_model(ud_model$file_model)
es.text.ud <- hcandersen_es %>% # it contains Spanish names
  left_join(EK, by = c("book" = "name_es")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en) %>%
  set_names(c("text", "doc_id")) %>%
  udpipe(ud_model)

# German
ud_model <- udpipe_download_model(language = "german")
ud_model <- udpipe_load_model(ud_model$file_model)
de.text.ud <- hcandersen_de %>% # it contains German names
  left_join(EK, by = c("book" = "name_de")) %>%
  filter(name_en %in% book.lang.all) %>%
  select(text, name_en) %>%
  set_names(c("text", "doc_id")) %>%
  udpipe(ud_model)

Get the length of the books where the length is the number of tokens.

en.c <- en.text.ud %>%
  count(doc_id)
da.c <- da.text.ud %>%
  count(doc_id)
fr.c <- fr.text.ud %>%
  count(doc_id)
es.c <- es.text.ud %>%
  count(doc_id)
dc.c <- de.text.ud %>%
  count(doc_id)
en.c %>%
  left_join(da.c, by = "doc_id") %>%
  left_join(fr.c, by = "doc_id") %>%
  left_join(es.c, by = "doc_id") %>%
  left_join(dc.c, by = "doc_id") %>%
  set_names(c("name", "English", "Danish", "French", "Spanish", "German"))
<table>
<thead>
<tr>
<th>Name</th>
<th>English</th>
<th>Danish</th>
<th>French</th>
<th>Spanish</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Something&quot;</td>
<td>3327</td>
<td>2679</td>
<td>3166</td>
<td>2942</td>
<td>3027</td>
</tr>
<tr>
<td>A cheerful temper</td>
<td>1877</td>
<td>1662</td>
<td>2000</td>
<td>1897</td>
<td>1731</td>
</tr>
<tr>
<td>A rose from Homer's grave</td>
<td>575</td>
<td>491</td>
<td>562</td>
<td>580</td>
<td>541</td>
</tr>
<tr>
<td>Clumsy Hans</td>
<td>1818</td>
<td>1557</td>
<td>1625</td>
<td>1790</td>
<td>1847</td>
</tr>
<tr>
<td>Everything in its proper place</td>
<td>3827</td>
<td>3337</td>
<td>3050</td>
<td>3774</td>
<td>3551</td>
</tr>
<tr>
<td>Five peas from a pod</td>
<td>1483</td>
<td>1330</td>
<td>1389</td>
<td>1446</td>
<td>1431</td>
</tr>
<tr>
<td>Little Claus and big Claus</td>
<td>5537</td>
<td>5206</td>
<td>4467</td>
<td>5020</td>
<td>5415</td>
</tr>
<tr>
<td>Little Ida's flowers</td>
<td>3548</td>
<td>3320</td>
<td>3093</td>
<td>3297</td>
<td>3449</td>
</tr>
<tr>
<td>Ole-Luk-Ole, the Dream-God</td>
<td>4970</td>
<td>4391</td>
<td>4377</td>
<td>4589</td>
<td>4712</td>
</tr>
<tr>
<td>Soup from a sausage skewer</td>
<td>6853</td>
<td>5295</td>
<td>6314</td>
<td>6227</td>
<td>5741</td>
</tr>
<tr>
<td>Sunshine stories</td>
<td>1460</td>
<td>1199</td>
<td>1261</td>
<td>1262</td>
<td>1236</td>
</tr>
<tr>
<td>The Nightingale</td>
<td>4348</td>
<td>4011</td>
<td>4477</td>
<td>4324</td>
<td>4128</td>
</tr>
<tr>
<td>The angel</td>
<td>1147</td>
<td>1014</td>
<td>1139</td>
<td>1079</td>
<td>1006</td>
</tr>
<tr>
<td>The bell</td>
<td>2275</td>
<td>2071</td>
<td>2859</td>
<td>2255</td>
<td>2173</td>
</tr>
<tr>
<td>The bottle neck</td>
<td>5079</td>
<td>4032</td>
<td>4525</td>
<td>4756</td>
<td>4416</td>
</tr>
<tr>
<td>The brave tin soldier</td>
<td>2035</td>
<td>1753</td>
<td>1943</td>
<td>1870</td>
<td>1875</td>
</tr>
<tr>
<td>The butterfly</td>
<td>1195</td>
<td>947</td>
<td>1064</td>
<td>1038</td>
<td>1068</td>
</tr>
<tr>
<td>The daisy</td>
<td>1781</td>
<td>1689</td>
<td>1648</td>
<td>1706</td>
<td>1775</td>
</tr>
<tr>
<td>The darning-needle</td>
<td>1554</td>
<td>1340</td>
<td>1548</td>
<td>1497</td>
<td>1390</td>
</tr>
<tr>
<td>The elderbush</td>
<td>3762</td>
<td>3301</td>
<td>3513</td>
<td>3422</td>
<td>3381</td>
</tr>
<tr>
<td>The emperor's new suit</td>
<td>1918</td>
<td>1817</td>
<td>1976</td>
<td>1883</td>
<td>1904</td>
</tr>
<tr>
<td>The farm-yard cock and the weather-cock</td>
<td>1067</td>
<td>886</td>
<td>1185</td>
<td>1043</td>
<td>1093</td>
</tr>
<tr>
<td>The fir tree</td>
<td>4053</td>
<td>3621</td>
<td>3331</td>
<td>4052</td>
<td>3767</td>
</tr>
<tr>
<td>The flax</td>
<td>2089</td>
<td>1734</td>
<td>1778</td>
<td>1840</td>
<td>1857</td>
</tr>
<tr>
<td>The flying trunk</td>
<td>2699</td>
<td>2333</td>
<td>1906</td>
<td>2547</td>
<td>2437</td>
</tr>
<tr>
<td>The garden of paradise</td>
<td>6604</td>
<td>5804</td>
<td>6261</td>
<td>6314</td>
<td>6233</td>
</tr>
<tr>
<td>The gardener and the noble family</td>
<td>2927</td>
<td>2567</td>
<td>2321</td>
<td>2871</td>
<td>2853</td>
</tr>
<tr>
<td>The happy family</td>
<td>1600</td>
<td>1347</td>
<td>1371</td>
<td>1476</td>
<td>1443</td>
</tr>
<tr>
<td>The jumper</td>
<td>793</td>
<td>700</td>
<td>799</td>
<td>771</td>
<td>767</td>
</tr>
<tr>
<td>The last dream of the old oak</td>
<td>2688</td>
<td>2205</td>
<td>2679</td>
<td>2589</td>
<td>2403</td>
</tr>
<tr>
<td>The little match-seller</td>
<td>1169</td>
<td>1019</td>
<td>1060</td>
<td>1089</td>
<td>1085</td>
</tr>
<tr>
<td>The little mermaid</td>
<td>10596</td>
<td>9369</td>
<td>9655</td>
<td>10287</td>
<td>9989</td>
</tr>
<tr>
<td>Poetry of old age</td>
<td>13140</td>
<td>11224</td>
<td>11094</td>
<td>11312</td>
<td>12355</td>
</tr>
</tbody>
</table>