Text Mining

Modern Data Mining

Contents

Objectives

This script borrows heavily from the lecture notes of STAT 471 taught at Wharton School of Business, University of Pennsylvania in the spring of 2022.

In modern data mining, we often encounter the situation where a text may contain important, useful information about response of interest. Can we predict how likely a lecture is given by a female based on the transript? One simple but effective way of learning from a text is through bag of words to convert raw text data into a numeric matrix. We first turn each text into a vector of frequency of words. The bag of words approach can be extended using the n-gram or k-skip-n-gram techniques to account for the semantics of word *ordering.* Then we apply existing methods that use numerical matrices to either extract useful information or carry out predictions. We will extend the regularization technique (LASSO) to classification problems.

In this `TED` case study, we will use the `tm` package to transform text into a word frequency matrix. We will build a classifier and conduct sentiment analysis. Finally we build a word cloud to exhibit words for male speakers' topics/transcripts and female speakers' topics/transcripts respectively.

Reference about text mining and web scraping: Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining, Wiley, by Munzert/Rubba/MeiBner/Nyhuis.

1. Introduction to Case Study: TED transcripts
2. LASSO for categorical response data
3. EDA
4. Data Cleaning and Transformation
    1. Corpus - Collection of documents
    2. Document cleansing
    3. Create words frequency table
        - Library: all the words among all documents
        - Rows: documents
        - Columns: word frequency
        - Output the matrix
5. Analyses
    1. Build a classifier using your favorite methods:
        - glmnet
        - glm
    2. Word clouds using glm results
    3. Predictions

1 Case Study: TED and gender

Founded in 1984, TED Conferences, LLC (Technology, Entertainment, Design) is an American-Canadian media organization that posts talks online for free distribution under the slogan "ideas worth spreading". TED was conceived by Richard Saul Wurman, who co-founded it with Harry Marks in February 1984 as a conference. It has been held annually since 1990. TED's early emphasis was on technology and design, consistent with its Silicon Valley origins. It has since broadened its perspective to include talks on many scientific, cultural, political, humanitarian, and academic topics.

Goal: How are the contents of TED talks related to speakers' genders? How well can we predict speakers' genders based on the content of talks?

Data Description: The dataset contain over 4,000 TED talks in many languages. Let us only look into talks whose native languages are in English.

Data needed:

- `ted_talks_en.csv`
- `NRC.csv`
- `ted_en_tm_freq_transcript.csv`
- `TextMining_glm_small_transcript.RDS`
- `TextMining_lasso_transcript.RDS`

1.1 Packages used

1. Text Mining Packages
   - `tm`: a popular text mining package
   - Note: Ingo Feinerer created text mining package `tm` in 2008 while he was a phd student at TU Vienna. Users can deploy `Hadoop` to handle large textual data.
   - `SnowballC`: For Stemming
2. Word Cloud Packages
   - `RColorBrewer`: a package for color palette
   - `wordcloud`: a package for creating wordcloud
3. LASSO and Random Forest packages
   - `glmnet`
   - `randomForest`
   - `ranger`
   - `stringr`: useful string package

My favorite packages (Of course all come from Hadley Wickham….): `ggplot2`, `dplyr`, `stringr` (string manipulations), `lubridate` (handling date-time). Most of them are aggregated into `tidyverse`.

2 LASSO for classification

The regularization techniques used in regression are readily applied to classification problems. Here we will penalize the coefficients while maximizing the likelihood function or minimizing the -loglikelihood function.

For a given lambda we minimize -loglikelihood. Here is the LASSO solutions:

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} -\frac{1}{n}\log(\mathcal{L}\rangle\|) + \lambda\{|\beta_1| + |\beta_2|, \ldots + |\beta_p|\}$$

Similarly we obtain the solution for elastic net using the general penalty functions:

$$\left(\frac{1-\alpha}{2}\right)\|\beta\|_2^2 + \alpha\|\beta\|_1$$

Remark:

1) To remain consistent in both binary and continuous responses, `glmnet()` uses the following penalized least squares for a continuous response variable.

$$\frac{1}{2n}RSS + \lambda\Big\{\Big(\frac{1-\alpha}{2}\Big)||\beta||_2^2 + \alpha||\beta||_1\Big\}$$

2) The properties of LASSO (and elastic-net) solutions for logistic regressions are similar to that of OLS.

3) `glmnet()` and `cv.glmnet()` are used in the logistic regression setup as well. We do need to specify `family=binomial`. Also we may choose different minimization criteria such as `Deviance`, `AUC` or `MCE`. The default setting is by `Deviance`, which means we try to minimize the CV -likelihood function.

For the remaining lecture, the first step is to do EDA as usual . Then we digitize a text into a large dimension of word frequency vectors. `glm` and `LASSO` methods will be used to build models of gender based on the topics/transcripts. Finally we report testing errors comparing different models.

Important remarks about this lecture:

- Because of large dimension, part of the analyses are run through an even smaller data set.
- The full analysis results have been done and saved. We will load the results directly for the purpose of demonstration. You may run some specific chunks to get the full results yourself but it will take some time and may crush your laptop. On the other hand once you are successful, output and save your results. Then set the chunk for analysis to `eval=F` and load the results properly.
- We have also included other methods such as `Random Forest`. You may keep them as references for later uses.

3 Exploratory Data Analysis (EDA)

3.1 Read data

Using package `data.table` to read and to manipulate data is much faster than using `read.csv()` especially when the dataset is large.

```
data.all <- fread("data/ted_talks_en.csv", stringsAsFactors = FALSE)
dim(data.all)
object.size(data.all)

nrc<- fread("data/NRC.csv")
afinn<- fread("data/Afinn.csv")
bing<- fread("data/Bing.csv")
```

```
options(warn = -1)
```

```
## [1] 4005   19
## 48003424 bytes
```

Let's first take a small piece of it to work through. We could use `fread` to only load `nrows` many rows to avoid loading the entire dataset.

```
data <- data.all
data = data %>% filter(native_lang=="en") # choose those whose native language is English
names(data)
n <- nrow(data)

data_sub<- data %>%
select(talk_id, title, speaker_1, occupations, about_speakers, recorded_date, published_date, duration,

str(data_sub%>%
select(-transcript, -description))

data_sub %>%
select(-transcript)%>%
head()
```

```
##  [1] "talk_id"        "title"          "speaker_1"      "all_speakers"
##  [5] "occupations"    "about_speakers" "views"          "recorded_date"
##  [9] "published_date" "event"          "native_lang"    "available_lang"
## [13] "comments"       "duration"       "topics"         "related_talks"
## [17] "url"            "description"    "transcript"
## Classes 'data.table' and 'data.frame':   3957 obs. of  11 variables:
##  $ talk_id       : int  1 92 7 53 66 49 86 94 71 55 ...
##  $ title         : chr  "Averting the climate crisis" "The best stats you've ever seen" "Simplicity s
##  $ speaker_1     : chr  "Al Gore" "Hans Rosling" "David Pogue" "Majora Carter" ...
##  $ occupations   : chr  "{0: ['climate advocate']}" "{0: ['global health expert; data visionary']}"
##  $ about_speakers: chr  "{0: 'Nobel Laureate Al Gore focused the world's attention on the global cli
##  $ recorded_date : IDate, format: "2006-02-25" "2006-02-22" ...
##  $ published_date: IDate, format: "2006-06-27" "2006-06-27" ...
##  $ duration      : int  977 1190 1286 1116 1164 1198 992 1485 1262 1538 ...
##  $ topics        : chr  "['alternative energy', 'cars', 'climate change', 'culture', 'environment',
##  $ views         : int  3523392 14501685 1920832 2664069 65051954 1208138 4636596 3781244 3998282 460
##  $ comments      : int  272 628 124 219 4931 48 980 919 930 59 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##    talk_id                                  title           speaker_1
## 1:       1            Averting the climate crisis             Al Gore
## 2:      92        The best stats you've ever seen        Hans Rosling
## 3:       7                       Simplicity sells         David Pogue
## 4:      53                    Greening the ghetto       Majora Carter
## 5:      66            Do schools kill creativity?    Sir Ken Robinson
## 6:      49 Behind the design of Seattle's library Joshua Prince-Ramus
##                                       occupations
## 1:                      {0: ['climate advocate']}
## 2: {0: ['global health expert; data visionary']}
## 3:                   {0: ['technology columnist']}
## 4:   {0: ['activist for environmental justice']}
## 5:                    {0: ['author', 'educator']}
## 6:                             {0: ['architect']}
```

```
##
## 1:
## 2:
## 3:                                                                                          {0: 'Davi
## 4:
## 5:
## 6: {0: 'Joshua Prince-Ramus is best known as architect of the Seattle Central Library, already being
##    recorded_date published_date duration
## 1:    2006-02-25     2006-06-27      977
## 2:    2006-02-22     2006-06-27     1190
## 3:    2006-02-24     2006-06-27     1286
## 4:    2006-02-26     2006-06-27     1116
## 5:    2006-02-25     2006-06-27     1164
## 6:    2006-02-23     2006-07-10     1198
##
## 1:     ['alternative energy', 'cars', 'climate change', 'culture', 'environment', 'global issues', '
## 2: ['Africa', 'Asia', 'Google', 'demo', 'economics', 'global issues', 'health', 'statistics', 'global
## 3:                ['computers', 'entertainment', 'interface design', 'media', 'music', 'performance'
## 4:                       ['MacArthur grant', 'activism', 'business', 'cities', 'environment', 'green
## 5:                                                       ['children', 'creativity', 'culture', 'danc
## 6:                                                                          ['architecture', 'collab
##
## 1: With the same humor and humanity he exuded in ""An Inconvenient Truth,"" Al Gore spells out 15 way
## 2:                                                              You've never seen data presented like
## 3:                                  New York Times columnist David Pogue takes aim at technology’s w
## 4:                       In an emotionally charged talk, MacArthur-winning activist Majora Carter de
## 5:                                                                                       Sir Ken Rob
## 6:                       Architect Joshua Prince-Ramus takes the audience on dazzling, dizzying virtu
##       views comments
## 1:  3523392      272
## 2: 14501685      628
## 3:  1920832      124
## 4:  2664069      219
## 5: 65051954     4931
## 6:  1208138       48
```

```
#to remove all the brackets, numbers and unnecessary punctuations in the occupations, transcript and to
data_sub<- data_sub%>%
mutate(occupations = str_replace_all(occupations, "\\{|\\#|\\:|\\[|\\'|\\;|\\,|\\]|\\}", "")) %>%
mutate(topics = str_replace_all(topics, "\\{|\\#|\\:|\\[|\\'|\\;|\\,|\\]|\\}", ""))%>%
mutate(about_speakers = str_replace_all(about_speakers, "\\{|\\#|\\:|\\[|\\'|\\;|\\,|\\]|\\}", ""))

#to remove numbers
data_sub$occupations<-data_sub$occupations%>%
removeNumbers()

data_sub$about_speakers<-data_sub$about_speakers%>%
removeNumbers()

# to convert duration from seconds to minutes
data_sub<- data_sub %>%
mutate(duration =round(duration/60, 2))

#to extract the year
```

```
data_sub<-data_sub %>%
mutate(year = year(recorded_date))
# published_date

#to remove the name of the speaker from the talk description
data_sub$description <- mapply(function(x,y)gsub(x,"",y) ,gsub(" ", "|",data_sub$speaker_1),data_sub$des

data_sub<- data_sub %>%
            relocate(year)%>%
            rename("speaker"="speaker_1")

data_sub %>%
select(-transcript)%>%  # not selecting the transcript column so that it is easy to view the data for n
head()
```

```
data_sub$first_name = word(data_sub$speaker, 1)
data_sub$first_name[word(data_sub$speaker, 1)=="Sir"] = word(data_sub$speaker[word(data_sub$speaker, 1)=

gender_df <- gender::gender(data_sub$first_name, method = "ssa", years = c(1960,2012)) %>%
  select(first_name = name, gender)
gender_df <- unique(gender_df)
data_sub_gender = left_join(data_sub, gender_df, by = "first_name")
```

3.2 label: gender

Let's make female out of the gender variable.

```
data_sub_gender$female <- c(0)
data_sub_gender$female[data_sub_gender$gender == "female"] <- 1
data_sub_gender$female <- as.factor(data_sub_gender$female)
summary(data_sub_gender$female) #str(data_sub_gender)
```

Proportion of female speakers:

```
prop.table(table(data_sub_gender$female))
```

```
##
##         0         1
## 0.6502401 0.3497599
```

Notice that 35% of the speakers are female.

3.3 How to handle date

Does talks relate to month or day of the weeks?

Dealing with `date` can be a challenging job. Should we treat them as continuous variables or categorical ones? This highly depends on the context and the goal of the study. In our situation it makes sense that we are interested in knowing if people tend to give lectures over the weekend.

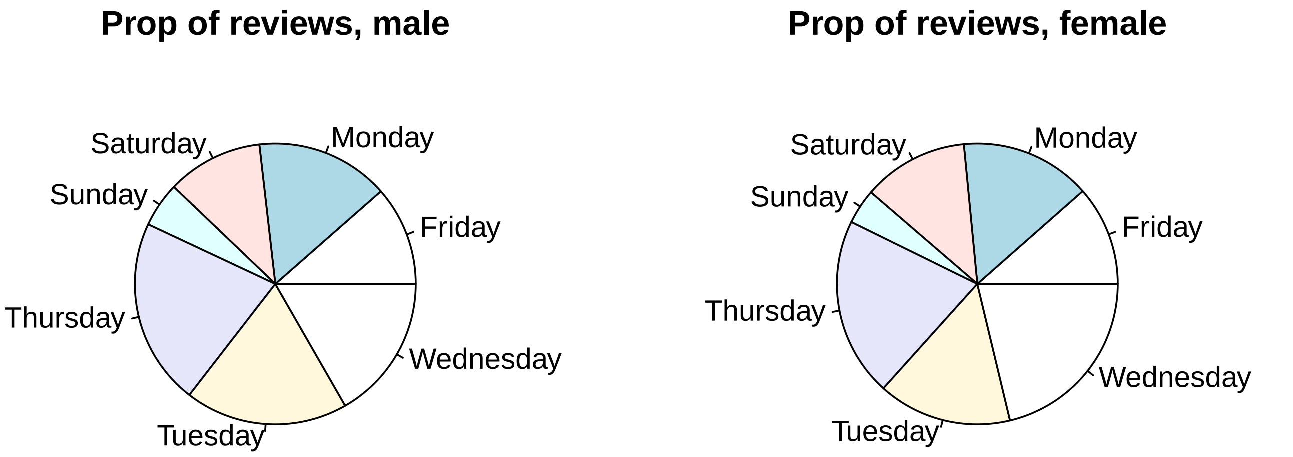
Let us use functions in `tidyverse` to format the dates and extract weekdays

```
weekdays <- weekdays(as.Date(data_sub_gender$recorded_date)) # get weekdays for each talk
weekdays_female <- weekdays(as.Date(data_sub_gender[data_sub_gender$female==1]$recorded_date)) # get we
weekdays_male <- weekdays(as.Date(data_sub_gender[data_sub_gender$female==0]$recorded_date)) # get week

months <- months(as.Date(data_sub_gender$recorded_date))    # get months
```

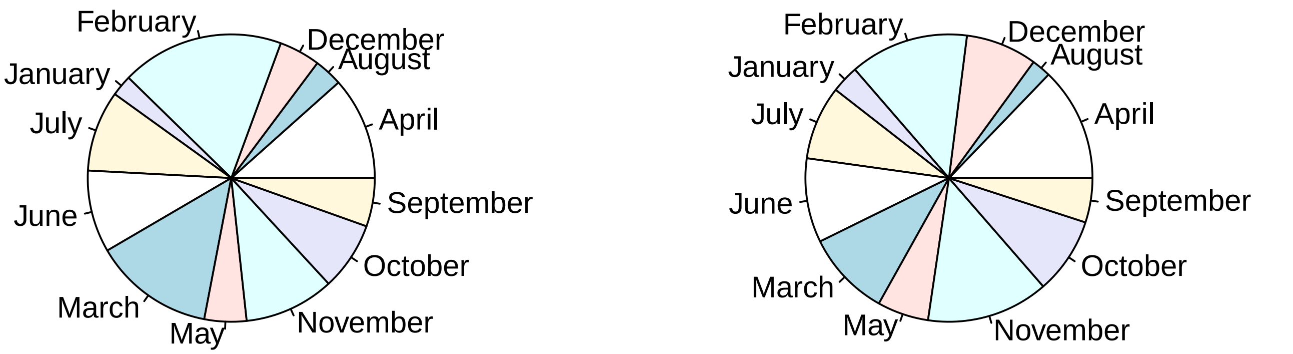
```
months_female <- months(as.Date(data_sub_gender[data_sub_gender$female==1]$recorded_date))   # get mont
months_male <- months(as.Date(data_sub_gender[data_sub_gender$female==0]$recorded_date))   # get months
```

Do people tend to give a talk over weekends? (months?)

```
par(mfrow=c(1,2))
pie(table(weekdays_male), main="Prop of reviews, male") # Pretty much evenly distributed
pie(table(weekdays_female), main="Prop of reviews, female") # Pretty much evenly distributed
```



```
par(mfrow=c(1,2))
pie(table(months_male))
pie(table(months_female))
```



Proportion of female talks: we don't really see any pattern.

```
prop.table(table(data_sub_gender$female, weekdays), 2)  # prop of the columns
prop.table(table(data_sub_gender$female, weekdays), 1)  # prop of the rows

prop.table(table(data_sub_gender$female, months), 2)  # prop of the columns
prop.table(table(data_sub_gender$female, months), 1)  # prop of the rows
```

```
##    weekdays
##        Friday    Monday  Saturday    Sunday  Thursday   Tuesday Wednesday
##   0 0.6497797 0.6550580 0.6283186 0.7037037 0.6599045 0.6939655 0.5939227
##   1 0.3502203 0.3449420 0.3716814 0.2962963 0.3400955 0.3060345 0.4060773
##    weekdays
##         Friday     Monday   Saturday     Sunday   Thursday    Tuesday
##   0 0.11465216 0.15351729 0.11037699 0.05169063 0.21492421 0.18771862
##   1 0.11496746 0.15039769 0.12147505 0.04049168 0.20607375 0.15401302
##    weekdays
##      Wednesday
##   0 0.16712009
##   1 0.21258134
##    months
##         April    August  December  February   January      July      June
##   0 0.6273684 0.7321429 0.5173913 0.7179878 0.5865385 0.6704871 0.6449864
##   1 0.3726316 0.2678571 0.4826087 0.2820122 0.4134615 0.3295129 0.3550136
##    months
##         March       May  November   October September
##   0 0.7240664 0.6039604 0.5787140 0.6246057 0.6698565
##   1 0.2759336 0.3960396 0.4212860 0.3753943 0.3301435
##    months
##           April     August   December   February    January       July
##   0 0.11581811 0.03186941 0.04624951 0.18305480 0.02370773 0.09094442
##   1 0.12798265 0.02169197 0.08026030 0.13376717 0.03109183 0.08315257
##    months
##            June      March        May   November    October  September
##   0 0.09249903 0.13563933 0.04741547 0.10143801 0.07695297 0.05441119
##   1 0.09472162 0.09616775 0.05784526 0.13738250 0.08604483 0.04989154
```

3.4 text mining

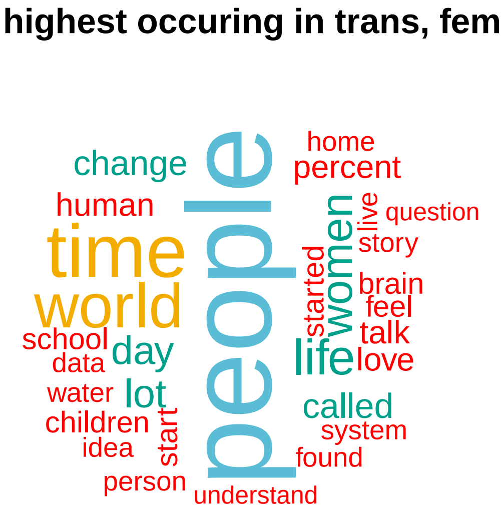
I will use the tidytext library to tokenize the transcripts to find which words have the highest frequency of occurance in the ted talks, with respect to each gender.

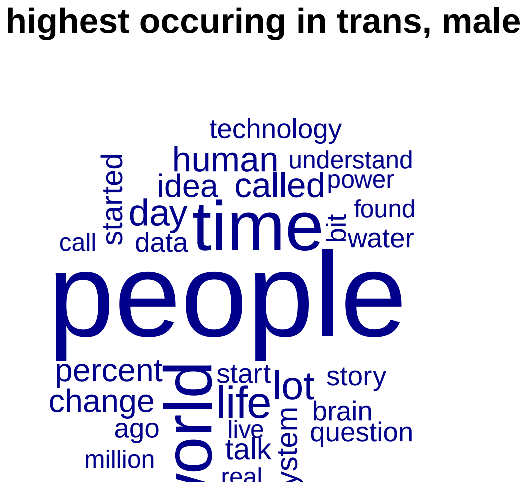
```
custom_stop<- tribble(
~word, ~lexicon,
"laughter", "custom",
"applause", "custom",
"chris", "custom",
"anderson", "custom")

stop_new<- stop_words%>%
rbind(custom_stop)
```

Let us use Wordcloud now to look for most frequent words¶. First let's see the most frequently used words in transcripts.

```
transcript_female <- data_sub_gender %>% filter(female == 1) %>%
  unnest_tokens(word, transcript)%>%
  anti_join(stop_new, by="word")

transcript_male <- data_sub_gender %>% filter(female == 0) %>%
  unnest_tokens(word, transcript)%>%
  anti_join(stop_new, by="word")

t_transcript_count_female <- transcript_female %>% count(word)
t_transcript_count_male <- transcript_male %>% count(word)

set.seed(1)
par(mfrow=c(1,2))

wordcloud(t_transcript_count_female$word, t_transcript_count_female$n, max.words=30, color=wes_palette(
title("highest occuring in trans, female")
options(repr.plot.width = 8, repr.plot.height = 8)
```

```
wordcloud(t_transcript_count_male$word, t_transcript_count_male$n, max.words=30, color="darkblue", scal
title("highest occuring in trans, male")
```

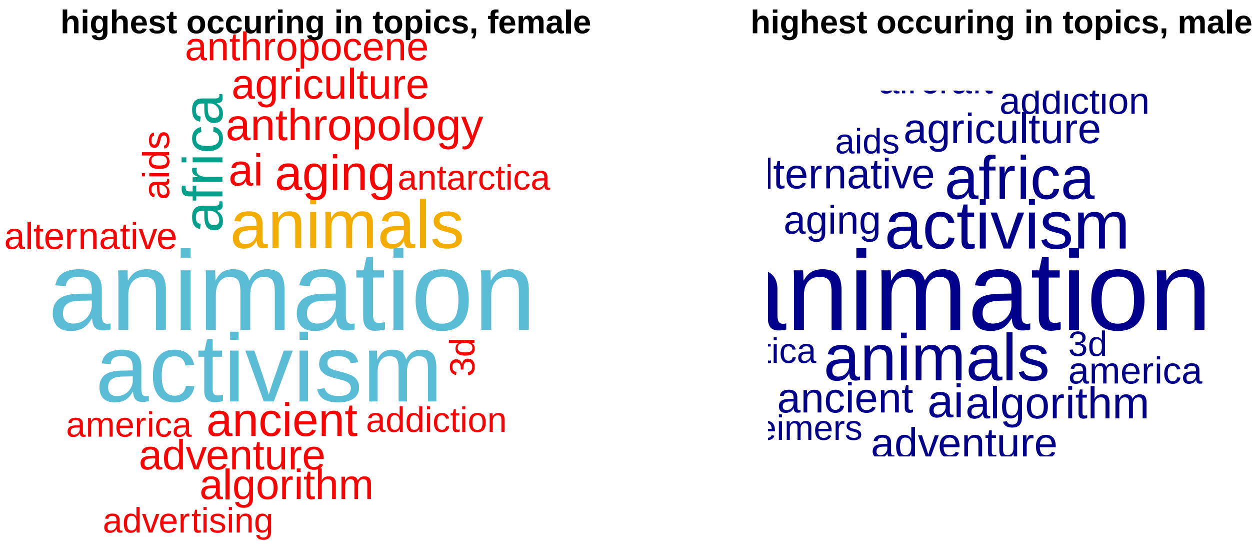




```
options(repr.plot.width = 8, repr.plot.height = 8)
```

Then let's see the most frequently used words in topics.

```
topics_female <- data_sub_gender %>% filter(female == 1) %>%
  unnest_tokens(word, topics)%>%
  anti_join(stop_new, by="word")

topics_male <- data_sub_gender %>% filter(female == 0) %>%
  unnest_tokens(word, topics)%>%
  anti_join(stop_new, by="word")

t_topics_count_female <- topics_female %>% count(word)
t_topics_count_male <- topics_male %>% count(word)

set.seed(1)
par(mfrow=c(1,2))

wordcloud(t_topics_count_female$word[1:20], t_topics_count_female$n[1:20], max.words=100, color=wes_pal
title("highest occuring in topics, female")
options(repr.plot.width = 8, repr.plot.height = 8)

wordcloud(t_topics_count_male$word[1:20], t_topics_count_male$n[1:20], max.words=100, color="darkblue",
title("highest occuring in topics, male")
```


highest occuring in topics, female

highest occuring in topics, male

```
options(repr.plot.width = 8, repr.plot.height = 8)
```

4 Bag of words and term frequency

How should we use a transcript or a topic list as predictors? Obviously, some sentences are informative. Also, words used to provide information, as well as sentiments, will show how people feel as well. We will turn a text into a vector of features, each of which represents the words that are used. The specific value of the feature for a given document tells us the frequency (how many occurrences) of that word in the document.

We do this by first collect all possible words (referred to as a library or bag of all words). We will then record frequency of each word used in the text.

4.1 Word term frequency table using `tm`

Use word term frequency table to transform texts into word frequencies matrix.

- First form a `bag of words`: all the words appeared in the documents say `N` (in general, very large)
- For each document (row), record the frequency (count) of each word in the bag which gives us `N` values (notice: most of the entries are 0, as most words will *not* occur in every document)
- Output the document term matrix (`dtm`) as an input to a later model

Let's first take a look at the texts we have:

```
transcript.text <- data_sub_gender$transcript   # take the text out
topics.text <- data_sub_gender$topics   # take the text out

length(transcript.text)
typeof(transcript.text)
```

We now extract text into word term frequency table.

4.1.1 Corpus: a collection of text

- `VCorpus()`: create Volatile Corpus
- `inspect()`: display detailed info of a corpus

```
transcript.corpus <- VCorpus(VectorSource(transcript.text))
topics.corpus <- VCorpus(VectorSource(topics.text))
# transcript.corpus
typeof(transcript.corpus)   ## It is a list
typeof(topics.corpus)   ## It is a list
```

```
## [1] "list"
## [1] "list"
```

Inspect the corpus, say documents number 4 and 5.

4.1.2 Data cleaning using `tm_map()`

Before transforming the text into a word frequency matrix, we should transform the text into a more standard format and clean the text by removing punctuation, numbers and some common words that do not have predictive power (a.k.a. stopwords; e.g. pronouns, prepositions, conjunctions). This can be tedious but is necessary for quality analyses. We use the `tm_map()` function with different available transformations including `removeNumbers()`, `removePunctuation()`, `removeWords()`, `stemDocument()`, and `stripWhitespace()`. It works like the `apply()` class function to apply the function to corpus. Details of each step are in the appendix. These cleaning techniques are not one-size-fits-all, and the techniques appropriate for your data will vary based on context.

```
# Converts all words to lowercase
transcript.corpus_clean <- tm_map(transcript.corpus, content_transformer(tolower))
```

```r
# Removes common English stopwords (e.g. "with", "i")
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removeWords, stopwords("english"))

# Removes any punctuation
# NOTE: This step may not be appropriate if you want to account for differences
#       on semantics depending on which sentence a word belongs to if you end up
#       using n-grams or k-skip-n-grams.
#       Instead, periods (or semicolons, etc.) can be replaced with a unique
#       token (e.g. "[PERIOD]") that retains this semantic meaning.
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removePunctuation)

# Removes numbers
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removeNumbers)

# Stem words
transcript.corpus_clean <- tm_map(transcript.corpus_clean, stemDocument, lazy = TRUE)

lapply(transcript.corpus_clean[4:5], as.character)
```

```
## $`4`
## [1] "today - happi - heard sustain develop will save us howev ted often told real sustain polici agen
##
## $`5`
## [1] "good morn audienc good great blown away whole thing fact leav laughter three theme run confer r
```

```r
# for topics

# Converts all words to lowercase
topics.corpus_clean <- tm_map(topics.corpus, content_transformer(tolower))

# Removes common English stopwords (e.g. "with", "i")
topics.corpus_clean <- tm_map(topics.corpus_clean, removeWords, stopwords("english"))

# Removes any punctuation
# NOTE: This step may not be appropriate if you want to account for differences
#       on semantics depending on which sentence a word belongs to if you end up
#       using n-grams or k-skip-n-grams.
#       Instead, periods (or semicolons, etc.) can be replaced with a unique
#       token (e.g. "[PERIOD]") that retains this semantic meaning.
topics.corpus_clean <- tm_map(topics.corpus_clean, removePunctuation)

# Removes numbers
topics.corpus_clean <- tm_map(topics.corpus_clean, removeNumbers)

# Stem words
topics.corpus_clean <- tm_map(topics.corpus_clean, stemDocument, lazy = TRUE)

lapply(topics.corpus_clean[4:5], as.character)
```

```
## $`4`
## [1] "macarthur grant activ busi citi environ green inequ polit pollut"
##
## $`5`
## [1] "children creativ cultur danc educ parent teach"
```

4.1.3 Word frequency matrix

Now we transform each transcript into a word frequency matrix using the function `DocumentTermMatrix()`.

```
transcript.dtm1 <- DocumentTermMatrix( transcript.corpus_clean )   ## library = collection of words for
class(transcript.dtm1)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
inspect(transcript.dtm1) # typeof(dtm1)  #length(dimnames(dtm1)$Terms)
```

```
## <<DocumentTermMatrix (documents: 3957, terms: 52898)>>
## Non-/sparse entries: 1583939/207733447
## Sparsity           : 99%
## Maximal term length: 42
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   can get just know like now one peopl thing think
##   2318  41  17   40   22   40  31  25    75    30    71
##   2349  27  20   41   11   39  16  22    77    18    68
##   348   26  27   13   24   22  37  38    65    34    21
##   3723  60  15   29   26   22  25  33    47    24    29
##   3913  43  58   29   70   24  26  26    80    41    16
##   3918  68  58   23   43   38  43  40    63    26    63
##   3919  37  32   33   47   32  62  38    79    35    74
##   3926  90  25   49   32   33  60  43    62    37    80
##   3930  23  40   35   29   32  12  25    60    41    75
##   3935  40  36   25   56   31  20  22    44    36    58
```

```
topics.dtm1 <- DocumentTermMatrix( topics.corpus_clean )   ## library = collection of words for all doc
class(topics.dtm1)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
inspect(topics.dtm1) # typeof(dtm1)  #length(dimnames(dtm1)$Terms)
```

```
## <<DocumentTermMatrix (documents: 3957, terms: 459)>>
## Non-/sparse entries: 34492/1781771
## Sparsity           : 98%
## Maximal term length: 14
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   anim chang cultur design global health human scienc social technolog
##   1134    0     0      0      2      0      1     0      2      0         1
##   1270    0     1      0      1      0      0     0      1      0         1
##   2090    0     1      1      0      2      0     1      0      0         0
##   2108    0     1      1      0      2      0     1      0      1         0
##   2118    0     2      0      0      2      0     1      0      1         1
##   2211    0     2      0      0      2      0     0      0      1         1
##   2241    0     0      0      0      2      0     0      0      0         0
##   2302    0     0      1      0      1      0     2      1      0         1
##   2367    0     1      0      3      1      0     1      1      1         1
##   2401    0     1      0      0      2      1     1      0      2         0
```

Take a look at the dtm.

```
colnames(transcript.dtm1)[7150:7161] # the last a few words in the bag
# another way to get list of words
# dimnames(dtm1)$Terms[7000:7161]
dim(as.matrix(transcript.dtm1))  # we use 7161 words as predictors

colnames(topics.dtm1)[300:310] # the last a few words in the bag
# another way to get list of words
# dimnames(dtm1)$Terms[7000:7161]
dim(as.matrix(topics.dtm1))  # we use 7161 words as predictors
```

Document 1, which is row1 in the dtm.

```
inspect(transcript.dtm1[1,])  #Non-/sparse entries: number of non-zero entries vs. number of zero entri
inspect(topics.dtm1[1,])  #Non-/sparse entries: number of non-zero entries vs. number of zero entries
```

transcript.dtm1 has 42 distinctive words; in other words, there are 42 non-zero cells out of 52898 bag of words. topics.dtm1 has 14 distinctive words; in other words, there 14 non-zero cells out of 459 bag of words.

```
as.matrix(transcript.dtm1[1, 1:50])  # most of the cells are 0
as.matrix(topics.dtm1[1, 1:50])  # most of the cells are 0
```

This is because review 1 only consists of 28 words after all the cleansing.

```
sum(as.matrix(transcript.dtm1[1,]))
sum(as.matrix(topics.dtm1[1,]))
```

We may

```
colnames(as.matrix(transcript.dtm1[1, ]))[which(as.matrix(transcript.dtm1[1, ]) != 0)]
colnames(as.matrix(topics.dtm1[1, ]))[which(as.matrix(topics.dtm1[1, ]) != 0)]
```

```
as.character(transcript.corpus[[1]]) #original text
as.character(topics.corpus[[1]]) #original text
```

We inspect a few rows and columns of `dtm1`.

```
inspect(transcript.dtm1[1:5, 1000:1010]) # very sparse
inspect(topics.dtm1[1:5, 300:310]) # very sparse
```

4.1.4 Reduce the size of the bag

Many words do not appear nearly as often as others. If your cleaning was done appropriately, it will hopefully not lose much of the information if we drop such rare words. So, we first cut the bag to only include the words appearing at least 1% (or the frequency of your choice) of the time. This reduces the dimension of the features extracted to be analyzed.

```
transcript.threshold <- .005*length(transcript.corpus_clean)   # 1% of the total documents
transcript.words.05 <- findFreqTerms(transcript.dtm1, lowfreq=transcript.threshold)  # words appearing
length(transcript.words.05)
transcript.words.05[580:600]

topics.threshold <- .005*length(topics.corpus_clean)   # 1% of the total documents
topics.words.05 <- findFreqTerms(topics.dtm1, lowfreq=topics.threshold)  # words appearing at least amo
length(topics.words.05)
topics.words.05[200:210]
```

```
transcript.dtm.05 <- DocumentTermMatrix(transcript.corpus_clean, control = list(dictionary = transcript
dim(as.matrix(transcript.dtm.05))
```

```
colnames(transcript.dtm.05)[40:50]

topics.dtm.05 <- DocumentTermMatrix(topics.corpus_clean, control = list(dictionary = topics.words.05))
dim(as.matrix(topics.dtm.05))
colnames(topics.dtm.05)[40:50]
```

`removeSparseTerms()`:

Anther way to reduce the size of the bag is to use `removeSparseTerms`

```
transcript.dtm.05.2 <- removeSparseTerms(transcript.dtm1, 1-.005)  # control sparsity < .99
inspect(transcript.dtm.05.2)
# colnames(dtm.05.2)[1:50]
# words that are in dtm.05 but not in dtm.05.2
colnames(transcript.dtm.05)[!(colnames(transcript.dtm.05) %in% colnames(transcript.dtm.05.2))]

topics.dtm.05.2 <- removeSparseTerms(topics.dtm1, 1-.005)  # control sparsity < .99
inspect(topics.dtm.05.2)
# colnames(dtm.05.2)[1:50]
# words that are in dtm.05 but not in dtm.05.2
colnames(topics.dtm.05)[!(colnames(topics.dtm.05) %in% colnames(topics.dtm.05.2))]
```

We end up with two different bags because

- `findFreqTerms()`: counts a word multiple times if it appears multiple times in one document.
- `removeSparseTerms()`: keep words that appear at least once in X% of documents.

4.1.5 One step to get DTM

We consolidate all possible processing steps to the following clean `R-chunk`, turning texts (input) into `Document Term Frequency` which is a sparse matrix (output) to be used in the down-stream analyses.

All the `tm_map()` can be called inside `DocumentTermMatrix` under parameter called `control`. Here is how.

```
# Turn texts to corpus
transcript.corpus  <- VCorpus(VectorSource(transcript.text))


# Control list for creating our DTM within DocumentTermMatrix
# Can tweak settings based off if you want punctuation, numbers, etc.
control_list <- list( tolower = TRUE,
                      removePunctuation = TRUE,
                      removeNumbers = TRUE,
                      stopwords = stopwords("english"),
                      # stopwords = TRUE,
                      stemming = TRUE)
# dtm with all terms:
transcript.dtm.05.long  <- DocumentTermMatrix(transcript.corpus, control = control_list)
#inspect(dtm.05.long)

# kick out rare words
transcript.dtm.05<- removeSparseTerms(transcript.dtm.05.long, 1-.005)
inspect(transcript.dtm.05)
```

```
## <<DocumentTermMatrix (documents: 3957, terms: 6916)>>
## Non-/sparse entries: 1495946/25870666
## Sparsity           : 95%
## Maximal term length: 16
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   can get just know like now one peopl thing think
##   2318  41  17   40   22   40  31  25    75    30    71
##   2349  27  20   41   11   39  16  22    77    18    68
##   348   26  27   13   24   22  37  38    65    34    21
##   3723  60  15   29   26   22  25  33    47    24    29
##   3913  43  58   29   70   24  26  26    80    41    16
##   3918  68  58   23   43   38  43  40    63    26    63
##   3919  37  32   33   47   32  62  38    79    35    74
##   3926  90  25   49   32   33  60  43    62    37    80
##   3930  23  40   35   29   32  12  25    60    41    75
##   3935  40  36   25   56   31  20  21    44    36    58
```

```
# look at the document 1 before and after cleaning
# inspect(transcript.corpus[[1]])
# after cleaning
# colnames(as.matrix(transcript.dtm1[1, ]))[which(as.matrix(transcript.dtm1[1, ]) != 0)]





topics.corpus  <- VCorpus(VectorSource(topics.text))


# Control list for creating our DTM within DocumentTermMatrix
# Can tweak settings based off if you want punctuation, numbers, etc.
control_list <- list( tolower = TRUE,
                      removePunctuation = TRUE,
                      removeNumbers = TRUE,
                      stopwords = stopwords("english"),
                      # stopwords = TRUE,
                      stemming = TRUE)
# dtm with all terms:
topics.dtm.05.long  <- DocumentTermMatrix(topics.corpus, control = control_list)
#inspect(dtm.05.long)

# kick out rare words
topics.dtm.05<- removeSparseTerms(topics.dtm.05.long, 1-.005)
inspect(topics.dtm.05)
```

```
## <<DocumentTermMatrix (documents: 3957, terms: 288)>>
## Non-/sparse entries: 32698/1106918
## Sparsity           : 97%
## Maximal term length: 13
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   anim chang cultur design global health human scienc social technolog
##   1270    0     1      0      1      0      0     0      1      0         1
##   2090    0     1      1      0      2      0     1      0      0         0
##   2108     0     1      1      0      2      0     1      0      1         0
##   2118     0     2      0      0      2      0     1      0      1         1
##   2191     0     2      0      0      2      0     1      0      1         0
##   2302     0     0      1      0      1      0     2      1      0         1
##   2318     0     2      0      0      2      0     2      0      1         1
##   2367     0     1      0      3      1      0     1      1      1         1
##   2401     0     1      0      0      2      1     1      0      2         0
##   3234     1     0      2      0      0      0     1      0      0         0
```

```
# look at the document 1 before and after cleaning
# inspect(topics.corpus[[1]])
# after cleaning
# colnames(as.matrix(topics.dtm1[1, ]))[which(as.matrix(topics.dtm1[1, ]) != 0)]
```

5 N-grams and other extensions

We have included n-grams here as a reference. We also implemented how to get the DTM. But will skip this section.

5.1 What about word order?

The approach we have taken with bag of words may seem very naive. What we have done is essentially looked at the frequencies of word occurrences in each document while throwing out any of the grammatical structure related to word ordering, punctuation, etc.

Consider the following two sentences:

- "This restaurant is the worst, I can't stand the service here…"
- "I can't stand the worst service at other restaurants around here, but this place is an exception…"

When we read these, we realize that the *ordering* of the words plays an important role in the meaning. While both sentences might be treated similarly by bag-of-words, each having one occurrence of the word "worst", we know by the surrounding context that the first review is negative while the other is positive.

Fortunately, there are extensions to bag-of-words that let us account for these nuances related to word ordering and related semantics! Instead of looking at the frequencies of individual words, what if we look at the frequencies of pairs, triples, or even n consecutive words? N-grams do exactly this, typically looking at the number of occurrences of all n-tuples of consecutive words from all of our documents of interest. There are even further generalizations like skip-grams that let us look at consecutive sequences of words with some gaps in the middle, but this is beyond anything that you will be required to do in the class (though it could be interesting to try for your final project!).

While this gives the added benefit of capturing more nuances in the language, you can probably imagine that this would blow up our feature space really quickly, so in practice most people don't go beyond bi- or tri-grams. Like we had in the case of bag-of-words, most of our n-grams are also going to be pretty rare, especially for higher values of n, so we should keep in mind that our feature matrix is going to be even more sparse than before.

An example of bigrams on the sentence "I love data science" is "I love", "love data", and "data science". Generally, we also compute the 1-grams as well, which coincide with the bag-of-words from before.

5.2 N-grams in R

Using the `tm` package along with a custom tokenizer, we can now implement n-grams using R! As is, the tokenizer is set to produce bigrams but you should be able to change this by tweaking the variable `n`.

We next prepare a clean corpus

```
transcript.corpus  <- VCorpus(VectorSource(transcript.text))
topics.corpus  <- VCorpus(VectorSource(topics.text))

# control_list <- list( tolower = TRUE,
#                       removePunctuation = TRUE,
#                       removeNumbers = TRUE,
#                       stopwords = stopwords("english"),
#                       stemming = TRUE)
# # dtm with all terms:
# dtm.10.long  <- DocumentTermMatrix(transcript.corpus, control = control_list)


# Converts all words to lowercase
transcript.corpus_clean <- tm_map(transcript.corpus, content_transformer(tolower))
topics.corpus_clean <- tm_map(topics.corpus, content_transformer(tolower))

# Removes common English stopwords (e.g. "with", "i")
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removeWords, stopwords("english"))
topics.corpus_clean <- tm_map(topics.corpus_clean, removeWords, stopwords("english"))

# Removes any punctuation
# NOTE: This step may not be appropriate if you want to account for differences
#       on semantics depending on which sentence a word belongs to if you end up
#       using n-grams or k-skip-n-grams.
#       Instead, periods (or semicolons, etc.) can be replaced with a unique
#       token (e.g. "[PERIOD]") that retains this semantic meaning.
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removePunctuation)
topics.corpus_clean <- tm_map(topics.corpus_clean, removePunctuation)

# Removes numbers
transcript.corpus_clean <- tm_map(transcript.corpus_clean, removeNumbers)
topics.corpus_clean <- tm_map(topics.corpus_clean, removeNumbers)

# Stem words
transcript.corpus_clean <- tm_map(transcript.corpus_clean, stemDocument, lazy = TRUE)
topics.corpus_clean <- tm_map(topics.corpus_clean, stemDocument, lazy = TRUE)




transcript.text_clean <- data.frame(text = unlist(sapply(transcript.corpus_clean, `[`, "content")),
    stringsAsFactors=F)

topics.text_clean <- data.frame(text = unlist(sapply(topics.corpus_clean, `[`, "content")),
    stringsAsFactors=F)

data_sub_gender_clean = data.frame(data_sub_gender, transcript.text_clean, topics.text_clean)
data_sub_gender_clean = data_sub_gender_clean %>% mutate(transcript_clean = text, topics_clean = text.1
names(data_sub_gender_clean)
```

The following part runs too slow. So I dropped the bigram idea.

```
# data_sub_gender_clean_trans2gram <- data_sub_gender_clean %>%
#   unnest_tokens(bigram, transcript_clean, token = "ngrams", n = 2) %>% count(talk_id,bigram) %>% spre
```

```
#
#
# # colSums(!is.na(data_sub_gender_clean_trans2gram))
#
#
# data_sub_gender_clean_trans2gram[,2:dim(data_sub_gender_clean_trans2gram)[2]][is.na(data_sub_gender_c
# data_sub_gender_clean_trans2gram[,2:dim(data_sub_gender_clean_trans2gram)[2]][is.na(data_sub_gender_c
#
#
# thresh = dim(data_sub_gender_clean_trans2gram)[1]*0.005
#
# temp = data_sub_gender_clean_trans2gram %>% colSums()
# bigrams_trans = data_sub_gender_clean_trans2gram %>% select(which(temp[2:length(temp)] > thresh)+1)
#
# colnames(bigrams_trans)
```

6 Analyses

Once we have turned a text into a vector, we can then apply any methods suitable for the settings. In our case we will use logistic regression models and LASSO to explore the relationship between `female` and `text`.

6.1 Data preparation

The following chunk output a data frame called data2, combining “user_id”, “stars”, “date”, “female” and all the tf’s of each word.

```
names(data_sub_gender)



# Combine the original data with the text matrix
# transcript.data1.temp <- data.frame(data_sub_gender, as.matrix(transcript.dtm.05), bigrams_trans)
transcript.data1.temp <- data.frame(data_sub_gender, as.matrix(transcript.dtm.05))
dim(transcript.data1.temp)
names(transcript.data1.temp)[1:30]
#str(data1.temp)

# data2 consists of date, female and all the top 1% words
# data2 <- transcript.data1.temp[, c(1,7, 8,11, 14:ncol(transcript.data1.temp))]
# names(data2)[1:20]
# dim(data2)  ### remember we have only run 1000 rows

#### We have previously run the entire 100,000 rows and output the DTM out.
### if not, run and write as csv
if(!file.exists("data/ted_en_tm_freq_transcript.csv")) {
  fwrite(transcript.data1.temp, "data/ted_en_tm_freq_transcript.csv", row.names = FALSE)
}




# Combine the original data with the text matrix
topics.data1.temp <- data.frame(data_sub_gender,as.matrix(topics.dtm.05))
dim(topics.data1.temp)
names(topics.data1.temp)[1:30]
```

```r
#str(data1.temp)

# data2 consists of date, female and all the top 1% words
# data2 <- topics.data1.temp[, c(1,7, 8,11, 14:ncol(topics.data1.temp))]
# names(data2)[1:20]
# dim(data2)  ### remember we have only run 1000 rows

#### We have previously run the entire 100,000 rows and output the DTM out.
### if not, run and write as csv
if(!file.exists("data/ted_en_tm_freq_topics.csv")) {
  fwrite(topics.data1.temp, "data/ted_en_tm_freq_topics.csv", row.names = FALSE)
}
```

6.2 Splitting data

Let's first read in the processed data with text being a vector.

```r
transcript.data2 <- fread("data/ted_en_tm_freq_transcript.csv")  #dim(data2)
names(transcript.data2)[1:20] # notice that user_id, stars and date are in the data2
dim(transcript.data2)
transcript.data2$female <- as.factor(transcript.data2$female)
table(transcript.data2$female)
#str(transcript.data2)  object.size(transcript.data2)  119Mb




topics.data2 <- fread("data/ted_en_tm_freq_topics.csv")  #dim(data2)
names(topics.data2)[1:20] # notice that user_id, stars and date are in the data2
dim(topics.data2)
topics.data2$female <- as.factor(topics.data2$female)
table(topics.data2$female)
#str(topics.data2)  object.size(topics.data2)  49Mb
```

As one standard machine learning process, we first split data into two sets one training data and the other testing data. We use training data to build models, choose models etc and make final recommendations. We then report the performance using the testing data.

Reserve 10000 randomly chosen rows as our test data (`data2.test`) and the remaining 90000 as the training data (`data2.train`)

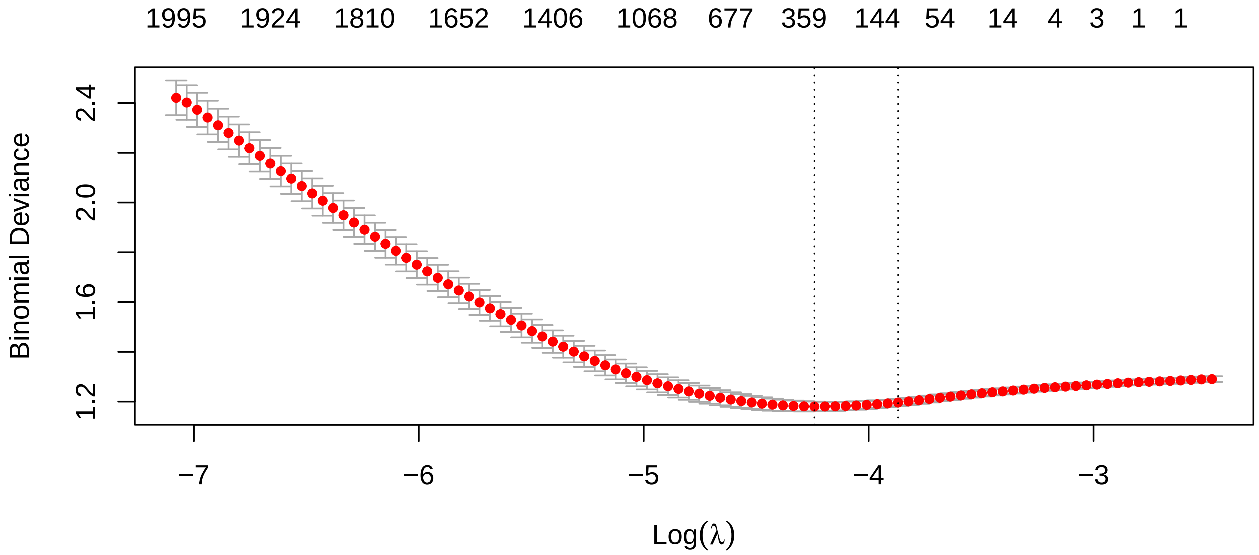
```r
set.seed(911)  # for the purpose of reporducibility
transcript.n <- nrow(transcript.data2)
transcript.test.index <- sample(transcript.n, round(0.1*transcript.n))
# length(test.index)
transcript.data2.test <- transcript.data2[transcript.test.index, -c(1:16)] # only keep female and the t
transcript.data2.train <- transcript.data2[-transcript.test.index, -c(1:16)]
dim(transcript.data2.train)

set.seed(1)  # for the purpose of reporducibility
topics.n <- nrow(topics.data2)
topics.test.index <- sample(topics.n, round(0.1*transcript.n))
# length(test.index)
topics.data2.test <- topics.data2[topics.test.index, -c(1:16)] # only keep female and the texts
topics.data2.train <- topics.data2[-topics.test.index, -c(1:16)]
dim(topics.data2.train)
```

6.3 Analysis 1: LASSO

We first explore a logistic regression model using LASSO. The following R-chunk runs a LASSO model with $\alpha = .99$. The reason we take an elastic net is to enjoy the nice properties from both `LASSO` (impose sparsity) and `Ridge` (computationally stable).

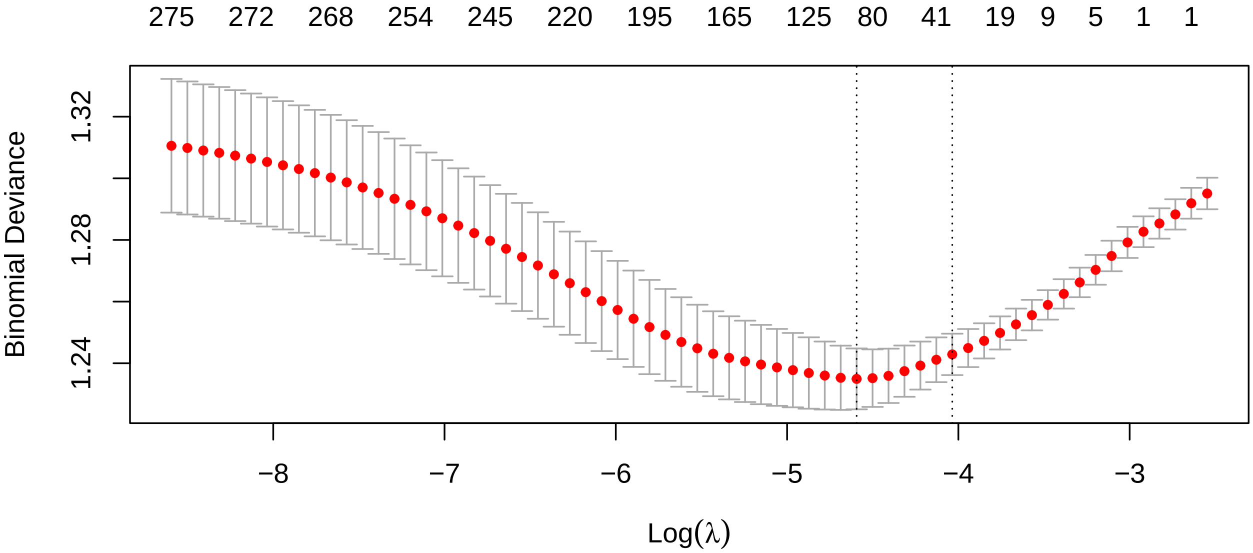
LASSO takes sparse design matrix as an input. So make sure to extract the sparse matrix first as the input in cv.glm(). It takes about 1 minute to run cv.glm() with sparse matrix or 11 minutes using the regular design matrix.

```r
### or try `sparse.model.matrix()` which is much faster
y <- transcript.data2.train$female
X1 <- sparse.model.matrix(female~., data=transcript.data2.train)[, -1]

set.seed(2)
transcript.result.lasso <- cv.glmnet(X1, y, alpha=.99, family="binomial")
# 1.25 minutes in my MAC
plot(transcript.result.lasso)
```



```r
# this this may take you long time to run, we save result.lasso
saveRDS(transcript.result.lasso, file="data/TextMining_lasso_transcript.RDS")
# result.lasso can be assigned back by
# result.lasso <- readRDS("data/TextMining_lasso.RDS")

# number of non-zero words picked up by LASSO when using lambda.1se
coef.1se <- coef(transcript.result.lasso, s="lambda.1se")
lasso.words <- coef.1se@Dimnames[[1]] [coef.1se@i][-1] # non-zero variables without intercept.
summary(lasso.words)


# or our old way
coef.1se <- coef(transcript.result.lasso, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
lasso.words <- rownames(as.matrix(coef.1se))[-1]
summary(lasso.words)
```

```
y <- topics.data2.train$female
X1 <- sparse.model.matrix(female~., data=topics.data2.train)[, -1]

set.seed(2)
topics.result.lasso <- cv.glmnet(X1, y, alpha=.99, family="binomial")
# 1.25 minutes in my MAC
plot(topics.result.lasso)
```

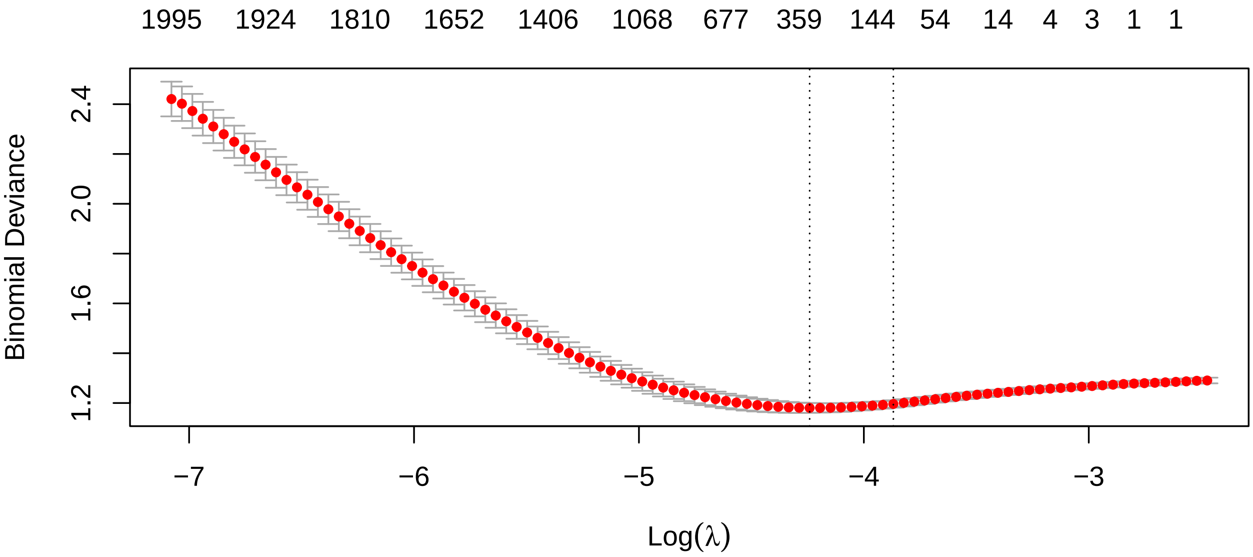


```
# this this may take you long time to run, we save result.lasso
saveRDS(topics.result.lasso, file="data/TextMining_lasso_topics.RDS")
# result.lasso can be assigned back by
# result.lasso <- readRDS("data/TextMining_lasso.RDS")


### cv.glmnt with the non-sparse design matrix takes much longer
# X <- as.matrix(data2.train[, -1]) # we can use as.matrix directly her
#### Be careful to run the following LASSO.
#set.seed(2)
#result.lasso <- cv.glmnet(X, y, alpha=.99, family="binomial")
# 10 minutes in my MAC
#plot(result.lasso)
```

Try to kick out some not useful words (Warning: this may crash your laptop!!!) Because of the computational burden, I have saved the LASSO results and other results into `TextMining_lasso.RDS` and `TextMining_glm.RDS`.

We resume our analyses by loading the `LASSO` results here. We extract useful variables using `lambda.1se`

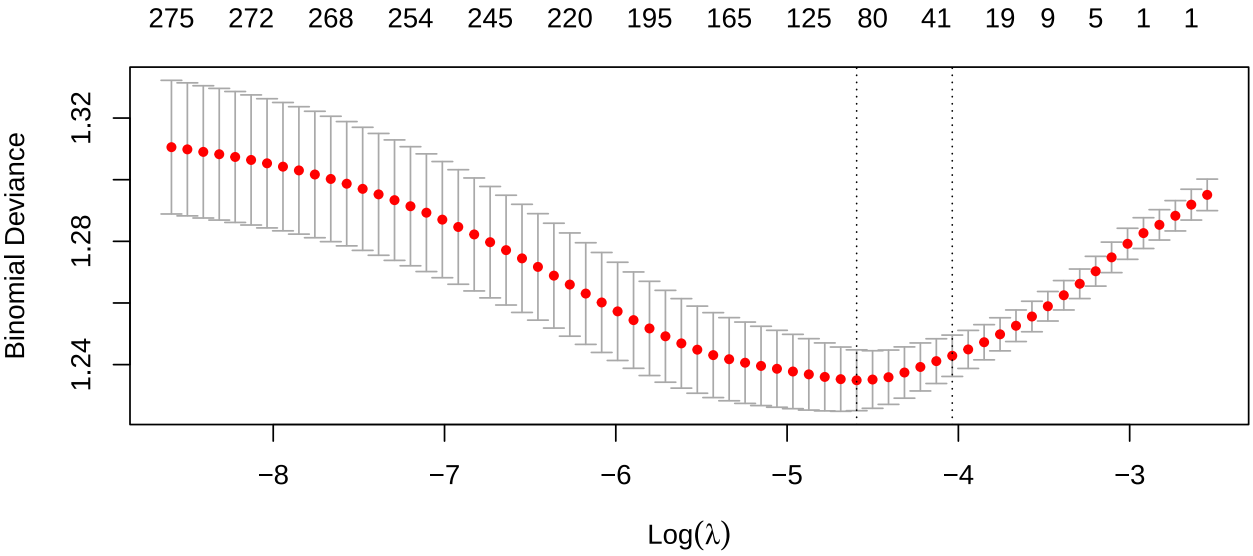
```
transcript.result.lasso <- readRDS("data/TextMining_lasso_transcript.RDS")
plot(transcript.result.lasso)
```



```
coef.1se <- coef(transcript.result.lasso, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
transcript.lasso.words <- rownames(as.matrix(coef.1se))[-1]
summary(transcript.lasso.words)
```

```
##    Length     Class      Mode 
##       106 character character 
```

```
# lasso.words[100:120]

topics.result.lasso <- readRDS("data/TextMining_lasso_topics.RDS")
plot(topics.result.lasso)
```



```
coef.1se <- coef(topics.result.lasso, s="lambda.1se")
coef.1se <- coef.1se[which(coef.1se !=0),]
topics.lasso.words <- rownames(as.matrix(coef.1se))[-1]
summary(topics.lasso.words)
```

```
##    Length     Class      Mode 
##        34 character character 
```

6.4 Analysis 2: Relaxed LASSO

As an alternative model we will run our relaxed LASSO. Input variables are chosen by LASSO and we get a regular logistic regression model. Once again it is stored as result.glm in TextMining.RData.

```
transcript.sel_cols <- c("female", transcript.lasso.words)
# use all_of() to specify we would like to select variables in sel_cols
transcript.data_sub <- transcript.data2.train %>% select(all_of(transcript.sel_cols))
transcript.result.glm <- glm(female~., family=binomial, transcript.data_sub) # takes 3.5 minutes
## glm() returns a big object with unnecessary information
# saveRDS(result.glm,
#      file = "data/TextMining_glm.RDS")

## trim the glm() fat from
## https://win-vector.com/2014/05/30/trimming-the-fat-from-glm-models-in-r/
stripGlmLR = function(cm) {
  cm$y = c()
  cm$model = c()

  cm$residuals = c()
  cm$fitted.values = c()
  cm$effects = c()
  cm$qr$qr = c()
  cm$linear.predictors = c()
  cm$weights = c()
  cm$prior.weights = c()
  cm$data = c()


  cm$family$variance = c()
  cm$family$dev.resids = c()
  cm$family$aic = c()
  cm$family$validmu = c()
  cm$family$simulate = c()
  attr(cm$terms,".Environment") = c()
  attr(cm$formula,".Environment") = c()

  cm
}

transcript.result.glm.small <- stripGlmLR(transcript.result.glm)

saveRDS(transcript.result.glm.small,
     file = "data/TextMining_glm_small_transcript.RDS")




topics.sel_cols <- c("female", topics.lasso.words)
# use all_of() to specify we would like to select variables in sel_cols
topics.data_sub <- topics.data2.train %>% select(all_of(topics.sel_cols))
topics.result.glm <- glm(female~., family=binomial, topics.data_sub) # takes 3.5 minutes
## glm() returns a big object with unnecessary information
# saveRDS(result.glm,
#      file = "data/TextMining_glm.RDS")
```

```
topics.result.glm.small <- stripGlmLR(topics.result.glm)

saveRDS(topics.result.glm.small,
     file = "data/TextMining_glm_small_topics.RDS")
```

6.5 Analysis 3: Word cloud! (Sentiment analysis)

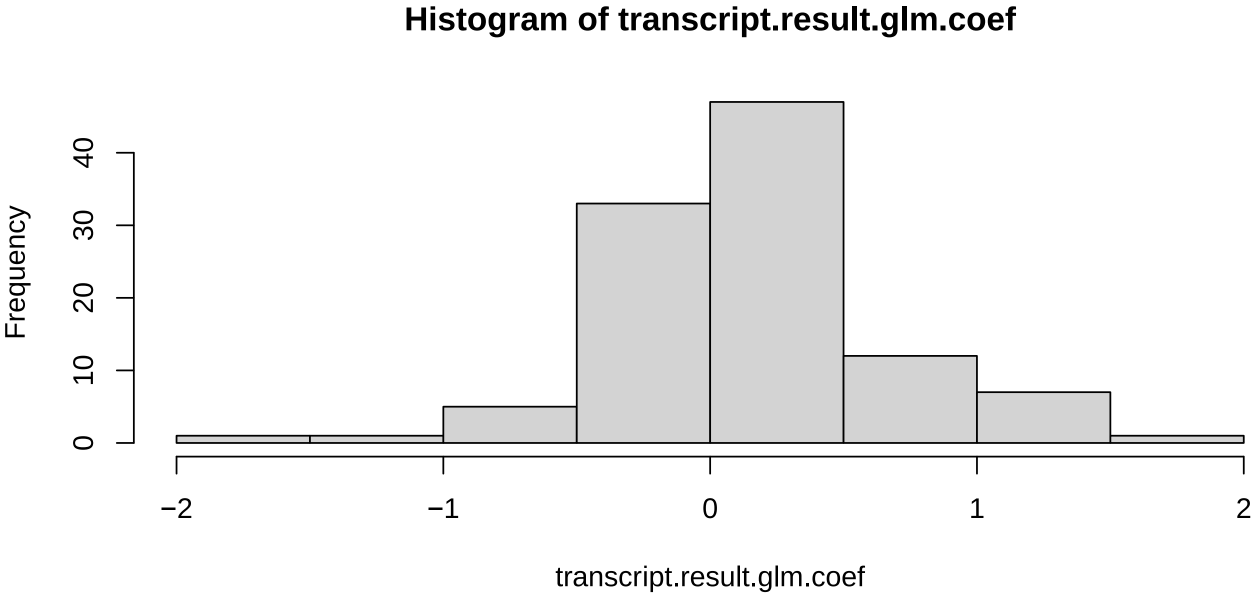
Logistic regression model connects the chance of `being female` given a text What are the `female` (or feminine) words and how much it influence the chance being female? In addition to explore the set of good words we also build word clouds to visualize the correlation between positive words and negative words.

1. Order the glm positive coefficients (positive words). Show them in a word cloud. The size of the words indicates the strength of positive correlation between that word and the chance being a female speaker
2. Order the glm negative coefficients (negative words)

TIME TO PLOT A WORD CLOUD!! Plot the world clouds, the size of the words are prop to the logistic reg coef's

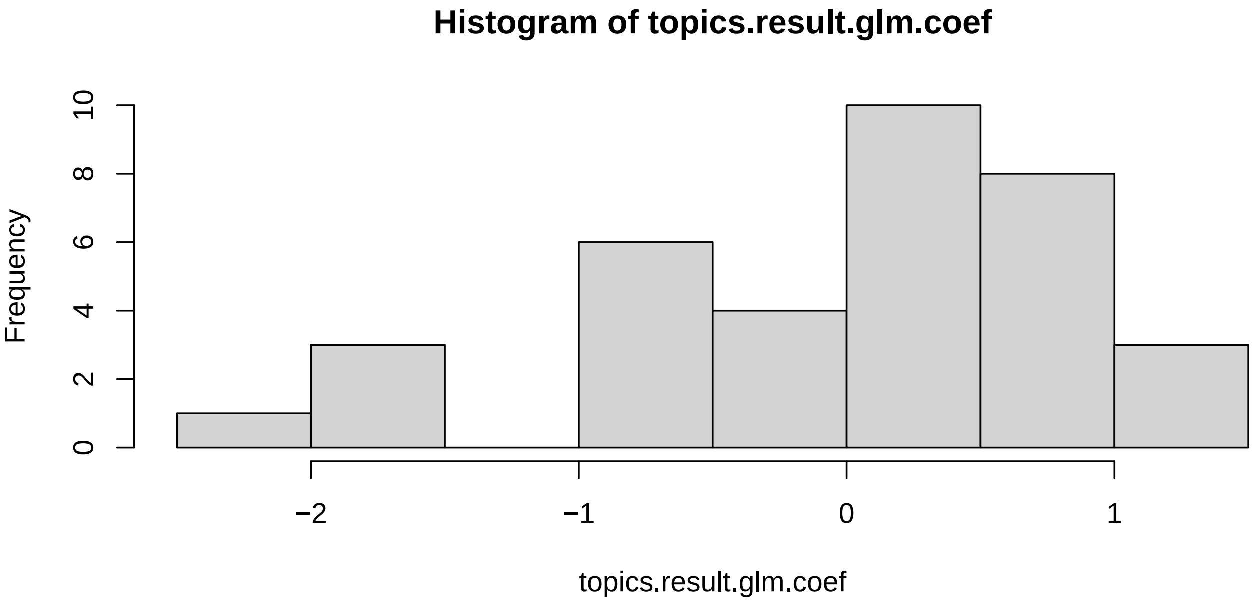
Positive word cloud:

```
transcript.result.glm <- readRDS("data/TextMining_glm_small_transcript.RDS")
transcript.result.glm.coef <- coef(transcript.result.glm)
transcript.result.glm.coef[200:250]
hist(transcript.result.glm.coef)
```



```
# pick up the positive coef's which are positively related to the prob of being a good review
transcript.female.glm <- transcript.result.glm.coef[which(transcript.result.glm.coef > 0)]
# female.glm <- female.glm[-1]  # took intercept out
names(transcript.female.glm)[1:20]  # which words are positively associated with being female

transcript.female.fre <- sort(transcript.female.glm, decreasing = TRUE) # sort the coef's
round(transcript.female.fre, 4)[1:20] # leading 20 positive words, amazing!
length(transcript.female.fre)  # 390 good words

# hist(as.matrix(female.fre), breaks=30, col="red")
transcript.female.word <- names(transcript.female.fre)  # good words with a decreasing order in the coe
```

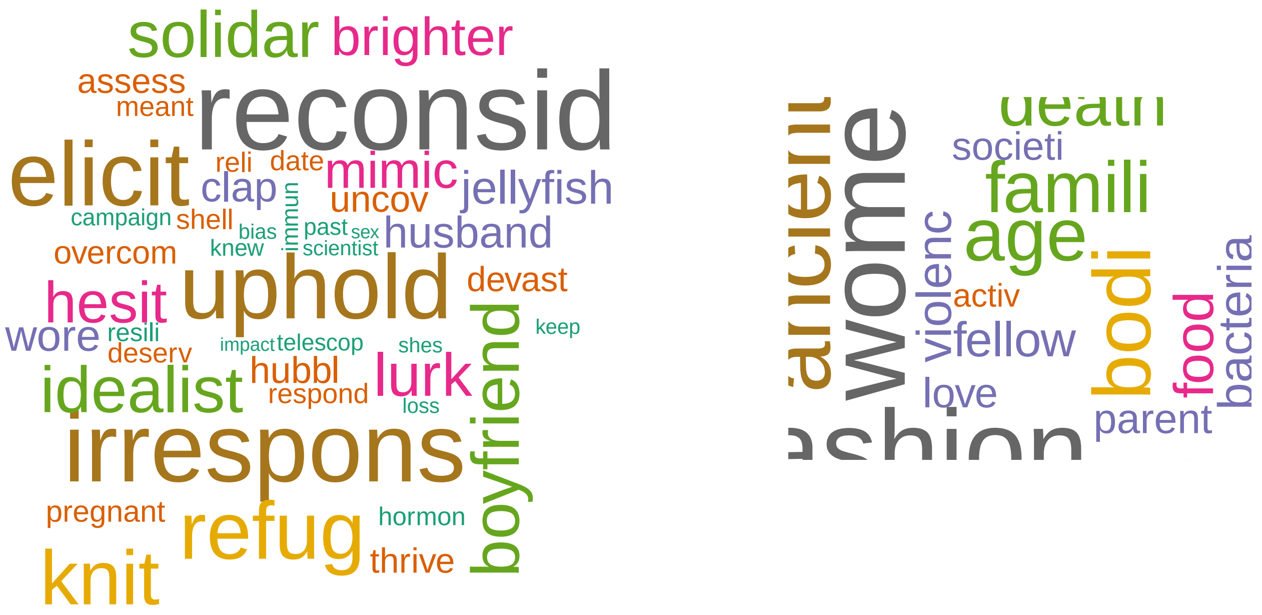
```r
topics.result.glm <- readRDS("data/TextMining_glm_small_topics.RDS")
topics.result.glm.coef <- coef(topics.result.glm)
topics.result.glm.coef[200:250]
hist(topics.result.glm.coef)
```



```r
# pick up the positive coef's which are positively related to the prob of being a good review
topics.female.glm <- topics.result.glm.coef[which(topics.result.glm.coef > 0)]
# female.glm <- female.glm[-1]  # took intercept out
names(topics.female.glm)[1:20]  # which words are positively associated with female speakers

topics.female.fre <- sort(topics.female.glm, decreasing = TRUE) # sort the coef's
round(topics.female.fre, 4)[1:20] # leading 20 positive words, amazing!
length(topics.female.fre)  # 390 good words

# hist(as.matrix(female.fre), breaks=30, col="red")
topics.female.word <- names(topics.female.fre)  # good words with a decreasing order in the coeff's
```

The above chunk shows in detail about the weight for positive words. We only show the positive word-cloud here. One can tell the large positive words are making sense in the way we do expect the collection of large words should have a positive tone towards the restaurant being reviewed.

```r
set.seed(1)
par(mfrow=c(1,2))

cor.special <- brewer.pal(8,"Dark2")  # set up a pretty color scheme
wordcloud(transcript.female.word[1:50], transcript.female.fre[1:50],  # make a word cloud
          colors=cor.special, ordered.colors=F)

wordcloud(topics.female.word[1:20], topics.female.fre[1:20],  # make a word cloud
          colors=cor.special, ordered.colors=F)
```



Concern: Many words got trimmed due to stemming? We may redo dtm without stemming?

Negative word cloud:

Similarly to the negative coef's which is positively correlated to the prob. of being a bad review

```
transcript.male.glm <- transcript.result.glm.coef[which(transcript.result.glm.coef < 0)]
transcript.male.glm <- transcript.male.glm[-1]  # took intercept out
# names(male.glm)[1:50]

cor.special <- brewer.pal(6,"Dark2")
transcript.male.fre <- sort(-transcript.male.glm, decreasing = TRUE)
round(transcript.male.fre, 4)[1:40]
```

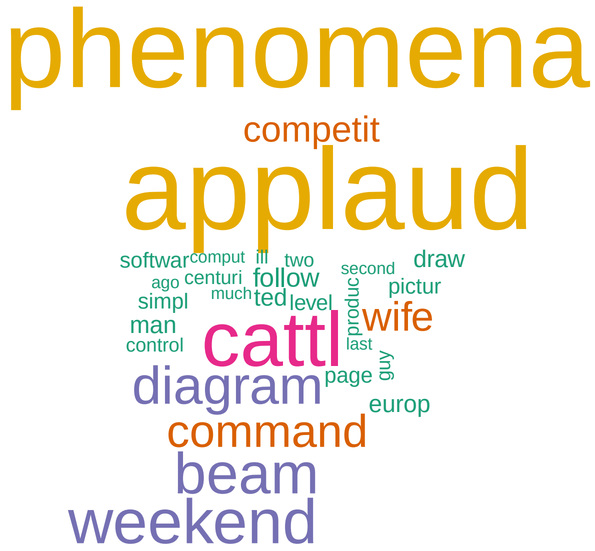
```
##   applaud phenomena     cattl   weekend      beam   diagram   command      wife
##    1.5747    1.4933    0.9892    0.7666    0.6751    0.5945    0.4745    0.4022
##  competit    follow      draw       ted     europ       man     level      page
##    0.3334    0.1750    0.1431    0.1385    0.1383    0.1316    0.1268    0.1249
##   softwar     simpl    pictur       ill   control   centuri    produc       guy
##    0.1188    0.1085    0.1069    0.0870    0.0819    0.0746    0.0726    0.0651
##       two       ago      much    second      last    comput   problem      will
##    0.0573    0.0560    0.0558    0.0476    0.0461    0.0445    0.0426    0.0365
##    someth      move      done      that       got       let       now      <NA>
##    0.0352    0.0348    0.0308    0.0134    0.0095    0.0092    0.0025        NA
```

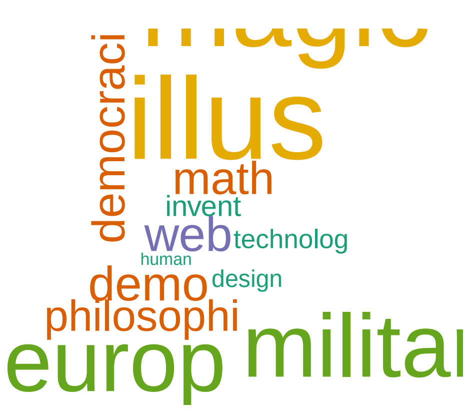
```
par(mfrow=c(1,2))


# hist(as.matrix(bad.fre), breaks=30, col="green")
set.seed(1)
transcript.male.word <- names(transcript.male.fre)
wordcloud(transcript.male.word[1:30], transcript.male.fre[1:30],
          color=cor.special, ordered.colors=F)
```

```
topics.male.glm <- topics.result.glm.coef[which(topics.result.glm.coef < 0)]
topics.male.glm <- topics.male.glm[-1]  # took intercept out
# names(male.glm)[1:50]

topics.male.fre <- sort(-topics.male.glm, decreasing = TRUE)
round(topics.male.fre, 4)[1:20]
```

```
##      illus      magic   militari      europ        web       demo       math
##     2.1240     1.9898     1.5608     1.5221     0.7243     0.7002     0.6766
##   democraci philosophi    invent  technolog     design      human       <NA>
##     0.6679     0.6284     0.3089     0.2672     0.2202     0.0486         NA
##       <NA>       <NA>       <NA>       <NA>       <NA>       <NA>
##         NA         NA         NA         NA         NA         NA
```

```
# hist(as.matrix(bad.fre), breaks=30, col="green")
set.seed(1)
topics.male.word <- names(topics.male.fre)
wordcloud(topics.male.word, topics.male.fre,
          color=cor.special, ordered.colors=F)
```

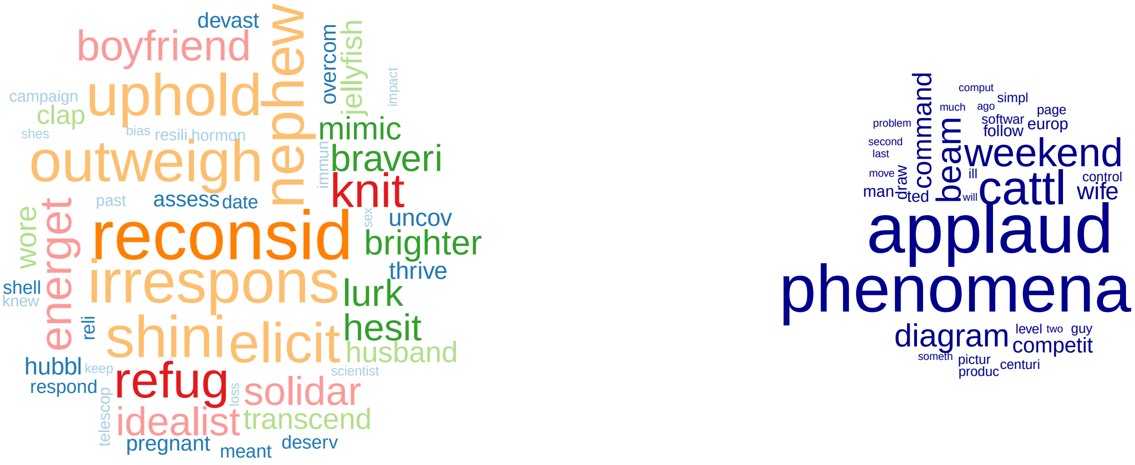




Put two clouds (female and male) together:

It seems that judging from the transcript, the most characterizing feature of a male TED talk is the "applaud" it receives. The "applaud" is positively related to the speaker being male.

```
# wordcloud(lords, scale=c(5,0.5), max.words=100, random.order=FALSE, rot.per=0.35, use.r.layout=FALSE,




set.seed(1)
par(mfrow=c(1,2))
# cor.special <- brewer.pal(8,"Dark2")
cor.female <- brewer.pal(8, "Paired")
cor.male <- brewer.pal(8, "Blues")
wordcloud(transcript.female.word[1:50], transcript.female.fre[1:50],
          colors=cor.female, ordered.colors=F, scale=c(2.2,0.3), max.words=100, random.order=FALSE, rot
wordcloud(transcript.male.word[1:34], transcript.male.fre[1:34],
          color="darkblue", ordered.colors=F, scale=c(2.2,0.3), max.words=100, random.order=FALSE, rot.
```



```
transcript.female.word
```

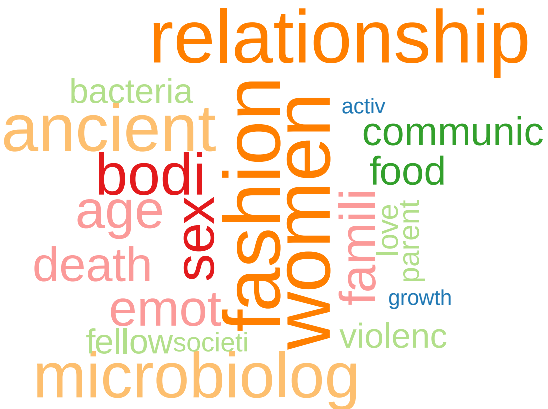
```
##  [1] "reconsid"  "irrespons" "nephew"    "outweigh"  "shini"     "uphold"
##  [7] "elicit"    "refug"     "knit"      "energet"   "boyfriend" "idealist"
## [13] "solidar"   "lurk"      "braveri"   "hesit"     "brighter"  "mimic"
## [19] "transcend" "jellyfish" "wore"      "husband"   "clap"      "uncov"
## [25] "hubbl"     "thrive"    "assess"    "devast"    "overcom"   "pregnant"
## [31] "shell"     "respond"   "reli"      "date"      "meant"     "deserv"
## [37] "resili"    "hormon"    "past"      "telescop"  "campaign"  "immun"
## [43] "knew"      "scientist" "keep"      "bias"      "loss"      "shes"
## [49] "impact"    "sex"       "studi"     "convers"   "help"      "woman"
## [55] "mom"       "word"      "bodi"      "women"     "food"      "often"
## [61] "babi"      "research"  "love"      "support"   "even"      "girl"
## [67] "sexual"
```

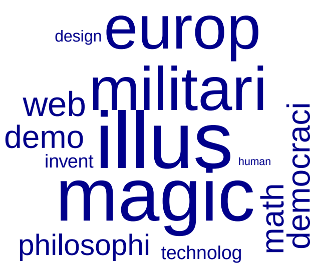
```
transcript.male.word
```

```
##  [1] "applaud"   "phenomena" "cattl"     "weekend"   "beam"      "diagram"
##  [7] "command"   "wife"      "competit"  "follow"    "draw"      "ted"
## [13] "europ"     "man"       "level"     "page"      "softwar"   "simpl"
## [19] "pictur"    "ill"       "control"   "centuri"   "produc"    "guy"
## [25] "two"       "ago"       "much"      "second"    "last"      "comput"
## [31] "problem"   "will"      "someth"    "move"      "done"      "that"
## [37] "got"       "let"       "now"
```

Judging from the topics, the most predicting topics of female talks are mostly relationship-oriented, whereas of the male talks are mostly object-oriented.

```
set.seed(1)

par(mfrow=c(1,2))
wordcloud(topics.female.word, topics.female.fre,
          colors=cor.female, ordered.colors=F, scale=c(2.5,0.3), max.words=100, random.order=FALSE, rot
wordcloud(topics.male.word, topics.male.fre,
          color="darkblue", ordered.colors=F, scale=c(2.5,0.3), max.words=100, random.order=FALSE, rot.
```





```
# par(mfrow=c(1,1))
```

6.6 Analysis 4: Predictions

We have obtained two sets of models one from `LASSO` the other from `relaxed LASSO`. To compare the performance as classifiers we will evaluate their `mis-classification error` and/or `ROC` curves using `testing data`.

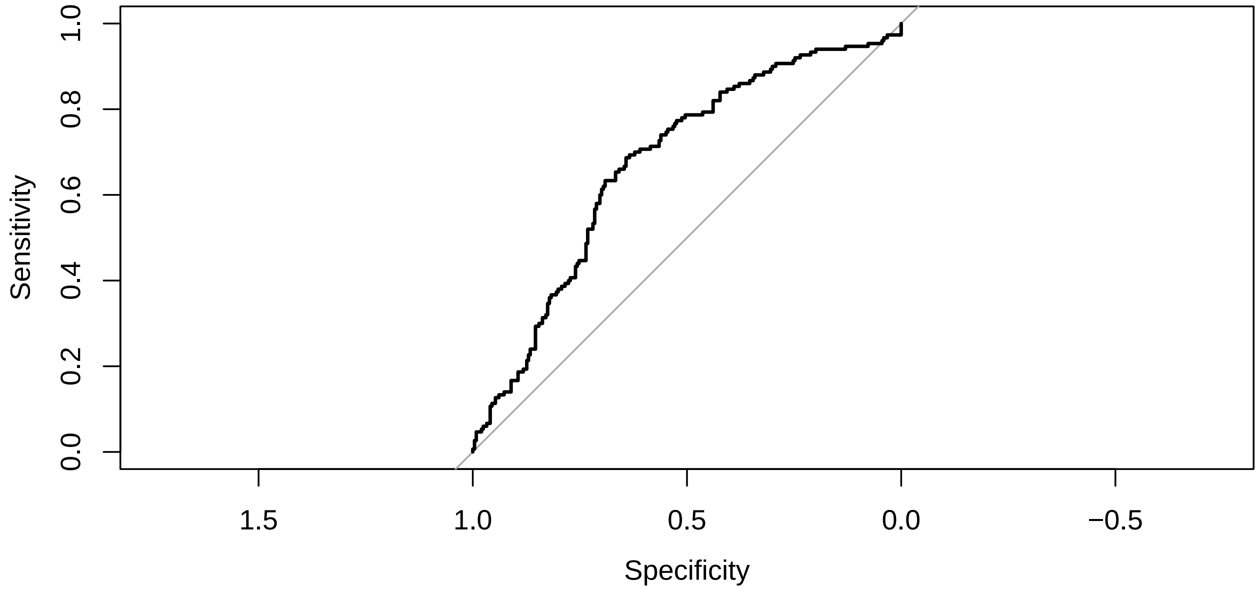
6.6.1 1) How does glm do in terms of classification?

Use the testing data we get mis-classification errors for one rule: majority vote.

```
transcript.predict.glm <- predict(transcript.result.glm, transcript.data2.test, type = "response")
transcript.class.glm <- ifelse(transcript.predict.glm > .5, "1", "0")
# length(class.glm)

transcript.testerror.glm <- mean(transcript.data2.test$female != transcript.class.glm)
transcript.testerror.glm   # mis classification error is 0.34

pROC::roc(transcript.data2.test$female, transcript.predict.glm, plot=T) # AUC=0.66
```

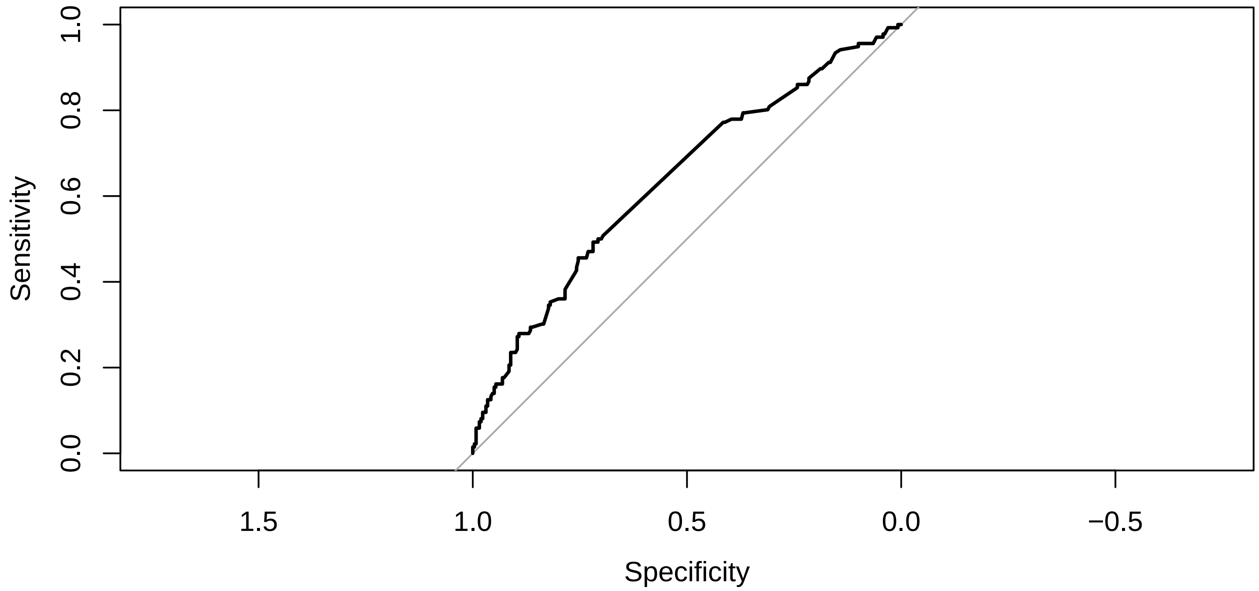
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```r
topics.predict.glm <- predict(topics.result.glm, topics.data2.test, type = "response")
topics.class.glm <- ifelse(topics.predict.glm > .5, "1", "0")
# length(class.glm)

topics.testerror.glm <- mean(topics.data2.test$female != topics.class.glm)
topics.testerror.glm   # mis classification error is 0.33

pROC::roc(topics.data2.test$female, topics.predict.glm, plot=T) # AUC=0.61
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
## [1] 0.3560606
##
## Call:
## roc.default(response = transcript.data2.test$female, predictor = transcript.predict.glm,     plot =
##
```

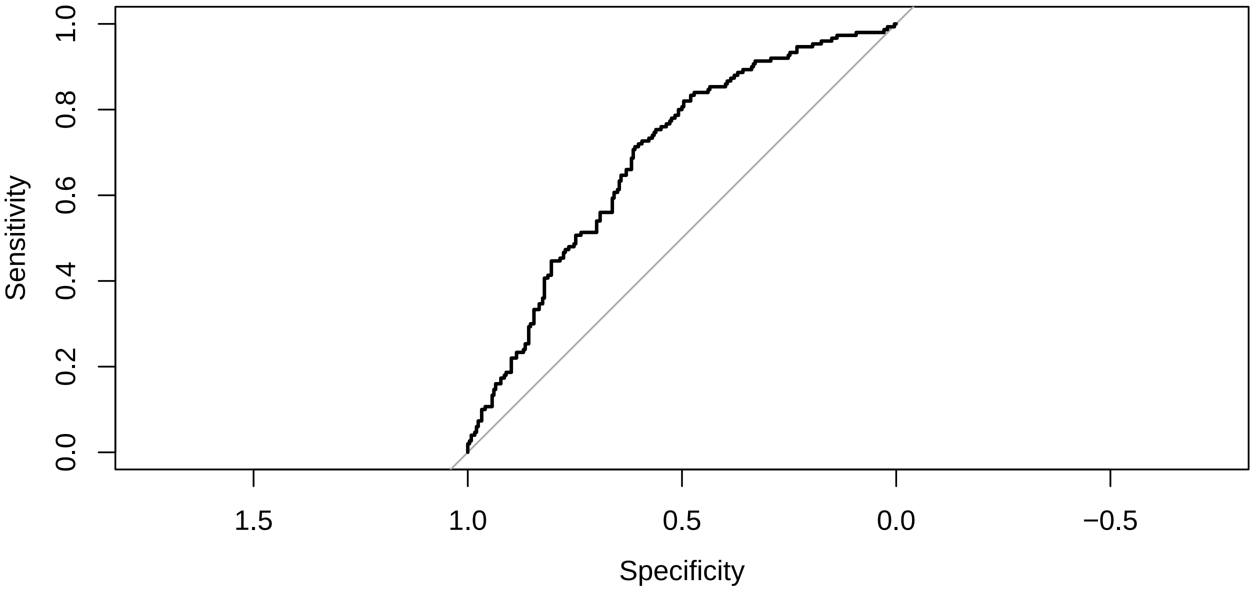
```
## Data: transcript.predict.glm in 246 controls (transcript.data2.test$female 0) < 150 cases (transcript
## Area under the curve: 0.6733
## [1] 0.3257576
##
## Call:
## roc.default(response = topics.data2.test$female, predictor = topics.predict.glm,     plot = T)
##
## Data: topics.predict.glm in 260 controls (topics.data2.test$female 0) < 136 cases (topics.data2.test$
## Area under the curve: 0.6355
```

6.6.2 2) LASSO model using `lambda.1se`

Once again we evaluate the testing performance of `LASSO` solution.

```
transcript.predict.lasso.p <- predict(transcript.result.lasso, as.matrix(transcript.data2.test[, -1]), t
  # output lasso estimates of prob's
transcript.predict.lasso <- predict(transcript.result.lasso, as.matrix(transcript.data2.test[, -1]), typ
  # output majority vote labels

# LASSO testing errors
mean(transcript.data2.test$female != transcript.predict.lasso)   # 0.36

# ROC curve for LASSO estimates

pROC::roc(transcript.data2.test$female, transcript.predict.lasso.p, plot=TRUE) # AUC=0.67
```

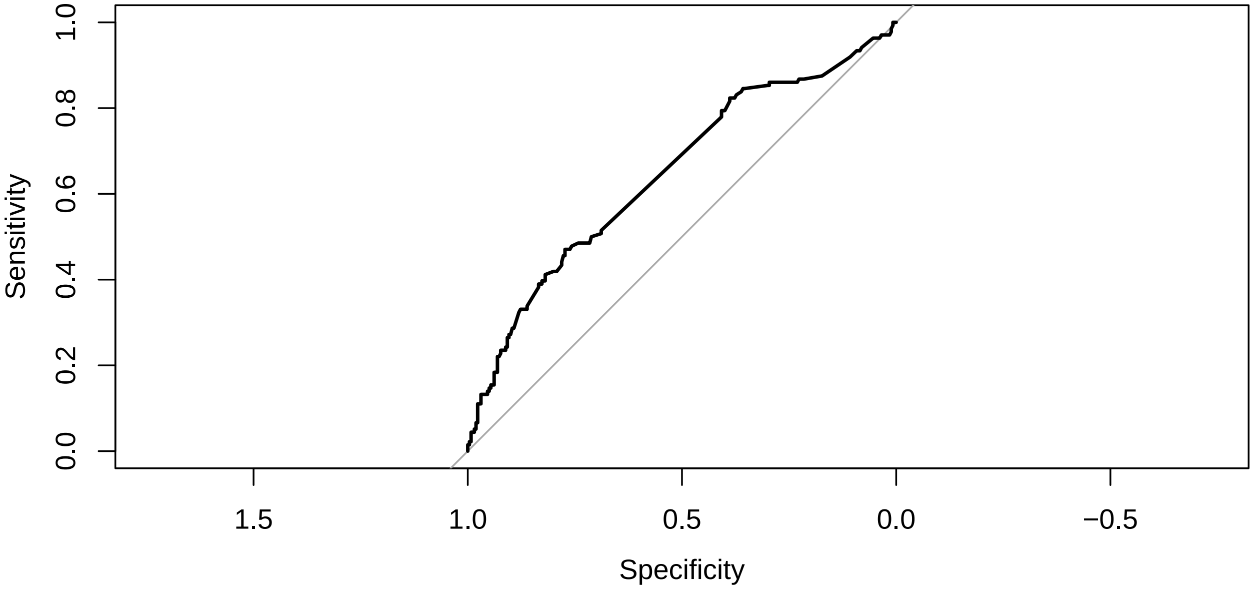
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
topics.predict.lasso.p <- predict(topics.result.lasso, as.matrix(topics.data2.test[, -1]), type = "resp
  # output lasso estimates of prob's
topics.predict.lasso <- predict(topics.result.lasso, as.matrix(topics.data2.test[, -1]), type = "class"
  # output majority vote labels

# LASSO testing errors
mean(topics.data2.test$female != topics.predict.lasso)   # 0.33
```

```
# ROC curve for LASSO estimates

pROC::roc(topics.data2.test$female, topics.predict.lasso.p, plot=TRUE) # AUC=0.63
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
## [1] 0.3611111
##
## Call:
## roc.default(response = transcript.data2.test$female, predictor = transcript.predict.lasso.p,     plo
##
## Data: transcript.predict.lasso.p in 246 controls (transcript.data2.test$female 0) < 150 cases (transc
## Area under the curve: 0.6889
## [1] 0.3232323
##
## Call:
## roc.default(response = topics.data2.test$female, predictor = topics.predict.lasso.p,     plot = TRUE
##
## Data: topics.predict.lasso.p in 260 controls (topics.data2.test$female 0) < 136 cases (topics.data2.t
## Area under the curve: 0.6467
```

Comparing the two predictions through testing errors/ROC we do not see much of the difference. We could use either final models for the purpose of the prediction.

6.7 Analysis 4: Predictions

We have obtained two sets of models one from `LASSO` the other from `relaxed LASSO`. To compare the performance as classifiers we will evaluate their `mis-classification error` and/or `ROC` curves using `testing data`.

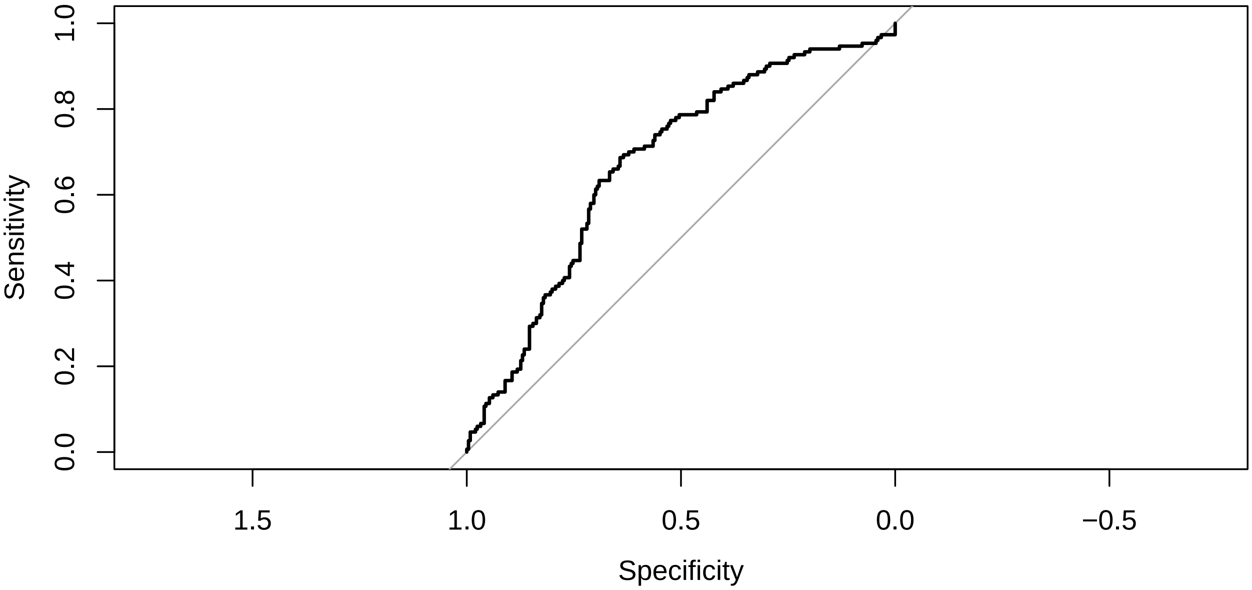
6.7.1 1) How does glm do in terms of classification?

Use the testing data we get mis-classification errors for one rule: majority vote.

```
transcript.predict.glm <- predict(transcript.result.glm, transcript.data2.test, type = "response")
transcript.class.glm <- ifelse(transcript.predict.glm > .5, "1", "0")
# length(class.glm)

transcript.testerror.glm <- mean(transcript.data2.test$female != transcript.class.glm)
transcript.testerror.glm   # mis classification error is 0.34

pROC::roc(transcript.data2.test$female, transcript.predict.glm, plot=T) # AUC=0.66
```

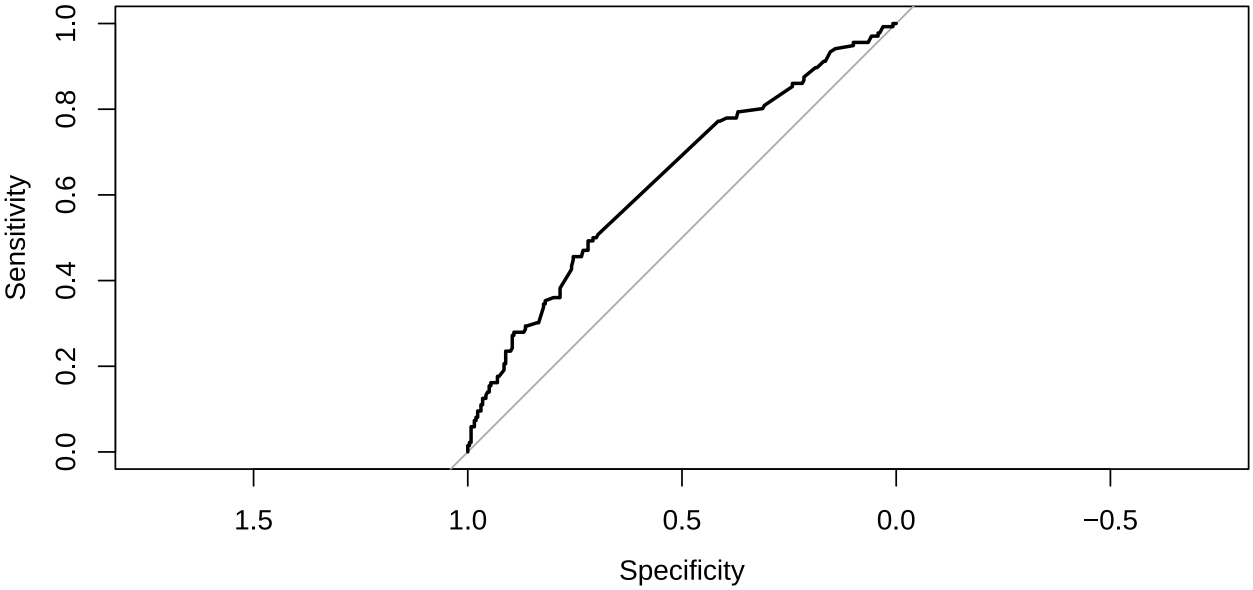
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
topics.predict.glm <- predict(topics.result.glm, topics.data2.test, type = "response")
topics.class.glm <- ifelse(topics.predict.glm > .5, "1", "0")
# length(class.glm)

topics.testerror.glm <- mean(topics.data2.test$female != topics.class.glm)
topics.testerror.glm   # mis classification error is 0.33

pROC::roc(topics.data2.test$female, topics.predict.glm, plot=T) # AUC=0.61
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
## [1] 0.3560606
##
## Call:
## roc.default(response = transcript.data2.test$female, predictor = transcript.predict.glm,     plot = T
##
## Data: transcript.predict.glm in 246 controls (transcript.data2.test$female 0) < 150 cases (transcrip
## Area under the curve: 0.6733
## [1] 0.3257576
##
## Call:
## roc.default(response = topics.data2.test$female, predictor = topics.predict.glm,     plot = T)
##
## Data: topics.predict.glm in 260 controls (topics.data2.test$female 0) < 136 cases (topics.data2.test$
## Area under the curve: 0.6355
```

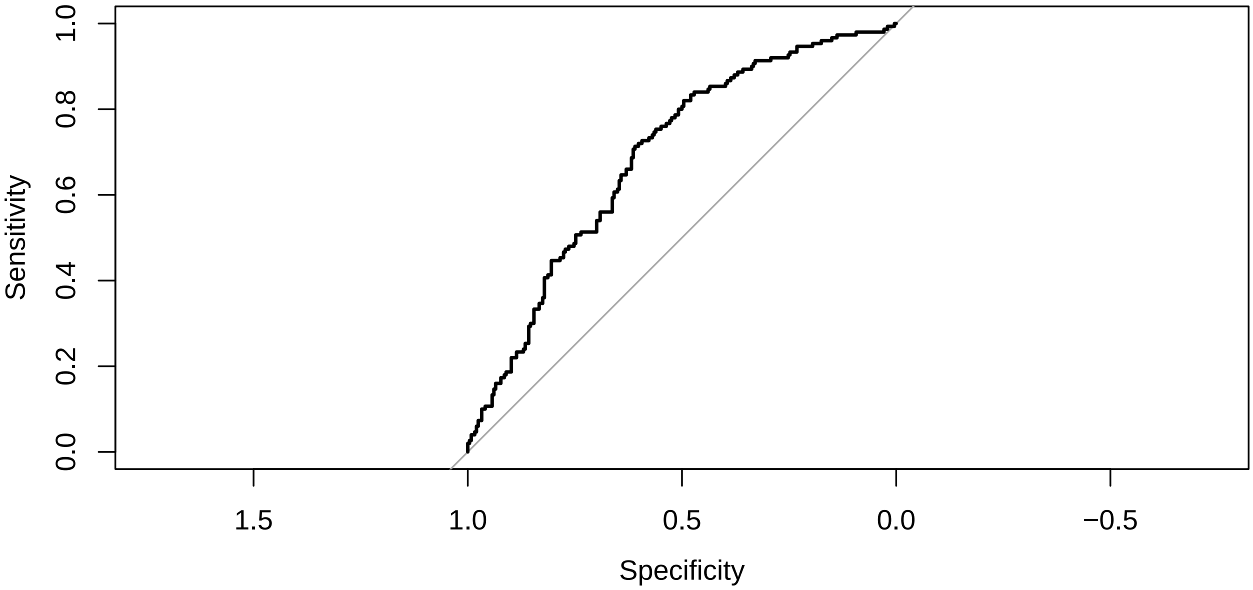
6.7.2 2) LASSO model using `lambda.1se`

Once again we evaluate the testing performance of `LASSO` solution.

```
transcript.predict.lasso.p <- predict(transcript.result.lasso, as.matrix(transcript.data2.test[, -1]), t
  # output lasso estimates of prob's
transcript.predict.lasso <- predict(transcript.result.lasso, as.matrix(transcript.data2.test[, -1]), typ
# output majority vote labels

# LASSO testing errors
mean(transcript.data2.test$female != transcript.predict.lasso)   # 0.36

# ROC curve for LASSO estimates

pROC::roc(transcript.data2.test$female, transcript.predict.lasso.p, plot=TRUE) # AUC=0.67
```

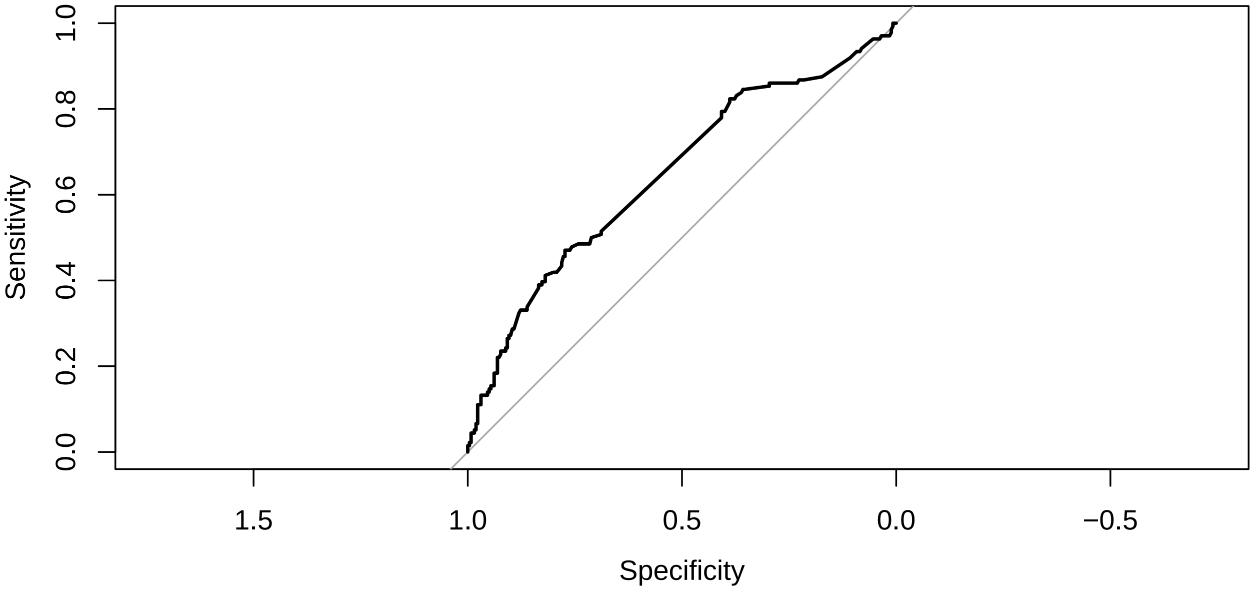
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
topics.predict.lasso.p <- predict(topics.result.lasso, as.matrix(topics.data2.test[, -1]), type = "respo
# output lasso estimates of prob's
topics.predict.lasso <- predict(topics.result.lasso, as.matrix(topics.data2.test[, -1]), type = "class"
# output majority vote labels

# LASSO testing errors
mean(topics.data2.test$female != topics.predict.lasso)   # 0.33

# ROC curve for LASSO estimates

pROC::roc(topics.data2.test$female, topics.predict.lasso.p, plot=TRUE) # AUC=0.63
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
## [1] 0.3611111
##
```

```
## Call:
## roc.default(response = transcript.data2.test$female, predictor = transcript.predict.lasso.p,     plo
##
## Data: transcript.predict.lasso.p in 246 controls (transcript.data2.test$female 0) < 150 cases (trans
## Area under the curve: 0.6889
## [1] 0.3232323
##
## Call:
## roc.default(response = topics.data2.test$female, predictor = topics.predict.lasso.p,     plot = TRUE
##
## Data: topics.predict.lasso.p in 260 controls (topics.data2.test$female 0) < 136 cases (topics.data2.
## Area under the curve: 0.6467
```

Comparing the two predictions through testing errors/ROC we do not see much of the difference. We could use either final models for the purpose of the prediction.

7 Conclusion

In this lecture, we apply LASSO to classify good/bad review based on the text. The core technique for text mining is a simple bag of words, i.e. a word frequency matrix. The problem becomes a high-dimensional problem. Using LASSO, we reduce dimension and train a model with high predictive power. Based on the model, we find out the positive/negative words and build a word cloud.

One natural question to ask about `bag of words` is to take certain weight for each word or a document length. There are some attempts to make adjustment. One way is called `TF-IDF`. We have tried this and it did not really help in reducing the errors. We left it in the Appendices.

There are many other models we can build. For example `trees` through `Random Forest` or `boosting` . We could also try to transform the features by `PCA` first then apply the methods mentioned here. We could also apply deep learning techniques to expand logistic regression models. Some methods are implemented and we have left these analyses into the Appendices as reference for you.