Plotting with R

Martin Schweinberger

December 29, 2017

This tutorial exemplifies how to use R to visualize, i.e. to plot data.

1 Line graphs

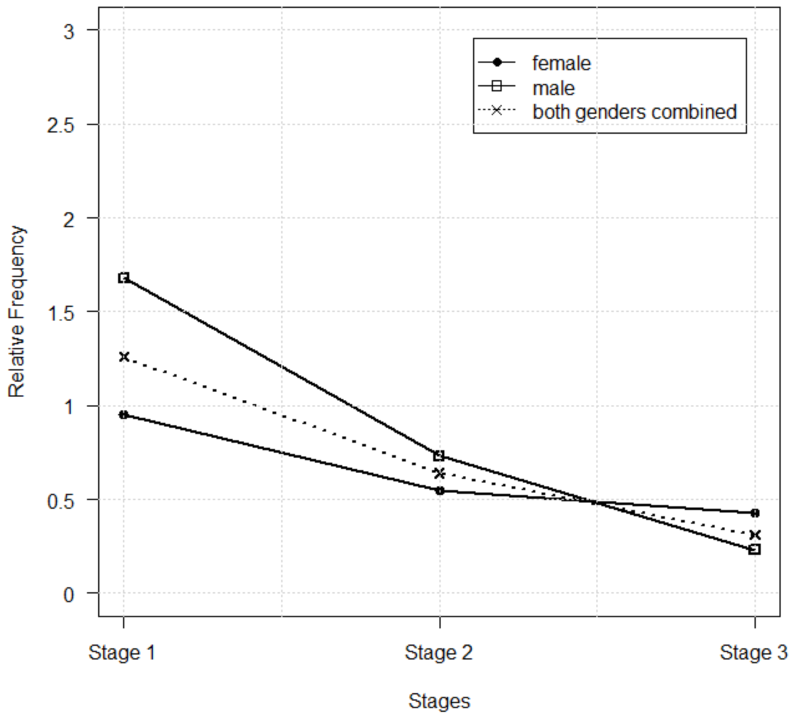
R is extremely versatile when it comes to plotting data but it can be troublesome to use R for visualizations - particularly when you are not yet as used to R. In the following I will show you how to set up a line graph with three lines representing different mean frequencies of three groups during three stages of a process. I frequently use line plots and as a colleague struggled to set one up in R I thought that including an example may be of interest for some of you.

Creating a line graph is pretty straight forward and rather easy. However, you have a lot of options that can and sometimes have to specify - you can find a very nice and VERY handy overview of the graphical parameters you can customize at `http://www.statmethods.net/advgraphs/parameters.html`. We are going to set this graph up in three steps: first, we plot one vector, then we plot the other two vectors, and , finally, we will define and plot the axes. Have a look at the code below to see how it can be done.

```
# Remove all lists from the current workspace
rm(list=ls(all=T))
# create vectors with some data points
male <- c(1.6829871, 0.7339867, 0.2307310)
female <- c(0.9524596, 0.5481039, 0.4289967)
combined <- c(1.2639248, 0.6427771, 0.3125549)
# round vectors so they look nicer in the plot
male <- round(male, 2)
female <- round(female, 2)
combined <- round(combined, 2)
# setting up line graph
```

```r
plot(female, # plot the vector called "female"
  type = "o", # use plot type "o" (overplotted points and lines)
  lwd = 2, # use double line width (4 for 4 times the normal line width)
  lty = 1, # use line type 1 (striaght line)
  pch = 19, #  use point character 19 (filled circle)
  ylim = c(0, 3), # the y-axis should be drawn from 0 to 3
  ylab = "Relative Frequency", # the y-axis label should read "Relative Frequency"
  xlab = "Stages",  # the x-axis label should read "Stages"
  axes = F, # don't draw axes yet!
  cex = 1) # all writing should be of normal size (.5 for half size)
# add aline for the data points in vector male (pch = 0: use empty squares as point characters)
lines(male, type = "o", lwd = 2,  lty = 1, pch = 0,  cex = 1)
# add aline for the data points in vector combined
# (lty = 3: use a dotted line instead of a straight line and
# pch = 4: use x marks as point characters)
lines(combined, type = "o", lwd = 2,  lty = 3, pch = 4,  cex = 1)
# add x-axes with specified labels at specified intervals
axis(1, at = 0:4, lab = c("", "Stage 1", "Stage 2", "Stage 3", ""))
# add y-axes with specified labels at specified intervals
axis(2, at = seq(0, 3, .5), las = 1, lab = seq(0, 3, .5))
# create a legend
# define vector with linetypes
linetype <- c(1, 1, 3)
# define vector with point charaters
plotchar <- c(19, 0, 4)
# set up legend
legend("topright", inset = .05, c("female", "male", "both genders combined"),
  horiz = F,  pch = plotchar, lty = linetype)
# create a box around the plot
box()
# create a grid in the plot
grid()
```

Below is what the code produces.



2 Scatter plots and histograms

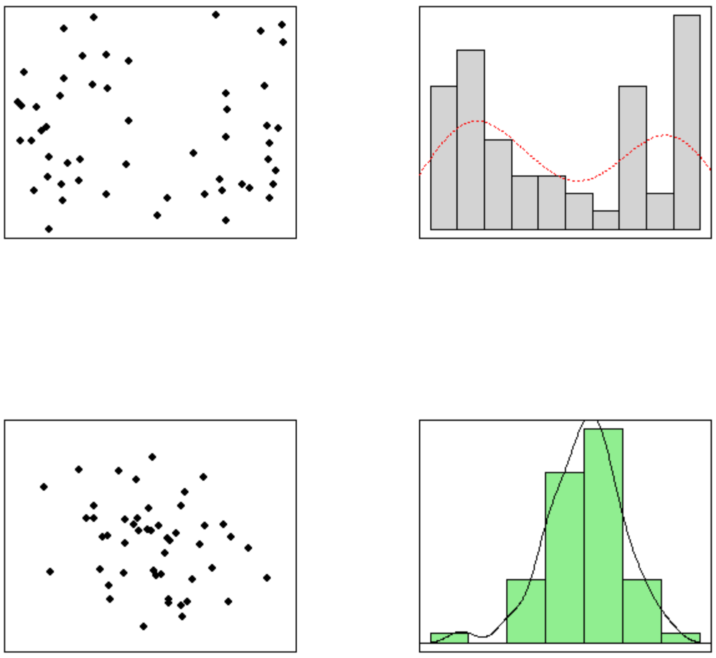
A friend asked me to write some code which produces a bimodal and a monomodal distribution and I thought someone might find the code useful.

My friend needed two plots without annotation: one showing a scatterplot with a cloud of dots in the center of the coordinate system and another plot with two clusters of dots that are close to the extremes of the x-axis. I wrote some code showing these two graphs but have continued to dabble with the code to also include histograms with their density function plotted as a line onto the bars. So, here is the code and the plots it produces.

```
### --- Prepare data
# Remove all lists from the current workspace
rm(list=ls(all=T))

# set graphic's parameters: four plots in one window
par(mfrow = c(2, 2))

# generate data: bimodal distribution
```

```
x1a <- rnorm(50, 0, 2) # create vector x1a with length 50, mean = 0 and sd = 2
x1b <- rnorm(50, 10, 2) # create vector x1b with length 50, mean = 10 and sd = 2
y1 <- rnorm(100, 20, 20) # create vector y1 with length 100, mean = 20 and sd = 20
# combine vectors into a table & convert table into a data frame
tb1 <- as.data.frame(cbind(c(x1a, x1b), y1))
colnames(tb1) <- c("x.coordinate", "y.coordinate") # add column names
tb1 <- tb1[tb1[, 1] >= 0 & tb1[, 1] <=10, ] # remove rows where x is smaller or equal to 0 greater or eqal to 10

# generate scatterplot
plot(tb1, # plot data in tb1
  xlim = c(0, 10), # x axis range (from 0 to 10)
  axes = F, # do not plot the axis
  xlab = "", # do not add an x-axis label
  ylab = "", # do not add an y-axis label
  pch = 19) # add filled dots (instead of empty dots)
box() # plot a box around the distribution

# histogram of the first column of tb1
hist(tb1[, 1],
  xlim = c(0, 10),
  axes = F,
  xlab = "",
  ylab = "",
  breaks = 10,
  main = "",
  col = "lightgrey",
  freq = F)
box()
tb1.d <- density(tb1[, 1])
lines(tb1.d, col = "red", lty = 3)

# generate data: monomodal distribution
x2 <- rnorm(50, 10, 2)
y2 <- rnorm(50, 10, 2)
# combine vectors into a table & convert table into a data frame
tb2 <- as.data.frame(cbind(x2, y2))
colnames(tb2) <- c("x.coordinate", "y.coordinate") # add column names

# generate scatterplot of tb2
plot(tb2, xlim = c(5, 15), ylim = c(5, 15), axes = F, xlab = "", ylab = "", pch = 19)
```

```r
box()

# generate histogram of tb2
hist(tb2$x.coordinate, axes = F, xlab = "", ylab = "", main =
    "", col = "lightgreen", freq = F)
box()
tb2.d <- density(tb2$x.coordinate)
lines(tb2.d, col = "black", lty = 1, lwd = 1)

# restore original graphic's parameters
par(mfrow = c(1, 1))
```



3 Boxplots

Probably my favorite way to display data are boxplots. Boxplots are used if you want to display one numeric vector or when you have a categorical and a numeric variable, e.g. you are looking at reaction times cross different groups are frequencies across the sex and age. The advantage over other displays lies in the fact that boxplots show aspects of the underlying distribution and also allows statistical inferences directly from the display. `Quick R` offers a very nice introduction to boxplots and I highly recommend you have a look at the link.

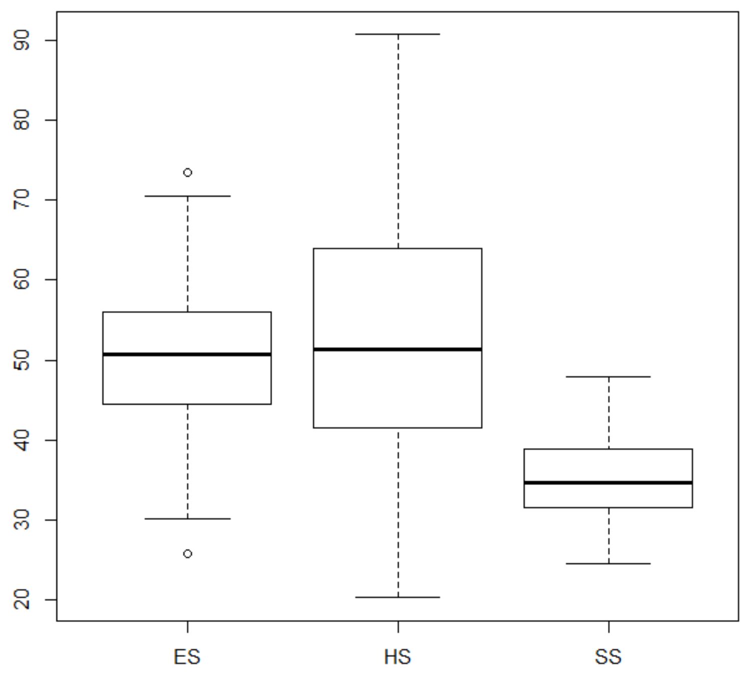
The example I chose is very complex but you can easily adapt it to your needs and delete code which produces things you don't want or need. In fact, like always with R, there are a lot of options that can specify - simply modify the code to match your needs. But let's start and set up the boxplots: In a first step, we are going to generate some data and set up a data frame called *df*:

```r
# Remove all lists from the current workspace
rm(list=ls(all=T))
# set up fictitious data
ES <- rnorm(100, 50, 10)
HS <- rnorm(100, 50, 15)
SS <- rnorm(100, 35, 5)
duration <- c(ES, HS, SS)
speakers <- c(rep("ES", 100), rep("HS", 100), rep("SS", 100))
df <- data.frame(speakers, duration)
df[, 2] <- as.numeric(df[, 2])
# inspect data
head(df)

# and this is what the first rows of the data frame look like
    :

#>  speakers duration
#>1        ES 58.58587
#>2        ES 45.10878
#>3        ES 70.49455
#>4        ES 51.82427
#>5        ES 51.55624
#>6        ES 57.09725
```

In a next step, we are going to create the simplest boxplot possible (it doesn't look very fancy yet, but we are going to customize it later on...) The function we use to set up a boxplot is simply called "boxplot" and it takes the variables to be plotted and the data set as mandatory arguments.

```r
# set up a first simple box plot
boxplot(duration ~ speakers, data = df)
```

Here is our first (very hmm let's say basic) boxplot:

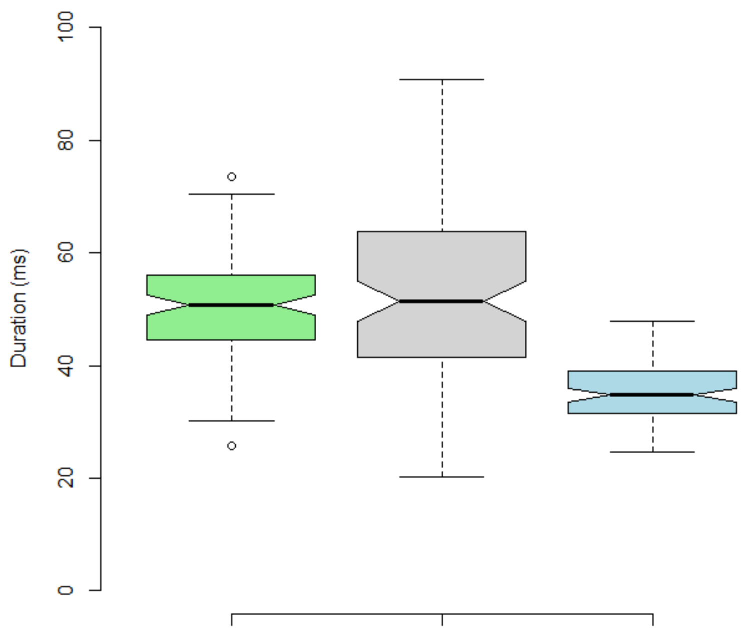


After having created a first very simple boxplot, we are going to customize it and make it look much nicer. To do so, we are going to make use of the in-build arguments that can be used to specify features of our boxplot. Something that is not really necessary but which allows you to specify and customize axes is to not draw them at first, but draw them separately from the plot - and this is exactly, what we are going to do now:

```
# set up a nicer box plot
boxplot(duration ~ speakers,
  data = df, # the data we want to display
  main = "", # you could specify a title here
  ylab = "Duration (ms)", # label of the y-axis
  ylim = c(0, 100), # label of the x-axis
  axes = F, # do not draw axes yet
  notch = T, # include notches
  col = c("lightgreen", "lightgrey", "lightblue")) # create
      boxplots with different colors

# now, we create the x-axis
```

```r
axis(1, # set up the x-axis (1 = x, 2 = y)
  at = 1:3, # we specify the locations where we want the
      tickmarks
  labels = c("", "", ""), # you could specify the text here
  lty = 1, # we define the linetype (1 = straight line)
  col = "black", # the tickmarks should be black
  las = .8) # the font size should be 80% of the normal size

# we now set up the y-axis
axis(2, # set up y-axis
  at = c(0, 20, 40, 60, 80, 100), # create tick marks at the
      specified locations
  labels= c("0", "20", "40", "60", "80", "100"), #create text
       at the specified locations
  lty = 1, # we define the linetype (1 = straight line)
  col = "black", # the tickmarks should be black
  las = .8) # the font size should be 80% of the normal size
```

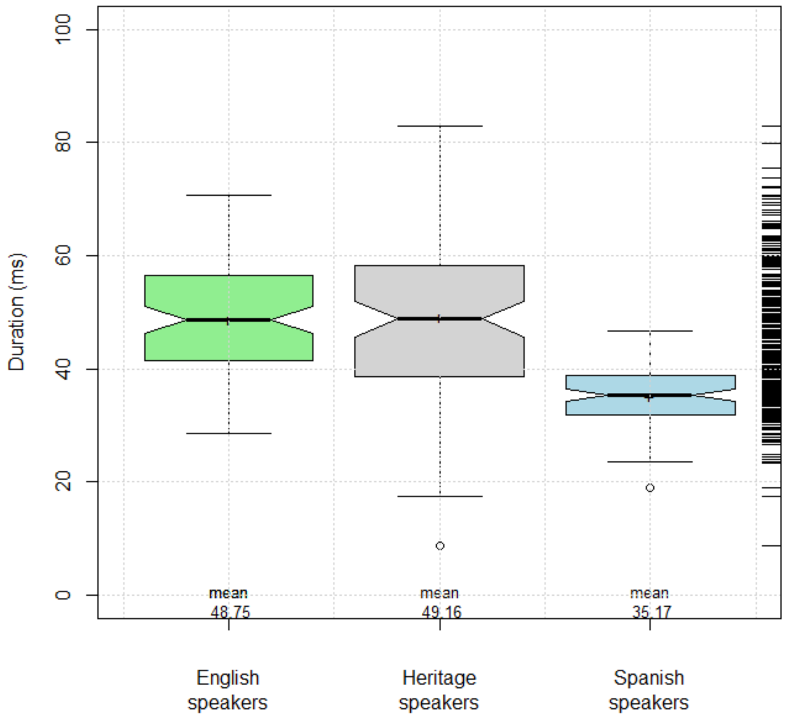
Here is our customized boxplot:



Now, we are goign to finish off our customized boxplot by including +-symbols at the location of the means and also add text which provides the values of the means for each group.

```r
mtext(c("Group 1", "Group 2", "Group 3"), # create specified text
  side = 1, # put text along the x-axis
  line = 3, # place text at the 3rd line of the x-axis
  at = 1:3) # put text at location 1 to 3

text(1:3,
  c(as.vector(by(df$duration, df$speakers, mean))[1],
  as.vector(by(df$duration, df$speakers, mean))[2],
  as.vector(by(df$duration, df$speakers, mean))[3]),
  "+")

text(1:3,
  c(-1.0, -1.0, -1.0, -1.0),
  cex = 0.85,
  labels = paste("mean\n",
  c(round(as.vector(by(df$duration, df$speakers, mean))[1], 2),
    round(as.vector(by(df$duration, df$speakers, mean))[2], 2),
    round(as.vector(by(df$duration, df$speakers, mean))[3], 2),
    sep = "")))
rug(jitter(df$duration),
  side=4)
grid()
box()
```

Below is what the code produces.



4 Word clouds

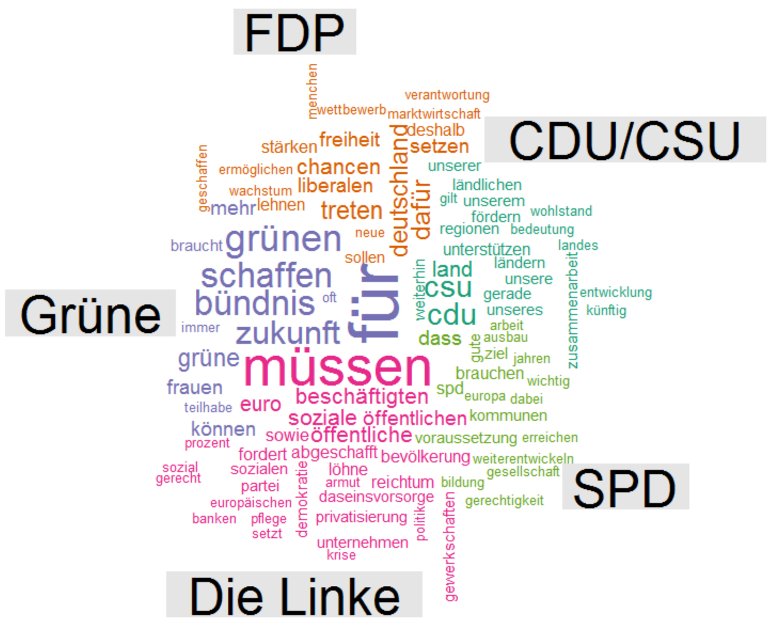
In this post I want to exemplify how to create word clouds in R.

Word clouds visualize word frequencies of either single corpora or they visualize different corpora. Although word clouds are not really used in academic linguistics, they are a neat way to display the themes - which may be thought of as the semantic content - of corpora. To exemplify how to use word clouds, we are going to have a look at the election programs (Wahlkampfprogramme) of German political parties for the Bundestag elections 2013.

```r
### --- Prepare data
# Remove all lists from the current workspace
rm(list=ls(all=T))

# Install packages we need or which may be useful
# (to activate just delete the #)
#install.packages("tm")
#install.packages("wordcloud")
#install.packages("Rstem")
#install.packages("stringr")
```

```
#install.packages("SnowballC")

# Initiate the packages
library(tm)
library(wordcloud)
library(Rstem)
library(stringr)
library(SnowballC)

# Read in data
corp  &lt;- Corpus(DirSource("C:\\Corpora\\original versions
    \\Wahlkampfprogramme Bundestagswahl 2013\\corpus"),
    readerControl = list(language = "german")) #specifies the
    exact folder where my text file(s) is for analysis with tm
    .
##
    ###########################################################

corp <- Corpus(VectorSource(corp)) # Create a corpus from the
     vectors
#corp &lt;- tm_map(corp, stemDocument, language = "german") #
     stem words (inactive because I want intakt words)
corp <- tm_map(corp, removePunctuation) # remove punctuation
corp <- tm_map(corp, tolower) # convert all words to lower
    case
corp <- tm_map(corp, removeNumbers) # remove all numerals
corp <- tm_map(corp, function(x)removeWords(x, stopwords("
    german"))) # remove grammatical words such as "ein", "ist
    ", "war", etc.

# clean corpus content
corp <- sapply(corp, function(x) {
  x <-  gsub("ue", "\{"u}", x)
  x <-  gsub("a\{"u}n", "auen", x)
  x <-  gsub("e\{"u}n", "euen", x)
  x <-  gsub("e\{"u}", "eue", x)
  x <-  gsub("oe", "\{"o}", x)
  x <-  gsub("ae", "\{"a}", x)  }  )

corp <- Corpus(VectorSource(corp))  # convert vectors back
    into a corpus

# Create a term document matrix
term.matrix <- TermDocumentMatrix(corp)  # crate a term
    document matrix
term.matrix <- removeSparseTerms(term.matrix, 0.5) # remove
    infrequent words
term.matrix <- as.matrix(term.matrix)
```

```
colnames(term.matrix) <- c("CDU/CSU", "FDP", "Gr\{"u}ne", "
   Die Linke", "SPD") # add column labels to tdm
# clean row names

# normalize absolute frequencies: convert absolute
   frequencies
# to relative freqeuncies (per 1,000 words)
#colSums(term.matrix)
term.matrix[, 1] <- as.vector(unlist(sapply(term.matrix[, 1],
    function(x) round(x/colSums(term.matrix)[1]*1000, 0) )))
term.matrix[, 2] <- as.vector(unlist(sapply(term.matrix[, 2],
    function(x) round(x/colSums(term.matrix)[2]*1000, 0) )))
term.matrix[, 3] <- as.vector(unlist(sapply(term.matrix[, 3],
    function(x) round(x/colSums(term.matrix)[3]*1000, 0) )))
term.matrix[, 4] <- as.vector(unlist(sapply(term.matrix[, 4],
    function(x) round(x/colSums(term.matrix)[4]*1000, 0) )))
term.matrix[, 5] <- as.vector(unlist(sapply(term.matrix[, 5],
    function(x) round(x/colSums(term.matrix)[5]*1000, 0) )))
#colSums(term.matrix)

# Create word clouds
#wordcloud(corp, max.words = 100, colors = brewer.pal(6, "
   Dark2"), random.order = FALSE)
comparison.cloud(term.matrix, max.words = 100, random.order =
    FALSE, colors = brewer.pal(8, "Dark2"))
#commonality.cloud(term.matrix, max.words = 100, random.order
    = FALSE)
```



The plot above shows a comparative word cloud which highlights distinctive words in the election programs of German political parties for the Bundestag election 2013.

At first I thought that word clouds are simply a fancy but not very helpful way to inspect language data but I have to admit that word clouds really surprised me as they do appear to possess potential to offer an idea of what groups of people are talking about.

The comparative word cloud shows that the FDP stresses concepts like "wettbewerb", "freiheit", "chancen", "liberal" thereby stressing their liberal outlook (they didn't make it and didn't deserve it by the way - just my opinion). Die Grünen support every nonsense as they are "für" everything and relied more on emphasizing "frauen", "zukunft", and "teilhabe" which is in line with their feel-good philosophy. Die Linke rallied on about what we has to be done ("müssen"), and used words like "sozial", "beschäftigten", and "öffentlich" a lot showing their emphasis on economic issues. The social democrats (SPD) addressed topics like "kommunen", "arbeit", "gesellschaft", "bildung", and "gerechtigkeit" - so they essentially used their typical buzz words (just sayin'). Finally, the CDU/CSU mentioned "ländlich", "wohlstand", "unser*, "and "weiterhin" to suggest that they will just continue with whatever nothing that hve been doing over the past years.

In conclusion, I honestly didn't think that I would get meaningful results but the comparative word cloud does a rather good job at that. So that was it on word clouds in R.

References http://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf