Excersice Sheet 11

Block 1: Develop a Naive Bayesian classifier that can detect spam SMS. The learning record contains the text and the label for each SMS: Spam SMS are marked as `spam` and normal SMS as `ham`. The record is to be converted into a Document-Term Matrix[1], which serves as input for the Naive Bayes classifier.

1. Determine the number of `spam` and `ham` messages in the record. Perform a word tokenization[2]. For example, you can use `tidytext::unnest_tokens()`. Convert all uppercase letters to lowercase letters and remove punctuation marks like ".", "," and ";". Remove stop words like "and", "of" and "or" from the SMS text. You can use stop dictionaries like `tidytext::stop_words` or `tm::stopwords()`.
2. Identify the 10 most common words for Spam and Ham SMS. Remove words that occur less than 2 times in total in all SMS. Create a Document-Term Matrix. The rows of the matrix correspond to the SMS and the columns correspond to all words that occur in all SMS. Each value in the matrix indicates whether a particular word occurs in a particular SMS (`TRUE`/`FALSE`).
3. Divide the data set into a training and a test quantity in the ratio 70%:30%. Make sure that the distribution of `spam` and `ham` is approximately the same in both quantities. Use `set.seed()` for reproducibility. Learn a Naive Bayes classifier on the training set, e.g. with `e1071:naiveBayes()`. Use the learned model to predict spam in the test set. Create a Confusion Matrix and calculate Accuracy, Sensitivity and Specificity. Calculate the improvement or deterioration in accuracy, sensitivity and specificity of the model compared to a simpler classifier that would always predict the majority class (`ham`) for each SMS.

```r
sms_raw <- read_csv(str_c(getwd(), "/spam.csv"))
head(sms_raw)
```

```
## # A tibble: 6 x 2
##   type  text
##   <chr> <chr>
## 1 ham   Go until jurong point, crazy.. Available only in bugis n great wor~
## 2 ham   Ok lar... Joking wif u oni...
## 3 spam  Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005~
## 4 ham   U dun say so early hor... U c already then say...
## 5 ham   Nah I don't think he goes to usf, he lives around here though
## 6 spam  "FreeMsg Hey there darling it's been 3 week's now and no word back~
```

```r
table(sms_raw$type)
```

```
##
##  ham spam
## 4825  747
```

```r
library(tidytext)
data("stop_words")
glimpse(stop_words)
```

```
## Observations: 1,149
## Variables: 2
```

```
## $ word    <chr> "a", "a's", "able", "about", "above", "according", "ac...
## $ lexicon <chr> "SMART", "SMART", "SMART", "SMART", "SMART", "SMART", ...
```

```
sms_cleaned <- sms_raw %>%
  mutate(.id = row_number()) %>% #.id identifies SMS, needed for word tokenization
  unnest_tokens(word, text, to_lower = TRUE) %>% # automatic lowercase conversion and punctuati
  anti_join(stop_words %>% select(word), by = "word") %>% # Remove stop words
  mutate(numb = as.numeric(word)) %>% # numbers are removed
  filter(is.na(numb)) %>%
  select(-numb)
```

```
sms_cleaned %>%
  group_by(type, word) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  arrange(-n) %>%
  group_by(type) %>%
  filter(nchar(word) > 2) %>% # Minimum word length of 3
  slice(1:10) %>%
  ungroup()
```

```
## # A tibble: 20 x 3
##    type  word       n
##    <chr> <chr>  <int>
##  1 ham   call     231
##  2 ham   day      200
##  3 ham   time     198
##  4 ham   love     191
##  5 ham   lor      162
##  6 ham   home     161
##  7 ham   dont     129
##  8 ham   send     125
##  9 ham   pls      115
## 10 ham   night    108
## 11 spam  call     355
## 12 spam  free     223
## 13 spam  txt      160
## 14 spam  mobile   127
## 15 spam  text     125
## 16 spam  stop     121
## 17 spam  claim    113
## 18 spam  reply    104
## 19 spam  prize     92
## 20 spam  cash      76
```

```
sms_cleaned_min_Count <- sms_cleaned %>%
  group_by(.id, type, word) %>%
  summarize(.count = n()) %>% #Calculate number of word occurrences
  ungroup() %>%
  rename(.type = type) %>% # word 'type' occurs in SMS text, name of target variable is renamed
```

```r
  mutate(.type = factor(.type)) %>%
  mutate(word_occurs = factor(TRUE, levels = c(FALSE, TRUE)))

# Create Document-Term-Matrix
# rows correspond to SMS (documents), columns correspond to words (terms)
# Values correspond to the number of occurrences of the word in the corresponding SMS.

library(tidyr)
sms_dtm <- sms_cleaned_min_Count %>%
  spread(word, word_occurs) %>%
  select(-.id, -.count)
sms_dtm[is.na(sms_dtm)] <- FALSE

# remove words (coloumns) that occur less than 2 times in all SMS
idx <- map_lgl(sms_dtm %>% select(-.type), ~ sum(as.logical(.)) > 1)
sms_dtm <- sms_dtm %>% select(.type) %>% bind_cols(sms_dtm[, which(idx)+1])

# Create training and test sets
set.seed(123)
trainIndex <- sample(c(FALSE,TRUE), size = nrow(sms_dtm), prob = c(.3,.7), replace = TRUE)
sms_train <- sms_dtm[trainIndex, ]
sms_test <- sms_dtm[!trainIndex, ]

# Check that distribution of target variables is approximately the same in both sets.
prop.table(table(sms_train$.type))
```

```
##
##       ham      spam
## 0.8361594 0.1638406
```

```r
prop.table(table(sms_test$.type))
```

```
##
##       ham      spam
## 0.8317708 0.1682292
```

```r
# Create Naive Bayes classifier
library(e1071)
m <- naiveBayes(sms_train %>% select(-.type), sms_train$.type, laplace = 0)
# laplace is a number to control the Laplace estimator --> avoid zero
# values for probability value
p <- predict(m, sms_test %>% select(-.type), type = "class")

library(caret)
cm <- confusionMatrix(sms_test$.type, p, dnn = c("True Label", "Predicted Label"))
cm$table
```

```
##           Predicted Label
## True Label  ham spam
##       ham  1580   17
##       spam   93  230
```

```r
# Accuracy
cm$overall[1]
```

```
##  Accuracy
## 0.9427083
```

```r
# Sensitivity and specificity (assuming that 'spam' is the positive class)
cm$byClass[1:2]
```

```
## Sensitivity Specificity
##   0.9444112   0.9311741
```

```r
# How much better is our classifier in terms of Accuracy compared to
# a simple heuristic that always predicts the majority class ('ham')?
cm$overall[1] - sum(sms_test$.type == "ham")/nrow(sms_test)
```

```
##  Accuracy
## 0.1109375
```

```r
# Sensitivity improvement corresponds to the sensitivity, since the
# sensitivity of the maj. vote is 0.
cm$byClass[1] - 0
```

```
## Sensitivity
##   0.9444112
```

```r
# The specificity of the Maj. voting is 1, therefore the value for our
# classifier is slightly worse
cm$byClass[2] - 1
```

```
## Specificity
## -0.06882591
```

Block 2: Since 1946, all member states of the United Nations have come together at the United Nations General Assembly to discuss and vote on resolutions, among other things. Currently 193 states belong to the United Nations. Each of these member states has exactly one vote in the General Assembly's resolution votes on issues such as disarmament, international security, humanitarian aid and human rights.
The record for this task contains the complete voting process at the General Assembly of each country. Is it possible to predict whether Germany will vote "yes" or "no" in a resolution vote?

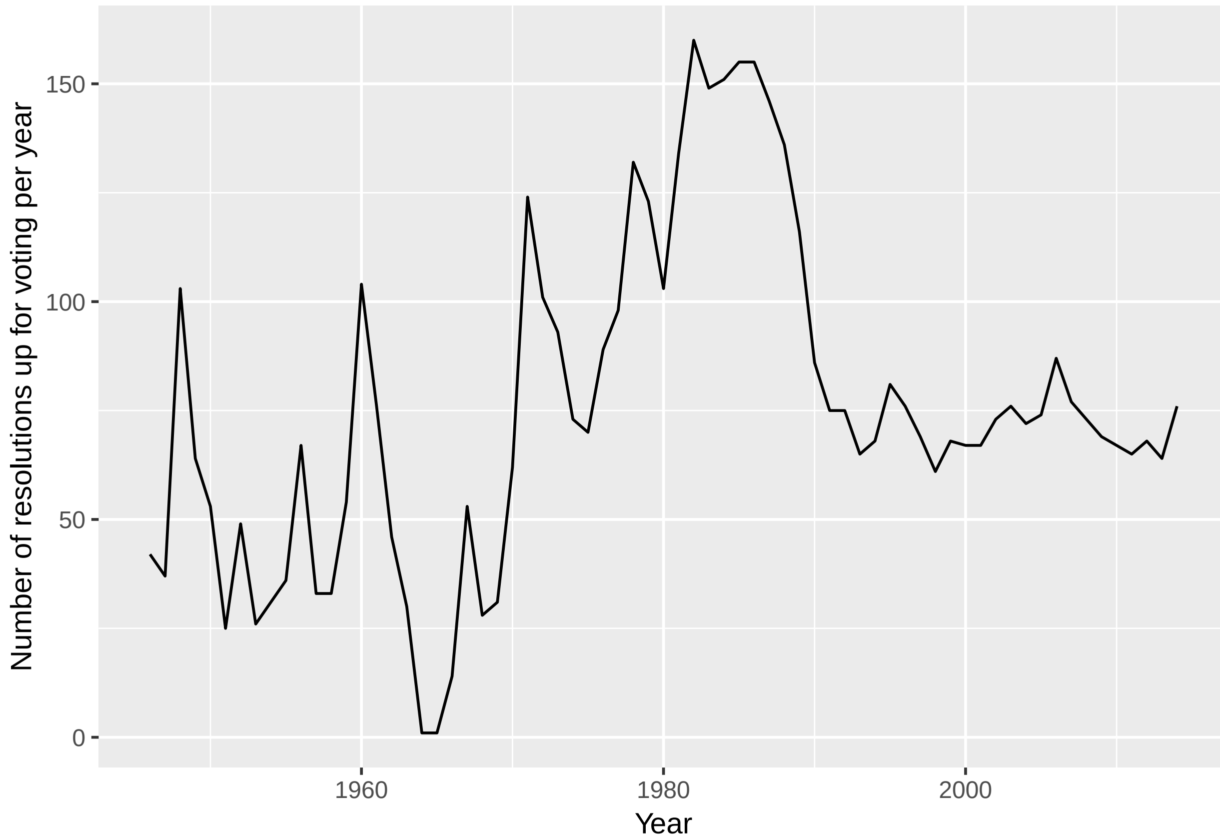
4. Display the number of resolutions voted on each year in a line chart. In which year were there the most votes and how many were there? Calculate between Germany and the USA for each year the proportion of equal votes (variable `vote`) for resolutions, hereinafter referred to as `agreement`. For the year 2006, the agreement between the two states was only about 25% of a total of 87 votes. (*Note: until 1989 "Federal Republic of Germany"; from 1989 "Germany"*)
5. Create a linear regression model that predicts the agreement between the two states based on the year (`agreement ~ year`). Interpret the trend and the p-value of the regression coefficient for `year`. Check the statement of the model graphically. Create a distance matrix between all pairs of states based on their voting history. Only consider states that have cast a vote in at least 70% of all votes. Determine the 5 states that are most similar or most dissimilar to Germany with regard to the voting history at UN General Assemblies.

6. Divide the data set into a training and test set at a ratio of 75%:25%. Create a kNN classifier with $k = 3$ (`caret::knn3Train()`) to predict the vote of Germany in a vote based on the votes of the countries `'Italy', 'Netherlands', 'United States of America', 'Israel', 'Cuba', 'India'`. Remove votes in which Germany abstained (`vote=2` ("Abstain")) to get a binary target variable for `vote=1` ("Yes") and `vote=0` ("No"). Create the Confusion Matrix and calculate the Accuracy for the model. On the same data, create a logistic regression model (`glm(..., family = "binomial")`) and compare the accuracy with that of the kNN classifier.

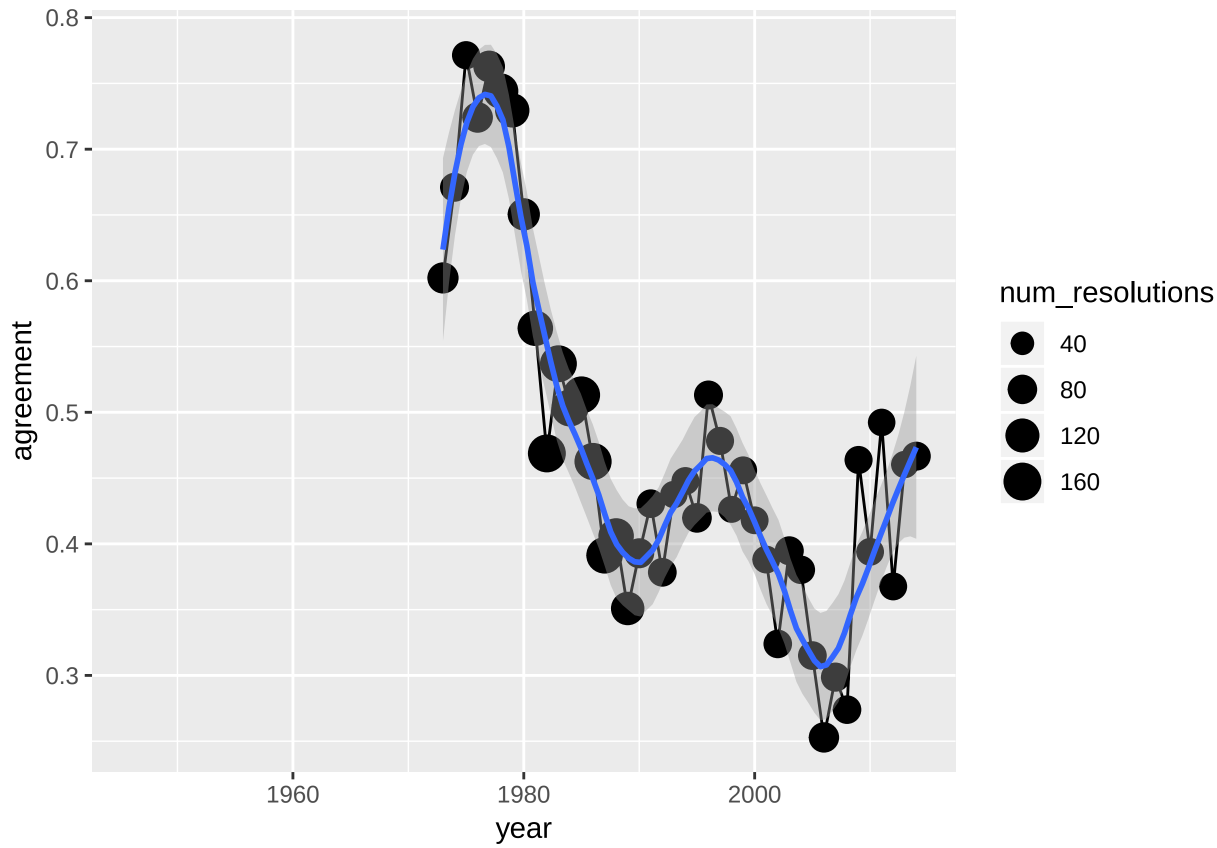
```
votes <- read_rds(str_c(getwd(), "/UNVotes.rds"))
votes
```

```
## # A tibble: 724,863 x 20
##     rcid session  vote ccode member importantvote date  unres    me    nu
##    <int>   <int> <int> <int>  <int>         <int> <I(c> <I(c> <int> <int>
##  1     3       1     1     2      1             0 1/1/~ R/1/~     0     0
##  2     3       1     3    20      1             0 1/1/~ R/1/~     0     0
##  3     3       1     1    40      1             0 1/1/~ R/1/~     0     0
##  4     3       1     1    41      1             0 1/1/~ R/1/~     0     0
##  5     3       1     1    42      1             0 1/1/~ R/1/~     0     0
##  6     3       1     1    70      1             0 1/1/~ R/1/~     0     0
##  7     3       1     1    90      1             0 1/1/~ R/1/~     0     0
##  8     3       1     1    91      1             0 1/1/~ R/1/~     0     0
##  9     3       1     1    92      1             0 1/1/~ R/1/~     0     0
## 10     3       1     1    93      1             0 1/1/~ R/1/~     0     0
## # ... with 724,853 more rows, and 10 more variables: di <int>, hr <int>,
## #   co <int>, ec <I(chr)>, year <dbl>, country <chr>, amend <int>,
## #   para <int>, short <chr>, descr <chr>
```

```
# Task 4
# Number of resolution votes per year
votes %>%
  group_by(year, unres) %>%
  slice(1) %>%
  ungroup() %>%
  count(year) %>%
  ggplot(aes(year, n)) +
  geom_line() +
  labs(x = "Year", y = "Number of resolutions up for voting per year")
```




```
votes <- votes %>%
  mutate(country = str_replace(country, "Federal Republic of Germany", "Germany")) %>%
  mutate(country = str_replace(country, "United States of America", "USA")) %>%
  mutate(country = str_replace(country, "United Kingdom of Great Britain and Northern Ireland"

y <- votes %>%
  filter(country %in% c("Germany", "USA")) %>%
  select(rcid, country, year, vote) %>%
  spread(country, vote) %>%
  group_by(year) %>%
  summarize(agreement = mean(Germany == USA, na.rm = T),
            num_resolutions = n())

y %>%
  ggplot(aes(year, agreement)) +
  geom_path() +
  geom_point(aes(size = num_resolutions)) +
  geom_smooth(span = 1/3)
```




```
# Task 5
lm_fit <- y  %>% lm(agreement ~ year, dat = .)

library(broom)
tidy(lm_fit)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic      p.value
##   <chr>          <dbl>     <dbl>     <dbl>        <dbl>
## 1 (Intercept)  16.4      2.35         6.98 0.0000000203
## 2 year        -0.00800   0.00118     -6.77 0.0000000388
```

```
#summary
summary(lm_fit)
```

```
##
## Call:
## lm(formula = agreement ~ year, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15947 -0.06761 -0.01678  0.05697  0.15788
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.415231   2.353403   6.975 2.03e-08 ***
## year        -0.007996   0.001181  -6.774 3.88e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09273 on 40 degrees of freedom
##   (27 observations deleted due to missingness)
## Multiple R-squared:  0.5342, Adjusted R-squared:  0.5226
## F-statistic: 45.88 on 1 and 40 DF,  p-value: 3.88e-08
```

```
# printed model
print(lm_fit)
```

```
##
## Call:
## lm(formula = agreement ~ year, data = .)
##
## Coefficients:
## (Intercept)         year
##   16.415231    -0.007996
```

```
# Number of resolution votes -> one-liner alternative: length(unique(votes$rcid))
num_votes <- votes %>%
  group_by(rcid) %>%
  slice(1) %>%
  ungroup() %>%
  nrow()

# Countries that have taken part in at least 70% of all votes
countries <- votes %>%
  group_by(country) %>%
  summarize(p = n()/num_votes) %>%
  filter(p >= 0.7) %>%
  .$country

# Create voting country matrix
# (rows correspond to votes, columns countries, values vote of the
# respective country at the respective vote)
tmp <- votes %>%
    filter(country %in% countries) %>%
    select(rcid, country, vote) %>%
    spread(country, vote)

X <- as.matrix(tmp[,-1])
rownames(X) <- tmp$rcid
```

```
d <- dist(t(X))

dist_from_de <- as.matrix(d)["Germany",]
```

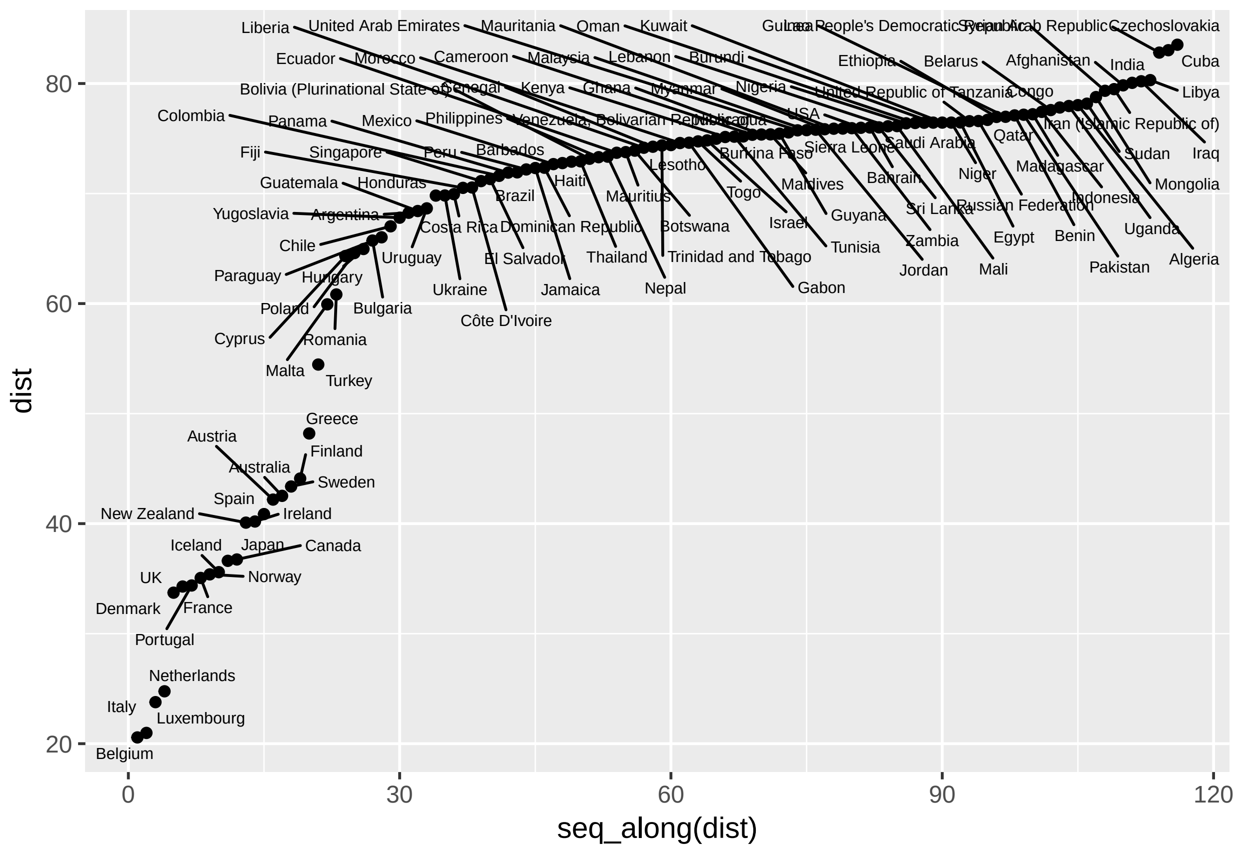
```r
library(ggrepel)
dist_from_de_tibble <- tibble(country = names(dist_from_de),
                              dist = dist_from_de) %>%
  filter(country != "Germany") %>%
  arrange(dist)

# 5 countries with the smallest distance to Germany
dist_from_de_tibble %>%
  slice(1:5)
```

```
## # A tibble: 5 x 2
##   country      dist
##   <chr>       <dbl>
## 1 Belgium      20.6
## 2 Luxembourg   21.0
## 3 Italy        23.8
## 4 Netherlands  24.8
## 5 Denmark      33.7
```

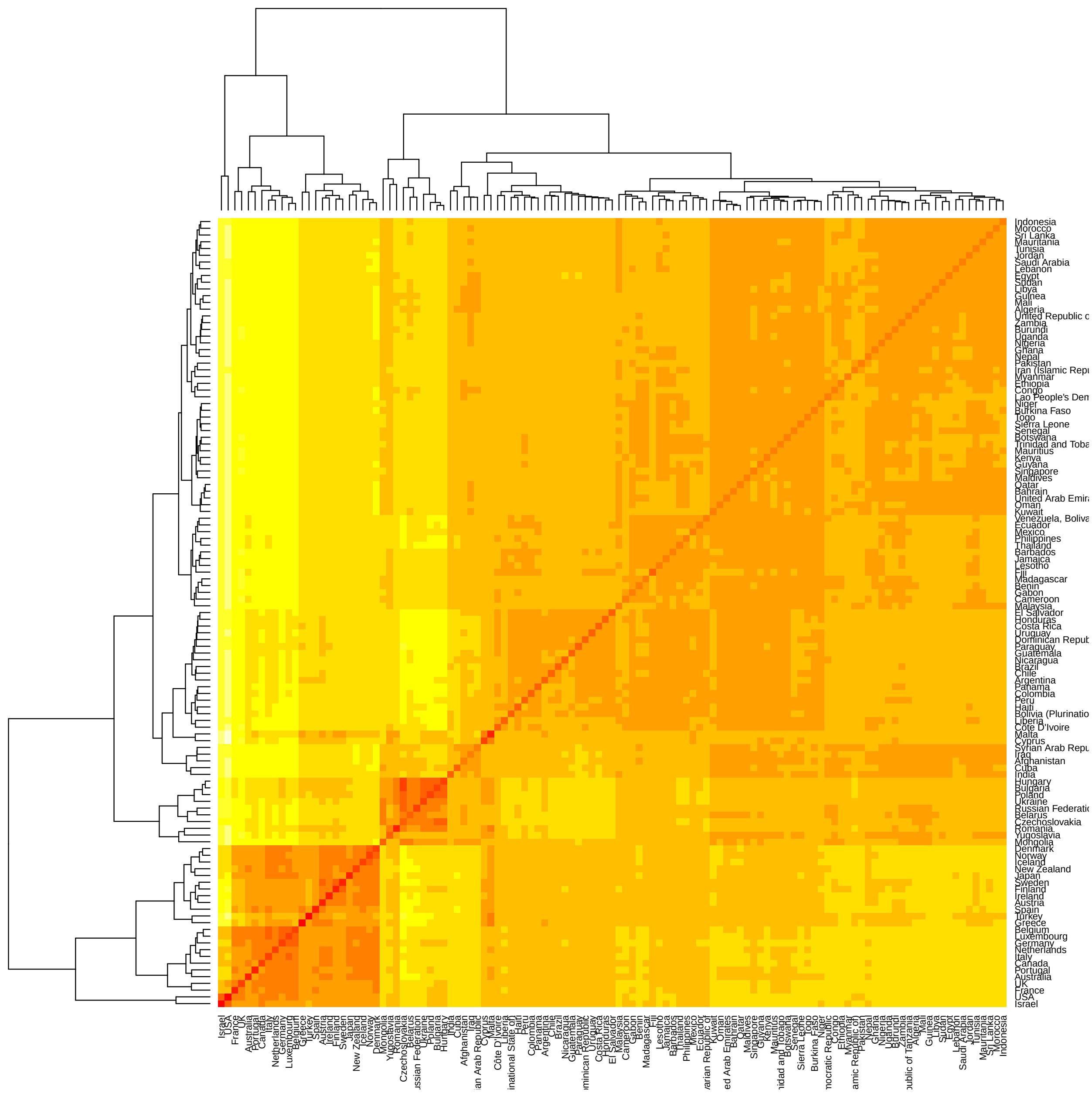
```r
# 5 countries with the greatest distance to Germany
dist_from_de_tibble %>%
  arrange(-dist) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 2
##   country               dist
##   <chr>                <dbl>
## 1 Czechoslovakia        83.5
## 2 Cuba                  83.0
## 3 Syrian Arab Republic  82.8
## 4 Libya                 80.3
## 5 Iraq                  80.2
```

```r
dist_from_de_tibble %>%
  ggplot(aes(x = seq_along(dist), y = dist, label = country)) +
  geom_point() +
  geom_text_repel(size = 2)
```



```
heatmap(as.matrix(d))
```




```
# Task 6
countries <- c("Germany",  "Italy", "Netherlands", "USA", "Israel", "Cuba", "India")

# Filter only polls of countries in 'countries
# Create binary target variable -> German Yes (1) vs. German No (0)
dat <- votes %>%
        filter(country %in% countries) %>%
        select(rcid, country, vote) %>%
        spread(country, vote, fill = 2) %>%
        rename(y = Germany) %>%
        filter(y != 2) %>%
        select(-rcid) %>%
```

```
        mutate(y = ifelse(y == 1, 1, 0))
library(caret)

# Create training and test sets
set.seed(123)
trainIndex <- sample(c(FALSE,TRUE), size = nrow(dat), prob = c(.25,.75), replace = TRUE)

train_set <- dat[trainIndex, ]
test_set <- dat[!trainIndex, ]

# Learn Logistic Regression Model
fit <- glm(y ~ ., data = train_set, family = "binomial")
pred <- predict(fit, newdata = test_set, type = "response")
tab <- table(actual = test_set$y, predicted = round(pred))
cm1 <- confusionMatrix(tab)
cm1
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual   0   1
##      0 150   6
##      1   5 471
##
##                Accuracy : 0.9826
##                  95% CI : (0.9691, 0.9913)
##     No Information Rate : 0.7547
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9531
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9677
##             Specificity : 0.9874
##          Pos Pred Value : 0.9615
##          Neg Pred Value : 0.9895
##              Prevalence : 0.2453
##          Detection Rate : 0.2373
##    Detection Prevalence : 0.2468
##       Balanced Accuracy : 0.9776
##
##        'Positive' Class : 0
##
```

```
# Learn KNN classifier
fit <- knn3Train(train_set %>% select(-y), test_set %>% select(-y), cl = train_set$y,
                 k = 3, prob = F)
```

```r
tab <- table(actual = as.character(test_set$y), predicted = fit)
cm2 <- confusionMatrix(tab)
cm2
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual   0   1
##      0 154   2
##      1   5 471
##
##                Accuracy : 0.9889
##                  95% CI : (0.9773, 0.9955)
##     No Information Rate : 0.7484
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9704
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9686
##             Specificity : 0.9958
##          Pos Pred Value : 0.9872
##          Neg Pred Value : 0.9895
##              Prevalence : 0.2516
##          Detection Rate : 0.2437
##    Detection Prevalence : 0.2468
##       Balanced Accuracy : 0.9822
##
##        'Positive' Class : 0
##
```

```r
# Difference with regard to accuracy between log. Regression and KNN
cm2$overall["Accuracy"] - cm1$overall["Accuracy"]
```

```
##    Accuracy
## 0.006329114
```

Dataset for Block 1: http://isgwww.cs.uni-magdeburg.de/cv/lehre/VisAnalytics/material/exercise/datasets/spam.csv
(adaptiert von http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/)

Dataset for Block 2: http://isgwww.cs.uni-magdeburg.de/cv/lehre/VisAnalytics/material/exercise/datasets/UNVotes.rds
(adapted by https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:1902.1/12379)
- Data Dictionary / Codebook: http://isgwww.cs.uni-magdeburg.de/cv/lehre/VisAnalytics/material/exercise/datasets/UNVotes_Codebook.pdf

[1] https://en.wikipedia.org/wiki/Document-term_matrix

[2] https://de.wikipedia.org/wiki/Tokenisierung, http://tidytextmining.com/tidytext.html