# Text Analysis with R

24 March 2022

Dani Madrid-Morales

**Installing packages, and setting the enrvironment up**

The goal of this lab is to learn how to load different types of texts to `quanteda`, transform them into a corpus, clean them, re-shape them, describe them, and create simple visualizations.

For this lab session, we will use a dataset that includes a random sample of news stories published by two English-language news organizations from North Korea: the Korean Central News Agency (KCNA) and the Pyongyang Times (PT). The goal for today's lab is to answer two research questions:
1. What are the most frequently used words in news stories by KCNA and PT between 1997 and 2014?
2. Are there differences in the words used by KCNA and PT in news stories mentioning Russia and Japan?

**Part 1 - Importing some documents and creating a corpus with some additional metadata**

There are commonly seven steps in any computational text analysis project:
1. Selecting texts and defining a corpus. [part 1]
2. Converting the texts into a common format. [part 1]
3. Deciding the documentary unit. [part 1]
4. Defining and refining the features. [part 2]
5. Converting features to a quantitative matrix. [part 3]
6. Extracting information from the matrix statistically. [part 4]
7. Summarizing & interpreting the results. [part 4]

My goal with this lab is to cover the whole process in four parts, so that you get a sense of how *easy* it is to do basic descriptive text analysis with *quanteda*, but also to highlight that many of the steps involve quite a bit of human involvement and, therefore, often require us to go through different rounds of trial and error.

**Steps 1 & 2: selecting texts, defining a corpus and converting them to a common format** Using the `readtext` package is probably the most convenient way to import texts into R to create a corpus. It was developed by the same team that developed `quanteda`, and therefore the package is able to handle multiple situations, such as reading text stored in rows in Excel and csv format, or to read files such as docx and pdf (as long as the pdf file has ben OCRed first).

I'd like to show you two common approaches in importing data:
1. Reading a CSV file that has one column that stores each text in one row, and has additional columns with "metadata" (i.e. additional information about each file, such as the author, the date...).
2. Reading text from a folder with lots of files, where each file has one text, and the filename includes the mmetadata.

You can read how to handle other situations in the documentation of the `readtext` package.

Let's start with the simpler approach: importing data from a csv file. `readtext` has a single function, `readtext` that requires us to specify the name of the column in the csv file that has the text data. In our case, it is the `TX`column.

The resulting object is a `readtext` object with four variables: `text`, a new variable called `docid` which will be used by `quanteda` to identify each text in our dataset, and the two additional variables (metadata) that were included in the original final. In `quanteda` speak, we will call metadata (the additional information about each text) *docvars*. Docvars are very useful, as they allows us to separate the corpus (or group it) according to some theoretically-driven characteristics of our data.

Data from The Pyongyang Times (PT) is stored in a folder called "PT". Each article is saved as a txt file, and the filename includes the docvars we neeed (date and source). This is a scenario that `readtext` can handle rather easily: it reads each file in the folder and identifies the variables from the file name automatically "2005-08-29_PT_143.txt". All we need to specificy are the names of each variable.

```
df_pt <- readtext(file = "data/PT/", # We pass the name of the folder that has the files
                  docvarsfrom = "filename", # Refer to th filename to find the docvars
                  docvarnames = c("DE", "SC", "NU")) # Provide a vector with the docvars names

# We don't need the "NU" docvar (that's just a file index), so we drop it
df_pt$NU <- NULL
```

Document variables are really important as they help us make comparisons betweeen common elements in a corpus. For example, in RQ2 for this lab, we want to know whether there are differences in word frequencies in stories that mention Russia and stories that mention Japan. We could create a new docvar that indicates whether an article mentions either of the two countries (or both). This can be easily done with functions from the `stringr` package.

We can search for a keyword (say, "Japan" or "Russia") in each text. Whenever we find a match, we can mark that row as mentioning either of the two countries (as either `TRUE` or `FALSE`). We can store this information in a vector for each country, and then we can save those as new columns in our `df`. This information can then be used in `quanteda` as docvar to compare articles mentioning one country or the other.

```
#install.packages("stringr")
library(stringr)

# We merge both datasets into a single data frame
df <- rbind(df_kcna, df_pt)

# The str_detect command finds instances of a string within a string
mention_japan <- str_detect(string = df$text, pattern = "Japan")
mention_russia <- str_detect(string = df$text, pattern = "Russia")
df$japan <- mention_japan
df$russia <- mention_russia

# We add one column for articles that mention both countries
df$both <- with(df, japan & russia)
```

With all our docvars in place, we are ready to create our first `quanteda` corpus (step 1). All we need is the `corpus` function.

```
# Create a corpus with quanteda
nk_corpus <- corpus(df)

# We can see what's inside our corpus using the `summary` command
# The default number of entries to display is 100
summary(nk_corpus, 10)
```

```
## Corpus consisting of 3031 documents, showing 10 documents:
##
##          Text Types Tokens Sentences         DE   SC japan russia  both
##    KCNA.csv.1   124    226         9 1997-01-18 KCNA  TRUE  FALSE FALSE
##    KCNA.csv.2    69    122         4 1997-01-21 KCNA FALSE  FALSE FALSE
```

```
##    KCNA.csv.3   120    296         15 1997-01-25 KCNA FALSE  FALSE FALSE
##    KCNA.csv.4    38     59          4 1997-01-25 KCNA FALSE   TRUE FALSE
##    KCNA.csv.5    56     78          2 1997-01-28 KCNA FALSE  FALSE FALSE
##    KCNA.csv.6   132    304         10 1997-01-31 KCNA FALSE  FALSE FALSE
##    KCNA.csv.7   122    213          6 1998-01-08 KCNA FALSE  FALSE FALSE
##    KCNA.csv.8    28     48          2 1998-01-16 KCNA FALSE   TRUE FALSE
##    KCNA.csv.9   181    355         15 1998-01-16 KCNA FALSE  FALSE FALSE
##   KCNA.csv.10   133    255          9 1998-01-16 KCNA FALSE  FALSE FALSE
```

As you might recall from our lecture, we differentiate betwween "types" (unique words) and "tokens" (all words) in a document. To be more precise, both type and token also include punctuation and special symbols.

**Step 3: Defining documentary unit** When creating a corpus with `quanteda`, by running the `summary` command, we also get the number of sentences in a document. Depending on what it is that we are studying, we might determine that, the best documentary unit (i.e. how do we want to break down the corpus) is a sentence, or a paragraph, or the full document. We can make this transformations easily with the `corpus_reshape` command.

```r
# You could change the unit of text (defaults to "document") to sentences
nk_sent_corpus <- corpus_reshape(nk_corpus, to = 'sentences')
ndoc(nk_sent_corpus)
```

```
## [1] 32473
```

```r
summary(nk_sent_corpus, 3)
```

```
## Corpus consisting of 32473 documents, showing 3 documents:
##
##           Text Types Tokens Sentences         DE   SC japan russia  both
##   KCNA.csv.1.1    31     38         1 1997-01-18 KCNA   TRUE  FALSE FALSE
##   KCNA.csv.1.2    10     10         1 1997-01-18 KCNA   TRUE  FALSE FALSE
##   KCNA.csv.1.3    14     17         1 1997-01-18 KCNA   TRUE  FALSE FALSE
```

```r
# Or back to documents (in our case, each article)
nk_corpus <- corpus_reshape(nk_sent_corpus, to = 'documents')
ndoc(nk_corpus)
```

```
## [1] 3031
```

```r
summary(nk_corpus, 3)
```

```
## Corpus consisting of 3031 documents, showing 3 documents:
##
##         Text Types Tokens Sentences         DE   SC japan russia  both
##   KCNA.csv.1   124    226         9 1997-01-18 KCNA   TRUE  FALSE FALSE
##   KCNA.csv.2    69    122         4 1997-01-21 KCNA FALSE  FALSE FALSE
##   KCNA.csv.3   120    296        15 1997-01-25 KCNA FALSE  FALSE FALSE
```

Determining the best documentary unit really depends on your RQs or Hs. In our case, we are not interested in the granularity provided by a sentence-by-sentence analysis, so we will just keep our full corpus.

### Part 2 - Pre-processing documents in a corpus

As you might recall from our lecture, it is during the pre-processing stage that we make some of the most consequential decissions in the analysis of text. This is the stage in which we decide what features to include, and what features to transform. This is also the stage that tends to bring most human involvement (the 'qualitative' dimension). We will see how different choices impact our outcome by comparing different pre-processing choices.

**Step 4: Defining and refining features** While there isn't a single best workflow to pre-process our data, we generally follow these steps. In some cases, you might need/want to skip some of them (e.g. sometimes, capitalized words matter, and therefore we would not lowecarse our corpus). 1. Tokenize - we break down each text in the corpus into tokens. 2. Remove punctuation & capitalization. 3. Discard stopwords - we can use existing lists, or create our own lists. 4. Stem & lemmatize

Before we do that with our corpus, let's start with a short character vector to see how the process works. After tokenizing it, we are going to use the Porter stemmer for English to stem it. Remember that stemming remmoves rather bluntly the suffix of a word, and might lead to unwanted consequences. However, it is the easiest and fastest way to reduce the number of features (the dimensionality) in a corpus.

```
sampletxt <- "The police with their policing instruments created a policy of fear."

tokenized_text <- tokens(sampletxt)
tokenized_text
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "The"         "police"      "with"        "their"       "policing"
##  [6] "instruments" "created"     "a"           "policy"      "of"
## [11] "fear"        "."
```

```
stems <- tokens_wordstem(tokenized_text)
stems
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "The"        "polic"      "with"       "their"      "polic"
##  [6] "instrument" "creat"      "a"          "polici"     "of"
## [11] "fear"       "."
```

In our example, the un-stemmed sentence leaves us with 12 tokens and 12 features, while the stemmed version has 11 features. The Porter stemmer is able to differentiate between `polic` (police, and policing) and `polici` (policy).

Currently, quanteda uses the stemmer in the `SnowballC` package, and is is able to handle stemming for the following languages:

```
#install.packages(SnowballC)
library(SnowballC)
getStemLanguages()
```

```
##  [1] "arabic"     "basque"     "catalan"    "danish"     "dutch"
##  [6] "english"    "finnish"    "french"     "german"     "greek"
## [11] "hindi"      "hungarian"  "indonesian" "irish"      "italian"
## [16] "lithuanian" "nepali"     "norwegian"  "porter"     "portuguese"
## [21] "romanian"   "russian"    "spanish"    "swedish"    "tamil"
## [26] "turkish"
```

For languages in which character spaces are not used, quanteda uses different approaches for tokenization. For Japanese and Chinese, the `tokens()` function will automatically detect word boundaries using a dictionary with frequency information as explained here.

This isn't a clean approach and is prone to errors. There are other options for Japanese, as explained [here](https://tutorials.quanteda.io/language-specific/japanese/). For more info on quanteda & Chinese, you can read this. There's an excellent presentation on the topic of Asian languages and computational text analysis by Kohei Watanabe.

Now that you have seen how tokenization works, let's use the power of `quanteda` to pre-process textual data in a corpus. We can do most of the pre-processing (e.g. lowercasering, removing stopwords, tokenizing. . . ) with just a few lines of code.

```
# 1 - Tokenize corpus & remove punctuation
nk_tokens <- tokens(nk_corpus,
                    remove_punct = TRUE,
                    remove_numbers = TRUE,
                    remove_symbols = TRUE) # For even more options, see ?tokens
head(nk_tokens[[7]], 20) # Gives me 50 tokens from first document in corpus
```

```
##  [1] "Rodong"         "Sinmun"         "today"          "comments"
##  [5] "on"             "the"            "unjustifiable" "agreement"
##  [9] "of"             "the"            "National"       "Congress"
## [13] "for"            "New"            "Politics"       "the"
## [17] "United"         "Liberal"        "Democrats"      "and"
```

```
# 2- Lowercase the corpus
nk_lower_tokens <- tokens_tolower(nk_tokens)
head(nk_lower_tokens[[7]], 20)
```

```
##  [1] "rodong"         "sinmun"         "today"          "comments"
##  [5] "on"             "the"            "unjustifiable" "agreement"
##  [9] "of"             "the"            "national"       "congress"
## [13] "for"            "new"            "politics"       "the"
## [17] "united"         "liberal"        "democrats"      "and"
```

Your next choice is between discarding or not discarding words from the tokenized version of the corpus using a list of stopwords or by passing your own list of words. In either case, you will want to use the `tokens_remove()` command.

The `stopwords` package, which is used by `quanteda`, includes a good array of lists of commonly used words for many languages. The package includes lists from different sources, and for each source, there are lists for different languages. You can get the lists of sources and languages with specific commands as detailed below. Once you have identified the source and language you want, you can print the list of words.

```
# 3 - Remove stopwords
#install.packages("stopwords")
library(stopwords)
# Prints a list of available sources for stopwords
stopwords_getsources()
```

```
## [1] "snowball"      "stopwords-iso" "misc"          "smart"
## [5] "marimo"        "ancient"       "nltk"          "perseus"
```

```
# Prints a list of languags for a given source
stopwords_getlanguages("marimo")
```

```
## [1] "en"    "de"    "ar"    "he"    "zh_tw" "zh_cn" "ko"    "ja"
```

```
stopwords("en", "snowball")
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
```

```
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"       "will"
```

In addition, you could create your own list of words by simply creating a vector of words (or importing a list from an external file).

```r
# Create a list of words to exclude
days_week <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

# Exclude words from stopwords list
nk_tokens_no_stopwords <- tokens_remove(nk_lower_tokens, stopwords("en", "snowball"))
head(nk_lower_tokens[[7]], 20) # with stopwords
```

```
##  [1] "rodong"         "sinmun"         "today"          "comments"
##  [5] "on"             "the"            "unjustifiable"  "agreement"
##  [9] "of"             "the"            "national"       "congress"
## [13] "for"            "new"            "politics"       "the"
## [17] "united"         "liberal"        "democrats"      "and"
```

```r
head(nk_tokens_no_stopwords[[7]], 20) # without stopwords
```

```
##  [1] "rodong"        "sinmun"        "today"         "comments"
##  [5] "unjustifiable" "agreement"     "national"      "congress"
##  [9] "new"           "politics"      "united"        "liberal"
## [13] "democrats"     "grand"         "national"      "party"
## [17] "south"         "korea"         "push"          "ahead"
```

```r
# Exclude words from custom made list
nk_tokens_no_stopwords <- tokens_remove(nk_tokens_no_stopwords, days_week)
```

The final step of the pre-processing stage involves stemming or lemmatizing your corpus. Both approaches reduce the size of our data, as words that would be considered different in an un-stemmed corpus (e.g. win, winner and winning), would become the same word. Stemming can be done faily quickly, but it is more prone to error. Lemmatizing is more computationally intensive, but much more accurate.

As we saw earlier, we can use the `tokens_wordstem()` command to stem a sentence, a text, or a `quanteda` corpus. By default, quanteda assumes we are stemming an English language text, but it is possible to use the argument `language` to specify an alternative language from the list you have above.

```
# Stemming
nk_tokens_stemmed <- tokens_wordstem(nk_tokens_no_stopwords)
head(nk_tokens_stemmed[[7]], 20)
```

```
##  [1] "rodong"    "sinmun"    "today"     "comment"   "unjustifi" "agreement"
##  [7] "nation"    "congress"  "new"       "polit"     "unit"      "liber"
## [13] "democrat"  "grand"     "nation"    "parti"     "south"     "korea"
## [19] "push"      "ahead"
```

Lemmatizing involves using previously trained models of a language that make it possible to identify what part of speech a given word is, or to disambiguate when a word might have different meanings. This is, as you might imagine, a much more computationally intense process than stemming, which we were able to complete rather fast. There's no function in `quanteda` to lemmatize a corpus, but we can lean on the `udppipe` package to do so. Because this is a somewhat more complex process, I will not be covering it in this lab.

### Part 3 - DFM creation

**Step 5: Converting features to quantitative matrices** In `quanteda`, the data structure used to fit statistical models for text analysis is the document feature matrix (DFM). This is just one way to represent data in the bag-of-words-approach. Let's first use the `dfm()` command to create a DFM from the stemmed tokens object that we saved early on.

```
# DFM fromm a stemmed tokens object
nk_dfm_stemmed <- dfm(nk_tokens_stemmed)
```

For illustration, we are going to create several DFMs to compare the impact of different types of pre-processig on their size. We will create 4 DFMs: `nk_tokens` (unprocessed tokenized version of our corpus) named `nk_dfm1`, `nk_lower_tokens` (tokenized version of the corpus in lower case) named `nk_dfm2`, `nk_tokens_no_stopwords` (tokenized version with no stop words) named `nk_dfm3`, and, finally, `nk_tokens_stemmed` (tokenized, pre-processed, stemmed without stopwords) named `nk_dfm`. Compare the number of features in each DFM

```
# Step 1 - Creates DFM from tokens objects
nk_dfm1 <- dfm(nk_tokens, tolower = FALSE)
nk_dfm2 <- dfm(nk_lower_tokens)
nk_dfm3 <- dfm(nk_tokens_no_stopwords)
nk_dfm <- dfm(nk_tokens_stemmed)

# Step 2 - Compare number of features
nfeat(nk_dfm1)
```

```
## [1] 29104
```

```
nfeat(nk_dfm2)
```

```
## [1] 25209
```

```
nfeat(nk_dfm3)
```

```
## [1] 25054
```

```
nfeat(nk_dfm)
```

```
## [1] 16830
```

At each step of the way, the number of features in our DFM has been reduced. There's one last step we can take to decrease the number of words to make our analysis faster and to avoid unnecessary noise: trimming the `dfm` object. When we trim a `dfm`, we remove features that either occur very frequently (e.g. 95% of documents) or very rarely (e.g. less than 1% of documents). The `dfm_trim` allows to specify these percentages, and use other criteria to limit the size of our `dfm`, such as the absolute maximum or minimum number of times a word occurs in the corpus.

To exemplify this, let's print the top 50 occurring words in the `nk_dfm` object by using the `topfeatures()` command.

```
# Most frequently occurring words
topfeatures(nk_dfm, 50)
```

```
##        korean        peopl          kim       nation         dprk
##          6069         5903         5457         4735         4329
##            il      countri        korea        south         jong
##          3938         3416         3251         3158         3037
##         parti          war       presid     militari         work
##          2383         2126         2024         1891         1849
##          said      develop        great         forc           u.
##          1843         1838         1801         1734         1722
##          year           us       reunif         sung     committe
##          1707         1663         1656         1643         1480
## revolutionari         made    pyongyang        japan        world
##          1476         1396         1395         1367         1327
##       general        organ        north        build         armi
##          1317         1299         1277         1268         1262
##        leader     independ         make         peac        power
##          1259         1253         1246         1224         1216
##           new       includ       intern        polit       worker
##          1189         1166         1154         1150         1112
##       japanes            x        relat      nuclear       offici
##          1101         1100         1073         1063         1051
```

Given that our corpus has 3,000 documents, some of these terms might be appearing on almost every single document. A word that appears in all documents is a word that has no discriminative power; the same applies for words that are so unique that only "describe" one document.

```
# Compare different trims of nk_dfm
nk_dfm_trimmed1 <- dfm_trim(nk_dfm,
                            max_docfreq = 1250)

nk_dfm_trimmed2 <- dfm_trim(nk_dfm,
                            min_docfreq = 0.1)

nk_dfm_trimmed3 <- dfm_trim(nk_dfm,
                            min_termfreq = 10,
                            max_termfreq = 100)

nk_dfm_trimmed4 <- dfm_trim(nk_dfm,
                            min_termfreq = 100,
                            max_termfreq = 1000)

nfeat(nk_dfm_trimmed1)
```

```
## [1] 16823
```

```r
nfeat(nk_dfm_trimmed2)
```

```
## [1] 16830
```

```r
nfeat(nk_dfm_trimmed3)
```

```
## [1] 3521
```

```r
nfeat(nk_dfm_trimmed4)
```

```
## [1] 927
```

```r
topfeatures(nk_dfm_trimmed1, 20)
```

```
##            il        south         jong        parti          war
##          3938         3158         3037         2383         2126
##        presid     militari         work         said      develop
##          2024         1891         1849         1843         1838
##         great         forc           u.         year           us
##          1801         1734         1722         1707         1663
##        reunif         sung     committe revolutionari         made
##          1656         1643         1480         1476         1396
```

```r
topfeatures(nk_dfm_trimmed2, 20)
```

```
##   korean    peopl      kim   nation     dprk       il  countri    korea
##     6069     5903     5457     4735     4329     3938     3416     3251
##    south     jong    parti      war   presid militari     work     said
##     3158     3037     2383     2126     2024     1891     1849     1843
##  develop    great     forc       u.
##     1838     1801     1734     1722
```

```r
topfeatures(nk_dfm_trimmed3, 20)
```

```
## anti-reunif    meanwhil        bear       shape     premier  taekwon-do
##         100         100         100         100         100         100
##      victim     mansuda      defens       remov        sacr        root
##         100          99          99          99          99          99
##     contain   juche-ori   literatur      intend     session       block
##          99          99          99          98          98          98
##        date     describ
##          98          98
```

```r
topfeatures(nk_dfm_trimmed4, 20)
```

```
##   product      time    govern      caus    effort   central      idea       day
##       993       985       972       961       947       941       931       922
##      juch     visit    achiev   revolut   present   foreign      unit    declar
##       922       919       916       914       905       902       898       894
##       one   perform secretari socialist
##       892       886       878       864
```

As shown in the top 20 most frequent words for each of the four trims we have identified, our choice of how to limit the size of the DFM will have quite an impact on the data we will be using to analyze our texts, and to fit our models. There is no one solution that fits all cases, so you will need to play around with the settings until you find one that fits best to the data that you have.

**Part 4 - Descriptive statistics for a corpus**

We have reached the final two steps in our seven step approach to using `quanteda` to analyze text data. As you will soon discover, the last two steps are often the 'easiest' ones.

**Steps 5 & 6: Analize text data and summarize/interpret the results** With `quanteda` you can compute several descriptive measures of your texts including word frequencies (absolute and relative, lexical diversity, feature similarity…). Instructions on how to compute some of these metrics can be found here.

To conclude this lab, we will come back to the RQs that we put forward at the very beginning, and use our data to provide an answer.

1. What are the most frequently used words in news stories by KCNA and PT between 1997 and 2014?

We already know that we can retrieve top words from a dfm using the `topfeatures()` command. We can get additional information, and we can retrieve data for two different groups (KCAN and PT, for example) by using the `texstat_frequency()` command. Because we spent some time at the very beginning of this lab adding metadata to our corpus, now we can use that metadata (quanteda's docvars) to summarize the data for us.

We are going to compare the two sources (metadata stored in a docvar called 'SC'), and for each source, we are going to get the top 20 features.

```r
#install.packages("quanteda.textstats")
#install.packages("quanteda.textplots")
library(quanteda.textstats)
library(quanteda.textplots)
tstat_freq <- quanteda.textstats::textstat_frequency(nk_dfm, n = 20, groups = SC)
head(tstat_freq, 40)
```

```
##      feature frequency rank docfreq group
## 1        kim      3109    1     995  KCNA
## 2     korean      3072    2    1101  KCNA
## 3      peopl      2919    3    1140  KCNA
## 4       dprk      2475    4     947  KCNA
## 5         il      2340    5     794  KCNA
## 6     nation      2200    6     867  KCNA
## 7      south      1859    7     581  KCNA
## 8       jong      1857    8     726  KCNA
## 9      korea      1848    9     928  KCNA
## 10        u.      1722   10     426  KCNA
## 11   countri      1537   11     832  KCNA
## 12     parti      1233   12     553  KCNA
## 13    presid      1075   13     595  KCNA
## 14      said      1054   14     683  KCNA
## 15      forc      1029   15     492  KCNA
## 16    reunif       996   16     362  KCNA
## 17       war       995   17     388  KCNA
## 18     great       965   18     528  KCNA
## 19  committe       914   19     504  KCNA
## 20      sung       890   20     496  KCNA
## 21    korean      2997    1     711    PT
## 22     peopl      2984    2     734    PT
## 23    nation      2535    3     670    PT
## 24       kim      2348    4     560    PT
## 25   countri      1879    5     678    PT
## 26      dprk      1854    6     568    PT
```

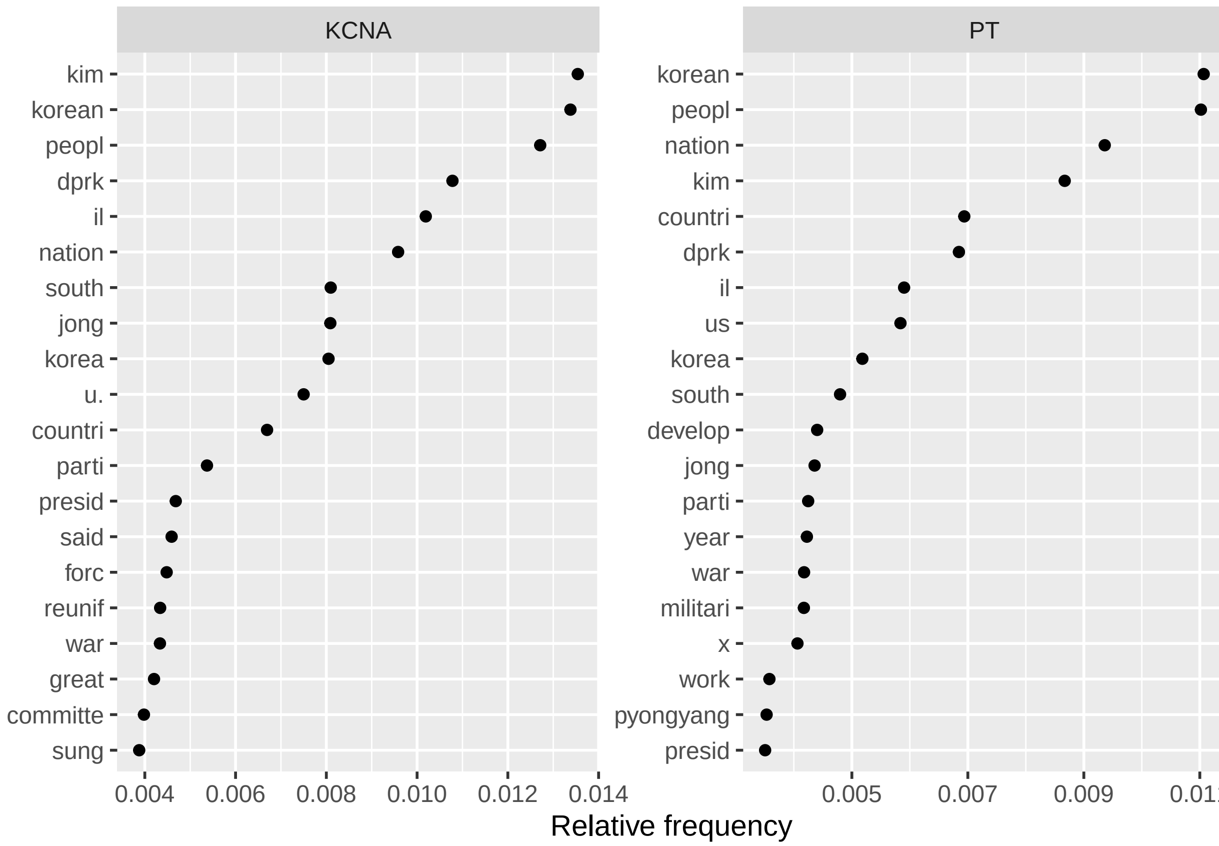
```
## 27        il      1598    7      451     PT
## 28        us      1581    8      324     PT
## 29     korea      1403    9      550     PT
## 30     south      1299   10      317     PT
## 31   develop      1192   11      488     PT
## 32      jong      1180   12      412     PT
## 33     parti      1150   13      359     PT
## 34      year      1144   14      518     PT
## 35       war      1131   15      313     PT
## 36  militari      1130   16      330     PT
## 37         x      1100   17     1100     PT
## 38      work       969   18      469     PT
## 39 pyongyang       956   19      433     PT
## 40    presid       949   20      346     PT
```

The table above provides both the number of times each feature is used, and the number of documents that contain each feature. We could use this information to compute relative frequencies, and plot them using the `ggplot` package. The chunk of code below weights the dfm (word frequency/total number of words), and uses that information to generate a plot that compares the top 20 words used by KCNA and PT.

```r
library(ggplot2)
nk_dfm_weighted <- nk_dfm %>%
  dfm_group(groups = SC) %>%
  dfm_weight(scheme = "prop")

relative_frequencies <- textstat_frequency(nk_dfm_weighted, n = 20, groups = SC)

ggplot(data = relative_frequencies, aes(x = factor(nrow(relative_frequencies):1), y = frequency)) +
  geom_point() +
  facet_wrap(~ group, scales = "free") +
  coord_flip() +
  scale_x_discrete(breaks = nrow(relative_frequencies):1,
                   labels = relative_frequencies$feature) +
  labs(x = NULL, y = "Relative frequency")
```




Unsurprisingly, there is very little difference in the most frequently used words of KCNA and PT, both of which are Party/State-controlled media. Any differences here in relative frequencies, would need to be tested statistically before we could make any inferences of the entire population. Remember, we only used a relatively small sample of articles for this analysis.

2. Are there differences in the words used by KCNA and PT in news stories mentioning Russia and Japan?

To answer this question, we are first going to use a simple visualization: a wordcloud of absolute frequencies to compare articles mentioning Russia to those mentioning Japan. The approach here is very similar to the one we used to plot KCNA and PT frequencies. First, we want to "group" our `nk_dfm` by a new docvar called `mentions` that tell us whether an article mentions Japan, Russia, both or neither. Earlier on we created columns with mentions for Russia and Japan. We can use these with the verb `mutate` and the command `case_when` to create the new variable based on four conditions.

```
df <- df %>%
  mutate(mentions = case_when(mention_japan == TRUE & mention_russia != TRUE ~ "Japan",
                              mention_japan != TRUE & mention_russia == TRUE ~ "Russia",
                              mention_japan == TRUE & mention_russia == TRUE ~ "Russia & Japan",
                              TRUE ~ "No mention"))
nk_dfm$mentions <- df$mentions # Adds the docvar to the dfm object
```

Now that we have this new variable, we can group the texts into one of these four categories. When we group a `dfm` we change the documentary unit from each article to each group. So, basically, we will have four very large documents, one with ALL articles that mention Japan, one with ALL the articles that mention Russia, and one each for those mentioning both countries, and those not mentioning either of them. We can see that when we use the `head()` command.

```
# Create a grouped dfm and compare groups
nk_dfm_compare <- dfm_group(nk_dfm, groups = mentions)
```

```r
head(nk_dfm_compare)
```

```
## Document-feature matrix of: 4 documents, 16,830 features (56.43% sparse) and 4 docvars.
##                   features
## docs                rodong sinmun today comment militarist utter made former
##   Japan                 50     50    73      12         54     7  300     57
##   No mention           162    164   192      39          1    39  973    107
##   Russia                 4      4    15       1          1     2   83      6
##   Russia & Japan         2      2     4       3          7     0   40     10
##                   features
## docs                director general
##   Japan                   57     279
##   No mention             218     886
##   Russia                  44     104
##   Russia & Japan           8      48
## [ reached max_nfeat ... 16,820 more features ]
```

As you can see, now we only have 4 'docs', one called Japan, one called Russia, one called Russia & Japan and one called No mention. Each "document" (group all the articles) contains the sum of all term frequencies. We can now use this DFM to create a comparative wordcloud.

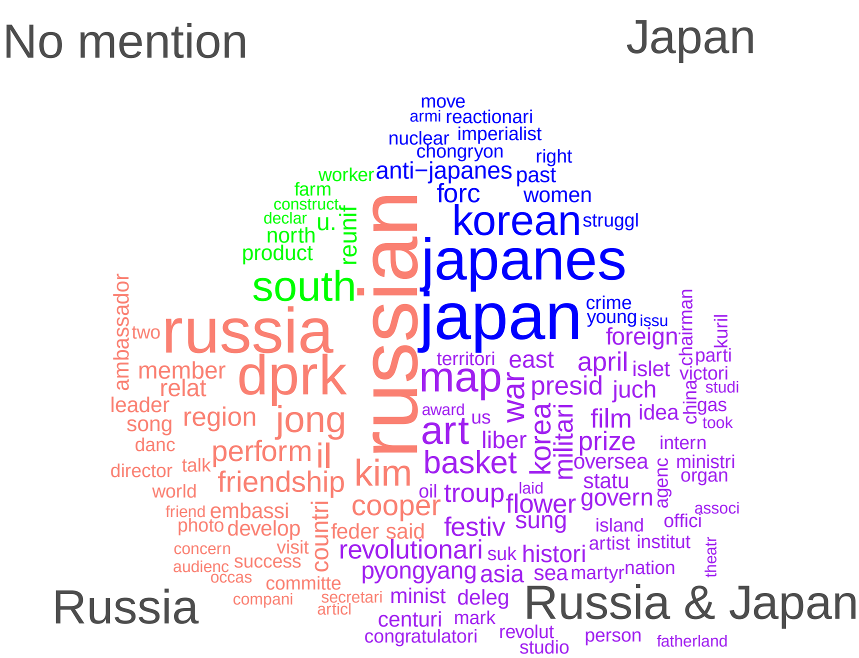
```r
# Create worcloud
set.seed(132)
textplot_wordcloud(nk_dfm_compare, comparison = TRUE, max_words = 130, color = c("blue", "green", "salm
```

```
## Warning in wordcloud_comparison(x, min_size, max_size, min_count, max_words, :
## leadership could not be fit on page. It will not be plotted.

## Warning in wordcloud_comparison(x, min_size, max_size, min_count, max_words, :
## front could not be fit on page. It will not be plotted.

## Warning in wordcloud_comparison(x, min_size, max_size, min_count, max_words, :
## polit could not be fit on page. It will not be plotted.

## Warning in wordcloud_comparison(x, min_size, max_size, min_count, max_words, :
## shale could not be fit on page. It will not be plotted.
```



If you look carefully, you can see that the words associated with Japan are much more belligerent (anti-japanese, aggressive, imperialisti, war, military…), while those used in articles about Russia are more amicable (cooperation, embassy, visit…).

You could now compare the actual counts of words, by using the `texstat_frequency` command we saw earlier.

```
# Get a table with frequencies
relative_frequencies <- textstat_frequency(nk_dfm_compare, n = 30, groups = mentions)
relative_frequencies
```

```
##           feature frequency rank docfreq        group
## 1          korean      1926    1       1        Japan
## 2           peopl      1415    2       1        Japan
## 3           japan      1245    3       1        Japan
## 4             kim      1229    4       1        Japan
## 5          nation      1154    5       1        Japan
## 6         japanes      1014    6       1        Japan
## 7           korea       962    7       1        Japan
## 8            dprk       905    8       1        Japan
## 9              il       899    9       1        Japan
## 10        countri       813   10       1        Japan
## 11            war       772   11       1        Japan
## 12       militari       725   12       1        Japan
## 13  revolutionari       595   13       1        Japan
## 14           forc       581   14       1        Japan
## 15           jong       577   15       1        Japan
## 16          south       561   16       1        Japan
## 17          parti       539   17       1        Japan
## 18             us       532   18       1        Japan
## 19           sung       494   19       1        Japan
## 20         presid       482   20       1        Japan
## 21          organ       419   21       1        Japan
## 22           armi       418   22       1        Japan
## 23           work       390   23       1        Japan
## 24             u.       372   24       1        Japan
## 25          great       359   25       1        Japan
## 26           year       353   26       1        Japan
## 27           said       352   27       1        Japan
## 28        revolut       334   28       1        Japan
## 29        nuclear       334   28       1        Japan
## 30           issu       332   30       1        Japan
## 31          peopl      4004    1       1   No mention
## 32         korean      3657    2       1   No mention
## 33            kim      3607    3       1   No mention
## 34         nation      3228    4       1   No mention
## 35           dprk      2969    5       1   No mention
## 36             il      2557    6       1   No mention
## 37          south      2509    7       1   No mention
## 38        countri      2231    8       1   No mention
## 39           jong      2072    9       1   No mention
## 40          korea      2054   10       1   No mention
## 41          parti      1653   11       1   No mention
## 42        develop      1396   12       1   No mention
## 43           work      1319   13       1   No mention
```

```
## 44          said     1298   14          1      No mention
## 45        presid     1297   15          1      No mention
## 46        reunif     1282   16          1      No mention
## 47         great     1279   17          1      No mention
## 48            u.     1278   18          1      No mention
## 49          year     1235   19          1      No mention
## 50           war     1200   20          1      No mention
## 51      committe     1069   21          1      No mention
## 52          forc     1047   22          1      No mention
## 53         north     1033   23          1      No mention
## 54      militari      980   24          1      No mention
## 55          sung      974   25          1      No mention
## 56          made      973   26          1      No mention
## 57     pyongyang      962   27          1      No mention
## 58            us      959   28          1      No mention
## 59         world      937   29          1      No mention
## 60          peac      917   30          1      No mention
## 61           kim      431    1          1          Russia
## 62          dprk      377    2          1          Russia
## 63         peopl      339    3          1          Russia
## 64            il      335    4          1          Russia
## 65       russian      304    5          1          Russia
## 66        korean      294    6          1          Russia
## 67          jong      282    7          1          Russia
## 68       countri      258    8          1          Russia
## 69        russia      242    9          1          Russia
## 70        nation      223   10          1          Russia
## 71        presid      157   11          1          Russia
## 72          said      144   12          1          Russia
## 73       develop      128   13          1          Russia
## 74       perform      127   14          1          Russia
## 75         korea      124   15          1          Russia
## 76        leader      122   16          1          Russia
## 77         parti      119   17          1          Russia
## 78         great      111   18          1          Russia
## 79      committe      109   19          1          Russia
## 80            us      109   19          1          Russia
## 81          work      106   21          1          Russia
## 82    friendship      105   22          1          Russia
## 83        intern      105   22          1          Russia
## 84          sung      105   22          1          Russia
## 85       general      104   25          1          Russia
## 86      militari      103   26          1          Russia
## 87        cooper      103   26          1          Russia
## 88         visit      100   28          1          Russia
## 89         world       99   29          1          Russia
## 90       foreign       98   30          1          Russia
## 91        korean      192    1          1 Russia & Japan
## 92           kim      190    2          1 Russia & Japan
## 93            il      147    3          1 Russia & Japan
## 94         peopl      145    4          1 Russia & Japan
## 95        nation      130    5          1 Russia & Japan
## 96         japan      122    6          1 Russia & Japan
## 97       countri      114    7          1 Russia & Japan
```

```
## 98          korea        111    8         1 Russia & Japan
## 99           jong        106    9         1 Russia & Japan
## 100           war         92   10         1 Russia & Japan
## 101        presid         88   11         1 Russia & Japan
## 102       japanes         87   12         1 Russia & Japan
## 103      militari         83   13         1 Russia & Japan
## 104          dprk         78   14         1 Russia & Japan
## 105         parti         72   15         1 Russia & Japan
## 106          sung         70   16         1 Russia & Japan
## 107 revolutionari         69   17         1 Russia & Japan
## 108           art         69   17         1 Russia & Japan
## 109            us         63   19         1 Russia & Japan
## 110     pyongyang         62   20         1 Russia & Japan
## 111        russia         61   21         1 Russia & Japan
## 112           map         58   22         1 Russia & Japan
## 113        leader         54   23         1 Russia & Japan
## 114       foreign         54   23         1 Russia & Japan
## 115         great         52   25         1 Russia & Japan
## 116          juch         51   26         1 Russia & Japan
## 117        basket         51   26         1 Russia & Japan
## 118          said         49   28         1 Russia & Japan
## 119        intern         49   28         1 Russia & Japan
## 120         organ         49   28         1 Russia & Japan
```

The list of absolute counts confirms what we could see in the wordcloud, words like "armi", "war", "militari" and "forc" are among the top 30 most frequently occurring words in texts about Japan, but most are missing from the list for Russia. Instead, we find words like "develop", "friendship", "visit" and "cooper".

We could now use these word counts (or relative word counts) to test whether these differences we observe in the sample are statistically significant, and thus descriptive of the entire population.