Dieses erste Kapitel liefert einen Überblick über zahlreiche Funktionen des Pakets quanteda, die gleichzeitig die Grundlage der automatisierten Inhaltsanalyse mit R bilden. Über quanteda hinaus werden im Verlauf dieser neunteiligen Einführung noch eine Reihe weiterer R-Bibliotheken verwendet, etwa für das Berechnen von Themenmodellen (Kapitel 5) und das überwachte maschinelle Lernen (Kapitel 6). In praktisch jeder Einheit relevant sind dabei die Pakete des tidyverse (vor allem ggplot, dplyr und stringr), durch die zahlreiche Funktionen wie Plotten, Textverarbeitung und Datenmanagement gegenüber den R-Basisfunktionen stark verbessert werden. Pakete für einzelne Teilbereiche, die erst später eine Rolle spielen werden, sind u.a. topicmodels und stm (Themenmodelle), RTextTools (überwachtes maschinelles Lernen), und spacyr (POS-Tagging und Named-Entity-Erkennung).

Die Basis der Analyse in diesem ersten Kapitel sind die beliebten Geschichten von Sherlock Holmes. Das Sherlock Holmes-Korpus besteht aus zwölf Erzählungen, die in dem 1892 erschienenem Band *The Adventures of Sherlock Holmes* zusammengefasst sind, und die man gemeinfrei unter anderem durch das Internet Archive herunterladen kann. Die für diese Einführung verwendete Fassung wurde zunächst dem Internet Archive entnommen und dann in zwölf Einzeldateien aufgeteilt. Natürlich können die vorgestellten Methoden auf die anderen hier behandelten Korpora angewandt werden – das Beispiel dient nur dazu, sich langsam an quanteda und die Grundlagen der computergestützen Inhaltsanalyse zu gewöhnen.

Sämtliche in dieser Einführung verwendeter Codebeispiele, Korpora und Lexika können hier heruntergeladen werden.

Installation und Laden der benötigten R-Bibliotheken

Zunächst werden die notwendigen Bibliotheken installiert (sofern noch nicht vorhanden) und anschließend geladen. Zudem wird vorbereitend die Theme-Einstellung für das Paket ggplot gesetzt (dies sorgt für hübschere Plots). Diesen Schritt wiederholen wir zu Beginn jedes Kapitels, daher wird auf ihn später nicht mehr weiter eingegangen. In einigen Kapiteln werden noch weiteren Pakete gelanden, etwa für eine erweiterte Farbpalette (RColorBrewer), Wortwolken (wordclouds) oder um URLs zu parsen (urltools).

```
# Installation und Laden der Bibliotheken
if(!require("quanteda")) install.packages("quanteda")
if(!require("readtext")) install.packages("readtext")
if(!require("tidyverse")) install.packages("tidyverse")
if(!require("RColorBrewer")) install.packages("RColorBrewer")
theme_set(theme_bw())
```

Einlesen der Daten und Anlegen eines Korpus

Nachdem alle notwendigen Pakete geladen wurden, können wir nun die Daten einlesen und daraus ein quanteda-Korpus erstellen. Für das Einlesen der Plaintext-Dateien wird die Funktion readtext aus dem gleichnamigen Paket verwendet, durch die sich eine Reihe von Dateiformaten erfolgreich importieren lassen (u.a. TXT, PDF und Word). Grundsätzlich sich Plaintext–Daten (i.d.R. mit der Endung ".txt") und Daten in Tabellenform (etwa im Format CSV oder auch als Excel–Datei) für readtext ohne größere Probleme lesbar, allerdings muss man beim Einlesen erklären, wie genau die einzelnen Datensätze von einander getrennt sind (bei Plaintext–Dateien wo nicht 1 Datei == 1 Text, was etwa bei Exporten aus Lexis Nexis der Fall sein kann), bzw. welche Felder die Primär– und welche Metadaten beinhalten (bei Tabellen).

In diesem Fall entspricht jede Datei einem Text, wodurch der Import sehr umkompliziert ausfällt. Wir entfernen die Endung ".txt" aus dem Dokumentnamen, um diese später in Plot–Beschriftungen verwenden zu können. Schließlich wird die Variable *korpus* aufgerufen, was uns die wichtigen Eckdaten Dokumentanzahl und Docvars (Metadaten zu den Texten im Korpus) zurückliefert.

```
daten.sherlock <- readtext("daten/sherlock/romane/[0-9]*.txt") # Dateiname beginnt mit Zahl und endet m
daten.sherlock$doc_id <- str_sub(daten.sherlock$doc_id, start = 4, end = -5) # Dateiendung weglassen
```

```
korpus <- corpus(daten.sherlock, docid_field = "doc_id") # Korpus anlegen
docvars(korpus, "Textnummer") <- sprintf("%02d", 1:ndoc(korpus)) # Variable Textnummer generieren
korpus
```

```
## Corpus consisting of 12 documents and 1 docvar.
```

In den folgenden Abschnitten werden häufig bereits vorbereitete Korpora geladen, d.h. der Befehl *corpus* wird hier nicht mehr explizit ausgeführt. Er ist aber im Vorfeld ausgeführt worden, um aus Textdatein auf der Festplatte oder Twitter-Daten in einem R-Data Frame ein quanteda-Korpus zu erstellen.

Die Funktionen ndoc, ntoken, ntype und nsentence geben die Anzahl der Dokumente, Tokens, Types und Sätze aus. Diese Statistiken können bequem gemeinsam mit Metadaten auf Dokumentebene durch die Funktion summary erstellt werden. Bei den meisten Korpora, die hier verwendet werden, liegt ein solcher Data Frame mit Statistiken zu jedem Text bereits bei. Notwendig ist dies allerdings nicht. Will man auf Korpus–Metadaten zurückgreifen oder diese verändern, kann man dies jederzeit über den Befehl docvars tun.

```
korpus.stats <- summary(korpus, n = 1000000)
korpus.stats$Text <- reorder(korpus.stats$Text, 1:ndoc(korpus), order = T)
korpus.stats
```

```
## Corpus consisting of 12 documents:
##
##                                  Text Types Tokens Sentences Textnummer
##                   A Scandal in Bohemia  2145  10542       669         01
##                 The Red-headed League  2087  11118       573         02
##                    A Case of Identity  1750   8506       396         03
##             The Boscombe Valley Mystery  2096  11499       636         04
##                   The Five Orange Pips  1925   8879       475         05
##            The Man with the Twisted Lip  2173  11160       586         06
##    The Adventure of the Blue Carbuncle  1926   9651       552         07
##     The Adventure of the Speckled Band  2232  11783       614         08
##  The Adventure of the Engineer's Thumb  1968   9999       508         09
##    The Adventure of the Noble Bachelor  1944   9987       540         10
##     The Adventure of the Beryl Coronet  1991  11669       626         11
##    The Adventure of the Copper Beeches  2110  12011       622         12
##
## Source: /Users/cp/Documents/GitHub/inhaltsanalyse-mit-r.de/* on x86_64 by cp
## Created: Wed Dec 26 13:19:04 2018
## Notes:
```

Das Funktionsargument n = 1000000 wird hier nur deshalb verwendet, weil die Funktion *summary* ansonsten nur maximal 100 Texte zusammenfasst. In diesem Fall reicht das zwar aus, aber bei größeren Datensätzen ist das eher unpraktisch. Technisch gesehen heißt diese Funktion *summary.corpus* und ist eine an Korpus-Objekte angepasste Variante der Basisfunktion *summary*, die auch sonst in R verwendet wird. Der Befehl reorder wird verwendet, um die Texte nach ihrer Reihenfolge in *The Adentures of Sherlock Holmes* zu sortieren, statt alphabetisch nach Titel.

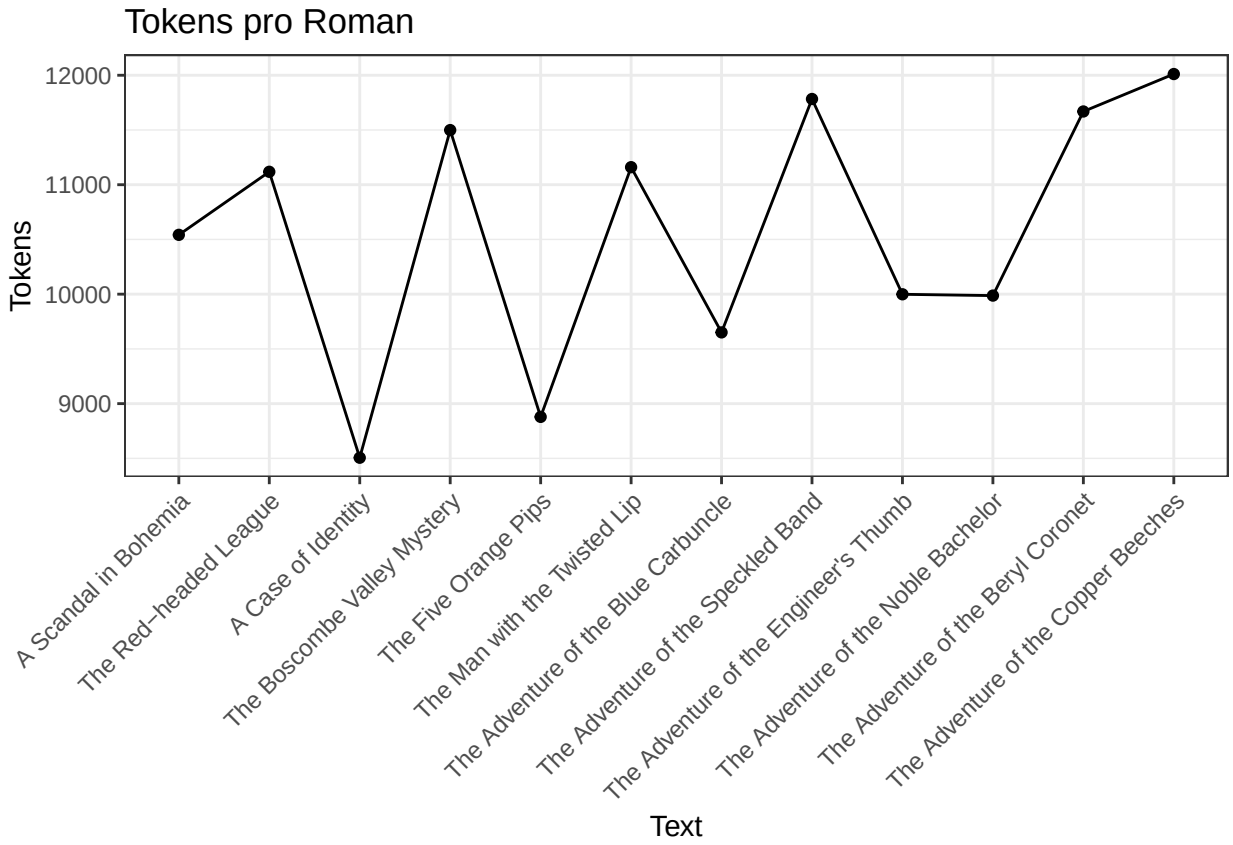
Basisstatistiken zu einem Korpus berechnen

Der Inhalt der Variable *korpus.stats* kann natürlich auch geplottet werden, um einen anschaulichen Eindruck von der Korpusbeschaffenheit zu geben. Die folgenden Zeilen liefern die Anzahl der Tokens (laufende Wörter), die Anzahl der Types (einmalige Wörter), und Sätze pro Roman zurück (vgl. dazu diese Einführung). Schließlich wird noch das Verhältnis von Typen zu Tokens (oder die sog. Typ-Token-Relation) geplottet.

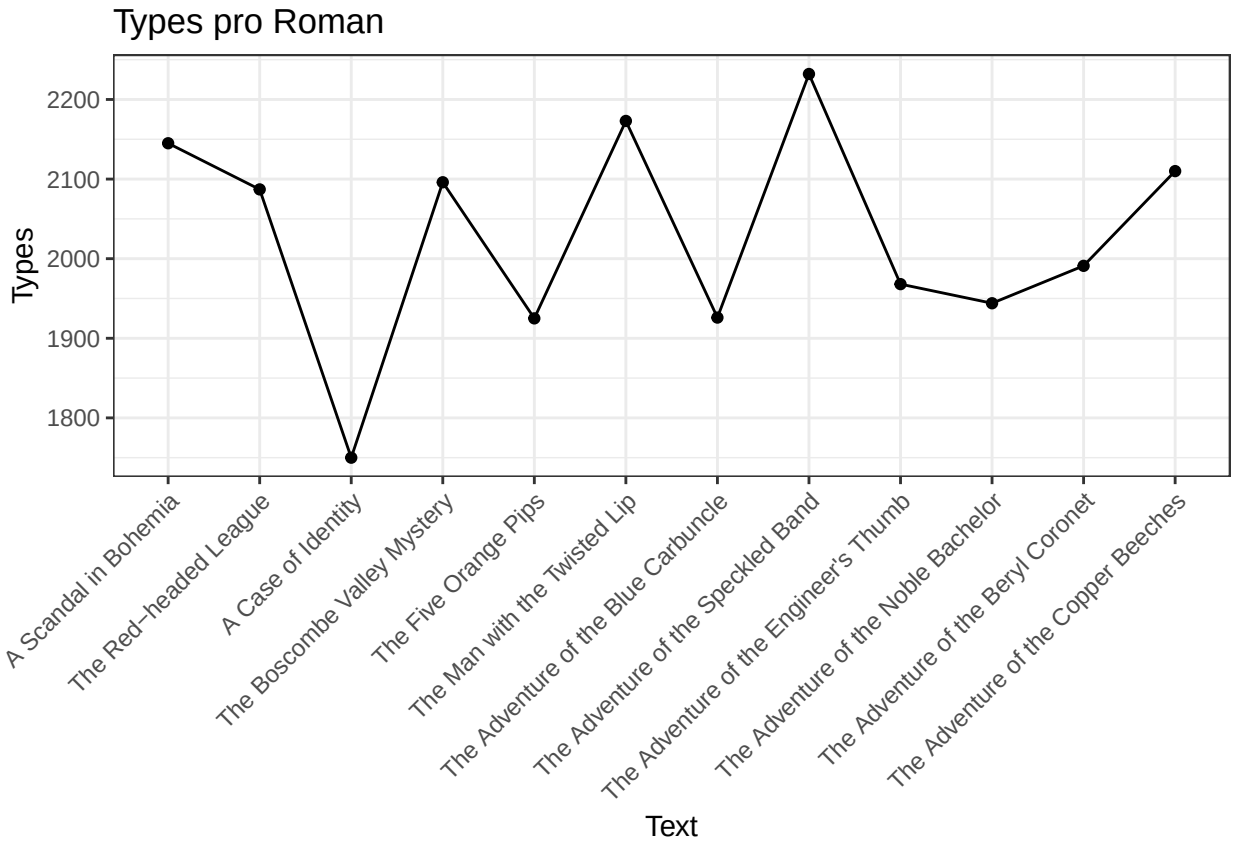
Grundlage solcher Plots sind praktisch immer Data Frame-Objekte (also Tabellen), die Informationen über Korpora, Texte, Wörter, Themen usw. enthalten, welche sich visuell darstellen lassen. Im Rest dieser

Einführung gehe ich nicht im Detail darauf ein, wie die jeweiligen Plots genau konstruiert werden, allerdings lassen sich die meisten Daten auch (etwas weniger ansprechend) mit der R-internen Funktion plot() darstellen. Eine hilfreiche deutschsprachige Einführung in das Plotten mit ggplot2 findet sich hier. Viele der hier vorgestellten Plots stammen zudem direkt aus quanteda (beginnend mit textplot_).

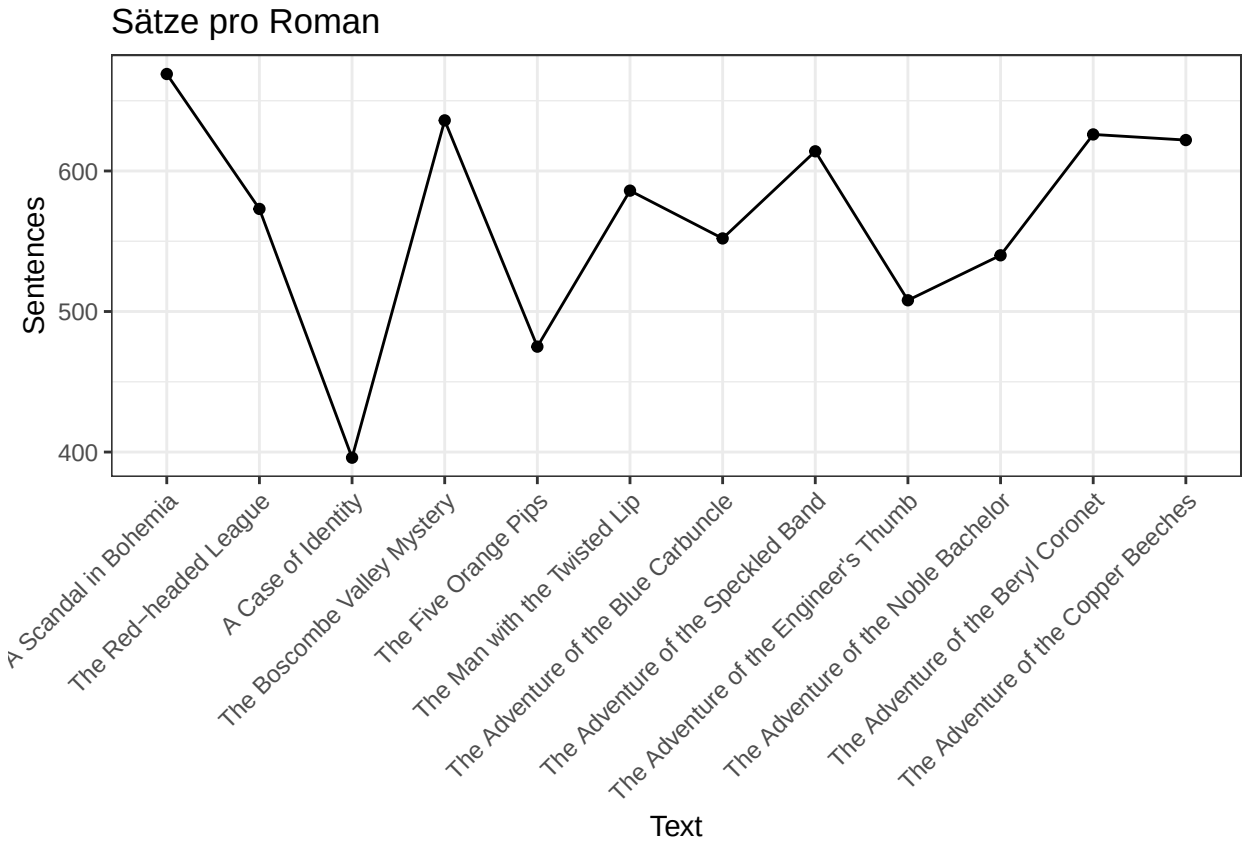
```
ggplot(korpus.stats, aes(Text, Tokens, group = 1)) + geom_line() + geom_point() + theme(axis.text.x = el
```



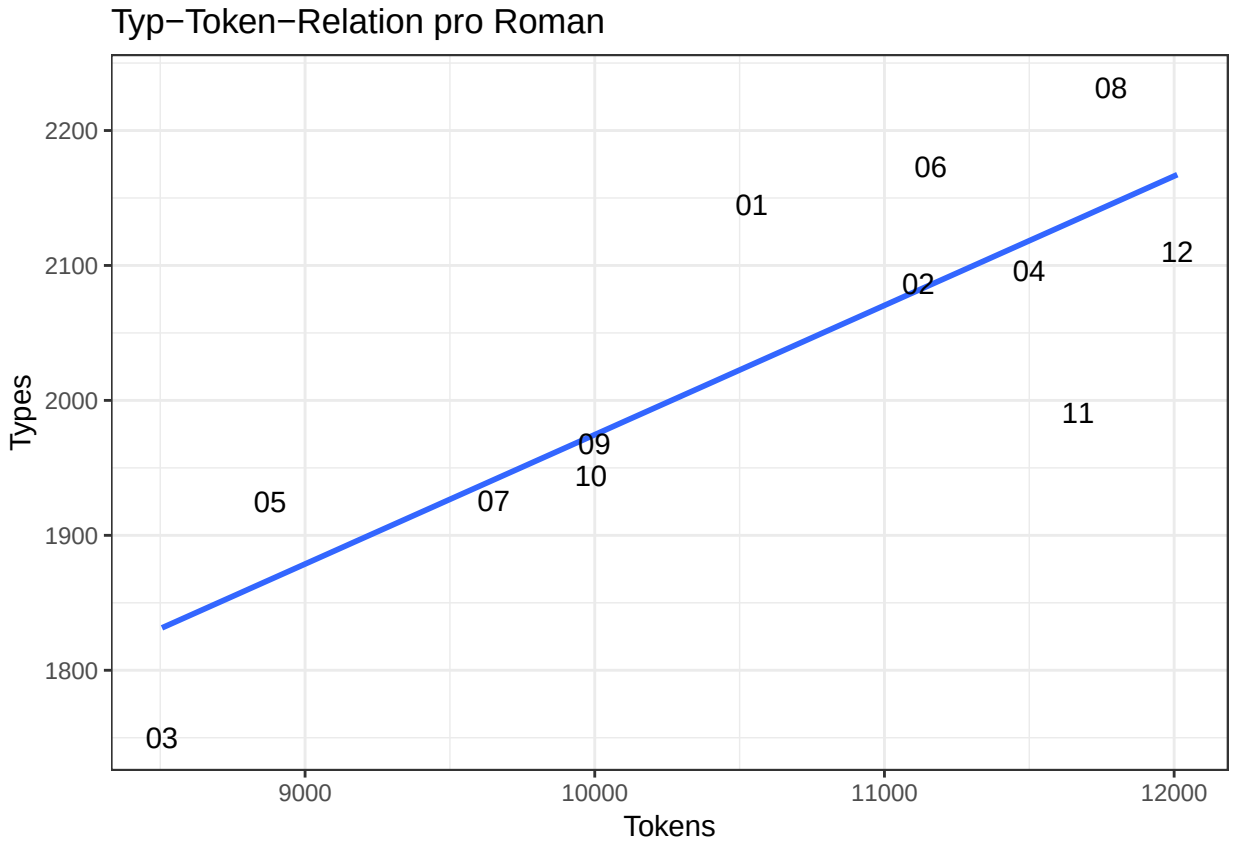

```
ggplot(korpus.stats, aes(Text, Types, group = 1)) + geom_line() + geom_point() + theme(axis.text.x = el
```




```
ggplot(korpus.stats, aes(Text, Sentences, group = 1)) + geom_line() + geom_point() + theme(axis.text.x =
```




```
ggplot(korpus.stats, aes(Tokens, Types, group = 1, label = Textnummer)) + geom_smooth(method = "lm", se
```



Diese Grafiken sind zunächst einmal nicht umwerfend informativ. Sie belegen lediglich, dass die Erzählungen 'A Case of Identity' und (in geringerem Maße) 'The Five Orangen Pips' deutlich kürzer sind als die anderen Texte, was sich auf allen drei Ebenen (Tokens, Types, Sätze) niederschlägt. Etwas interessanter wird es allerdings bei der Typ-Token-Relation: während drei Romane (mit den Nummern 3, 11 und 12) jeweils einen eher unterdurchschnittlichen TTR aufweisen, liegen weitere vier oberhalb der linearen Relation (1, 5, 6 und 8), während die verbleibenden sechs ziemlich genau dem Durchschnitt entsprechen. Über den TTR lassen sich Rückschlüssen über die Informationsdichte ziehen – dazu später noch mehr.

Mit Korpora arbeiten

Korpora lassen sich in quanteda sehr leicht samplen, umformen und mit zusätzlichen Metadaten versehen. Metadaten können wiederum genutzt werden, um das Korpus nach bestimmten Kriterien zu filtern. Der folgenden Aufruf zeit die erste 1.000 Wörter des ersten Romans.

```
str_sub(korpus[1], start = 1, end = 1000) # Anfang des ersten Romans wiedergeben
```

```
## [1] "A Scandal in Bohemia\n\n  To Sherlock Holmes she is always the woman. I have seldom\nheard him m
```

Jeder Text lässt sich also anhand seiner Indizierung aufrufen und auch ändern (etwa korpus[1] für den ersten Text). Gleiches funktioniert auch über die Funktion texts —- der Weg über die den Index ist lediglich die Kurzform von texts(korpus)[1].

Mittels corpus_reshape lässt sich ein Korpus so umformen, dass jeder Satz ein eigenes Dokument ergibt. Alternative Argumente sind "paragraphs" und "documents" (so lässt sich ein Satz-Korpus wieder in seinen Anfangszustand zurückversetzen). Die Erstellung von Satz-Korpora ist für die Sentimentanalyse und das überwachte maschinelle Lernen von Interesse.

Die Beschriftung des Beispiels besteht hier aus der Variable *docname* und einer angehängten Zahl (eine 1 für den ersten Satz).

```
korpus.saetze <- corpus_reshape(korpus, to = "sentences")
korpus.saetze[1]
```

```
##                                                  A Scandal in Bohemia.1
## "A Scandal in Bohemia    To Sherlock Holmes she is always the woman."
```

Mit corpus_sample() kann weiterhin ein zufälliges Sample aus einem Korpus gezogen werden. Wir wenden die Funktion hier auf das Satz-Korpus an.

```
zufallssatz <- corpus_sample(korpus.saetze, size = 1)
zufallssatz[1]
```

```
## A Scandal in Bohemia.233
##        "Holmes laughed."
```

Anhand von corpus_subset kann ein Korpus schließlich nach Metadaten gefiltert werden. Hier geschieht dies mittels der neu erstellten binären Variable *LangerSatz*, die dann TRUE ist, wenn ein Satz >= 25 Tokens enthält). So lässt sich ein Teilkorpus zu bilden, in dem nur längere Sätze enthalten sind. Das Beispiel soll lediglich verdeutlichen, dass mithilfe der von quanteda bereitgestellten Funktionen zahlreiche Schritte für die Bereinigung von Korpora möglich sind.

```
docvars(korpus.saetze, "Zeichenanzahl") <- ntoken(korpus.saetze)
docvars(korpus.saetze, "LangerSatz") <- ntoken(korpus.saetze)>=25
korpus.saetze_lang <- corpus_subset(korpus.saetze, LangerSatz == TRUE)
korpus.saetze_lang[1:3]
```

```
##
##                                          "He was, I take it, the most perfect reasoning and observi
##
## "But for the trained teasoner to admit such intrusions into his own delicate and finely adjusted temp
##
##                                            "Grit in a sensitive instrument, or a crack in one of hi
```

Schließlich lassen sich Korpora mithilfe von corpus_segment() auch nach bestimmten Kriterien aufspalten.

Tokenisierung

Unter Tokensierung versteht man die Aufspaltung eines Textes in laufende Wörter oder sog. N-Gramme, also Sequenzen mehrerer Wörter in Folge. Die Funktion tokens realisiert die Tokenisierung eines Korpus in quanteda. Zusätzlich versteht *tokens* auch unzählige Argumente für die Entfernung bestimmter Features.

```
meine.tokens <- tokens(korpus)
head(meine.tokens$`A Scandal in Bohemia`)
```

```
## [1] "A"        "Scandal"  "in"       "Bohemia"  "To"       "Sherlock"
```

Mittels der Funktion tokens lässt sich der Text über das Argument *ngrams* auch gleich in N-Gramme (Mehrwortsequenzen) aufspalten. Im folgenden Beispiel werden erst Bigramme vom Anfang des ersten Textes angezeigt, und dann alle Sequenzen von einem, zwei oder drei Begriffen extrahiert (durch die Anwendung von head sehen wir nur Trigramme, es sind aber auch kürzere Sequenzen vorhanden).

```
meine.tokens <- tokens(korpus, ngrams = 2)
head(meine.tokens$`A Scandal in Bohemia`)
```

```
## [1] "A_Scandal"       "Scandal_in"      "in_Bohemia"      "Bohemia_To"
## [5] "To_Sherlock"     "Sherlock_Holmes"
```

```
meine.tokens <- tokens(korpus, ngrams = 1:3)
head(meine.tokens$`A Scandal in Bohemia`)
```

```
## [1] "A"        "Scandal"  "in"       "Bohemia"  "To"       "Sherlock"
```

Hilfreich ist auch die Möglichkeit, bei der Tokenisierung bestimmte Begriffe zu entfernen oder zurückzubehalten.

```
meine.tokens <- tokens(korpus)
begriffe.behalten <- tokens_select(meine.tokens, c("holmes", "watson")) # Platzhalter mit padding = TRU
head(begriffe.behalten$`A Scandal in Bohemia`)
```

```
## [1] "Holmes" "Holmes" "Holmes" "Holmes" "Watson" "Watson"
```

```
begriffe.entfernen <- tokens_remove(meine.tokens, c("Sherlock", "in", "is", "the"))
head(begriffe.entfernen$`A Scandal in Bohemia`)
```

```
## [1] "A"       "Scandal" "Bohemia" "To"      "Holmes"  "she"
```

Wie bereits angedeutet akzeptiert die Funktion *tokens* eine Reihe von Argumenten, mit denen ganze Klassen von Zeichenketten (Zahlen, Interpunktion, Symbole usw.) ausgeschlossen oder zurückbehalten werden können. Folgend werden zunächst Zahlen, Interpunktion und Symbole entfernt, dann mittels tokens_tolower alle Wörter in Kleinschreibung umgewandelt und dann dann noch die Wörter "sherlock" und "holmes", sowie eine Reihe englischer Stoppwörter entfernt.

```
meine.tokens <- tokens(korpus, remove_numbers = TRUE, remove_punct = TRUE, remove_symbols = TRUE)
meine.tokens <- tokens_tolower(meine.tokens)
meine.tokens <- tokens_remove(meine.tokens, c(stopwords("english"), "sherlock", "holmes"))
head(meine.tokens$`A Scandal in Bohemia`)
```

```
## [1] "scandal" "bohemia" "always"  "woman"   "seldom"  "heard"
```

Das Resultat ist der Art von Daten, mit denen man bei Verfahren wie der Anwendung von Lexika (Kapitel 2), der Berechnung von Themenmodellen (Kapitel 3), und dem überwachten maschinellen Lernen (Kapitel 4) häufig arbeitet sehr ähnlich. Durch die Stoppwortentfernung und andere Schritte gehen syntaktische Informationen verloren, d.h. man kann nicht mehr nachvollziehen, wer was mit wem tut, oder wie der Text insgesamt argumentativ oder erzählerisch aufgebaut ist. Diese Informationen sind allerdings im "Bag-of-Words-Ansatz", der in der automatisierten Inhaltsanalyse nahezu immer verwendet wird, nicht unbedingt relevant.

Die in diesem Abschnitt beschriebenen Schritte sind zwar im Einzelfall nützlich, werden aber in den folgenden Kapitel praktisch nicht angewandt, weil die Daten dort schon als quanteda–Korpora vorliegen, und weil zudem häufig auch bis auf die Anwendung der Funktion *corpus* keine weiteren Schritte notwendig sind.

Dokument-Feature-Matrizen (DFMs) erstellen

Wir kommen nun zu einer zentralen Datenstruktur von quanteda, die im Gegensatz zu den zuvor vorgestellten Einheiten praktisch in jedem Projekt vorkommt: die Document Feature-Matrize (DFM). Üblicherweise wird direkt nachdem ein Korpus angelegt wurde eine DFM berechnet, zuweilen auch mehrere. Eine DFM ist eine Tabelle, deren Zeilen Texte und deren Spalten Wortfrequenzen enhalten. Dabei gehen Informationen darüber, wo in einem Text ein Wort vorkommt verloren (man spricht auch vom 'Bag-of-Words-Ansatz'). Immer dann, wenn wir uns für die Beziehung von Wörtern zu Texten (und umgekehrt) interessieren, berechnen wir eine DFM.

```
meine.dfm <- dfm(korpus, remove_numbers = TRUE, remove_punct = TRUE, remove_symbols = TRUE, remove = st
meine.dfm
```

```
## Document-feature matrix of: 12 documents, 8,489 features (79.1% sparse).
```

Wichtig: Hier wird implizit der uns schon vertraute Befehl tokens() angewandt, um bestimmte Features zu entfernen. Vieles funktioniert bei DFMs analog zur Erstellung eines Korpus. So zählen die Funktionen ndoc() und nfeat() Dokumente und Features (Wörter).

```
ndoc(meine.dfm)
```

```
## [1] 12
```

```
nfeat(meine.dfm)
```

```
## [1] 8489
```

Mittels der Funktionen docnames() und featnames() lassen sich die Namen der Dokumente und Features ausgeben.

```
head(docnames(meine.dfm)) # In der DFM enthaltene Dokumente
```

```
## [1] "A Scandal in Bohemia"         "The Red-headed League"
## [3] "A Case of Identity"           "The Boscombe Valley Mystery"
## [5] "The Five Orange Pips"         "The Man with the Twisted Lip"
```

```
head(featnames(meine.dfm), 50) # Features in chronologischer Reihenfolge
```

```
##  [1] "scandal"      "bohemia"      "sherlock"     "holmes"
##  [5] "always"       "woman"        "seldom"       "heard"
##  [9] "mention"      "name"         "eyes"         "eclipses"
## [13] "predominates" "whole"        "sex"          "felt"
## [17] "emotion"      "akin"         "love"         "irene"
## [21] "adler"        "emotions"     "one"          "particularly"
## [25] "abhorrent"    "cold"         "precise"      "admirably"
## [29] "balanced"     "mind"         "take"         "perfect"
## [33] "reasoning"    "observing"    "machine"      "world"
## [37] "seen"         "lover"        "placed"       "false"
## [41] "position"     "never"        "spoke"        "softer"
## [45] "passions"     "save"         "gibe"         "sneer"
## [49] "admirable"    "things"
```

Die tabellarische Ansicht illustriert den Inhalt der DFM als Text-Wort-Matrix am besten. Die sparsity ("Spärlichkeit") einer DFM beschreibt dabei den Anteil der leeren Zellen, also Wörter, die nur in sehr wenigen Texten vorkommen. Wie sich leicht ableiten lässt, werden DFMs sehr schnell sehr groß. Zum Glück macht sich quanteda eine Reihe von für den Nutzer unsichtbaren Funktionen aus anderen Paketen zunutze, um diesem Problem zu begegnen.

```
head(meine.dfm, n = 12, nf = 10) # Features und Texte als Matrix in chronologischer Reihenfolge
```

```
## Document-feature matrix of: 12 documents, 10 features (30.8% sparse).
## 12 x 10 sparse Matrix of class "dfm"
##                                      features
## docs                                  scandal bohemia sherlock holmes
##   A Scandal in Bohemia                      4       8       11     47
##   The Red-headed League                     0       0       10     51
##   A Case of Identity                        0       2        7     46
##   The Boscombe Valley Mystery               1       0       10     43
##   The Five Orange Pips                      1       0       10     25
##   The Man with the Twisted Lip              0       0       10     28
##   The Adventure of the Blue Carbuncle       0       0       10     34
##   The Adventure of the Speckled Band        0       0        9     55
##   The Adventure of the Engineer's Thumb     0       0        5     12
##   The Adventure of the Noble Bachelor       1       0        7     34
##   The Adventure of the Beryl Coronet        4       0        3     26
##   The Adventure of the Copper Beeches       0       1        2     42
##                                      features
```

```
## docs                                 always woman seldom heard mention
##   A Scandal in Bohemia                    5    12      3     8       1
##   The Red-headed League                   5     0      0    15       0
##   A Case of Identity                      7    10      0     5       0
##   The Boscombe Valley Mystery             5     1      0    10       0
##   The Five Orange Pips                    5     1      0     5       1
##   The Man with the Twisted Lip            4     5      0     8       0
##   The Adventure of the Blue Carbuncle     5     0      1     3       0
##   The Adventure of the Speckled Band      8     5      1    20       0
##   The Adventure of the Engineer's Thumb   0     6      1    11       0
##   The Adventure of the Noble Bachelor     3     8      0    11       0
##   The Adventure of the Beryl Coronet      3     5      0    12       0
##   The Adventure of the Copper Beeches     7     8      0     5       0
##                                      features
## docs                                  name
##   A Scandal in Bohemia                   6
##   The Red-headed League                  6
##   A Case of Identity                     1
##   The Boscombe Valley Mystery            3
##   The Five Orange Pips                   5
##   The Man with the Twisted Lip           4
##   The Adventure of the Blue Carbuncle   10
##   The Adventure of the Speckled Band     6
##   The Adventure of the Engineer's Thumb  3
##   The Adventure of the Noble Bachelor    6
##   The Adventure of the Beryl Coronet     8
##   The Adventure of the Copper Beeches    4
```

Gleich an den ersten Blick fällt auf, das die Wörter ‘sherlock’ und ‘holmes’ in allen Romanen vorkommen, also sehr wenig distinktiv sind, weshalb wir sie unter Umständen zu den Stoppwörtern für dieses Korpus hinzufügen sollten.

Die Funktion topfeatures() zählt Features in der gesamten DFM aus. Die Funktion textstat_frequency() liefert zusätzlich noch den Rang (rank), die Anzahl der Dokumente, in denen das Feature vorkommt (docfreq) sowie Metadaten, nach denen bei der Zählung gefiltert wurde.

```
topfeatures(meine.dfm) # Features nach Frequenz
```

```
##   said   upon holmes    one    man     mr little    now    see    may
##    485    465    443    372    290    275    269    234    229    197
```

```
worthaeufigkeiten <- textstat_frequency(meine.dfm) # Worthäufigkeiten
head(worthaeufigkeiten)
```

```
##   feature frequency rank docfreq group
## 1    said       485    1      12   all
## 2    upon       465    2      12   all
## 3  holmes       443    3      12   all
## 4     one       372    4      12   all
## 5     man       290    5      12   all
## 6      mr       275    6      12   all
```

Mit DFMs arbeiten

DFMs lassen sich mit dfm_sort leicht nach Dokument- und Feature-Frequenzen sortieren.

```r
head(dfm_sort(meine.dfm, decreasing = TRUE, margin = "both"), n = 12, nf = 10)
```

```
## Document-feature matrix of: 12 documents, 10 features (0% sparse).
## 12 x 10 sparse Matrix of class "dfm"
##                                      features
## docs                                  said upon holmes one man mr little
##   The Adventure of the Speckled Band    44   41     55  33  11  5     17
##   The Adventure of the Copper Beeches   47   33     42  36  34 44     37
##   The Boscombe Valley Mystery           37   42     43  31  41 24     25
##   The Man with the Twisted Lip          28   54     28  36  30 20     21
##   The Adventure of the Beryl Coronet    45   33     26  32  27 20     22
##   The Red-headed League                 51   50     51  29  25 55     25
##   A Scandal in Bohemia                  33   25     47  27  23  9     14
##   The Adventure of the Engineer's Thumb 47   38     12  33  17 11     25
##   The Adventure of the Noble Bachelor   33   29     34  31  10 17     26
##   The Adventure of the Blue Carbuncle   43   38     34  38  37 17     24
##   The Five Orange Pips                  32   47     25  29  19  3      5
##   A Case of Identity                    45   35     46  17  16 50     28
##                                      features
## docs                                  now see may
##   The Adventure of the Speckled Band   21  22  19
##   The Adventure of the Copper Beeches  18  17  21
##   The Boscombe Valley Mystery          16  24  19
##   The Man with the Twisted Lip         27  18  15
##   The Adventure of the Beryl Coronet   29  20  25
##   The Red-headed League                14  23   8
##   A Scandal in Bohemia                 17  15  21
##   The Adventure of the Engineer's Thumb 16  16   9
##   The Adventure of the Noble Bachelor  16  16  18
##   The Adventure of the Blue Carbuncle  33  27   7
##   The Five Orange Pips                 12  16  24
##   A Case of Identity                   15  15  11
```

Weiterhin lassen sich bestimmte Features einer DFM gezielt mittels dfm_select auswählen.

```r
dfm_select(meine.dfm, pattern = "lov*")
```

```
## Document-feature matrix of: 12 documents, 7 features (67.9% sparse).
## 12 x 7 sparse Matrix of class "dfm"
##                                      features
## docs                                  love lover lovely loves loved
##   A Scandal in Bohemia                   5     1      1     1     1
##   The Red-headed League                  1     0      0     0     0
##   A Case of Identity                     2     0      0     0     0
##   The Boscombe Valley Mystery            1     0      1     0     1
##   The Five Orange Pips                   1     0      0     0     0
##   The Man with the Twisted Lip           0     0      0     0     0
##   The Adventure of the Blue Carbuncle    2     0      0     0     0
##   The Adventure of the Speckled Band     1     0      1     0     0
##   The Adventure of the Engineer's Thumb  1     0      0     0     0
##   The Adventure of the Noble Bachelor    1     2      1     0     0
##   The Adventure of the Beryl Coronet     3     4      0     2     3
##   The Adventure of the Copper Beeches    0     0      1     1     0
##                                      features
## docs                                  lovers loving
```

```
##   A Scandal in Bohemia                       0      0
##   The Red-headed League                      0      0
##   A Case of Identity                         1      0
##   The Boscombe Valley Mystery                0      0
##   The Five Orange Pips                       0      0
##   The Man with the Twisted Lip               0      0
##   The Adventure of the Blue Carbuncle        0      0
##   The Adventure of the Speckled Band         0      0
##   The Adventure of the Engineer's Thumb      0      0
##   The Adventure of the Noble Bachelor        0      1
##   The Adventure of the Beryl Coronet         0      2
##   The Adventure of the Copper Beeches        0      0
```

Die Funktion dfm_wordstem() reduziert Wörter auf ihre Stammform. Diese Funktion existiert in quanteda derzeit nur für Englisch und ist auch dort nur begrenzt zuverlässig, was die folgende Ausgabe gut illustriert ('holm' ist kein Wortstamm).

```
meine.dfm.stemmed <- dfm_wordstem(meine.dfm)
topfeatures(meine.dfm.stemmed)
```

```
##  said  upon  holm   one   man    mr littl   see   now  come
##   485   465   460   383   304   275   269   253   234   207
```

Ebenso wie bei Wortfrequenzen in Korpora ist die Gewichtung einer DFM nach relativen Wortfrequenzen und Verfahren wie TF-IDF oftmals sinnvoll. Die Gewichtung einer DFM funktioniert immer aufgrund der Wort-Text-Relation, weshalb topfeatures() in Kombination mit dfm_weight() merkwürdige Resultate produziert. Relative Frequenzen und TF-IDF sind nur kontrastiv innerhalb der Text in einem Korpus sinnvoll (hier für 'A Scandal in Bohemia'), da für das gesamte Korpus relative Frequenz == absolute Frequenz

```
meine.dfm.proportional <- dfm_weight(meine.dfm, scheme = "prop")
topfeatures(meine.dfm) # absolute Frequenzen für das gesamte Korpus
```

```
##   said   upon holmes    one    man     mr little    now    see    may
##    485    465    443    372    290    275    269    234    229    197
```

```
topfeatures(meine.dfm.proportional) # ...ergibt wenig Sinn
```

```
##       said       upon     holmes        one        man         mr
## 0.12564554 0.12042666 0.11388128 0.09559304 0.07426168 0.07127181
##     little        now        see        may
## 0.06916820 0.06024412 0.05900373 0.05064617
```

```
topfeatures(meine.dfm.proportional[1,]) # ...ergibt mehr Sinn
```

```
##      holmes        said         one        upon         man         may
## 0.012339197 0.008663691 0.007088475 0.006563402 0.006038330 0.005513258
##  photograph      street        know         now
## 0.004988186 0.004725650 0.004725650 0.004463114
```

Im zweiten Beispiel sehen wir etwa, dass 'A Scandal in Bohemia' einen leicht höheren Anteil von Nennungen der Wortes 'holmes' hat, als dies im Gesamtkorpus der Fall ist. Dazu später noch etwas mehr.

Die Gewichtungsansätze Propmax und TF-IDF liefern relevante Wortmetriken, zum Beispiel für die Bestimmung von Stoppwörtern. Propmax skaliert die Worthäufigkeit relativ zum frequentesten Wort (hier 'holmes'). Funktional ähneln sich TF-IDF und der später vorgestellte Keyness-Ansatz – beide finden besonders distinktive Terme.

```
meine.dfm.propmax <- dfm_weight(meine.dfm, scheme = "propmax")
topfeatures(meine.dfm.propmax[1,])
```

```
##     holmes       said        one       upon        man        may
##  1.0000000  0.7021277  0.5744681  0.5319149  0.4893617  0.4468085
## photograph     street       know        now
##  0.4042553  0.3829787  0.3829787  0.3617021
```

```
meine.dfm.tfidf <- dfm_tfidf(meine.dfm)
topfeatures(meine.dfm.tfidf)
```

```
##    simon rucastle mccarthy  coronet lestrade   hosmer    clair        k
## 42.08807 36.69216 34.53380 29.13789 28.01345 24.82117 24.82117 22.66281
##   hunter   wilson
## 22.66281 21.58362
```

Schließlich lässt sich mit dfm_trim() noch eine reduzierten Dokument-Feature-Matrix erstellen. Das ist dann sinnvoll, wenn man davon ausgeht, dass beispielsweise nur solche Begriffe eine Rolle spielen, die mindestes X mal im Gesamtkorpus vorkommen. Auch eine Mindestzahl oder ein Maximum an Dokumenten, in denen ein Begriff vorkommen muss oder darf, kann bestimmt werden. Schließlich lassen sich beide Filteroptionen auch proportional anwenden (vgl. Beispiel).

```
meine.dfm.trim <- dfm_trim(meine.dfm, min_docfreq = 11) # Features, die mindestens in 11 Romanen vorkom
head(meine.dfm.trim, n = 12, nf = 10)
```

```
## Document-feature matrix of: 12 documents, 10 features (2.5% sparse).
## 12 x 10 sparse Matrix of class "dfm"
##                                      features
## docs                                  sherlock holmes always heard name
##   A Scandal in Bohemia                      11     47      5     8    6
##   The Red-headed League                     10     51      5    15    6
##   A Case of Identity                         7     46      7     5    1
##   The Boscombe Valley Mystery               10     43      5    10    3
##   The Five Orange Pips                      10     25      5     5    5
##   The Man with the Twisted Lip              10     28      4     8    4
##   The Adventure of the Blue Carbuncle       10     34      5     3   10
##   The Adventure of the Speckled Band         9     55      8    20    6
##   The Adventure of the Engineer's Thumb      5     12      0    11    3
##   The Adventure of the Noble Bachelor        7     34      3    11    6
##   The Adventure of the Beryl Coronet         3     26      3    12    8
##   The Adventure of the Copper Beeches        2     42      7     5    4
##                                      features
## docs                                  eyes whole felt one cold
##   A Scandal in Bohemia                   9     4    2  27    2
##   The Red-headed League                 10     9    4  29    1
##   A Case of Identity                     6     3    3  17    1
##   The Boscombe Valley Mystery            6     2    0  31    1
##   The Five Orange Pips                   5     2    2  29    1
##   The Man with the Twisted Lip          11     3    2  36    2
##   The Adventure of the Blue Carbuncle    2     0    4  38    5
##   The Adventure of the Speckled Band    11     1    2  33    3
##   The Adventure of the Engineer's Thumb  4     4    4  33    1
##   The Adventure of the Noble Bachelor    4     4    3  31    2
##   The Adventure of the Beryl Coronet    10     6    4  32    1
##   The Adventure of the Copper Beeches    9     7    2  36    1
```

```
meine.dfm.trim <- dfm_trim(meine.dfm, min_termfreq = 0.95, termfreq_type = "quantile") # Features im 95
head(meine.dfm.trim, n = 12, nf = 10)
```

```
## Document-feature matrix of: 12 documents, 10 features (4.17% sparse).
```

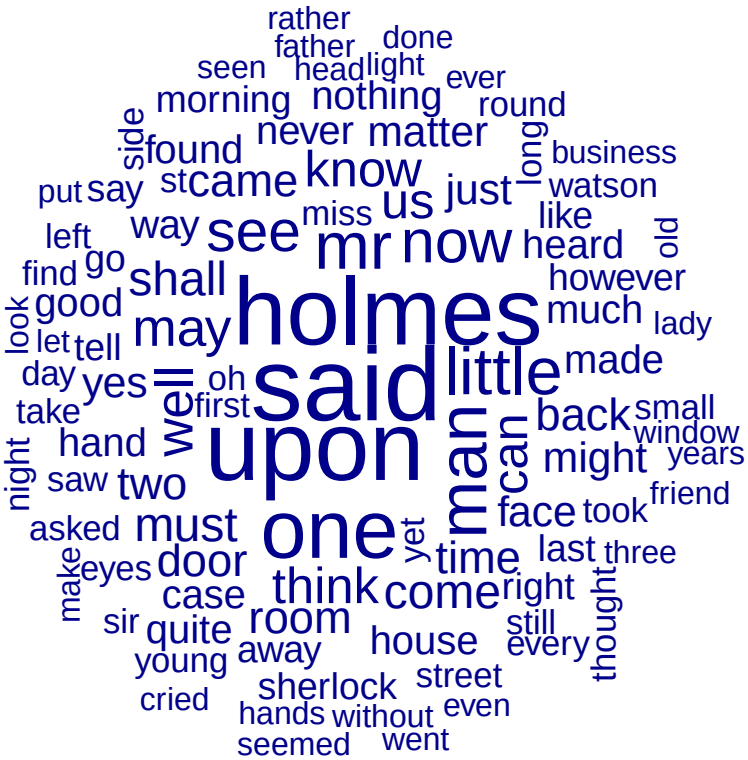
```
## 12 x 10 sparse Matrix of class "dfm"
##                                     features
## docs                                 sherlock holmes always woman heard
##   A Scandal in Bohemia                     11     47      5    12     8
##   The Red-headed League                    10     51      5     0    15
##   A Case of Identity                        7     46      7    10     5
##   The Boscombe Valley Mystery              10     43      5     1    10
##   The Five Orange Pips                     10     25      5     1     5
##   The Man with the Twisted Lip             10     28      4     5     8
##   The Adventure of the Blue Carbuncle      10     34      5     0     3
##   The Adventure of the Speckled Band        9     55      8     5    20
##   The Adventure of the Engineer's Thumb     5     12      0     6    11
##   The Adventure of the Noble Bachelor       7     34      3     8    11
##   The Adventure of the Beryl Coronet        3     26      3     5    12
##   The Adventure of the Copper Beeches       2     42      7     8     5
##                                     features
## docs                                 name eyes whole felt one
##   A Scandal in Bohemia                  6    9     4    2  27
##   The Red-headed League                 6   10     9    4  29
##   A Case of Identity                    1    6     3    3  17
##   The Boscombe Valley Mystery           3    6     2    0  31
##   The Five Orange Pips                  5    5     2    2  29
##   The Man with the Twisted Lip          4   11     3    2  36
##   The Adventure of the Blue Carbuncle  10    2     0    4  38
##   The Adventure of the Speckled Band    6   11     1    2  33
##   The Adventure of the Engineer's Thumb 3    4     4    4  33
##   The Adventure of the Noble Bachelor   6    4     4    3  31
##   The Adventure of the Beryl Coronet    8   10     6    4  32
##   The Adventure of the Copper Beeches   4    9     7    2  36
```

DFMs visualisieren

DFMs lassen sich u.a. auch als Wortwolke der häufigsten Begriffe darstellen.

```
textplot_wordcloud(meine.dfm, max_words = 100, scale = c(5,1))
```

```
## Warning: scale is deprecated; use min_size and max_size instead
```



Interessanter als die Darstellung des Gesamtkorpus ist auch hier der Vergleich. Das folgende Plot zeigt die distinktivsten Begriffe nach TF-IDF für vier Romane, wobei die Farbe den jeweiligen Roman kennzeichnet. Dass im Plot die Wortgröße nicht die absolute Frequenz anzeigt, sondern den TF-IDF-Wert, macht ein solches Plot für den unmittelbaren Vergleich nützlich.

```
textplot_wordcloud(meine.dfm.tfidf[1:4,], color = brewer.pal(4, "Set1"), comparison = T)
```

```
## Warning in dfm_trim.dfm(x, min_termfreq = min_count): dfm has been
## previously weighted
```