LEIPZIG UNIVERSITY
LINGUISTISCHE INFORMATIK ÜBUNG

Dr. Andreas Niekler
aniekler@informatik.uni-leipzig.de
Leipzig University

Zipf's law

In the exercise, a new R-Script is created. To do this, go to File -> New File -> R Script. In this script you will enter and execute all commands. If you want to run the complete script in RStudio, you can use Ctrl-A to select the complete source code and execute with Ctrl-Return. If you want to execute only one line, you can simply press Ctrl-Return on the respective line. You can also copy the source code directly into the console (2).

At the beginning of a script, different variables can be defined, whose values can always be used in a script.

```
dir <- "Your Choice"
setwd(dir)
```

The `read.csv` command reads a *CSV* (Comma Separated Value) file. Such files represent a table whose lines are represented by line breaks in the files and columns by a *separator*. Each row can therefore represent several columns of a table. The command can also be used to specify whether the CSV file contains a line with column names (header = TRUE or FALSE) and with which character set the file is to be loaded (encoding).

Now we read the textkorpus so that the raw text is available within **R**. This can be CSV, XML or pure text files. In our case, we have a CSV file with the format *ID \t DATE \t PUBLICATION \t TITLE \t TEXT*, which can be imported via the `read.csv` command. The individual columns are separated by a tabulator symbol (in R represented by "\t"), and the string of the document contents (which themselves could in turn divide the TAB separator) itself are encapsulated with quotation marks (**sep = "^"; quote="""**).

```
# Corpus file in CSV format
sourceFile <- "data/corpus.csv"
# Read the file into an object
textdata <- read.csv(
  sourceFile,
  header = TRUE,
  sep = "\t",
  quote="\"",
```

```
  enc = "UTF-8",
  stringsAsFactors = FALSE
)
```

The texts are now available in a data frame together with some metadata (ID, date of publication, publication, heading). Let us first see how many documents and metadata are available.

```
# Dimensions of the read-in data frame
dim(textdata)
```

```
## [1] 5078    5
```

```
# Column names of the metadata
colnames(textdata)
```

```
## [1] "ID"          "DATE"        "PUBLICATION" "TITLE"
## [5] "TEXT"
```

How many articles from ZEIT and SPIEGEL are available? This can be easily found with the command *table*, which can be used to count a cross table of different values. If we only apply it to the column *PUBLICATION* of our data frame, we get the counts of the documents in the individual newspapers.

```
table(textdata[, "PUBLICATION"])
```

```
##
## Spiegel    ZEIT
##    2912    2166
```

Now we want to transfer the loaded text source into a document-term-matrix object.

A further aim of this exercise is to provide a first statistical analysis of speech data. At the moment, the data has been read in **R**. However, the voice data are still present as string within the corpus object. To analyze which word forms the text contains, the text must be tokenized. This means that all the words in the texts are identified individually. Only in this way is it possible to count the frequency of individual word forms. A word form is also called "Type". The occurrence of a type in a text is a "token".

For further statistical analyzes, the text must also be available in a statistically evaluable form. This is only possible if the text is represented in numeric form. The reasoning in text mining is that the texts are represented as statistics on the contained words. All words are identified in a document. Furthermore, a data structure is created for a document in which a value can be set for each word form contained in the corpus. One calls such a data structure word vector a document. The dimension of a word vector corresponds to the size of the vocabulary. In this word vector, for each word contained in a document, the number of times this word appears in a document is entered. The word vectors have the same form for all documents in a corpus. Only the counts of the occurring words are distinguished. As a result, the entire body can be represented as a matrix by using all word vectors of the documents as lines of one. This data structure is called *document term matrix*.

The following source text example creates a document term matrix with the tm package. The function `DocumentTermMatrix` with the corpus object is called for this purpose. If this command is called without further parameters, the individual word forms are identified by using the "space" as the word separator.

```
# Create a document term matrix (can take a little time)
DTM <- processToDTM(processIds = 1:length(textdata$TEXT), data = textdata)
# Retrieve information about the document term matrix
dim(DTM)

stopwords <- union(readLines("./resources/stopwords_de.txt", encoding = "UTF-8"),
    tm::stopwords(kind = "german"))

# Stopword filtering, not necessary here because we want to observe language
# properties DTM <- DTM[,!(colnames(dtm) %in% stopwords)]

# delete vocabulary which has only 1 character
DTM <- DTM[, nchar(colnames(DTM)) > 1]
```

The word list provides a first access to the statistical distributions of a corpus. This is a data structure that contains all the word forms that exist in a corpus. In addition, the word list contains information about the frequency of a single word form. The texts or the corpus are available as a document term matrix. To create a list of words, you can sum over the columns of this matrix. This summarizes how often a word form is used in all documents.

A so-called *sparse matrix data structure* is used for the data structure of the document term matrix. Since most entries in a document term vector are 0, it would be very inefficient to actually store these values. A sparse data structure actually stores only the elements of a vector to which a value has also been assigned.

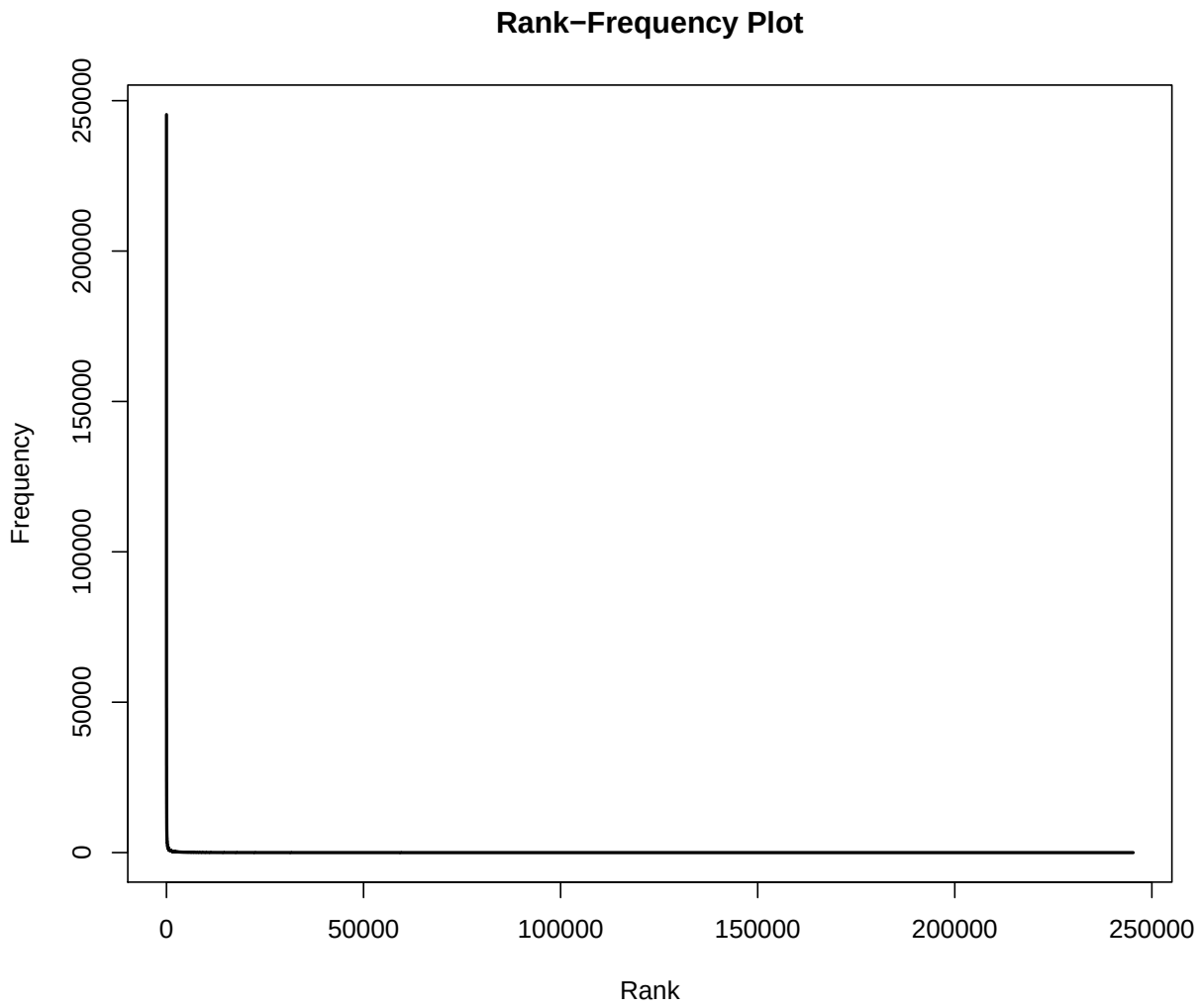
```
#Calculate the totals of all columns in the document term matrix
freqs <- colSums(DTM)
#Create a vector with the names of the words
words <- colnames(DTM)
#Create a data structure whose columns contain the word description and its frequency
wordlist <- data.frame(words, freqs)
#Descending sorting of the word list based on the frequencies
wordIndexes <- order(wordlist[, "freqs"], decreasing = TRUE)
wordlist <- wordlist[wordIndexes, ]
#Display the 100 most common words
head(wordlist, 30)
```

```
##       words  freqs
## die     die 245370
## der     der 245149
## und     und 148458
## in       in 127042
## den     den  87057
## das     das  74093
```

```
## zu       zu  69798
## von     von  63964
## nicht nicht  61208
## sich   sich  57662
## mit     mit  56203
## im       im  52813
## des     des  52724
## ein     ein  50940
## er       er  49717
## dem     dem  48723
## ist     ist  47767
## für     für  47295
## sie     sie  46676
## es       es  45762
## auf     auf  44662
## als     als  43385
## eine   eine  40398
## auch   auch  40151
## an       an  30455
## aus     aus  29423
## nach   nach  29044
## daß     daß  28760
## wie     wie  28580
## so       so  26797
```

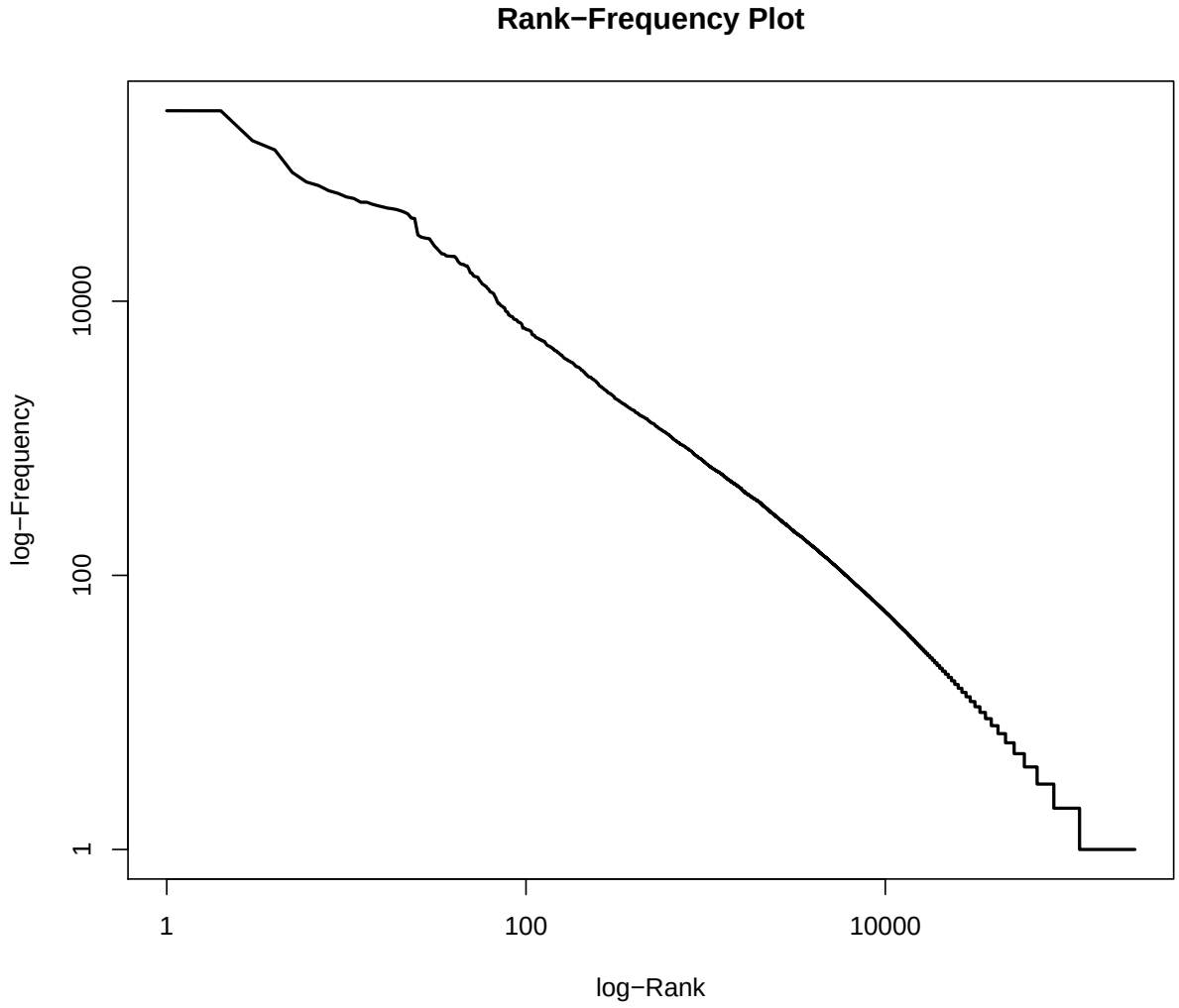
The words in this sorted list have a ranking depending on the position in this list. If the frequencies are converted into a representation in which all the ranks are plotted on the x axis and all frequencies on the y axis, the Zipf distribution is obtained. This is a typical property of natural language data, and this distribution is similar for all languages.

```
plot(wordlist$freqs, type = "l", lwd = 2, main = "Rank-Frequency Plot", xlab = "Rank",
    ylab = "Frequency")
```



The distribution is an extreme power distribution (very few words that occur very often, very many words that are very rare). The distribution says that the frequency of a word is reciprocal to its rank $(1/r)$. To make the display more accessible, the axes can be displayed logarithmized.

```
plot(wordlist$freqs, type = "l", log = "xy", lwd = 2, main = "Rank-Frequency Plot",
    xlab = "log-Rank", ylab = "log-Frequency")
```



In the illustrations, two extreme ranges can be determined. There are only words above the rank of 10,000 that can be counted less than 10 times. There are below the rank 100 words, which can be counted more than 1000 times in the documents. The aim of text mining is to allow structures in documents to be automatically identified. The mentioned extreme areas are not suitable for this because the word forms with an frequency <10 in too few documents to find. At the same time the word forms occur with an frequency> 1000 in almost all documents and are not suitable (i.d.R.), since they contribute little to the meaning of a text (usually stop words). The renouncement of very rare / frequent words has many advantages:

- The number of dimensions in the word vectors decreases, since stop words and the rare words do not have to be displayed
- This increases efficiency in calculations
- Procedures that semantic structures are to find in texts become more precise

To illustrate the amount of word to be used to carry out analyzes, a section is drawn in the following diagram, which marks the essential words of the text corpus. To display this area, a list of so-called stop words is loaded. These are words a language that normally does not contribute

to semantic information about a text. In addition, all the words in the word list that are less than 10 times are identified.

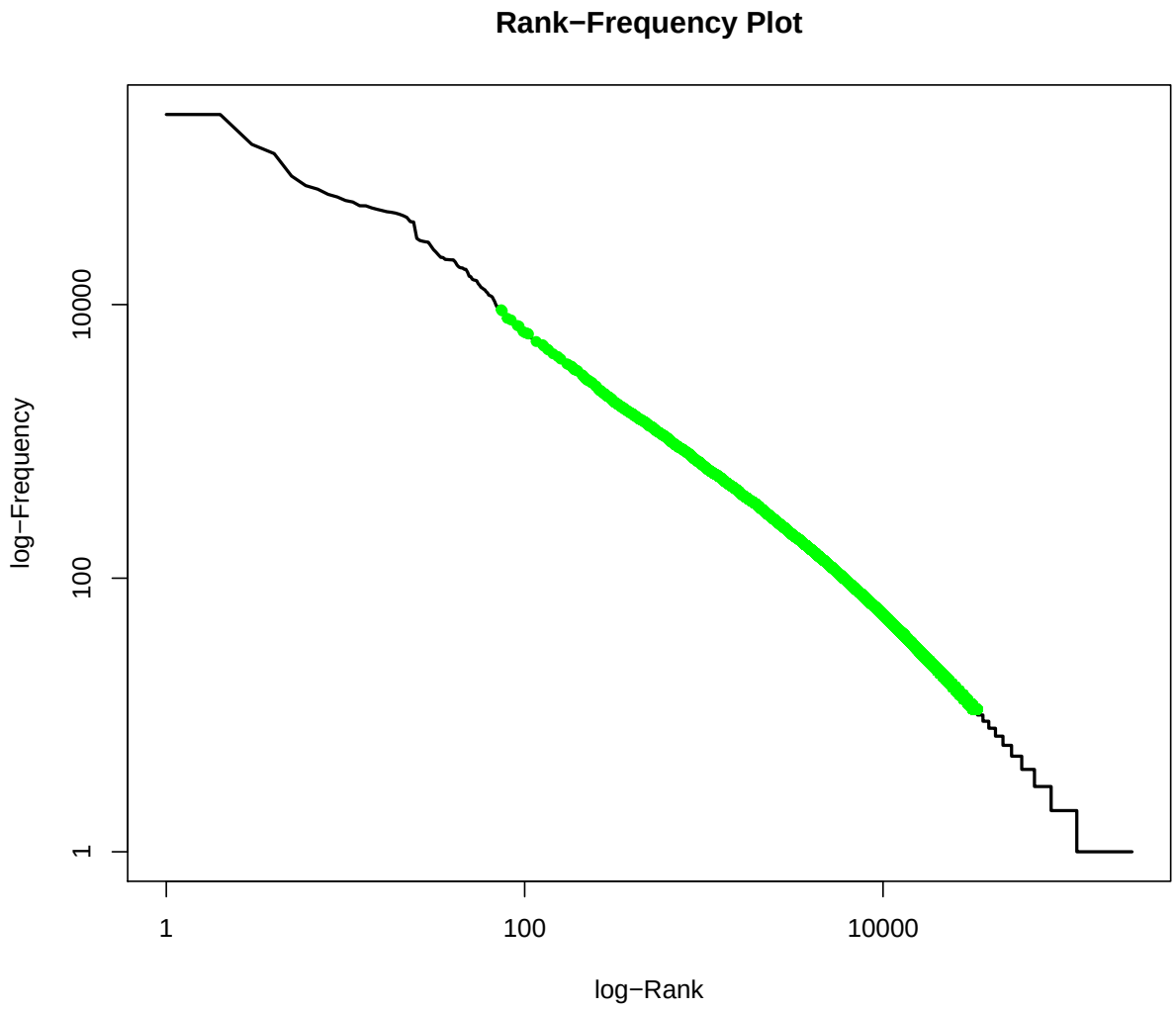
With the *%in%* comparison, two vectors can be compared. At this point, we compare the words in the word list with the loaded stopword list and an internal stopword list from the tm package. Result is a vector that contains values TRUE or FALSE for each word if they occur in the stop word list or do not occur.

A Boolean value (or a vector of boolean values) can be inverted with **!** (TRUE becomes FALSE and vice versa) so that all lines of the word list that do not exist in both lists are marked. With the *which* command, the indices of the rows in the worlist can be fetched.

We search for all words from the word list that are used more than 10 times. Then we compile an intersection of all indices in w and all the idioms of the words with a frequency> 10.

With the command "lines", the range of the resulting indices can be drawn into the representation.

```
plot(wordlist$freqs, type = "l", log = "xy", lwd = 2, main = "Rank-Frequency Plot",
    xlab = "log-Rank", ylab = "log-Frequency")
wordIndexes1 <- which(!wordlist$words %in% stopwords)
wordIndexes2 <- which(wordlist$freqs > 10)
intersectedIndexes <- intersect(wordIndexes1, wordIndexes2)
lines(intersectedIndexes, wordlist$freqs[intersectedIndexes], col = "green", lwd = 2,
    type = "p", pch = 20)
```



The green area shows in the display the word forms that occur more frequently than 10 times and can not be found in the loaded stopword lists.

Further tasks

1. Display the most common words in the word list without the rarest and without the stop words.
2. In the result of Exercise 1, check whether there are texts for this text and these body words, which should also be regarded as a stopword.
3. What is the proportion of words that occur only once in the body?
4. Calculate the type token ratio of the present corpus.