

# Scraping Wikipedia

Jan-Philipp Kolb

09 Mai 2017

# Einleitung

Im Folgenden werde ich zeigen, wie man Textinformationen aus Wikipedia herunterladen, verarbeiten und analysieren kann.

```
install.packages("NLP")  
install.packages("tm")  
install.packages("FactoMineR")
```

## Die verwendeten Pakete

- Das R-Paket `stringi` von Marek Gagolewski and Bartek Tartanus bietet Möglichkeiten zur String Verarbeitung.

```
library("stringi")
```

- `tm` ist ein R-Paket um Text Mining zu realisieren. Es wurde von Ingo Feinerer, Kurt Hornik, und David Meyer geschrieben.

```
library("tm")
```

- Und schließlich brauchen wir das `FactoMineR`-Paket, das von Sebastien Le, Julie Josse und Francois Husson zur Durchführung der Hauptkomponentenanalyse erstellt wurde.

```
library("FactoMineR")
```

# Die Text Daten herunterladen

- Als Beispiel verwenden wir Daten zu verschiedenen Krankheiten.
- In diesem Fall habe ich 7 deutsche Webseiten für Infektionskrankheiten ausgewählt.

```
wiki <- "http://de.wikipedia.org/wiki/"

titles <- c("Zika-Virus", "Influenza-A-Virus_H1N1",
            "Spanische_Grippe", "Influenzavirus",
            "Vogelgrippe_H5N1",
            "Legionellose-Ausbruch_in_Warstein_2013",
            "Legionellose-Ausbruch_in_Jülich_2014")
```

# Das Herunterladen der Seiten

- Zunächst wird ein Container erstellt um die Ergebnisse abzuspeichern
- Dann wird der Text für jeden Artikel heruntergeladen und in dem Container gespeichert.

```
articles <- character(length(titles))

for (i in 1:length(titles)){
  articles[i] <- stri_flatten(
    readLines(stri_paste(wiki, titles[i])), col = " ")
}

docs <- Corpus(VectorSource(articles))
```

# Die Daten vorbereiten

Das Folgende basiert auf einem Blogpost von Norbert Ryciak über die automatische Kategorisierung von Wikipedia-Artikeln.

- Eine Fehlermeldung ist aufgetreten, als ich den Code ausgewertet habe.
- Es war möglich, dieses Problem mit Hinweisen aus einer Diskussion auf Stackoverflow zu lösen.

```
docs2 <- tm_map(docs, function(x) stri_replace_all_regex(  
  x, "<.+?>", " "))  
docs3 <- tm_map(docs2, function(x) stri_replace_all_fixed(  
  x, "\t", " "))
```

# Den Text weiterverarbeiten

```
docs4 <- tm_map(docs3, PlainTextDocument)
docs5 <- tm_map(docs4, stripWhitespace)
docs6 <- tm_map(docs5, removeWords, stopwords("german"))
docs7 <- tm_map(docs6, removePunctuation)
docs8 <- tm_map(docs7, tolower)
docs8 <- tm_map(docs8, PlainTextDocument)
```

```
dtm <- DocumentTermMatrix(docs8)
```

# Principal Component Analysis

- Der folgende Code ist auf einem Blog post von Arthur Charpentier über das Mining von Wikipedia basiert.

```
dtm2 <- as.matrix(dtm)
frequency <- colSums(dtm2)
frequency <- sort(frequency, decreasing=TRUE)
words <- frequency[frequency>20]
s <- dtm2[1,which(colnames(dtm2) %in% names(words))]]

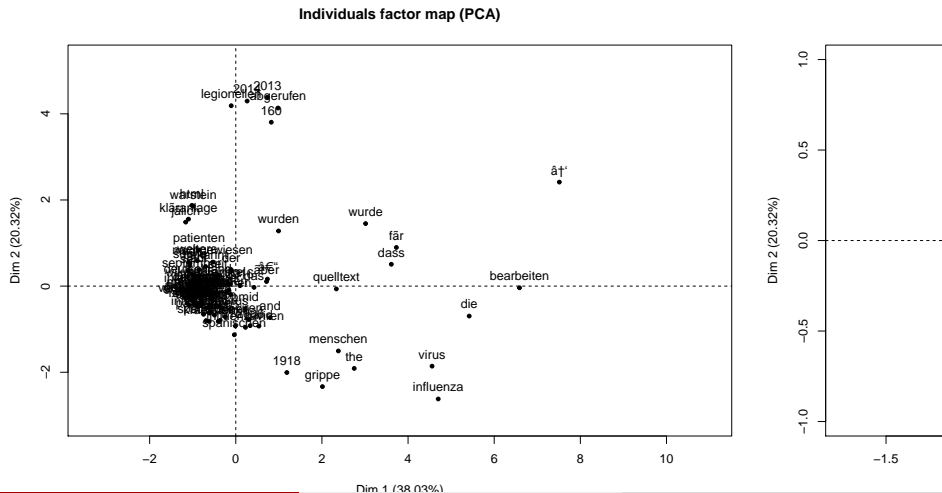
for(i in 2:nrow(dtm2)){
  s <- cbind(s,dtm2[i,which(colnames(dtm2) %in%
                           names(words))])
}

colnames(s) <- titles
```



# Ergebnis

## PCA(s)



- In der Factor Map sehen wir das erwartete Ergebnis.
- Die Seiten zur Legionnellen Krankheit sind sehr nah beianander, während die Seiten zur Influenza in einem anderen Teil sind.

# Das Dendrogramm

- Im Folgenden wird die Normalisierung durchgeführt und die Ergebnisse werden geplottet.

```
s0 <- s/apply(s,1,sd)
h <- hclust(dist(t(s0)), method = "ward")

plot(h, labels = titles, sub = "")
```

