Lab 4 - Machine Learning for Social Science

To be handed in no later than October 23rd, 10:00. The submission should include code, relevant output, as well as answers to questions. We recommend the use of RMarkdown to create the report.

Part 1: Topic modeling

In this lab, we will use a data set containing a random sample of public Facebook posts by members of the U.S. Congress from 2017.[1] Our broad objective in this first part of the lab is to explore what topics were discussed, and possible variation by party membership.

1. Begin by importing `fb-congress-data3.csv`. Report basic information about the data set; how many rows and column it has, as well as the name of the variables.

```
fb_congress <- fread('fb-congress-data3.csv')
dim(fb_congress)
```

```
## [1] 6752    4
```

```
colnames(fb_congress)
```

```
## [1] "doc_id"      "screen_name" "party"       "message"
```

The dataset contains 6752 rows and 4 columns. Each row describes a facebook message. In addition to the actual text of the post (*message* column), we also have three *metadata* columns: the document id (*doc_id*), the political party which the focal individual is a member of (*party*), and displayed screen name (*screen_name*).

2. As you may have noticed from your inspection in #1, this data set has yet to be pre-processed (it contains punctuation, etc.). Hence, that is what you shall do now. More specifically, perform the following steps:

   i. Use `quanteda`'s `corpus()` function to create a corpus of your data set. Hint: For the argument `x` select your data set, for the argument `text` select the column name which stores the text, for the argument `docid_field` select the id variable, and finally, add the names of remaining variables to the `meta` argument (in a list).
   ii. Tokenize your corpus using the `tokens()` function. This splits each document into a vector of so-called tokens. Make the following specifications (which will remove punctuation, numbers, non-alpha-numeric symbols, and urls):
      - `remove_punct = TRUE`

[1] Obtained from https://lse-my459.github.io/

- `remove_numbers = TRUE`
- `remove_symbols = TRUE`
- `remove_url = TRUE`
- `padding = FALSE`

iii. Exclude english stopwords using the `tokens_remove()` function. Setting `x` to the output from the previous step, setting the second argument to `stopwords("en")`, and setting `padding=FALSE`.

iv. To get a feel of how your data looks like now, print the first 3 texts by simple subsetting of the output from iii.

v. As mentioned in the lecture, topic models expect the data to be in a *document-term-matrix* form. Transform your tokens into a document-term-matrix using the `quanteda`'s function `dfm()`.

vi. As a last pre-processing step, we want to exclude (a) words which are very infrequent (below 5). and (b) documents which have very few words (below 10). When you have done a–b, report the dimensionality of your resulting document-term-matrix. Hint: To trim infrequent words, use `quanteda`'s function `dfm_trim()`. To exclude documents with too few words, you may use the following code (where `dtm` is the object in which you have stored your document-term-matrix):

```
# Pre-process
# =======================================
posts_corpus <- corpus(x = fb_congress,
                       text_field = "message",
                       meta = list("party"),
                       docid_field = 'doc_id')

# Tokenize & clean from particular types of words
mytokens <- tokens(x = posts_corpus,
                   remove_punct = TRUE,
                   remove_numbers = TRUE,
                   remove_symbols = TRUE,
                   remove_url = TRUE,
                   padding = FALSE)
mytokens <-   tokens_remove(x = mytokens,
                            stopwords("en"),
                            padding = FALSE)
mytokens <-   tokens_select(x = mytokens,selection = 'remove',
                            valuetype = 'glob',
                            pattern = '@',
                            padding = FALSE)

# Make tokens lowercase
mytokens <- tokens_tolower(x = mytokens)

# Create document term matrix
dtm <- dfm(x = mytokens)

# Exclude words with too low frequency
dtm <- dfm_trim(dtm, min_termfreq = 5)

# Exclude documents with too low frequency
rowsums <- rowSums(dtm)
keep_ids <- which(rowsums>=10)
dtm <- dtm[keep_ids,]

# Report final dimensionality of data set
```

```r
dim(dtm)
```

```
## [1] 5739 5461
```

3. Now we are ready to do some topic modeling! To do so, we will use the `topicmodels` package, and the function `LDA()`. Set `x` to your document-term-matrix and specify `method="Gibbs"` (note: Gibbs is the name of a particular estimation procedure; see the Appendix of the lecture for more details). Set the number of iterations to 1000, and specify a seed number to ensure replicability (hint: to specify iterations and seed number, use the `control` argument). Finally, set the number of topics, `K=50`.[2] With these settings specified, start the estimation. This could take a minute or two.

```r
# Fit LDA
set.seed(3)
mylda <- LDA(x = dtm,
             k = 50,
             method="Gibbs",
             control=list(iter = 1000,
                          seed = 1,
                          verbose = FALSE))
```

4. Once the estimation is finished, use the `get_terms()` function to extract the 15 words with the highest probability in each topic. In a real research setting, we would carefully examine each of the topics. Here, I only ask you to briefly skim them, and then focus on 5 that (*i*) you think are interesting, (*ii*) has a clear theme, and (*iii*) are clearly distinct from the other topics. Provide a label to each of those based on the top 15 words. Complementing your label, please also provide a bar chart displaying on the y-axis the top 15 words, and on the x-axis their topic probabilities. Hint: you can retrieve each topic's distribution over words using `topicmodels`'s function "posterior"'.[3] Lastly, please also report a general assessment—based on your skim—about the general quality of the topics; do most of them appear clearly themed and distinct, or are there a lot of "junk" topics?

```r
# Printing top 15 words from each topic
print(topicmodels::get_terms(mylda, 15))
```

```
##       Topic 1     Topic 2         Topic 3       Topic 4    Topic 5
##  [1,] "work"      "jobs"          "federal"     "women"    "bill"
##  [2,] "make"      "economic"      "government"  "men"      "act"
##  [3,] "together"  "job"           "congress"    "every"    "legislation"
##  [4,] "sure"      "workers"       "public"      "rights"   "house"
##  [5,] "hard"      "create"        "regulations" "day"      "passed"
##  [6,] "can"       "growth"        "agencies"    "today"    "bipartisan"
##  [7,] "better"    "economy"       "just"        "civil"    "introduced"
##  [8,] "come"      "america's"     "without"     "pay"      "h.r"
##  [9,] "making"    "workforce"     "process"     "women's"  "bills"
## [10,] "working"   "opportunities" "issue"       "life"     "system"
## [11,] "need"      "grow"          "regulatory"  "justice"  "support"
## [12,] "solutions" "development"   "already"     "equal"    "use"
## [13,] "way"       "ohio"          "transparency" "dr"      "pass"
## [14,] "find"      "training"      "act"         "defend"   "representatives"
```

[2] Note: As we discussed in the lecture, in real research settings, where we usually have a clear research question, we would likely explore a range of K and select K based on how well it enables us to address the research question.

[3] It provides both the (a) each document's distribution over topics, and (b) each topic's distribution over words.

```
## [15,] "good"      "industry"      "failed"       "equality" "today"
##       Topic 6      Topic 7         Topic 8       Topic 9        Topic 10
##  [1,] "security"   "states"        "state"       "live"         "investigation"
##  [2,] "national"   "united"        "secretary"   "morning"      "general"
##  [3,] "military"   "north"         "great"       "watch"        "director"
##  [4,] "defense"    "u.s"           "meeting"     "news"         "russia"
##  [5,] "u.s"        "korea"         "met"         "talk"         "russian"
##  [6,] "homeland"   "south"         "issues"      "discuss"      "intelligence"
##  [7,] "department" "policy"        "food"        "tonight"      "independent"
##  [8,] "border"     "world"         "discuss"     "check"        "sessions"
##  [9,] "guard"      "foreign"       "farmers"     "tomorrow"     "know"
## [10,] "social"     "international" "agriculture" "tune"         "fbi"
## [11,] "armed"      "nuclear"       "association" "yesterday"    "election"
## [12,] "air"        "allies"        "discussed"   "facebook"     "attorney"
## [13,] "nation"     "threat"        "farm"        "conversation" "democracy"
## [14,] "strategy"   "iran"          "importance"  "joined"       "special"
## [15,] "troops"     "sanctions"     "rural"       "another"      "must"
##       Topic 11      Topic 12       Topic 13   Topic 14      Topic 15
##  [1,] "office"      "tax"          "lives"    "law"         "members"
##  [2,] "information" "reform"       "day"      "enforcement" "rep"
##  [3,] "please"      "families"     "remember" "police"      "congressman"
##  [4,] "open"        "plan"         "lost"     "department"  "congressional"
##  [5,] "visit"       "cuts"         "never"    "officers"    "congress"
##  [6,] "county"      "taxes"        "prayers"  "safe"        "caucus"
##  [7,] "staff"       "code"         "thoughts" "line"        "friend"
##  [8,] "residents"   "middle"       "families" "local"       "good"
##  [9,] "call"        "class"        "attack"   "communities" "john"
## [10,] "hours"       "working"      "others"   "immigration" "sexual"
## [11,] "available"   "pay"          "victims"  "today"       "steve"
## [12,] "center"      "bill"         "first"    "laws"        "joined"
## [13,] "may"         "jobs"         "honor"    "protect"     "colleague"
## [14,] "website"     "gop"          "violence" "safety"      "fellow"
## [15,] "sign"        "corporations" "may"      "state"       "harassment"
##       Topic 16      Topic 17      Topic 18      Topic 19         Topic 20
##  [1,] "country"     "continue"    "senate"      "funding"        "small"
##  [2,] "across"      "keep"        "house"       "infrastructure" "businesses"
##  [3,] "home"        "fight"       "senator"     "critical"       "business"
##  [4,] "america"     "ensure"      "vote"        "important"      "local"
##  [5,] "nation"      "proud"       "colleagues"  "including"      "economy"
##  [6,] "communities" "support"     "floor"       "million"        "help"
##  [7,] "opportunity" "fighting"    "today"       "needs"          "support"
##  [8,] "give"        "safe"        "republican"  "appropriations" "make"
##  [9,] "daca"        "protect"     "spoke"       "project"        "communities"
## [10,] "dreamers"    "committed"   "republicans" "efforts"        "opportunity"
## [11,] "right"       "like"        "bipartisan"  "transportation" "owners"
## [12,] "immigrants"  "hold"        "now"         "development"    "like"
## [13,] "families"    "communities" "resolution"  "provides"       "helped"
## [14,] "us"          "every"       "democratic"  "programs"       "employees"
## [15,] "stand"       "safety"      "urge"        "grant"          "supporting"
##       Topic 21    Topic 22     Topic 23   Topic 24  Topic 25   Topic 26
##  [1,] "new"       "washington" "american" "can"     "week"     "national"
##  [2,] "state"     "great"      "people"   "get"     "last"     "public"
##  [3,] "many"      "office"     "put"      "help"    "read"     "park"
##  [4,] "work"      "thank"      "better"   "need"    "night"    "natural"
```

```
##  [5,] "york"      "district"  "time"      "now"     "letter"    "also"
##  [6,] "mexico"    "thanks"    "deserve"   "ways"    "full"      "lands"
##  [7,] "like"      "capitol"   "long"      "stop"    "west"      "read"
##  [8,] "impact"    "meet"      "americans" "problem" "virginia"  "alaska"
##  [9,] "jersey"    "d.c"       "past"      "getting" "statement" "part"
## [10,] "also"      "dc"        "political" "know"    "speaker"   "native"
## [11,] "need"      "yesterday" "years"     "find"    "joined"    "want"
## [12,] "state's"   "visit"     "process"   "better"  "ryan"      "resources"
## [13,] "see"       "tour"      "week"      "chance"  "urging"    "wildlife"
## [14,] "valley"    "staff"     "end"       "focus"   "paul"      "future"
## [15,] "thousands" "always"    "politics"  "going"   "sent"      "leaders"
##       Topic 27    Topic 28   Topic 29     Topic 30    Topic 31    Topic 32
##  [1,] "today"     "year"     "veterans"   "must"      "family"    "one"
##  [2,] "first"     "budget"   "va"         "us"        "today"     "just"
##  [3,] "see"       "next"     "care"       "america"   "day"       "right"
##  [4,] "us"        "dollars"  "benefits"   "world"     "happy"     "want"
##  [5,] "one"       "spending" "affairs"    "country"   "time"      "even"
##  [6,] "important" "increase" "receive"    "stand"     "everyone"  "back"
##  [7,] "step"      "just"     "veteran"    "nation"    "great"     "said"
##  [8,] "glad"      "proposal" "deserve"    "americans" "friends"   "know"
##  [9,] "take"      "billion"  "department" "around"    "hope"      "many"
## [10,] "come"      "debt"     "medical"    "still"     "birthday"  "people"
## [11,] "weekend"   "voted"    "access"     "face"      "wonderful" "can"
## [12,] "like"      "money"    "served"     "freedom"   "thanks"    "say"
## [13,] "coming"    "fiscal"   "service"    "standing"  "wish"      "things"
## [14,] "another"   "taxpayer" "choice"     "hate"      "grateful"  "made"
## [15,] "taken"     "way"      "ensure"     "values"    "thankful"  "going"
##       Topic 33    Topic 34    Topic 35     Topic 36     Topic 37
##  [1,] "need"      "congress"  "forward"    "health"     "financial"
##  [2,] "crisis"    "time"      "look"       "care"       "free"
##  [3,] "opioid"    "must"      "work"       "affordable" "internet"
##  [4,] "like"      "take"      "working"    "insurance"  "amendment"
##  [5,] "cancer"    "also"      "also"       "act"        "protect"
##  [6,] "help"      "action"    "looking"    "healthcare" "act"
##  [7,] "research"  "since"     "made"       "repeal"     "information"
##  [8,] "treatment" "address"   "move"       "coverage"   "access"
##  [9,] "drug"      "days"      "strong"     "obamacare"  "wall"
## [10,] "resources" "now"       "leadership" "access"     "open"
## [11,] "epidemic"  "people"    "see"        "costs"      "big"
## [12,] "helping"   "including" "back"       "system"     "protections"
## [13,] "health"    "took"      "group"      "plan"       "rules"
## [14,] "human"     "bring"     "colleagues" "premiums"   "street"
## [15,] "services"  "syria"     "moving"     "quality"    "consumers"
##       Topic 38    Topic 39 Topic 40     Topic 41      Topic 42    Topic 43
##  [1,] "service"   "life"   "years"      "committee"   "community" "program"
##  [2,] "honor"     "best"   "court"      "house"       "history"   "families"
##  [3,] "u.s"       "first"  "ago"        "today"       "proud"     "children"
##  [4,] "thank"     "story"  "old"        "hearing"     "month"     "need"
##  [5,] "honored"   "one"    "judge"      "chairman"    "great"     "services"
##  [6,] "academy"   "team"   "two"        "member"      "nation's"  "provide"
##  [7,] "war"       "place"  "gun"        "commerce"    "many"      "support"
##  [8,] "serve"     "many"   "supreme"    "subcommittee" "join"     "community"
##  [9,] "force"     "way"    "justice"    "space"       "part"      "working"
## [10,] "award"     "time"   "gorsuch"    "chamber"     "celebrate" "start"
```

```
## [11,] "military"  "came"   "one"        "today's"      "learn"     "kids"
## [12,] "ceremony"  "go"     "violence"   "oversight"    "black"     "like"
## [13,] "air"       "age"    "nomination" "judiciary"    "center"    "funding"
## [14,] "sacrifice" "young"  "sense"      "technology"   "honor"     "providing"
## [15,] "vietnam"   "free"   "common"     "next"         "around"    "head"
##       Topic 44        Topic 45         Topic 46         Topic 47
##  [1,] "students"      "president"      "water"          "join"
##  [2,] "school"        "trump"          "energy"         "questions"
##  [3,] "education"     "trump's"        "change"         "town"
##  [4,] "high"          "administration" "future"         "hall"
##  [5,] "congressional" "donald"         "climate"        "constituents"
##  [6,] "district"      "order"          "clean"          "share"
##  [7,] "young"         "executive"      "power"          "issues"
##  [8,] "college"       "j"              "rule"           "hear"
##  [9,] "student"       "president's"    "administration" "hope"
## [10,] "learn"         "obama"          "environment"    "meeting"
## [11,] "university"    "signed"         "environmental"  "please"
## [12,] "competition"   "white"          "epa"            "hosting"
## [13,] "schools"       "actions"        "progress"       "call"
## [14,] "programs"      "decision"       "approach"       "concerns"
## [15,] "career"        "ban"            "impact"         "asked"
##       Topic 48      Topic 49      Topic 50
##  [1,] "americans"   "county"      "assistance"
##  [2,] "bill"        "community"   "help"
##  [3,] "republicans" "city"        "federal"
##  [4,] "millions"    "texas"       "puerto"
##  [5,] "people"      "center"      "emergency"
##  [6,] "million"     "san"         "recovery"
##  [7,] "senate"      "de"          "u.s"
##  [8,] "families"    "building"    "rico"
##  [9,] "away"        "la"          "disaster"
## [10,] "republican"  "el"          "hurricane"
## [11,] "healthcare"  "university"  "relief"
## [12,] "trumpcare"   "congressman" "fire"
## [13,] "coverage"    "attended"    "fema"
## [14,] "take"        "area"        "management"
## [15,] "conditions"  "mayor"       "harvey"
```

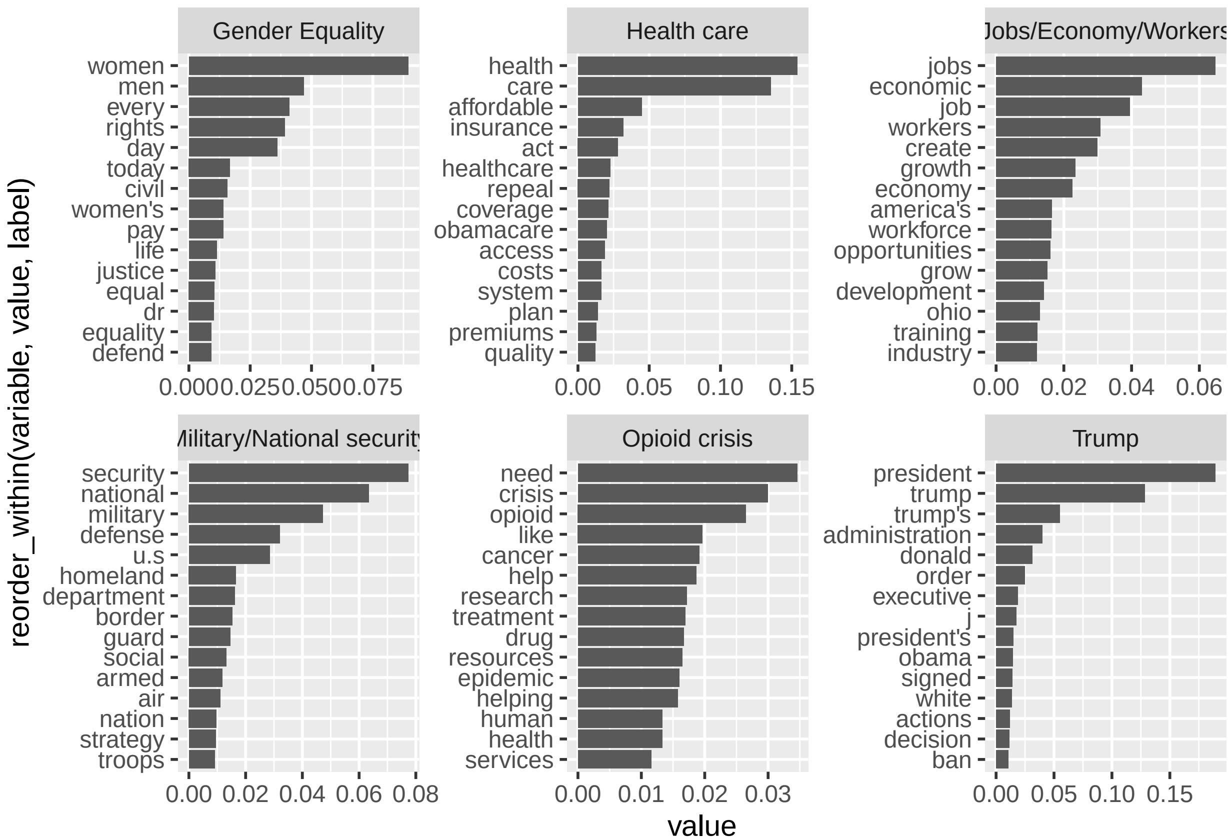
```
# To get a more granular view, extract the probabilities
mylda_posterior <- topicmodels::posterior(object = mylda)
topic_distr_over_words <- mylda_posterior$terms
topic_distr_over_words_dt <- data.table(topic=1:50,
                                        topic_distr_over_words)
topic_distr_over_words_dt <- melt.data.table(topic_distr_over_words_dt,
                                             id.vars = 'topic')

# data.table way of extracting top 10 rows by group
topic_distr_over_words_dt <- topic_distr_over_words_dt[order(value,decreasing = T)]
top15per_topic <- topic_distr_over_words_dt[,head(.SD,15),by='topic']

# Create topic labels
top15per_topic[topic==2, label := 'Jobs/Economy/Workers']
top15per_topic[topic==4, label := 'Gender Equality']
top15per_topic[topic==6, label := 'Military/National security']
```

```
top15per_topic[topic==33, label := 'Opioid crisis']
top15per_topic[topic==36, label := 'Health care']
top15per_topic[topic==45, label := 'Trump']

# Plot probability over words for a few topics.
library(tidytext) # To re-order x-axis by value within group
ggplot(top15per_topic[topic %in% c(2,4,6,33,36,45)],aes(y=reorder_within(variable,value,label),x=value))
  geom_bar(stat = 'identity', position = 'dodge') +
  scale_y_reordered() +
  facet_wrap(~label,scales = 'free')
```



The assigned labels are found in the facet-titles. Overall, these six topics are pretty clear-cut. The interpretation was also made easier by considering the relative weight (probabilities) of the words within the top 15. For some topics, the distribution within top 10/15 is rather equal (e.g., "Opiod crisis" topic), while for others it is very unequal (e.g., "Trump" topic). In terms of general assessment, I would say that most topics appear rather distinct and interpretable.[4] Some topics, like 21–22, are perhaps less interpretable. This is fine: what we usually care about is the extent to which a subset of the topics of our topic model captures something meaningful and distinct—then we can focus our analysis on those topics.

5. Out of the 5 topics that you labeled, select *two* which you think are particularly interesting. For these three, identify the three documents which have the highest proportion assigned of this topic (hint 1: use `topicmodels`'s `posterior()` to extract documents' distribution over topics | hint 2: to identify the document ids which correspond to each row of what you extract from `posterior()`, you

[4]As mentioned in the lecture, to refine and improve topics, one can use so-called *seeded topic models*. It turns out that this is actually possible to do using the *topicmodels* package. See the following link if you are interested: https://stats.stackexchange.com/questions/384183/seeded-lda-using-topicmodels-in-r .

can use `ldaobject@documents`. See help file for more details.), and do a qualitative inspection (= 2×3 documents to read). Does your readings corroborate your labels? Are they about what you expected?

```
# To validate labeling; identify documents with
# highest proportion on any given topic
doc_topic_proportions <- mylda_posterior$topics
doc_topic_proportions_dt <- data.table(doc_id=mylda@documents,
                                       doc_topic_proportions)
setnames(x = doc_topic_proportions_dt,
         old = 2:ncol(doc_topic_proportions_dt),
         new =  paste0('Topic',1:50))

# Select document ids to inspect for topic 6
t6_ids <- doc_topic_proportions_dt[order(Topic6,decreasing = T)][,.(doc_id,Topic6)][1:5,]

# Print the corresponding fb posts
print(fb_congress[doc_id==t6_ids$doc_id[1]]$message)
```

```
## [1] "The House of Representatives today passed the 2018 National Defense Authorization Act (NDAA), a
```

```
print(fb_congress[doc_id==t6_ids$doc_id[2]]$message)
```

```
## [1] "In Case You Missed It: Below are the key elements of the President’s military and diplomatic st
```

```
print(fb_congress[doc_id==t6_ids$doc_id[3]]$message)
```

```
## [1] "The passage of this year’s NDAA marks a critical step in rebuilding our nation’s military, furt
```

I only do the document inspection for one topic; the topic I labeled "Military/national security". I would say that reading these documents does indeed corroborate the chosen label: they are clearly about the military and national sectuity.

6. Now, estimate a topic model—as in #3—but with `K=3` instead. Extract the top 15 words from each topic, (try to) label each, and then make an assessment of the overall quality of them. To further explore the quality of this topic model, reconsider the documents you read in #5: extract the distribution over topics for these documents (from your new `K=3` model). How well does this topic model capture the theme of these documents? Based on your analysis, which of the two `K`’s do you prefer? Motivate.

```
# Fit LDA
set.seed(3)
mylda2 <- LDA(x = dtm,
              k = 3,
              method="Gibbs",
              control=list(iter = 1000,
                           seed = 1,
                           verbose = FALSE))
```

```
# Printing top 15 words from each topic
print(get_terms(mylda2, 15))
```

```
##       Topic 1          Topic 2     Topic 3
##  [1,] "president"      "health"    "today"
##  [2,] "trump"          "care"      "great"
##  [3,] "u.s"            "bill"      "day"
##  [4,] "national"       "act"       "community"
##  [5,] "federal"        "can"       "veterans"
##  [6,] "must"           "tax"       "office"
##  [7,] "law"            "new"       "week"
##  [8,] "house"          "americans" "thank"
##  [9,] "also"           "work"      "years"
## [10,] "security"       "families"  "service"
## [11,] "states"         "help"      "women"
## [12,] "congress"       "people"    "district"
## [13,] "committee"      "make"      "washington"
## [14,] "state"          "american"  "congressional"
## [15,] "administration" "need"      "students"
```

As expected, using $K = 3$ makes the topics less granular, and blends together different topics that we saw in the $K = 50$ topic model. For exampe, in the $K = 3$ model, topic 1 combines both Trump-related terms ("trump", "president") and militariy-related terms ("natinal","security"). This makes labeling—and the use of these topics for substantive research—difficult. Considering the topic proportions of the documents printed above, this assessment is corroborated. All three documents have topic 1 as its primary topic; which again contains both military- and Trump-related terms. The documents are not about Trump, however, but clearly about the military.

```r
# Inspecting the topic proportions for read docs
mylda2_posterior <- topicmodels::posterior(object = mylda2)
doc_topic_proportions2 <- mylda2_posterior$topics
doc_topic_proportions2_dt <- data.table(doc_id=mylda2@documents,
                                        doc_topic_proportions2)
setnames(x = doc_topic_proportions2_dt,
         old = 2:ncol(doc_topic_proportions2_dt),
         new =  paste0('Topic',1:3))

doc_topic_proportions2_dt[doc_id==t6_ids$doc_id[1]]
```

```
##    doc_id    Topic1    Topic2    Topic3
## 1:   3476 0.5210728 0.2796935 0.1992337
```

```r
doc_topic_proportions2_dt[doc_id==t6_ids$doc_id[2]]
```

```
##    doc_id   Topic1    Topic2    Topic3
## 1:   2176 0.564257 0.2570281 0.1787149
```

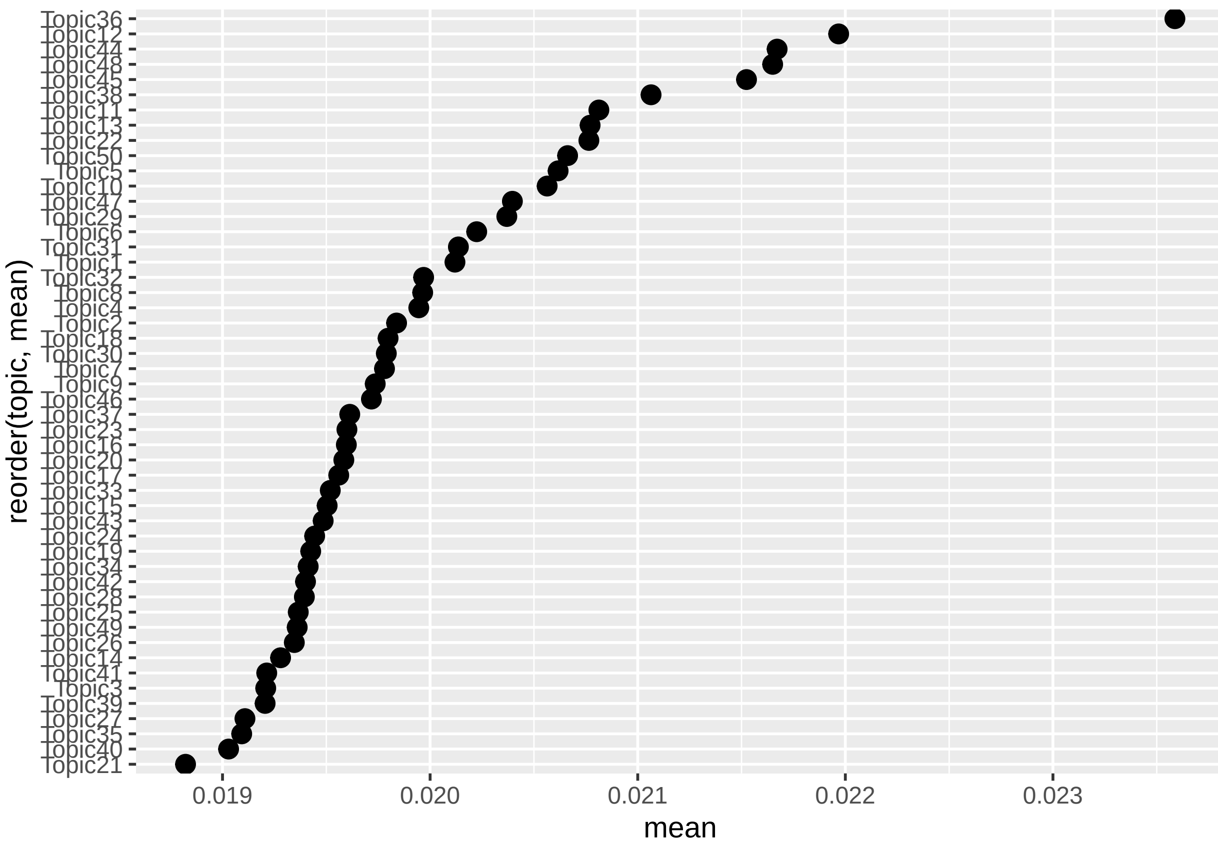
```r
doc_topic_proportions2_dt[doc_id==t6_ids$doc_id[3]]
```

```
##    doc_id    Topic1    Topic2    Topic3
## 1:    990 0.4947589 0.2620545 0.2431866
```

7. Continuing with the topic model you concluded the most appropriate, perform the following sets of analyses:

i. Compute the prevalence of each topic, across all documents. Report which is the most prevalent topic, overall, and then report—in the form of a single plot; e.g., a bar chart—the prevalence of the topics you labeled.
ii. Compare the prevalence on your labeled topics between *democrats* and *republicans*. You can for example fit a fractional regression model using `glm(family="quasibinomial")`[5] or using t-tests of difference in means. Interpret.

```r
# (i) Compute prevalence across documents (for model 1)
topic_means <- colMeans(doc_topic_proportions_dt[,c(2:51)])
topic_means <- topic_means[order(topic_means,decreasing = T)]
topic_means_dt <- data.table(topic=names(topic_means),
                             mean=topic_means)
ggplot(topic_means_dt,aes(x=mean,y=reorder(topic,mean))) +
  geom_point(size=3)
```



A first observation is that the different topics have a rather similar prevalence (min=0.019, max=0.024). Second, we find that the most prevalent topic is one of the topics that I labeled: topic 36 ("health care").

```r
# (ii) Compare the prevalence on your labeled topics between D and R
# - add party indicator
fb_congress[,doc_id := as.character(doc_id)] # Make sure both have same format
doc_topic_proportions_dt <- merge(x=doc_topic_proportions_dt,
                                  y=fb_congress[,.(doc_id,party)],
```

[5]If you go this route, note that you then also need to compute robust standard errors, which you can do using the `sandwich` package: `coeftest(myglm, vcov. = vcovHC(myglm, type = 'HC1'))`

```r
                                     by='doc_id')
# Example: health care topic
hc_glm <- glm(Topic36 ~ party,
              data=doc_topic_proportions_dt,
              family="quasibinomial")
# Using the fractional logit approach,
# (combining sandwich and lmtest pkgs
#  to get robust standard errors)
library(sandwich)
library(lmtest)
coeftest(hc_glm,
        vcov = vcovHC(hc_glm, type="HC1"))
```

```
##
## z test of coefficients:
##
##                   Estimate Std. Error   z value  Pr(>|z|)
## (Intercept)      -3.613789   0.021941 -164.7061 < 2.2e-16 ***
## partyRepublican -0.237000   0.031759   -7.4625 8.492e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

I illustrate a comparison for one of the topics I labeled ("health care"). It is talked about $1 - exp(-0.237) = 1 - 0.788 = 0.212 \sim 21\%$ less by republicans compared to democrats. This is in line with my surface-level understanding of US politics.

8. **BONUS** (not obligatory; suggestion; do after you have completed the rest of the lab). As a bonus exercise—to expose you to the traditional computer sciency way of selecting the number of topics, K—you shall consider a data-driven approach, relying on the measure of *hold-out likelihood* (or, *perplexity* as its also called). To do so, do the following:

   i. Split your document term matrix into two (a training and test set); 80/20 division.
   ii. Write a loop which in each iteration (a) estimates a topic model using a particular K, and then (b) computes (and stores) its perplexity using the `topicmodels` function `perplexity()`, which takes as input the model object and the test document-term-matrix (note: the document-term-matrix needs to be transformed into a particular format: use `...` for this).
   iii. Consider the following range of `K`: \$$\{3, 10, 25, 50, 75, 100, 200\}$\$, and run the loop. This may take a few minutes. Once the loop has finished, plot your results (x-axis: `K`, y-axis: `perplexity`). Interpret. Based on this, what is a reasonable `K`?

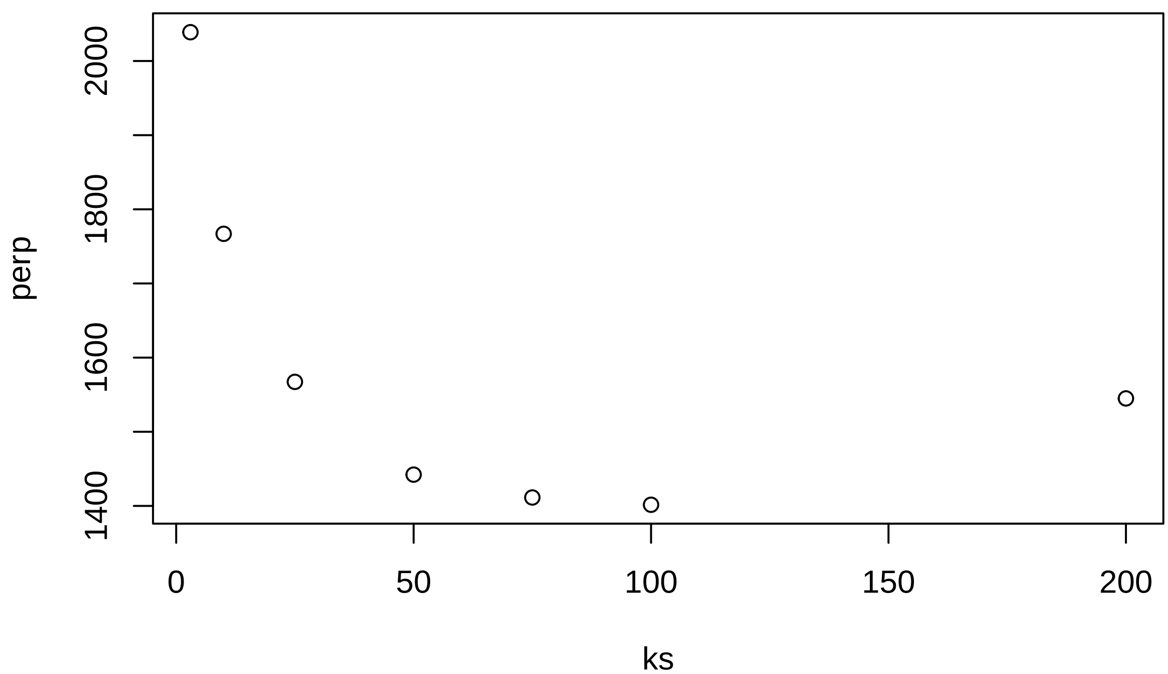
```r
# Data-driven way of selecting K (by perplexity)
# ===============================================
library(slam)
# - (i) split data into train/test
row_ids <- 1:nrow(dtm)
tr_ids <- sample(x = row_ids, size = 0.7*length(row_ids), replace = F)
tst_ids <- row_ids[!row_ids %in% tr_ids]
tr_dt <- dtm[tr_ids,]
tst_dt <- dtm[-tr_ids,]
dim(tst_dt)

# - (ii) loop over a set of possible topics and estimate LDA
ks <- c(3,10,25,50,75,100,200)
```

```r
perp <- c()
for(i in 1:length(ks)){

  # Train
  current_tm <- LDA(x = tr_dt,
                    k = ks[i],
                    method="Gibbs",
                    control=list(iter = 500,
                                 seed = 1,
                                 verbose = FALSE))
  # Compute perplexity
  perp[i] <- perplexity(object = current_tm,
                        newdata = as.simple_triplet_matrix(tst_dt))
}
```

```r
plot(x=ks,y=perp)
```



We see clear improvements in hold-out log likelihood (perplexity) up and till $K \sim 50$. After this, the fit improves only marginally, and at some point, the fit even becomes worse (e.g., as shown for $K = 200$). I would say that this plot confirms that $K = 50$ is a reasonable choice (from a pure data-driven stand point).

Part 2: Word embeddings

In this second part of the lab, we will continue with the data *U.S. Congress–Facebook posts* data set. However, now with a different focus: a focus on the word-level, using word embeddings instead of topic models.

1. Because word embeddings are not negatively affected by *stop words* or other highly frequent terms, your first task is to re-import the `fb-congress-data.csv` file, and re-process the data; performing step *i–ii* in task #2, but skipping #3. Here, we also *do not* want to transform our documents into a document-term matrix. Instead, after having tokenized and cleaned the documents, paste each back into a *single string* per document. Hint: for this, you could for example write: `sapply(mytokens,function(x)paste(x,collapse = " "))`. As a last pre-processing step, transform all your text into lowercase (hint: you can use the function `tolower()` for this).

```
fb_congress <- fread('fb-congress-data3.csv')
#fb_congress <- fread('fb-congress-data.csv')
```

```
# Pre-process
# ========================================
posts_corpus <- corpus(x = fb_congress,
                       text_field = "message",
                       meta = list("party"),
                       docid_field = 'doc_id')

# Tokenize & clean from particular types of words
mytokens <- tokens(x = posts_corpus,
                   remove_punct = TRUE,
                   remove_numbers = TRUE,
                   remove_symbols = TRUE,
                   remove_url = TRUE,
                   padding = FALSE)
mytokens <-   tokens_select(x = mytokens,
                            selection = 'remove',
                            valuetype = 'glob',
                            pattern = '@',
                            padding = FALSE)

# Make tokens lowercase
mytokens <- tokens_tolower(x = mytokens)

# Collapse into strings within documents
txt <- sapply(mytokens,function(x) paste(x,collapse = ' '))
txt <- tolower(txt)
```

2. Now we are set to fit word embeddings! To begin, let us fit one word embedding model to all documents—not separating posts by democrats and republicans. Use `word2vec`'s `word2vec()` function to fit a *cbow* model (`type="cbow"`) using 15 negative samples per real context/observation (`negative=15`), and setting `dim=50`, the number of dimensions of the word vetors/embeddings. This will take a minute or two.

```
# Fit word embeddings
library(word2vec)
set.seed(123456789)
system.time(w2v <- word2vec(x = txt,          # Your data
                            type = "cbow",    # Model type (the one we talked about in lecture)
                            window = 5,       # Context defined in terms of +-5 words.
                            dim = 50,         # Dimensionality of embeddings
                            iter = 50,        # Estimation iterations (higher means more time...)
                            hs = FALSE,       # Setting to FALSE here --> "negative sampling procedure"
                            negative = 15))   # Number of negative samples
```

```
##    user  system elapsed
##   37.55    0.17   62.16
```

3. When the estimation in #2 is finished, identify the 10 nearest terms to 3 focal (sufficiently frequent) words of your choice/interest. Hint: to retrieve the closest words in embedding/word vector space, you may use the following code: `predict(w2v,c("word2","word2","word3"),type="nearest",top_n = 10)`, where`wv2` is the object storing the *fitted model* of the `word2vec` function. Does the results you find makes sense? Why/why not?

```r
predict(w2v,c("military","muslim","science"),type="nearest",top_n = 10)
```

```
## $military
##       term1       term2 similarity rank
## 1  military    deployed  0.7391008    1
## 2  military      pilots  0.7254765    2
## 3  military      fallen  0.7240921    3
## 4  military         vso  0.7160951    4
## 5  military      troops  0.6947923    5
## 6  military   personnel  0.6914906    6
## 7  military       jrotc  0.6848000    7
## 8  military    honoring  0.6765638    8
## 9  military     uniform  0.6764978    9
## 10 military afghanistan  0.6763469   10
##
## $muslim
##     term1            term2 similarity rank
## 1  muslim unconstitutional  0.8346369    1
## 2  muslim         inhumane  0.7874492    2
## 3  muslim   discriminatory  0.7716109    3
## 4  muslim          immoral  0.7236773    4
## 5  muslim            visas  0.7057860    5
## 6  muslim            midst  0.7057691    6
## 7  muslim           pardon  0.7040161    7
## 8  muslim             kill  0.7013975    8
## 9  muslim           powers  0.6935122    9
## 10 muslim         denounce  0.6929017   10
##
## $science
##      term1        term2 similarity rank
## 1  science    institute  0.7675225    1
## 2  science     computer  0.7566153    2
## 3  science  exploration  0.7355672    3
## 4  science agricultural  0.7272885    4
## 5  science  engineering  0.7125544    5
## 6  science     exchange  0.7068127    6
## 7  science           ms  0.6961313    7
## 8  science        study  0.6953618    8
## 9  science    gathering  0.6911864    9
## 10 science    classroom  0.6871029   10
```

The closest words for "science" and "military" makes sense (with an expected bias towards "hard" science). The closest words for "muslim" would seem to come from two different types of posts; one exhibiting bias/discrimination ("dangerous","terrorist"; in top20) while the other discussing this bias/discrimination ("discriminatory"), including at the time much discussed "muslim ban" ("ban","unconstitutional").

4. What initially made people so excited about word embeddings was their surprising ability to solve seemingly complex analogy tasks. Your task now is to attempt to replicate one such classical analogy result, first with the embedding vectors that you have already estimated, and second using a pre-trained embedding model. To do so, please perform the following steps:

   i. Extract the whole embedding matrix: `embedding <- as.matrix(w2v)`.
   ii. Identify the rows in the embedding matrix which correspond to `king`, `man`, `woman`, and create a new R object `kingtowoman` which is equal to the vector for king, minus the vector for man, plus the vector for woman. Hint: to extract the row corresponding to a particular word (e.g., "king"), you may use `w2v[rownames(w2v)=="king",]`.
   iii. Use `word2vec`'s function `word2vec_similarity()` to identify the 20 most similar words to `kingtowoman`. Do you find "queen" in the top 20? Why do you think you get the result you do?
   iv. Next, we will consider a *pre-trained embedding model* (trained on all Wikipedia articles that existed in 2014 and about 5 million news articles). The embedding vectors from this model are stored in the file "glove6B200d.rds".[6] Note: this file is large; more than 300MB. Use `readRDS()` to import it, and stored it in an `R` object called `pretrained`. Each row stores the embedding vector for a particular word. With this info in mind, report how many embedding dimensions were used for this model, and how many words we have embedding vectors for.
   v. Repeat steps ii–iii for `pretrained`. Does "queen" appear in the top 20 here? What do you think explains this difference/similarity to the self-trained result?
   vi. Given the result in (v), what do you expect, if you were to construct a measure of *occupational gender bias* along the lines of Garg et al. (2018), that is by comparing the distance between different occupations and gendered words, for example: $\overrightarrow{occupationalbias} = \text{dist}(\overrightarrow{statistician}, \overrightarrow{man})$ – $\text{dist}(\overrightarrow{statistician}, \overrightarrow{woman})$, would this score be "more correct" than the one you would obtain from the same calculation on your facebook/congress model? Why/why not?

```r
# (i)
embedding <- as.matrix(w2v)
# (ii)
kingtowoman <- embedding[rownames(embedding)=="king",] -
               embedding[rownames(embedding)=="man",] +
               embedding[rownames(embedding)=="woman",]
# (iii)
word2vec::word2vec_similarity(x = matrix(kingtowoman,nrow = 1),
                              y = embedding,
                              top_n = 20)
```

```
##    term1      term2 similarity rank
## 1      A       king  0.8852698    1
## 2      A     luther  0.7958066    2
## 3      A      woman  0.7525811    3
## 4      A     church  0.7521880    4
## 5      A         dr  0.7482686    5
## 6      A   shoulder  0.7410657    6
## 7      A    borough  0.7367696    7
## 8      A    pursuit  0.7365691    8
## 9      A     martin  0.7313005    9
## 10     A    wichita  0.7244321   10
## 11     A    towards  0.7204262   11
## 12     A    singing  0.7203854   12
## 13     A    baptist  0.7142069   13
```

[6] Original source for this file: https://nlp.stanford.edu/projects/glove/

```
## 14     A centennial  0.7132192   14
## 15     A    slavery  0.7113341   15
## 16     A outpatient  0.7106864   16
## 17     A       born  0.7080638   17
## 18     A     system  0.7080564   18
## 19     A tirelessly  0.7070613   19
## 20     A  associate  0.7070007   20
```

```
# Not in the top 20.
w2v$data$n_vocabulary # small corpus; order of magnitude below rec min.
```

```
## [1] 363532
```

We do not find "queen" in the top 20. This lack of success is likely partly attributable to the fact that (a) our corpus size is considerably smaller than whas is recommended (350K « 3M), and (b) royalty may not be something which is prominently discussed in US politics (not a monarch like the UK). Let us next proceed with (iv) and the pre-trained embeddings.

```
# (iv)
pretrained <- readRDS(file = 'glove6B200d.rds')
dim(pretrained) # 400,000 unique tokens, and embedding dimensionality of 200
```

```
## [1] 400000    200
```

```
kingtowoman2 <- pretrained[rownames(pretrained)=="king",] -
                pretrained[rownames(pretrained)=="man",] +
                pretrained[rownames(pretrained)=="woman",]
# (iii)
word2vec::word2vec_similarity(x = matrix(kingtowoman2,nrow = 1),
                              y = pretrained,
                              top_n = 20)
```

```
##    term1     term2 similarity rank
## 1      A      king  0.4630444    1
## 2      A     queen  0.4272459    2
## 3      A  princess  0.4001166    3
## 4      A    prince  0.3994429    4
## 5      A    throne  0.3877838    5
## 6      A   emperor  0.3765438    6
## 7      A     royal  0.3732692    7
## 8      A  daughter  0.3721567    8
## 9      A   monarch  0.3719335    9
## 10     A   kingdom  0.3668355   10
## 11     A     crown  0.3652451   11
## 12     A    mother  0.3617875   12
## 13     A       her  0.3608811   13
## 14     A  marriage  0.3567054   14
## 15     A      wife  0.3565992   15
## 16     A elizabeth  0.3557918   16
## 17     A     woman  0.3553641   17
## 18     A   duchess  0.3549088   18
## 19     A adulyadej  0.3543496   19
## 20     A      duke  0.3536174   20
```

```
# Excluding "king", which is the focal word, "queen" is the best match.
```

Here the results are more convincing. Queen appears as number two in similarity, with only "king" above it (which, recall, is the focal word which we substracted from). The two key characteristics that differentiates this (pre-trained) embedding model and the self-trained one are: (1) size of corpus and (2) type of corpus. The pre-trained model was trained on a much larger data set, and it was trained on a general information corpus (Wikipedia).

5. Now we shall make a comparison between democrats and republicans. Split the data from step #1 into two based on party affiliation. Then, repeat 2–3, but now separately for republicans and democrats. For #3, select words which you expect might be used differently between the two political camps (but still are frequently used by both; for example “abortion”, “obamacare”). Do you find any differences? Do they align with your expectations?

```
# =======================================
# Pre-process for Republicans
# =======================================
R_posts_corpus <- corpus(x = fb_congress[party=="Republican",],
                       text_field = "message",
                       meta = list("party"),
                       docid_field = 'doc_id')

# Tokenize & clean from particular types of words
R_tokens <- tokens(x = R_posts_corpus,
                   remove_punct = TRUE,
                   remove_numbers = TRUE,
                   remove_symbols = TRUE,
                   remove_url = TRUE,
                   padding = FALSE)
R_tokens <-   tokens_select(x = R_tokens,
                            selection = 'remove',
                            valuetype = 'glob',
                            pattern = '@',
                            padding = FALSE)

# Make tokens lowercase
R_tokens <- tokens_tolower(x = R_tokens)

# Collapse into strings within documents
R_txt <- sapply(R_tokens,function(x) paste(x,collapse = ' '))
R_txt <- tolower(R_txt)

# =======================================
# Fit word2vec for Republicans
# =======================================
set.seed(123456789)
R_w2v <- word2vec(x = R_txt,
                  type = "cbow",
                  window = 5,
                  dim = 50,
                  iter = 50,
                  hs = FALSE,
                  negative = 15)
```

```r
# ========================================
# Pre-process for Democrats
# ========================================
D_posts_corpus <- corpus(x = fb_congress[party=="Democrat",],
                         text_field = "message",
                         meta = list("party"),
                         docid_field = 'doc_id')

# Tokenize & clean from particular types of words
D_tokens <- tokens(x = D_posts_corpus,
                   remove_punct = TRUE,
                   remove_numbers = TRUE,
                   remove_symbols = TRUE,
                   remove_url = TRUE,
                   padding = FALSE)
D_tokens <-   tokens_select(x = D_tokens,
                            selection = 'remove',
                            valuetype = 'glob',
                            pattern = '@',
                            padding = FALSE)

# Make tokens lowercase
D_tokens <- tokens_tolower(x = D_tokens)

# Collapse into strings within documents
D_txt <- sapply(D_tokens,function(x) paste(x,collapse = ' '))
D_txt <- tolower(D_txt)

# ========================================
# Fit word2vec for Democrats
# ========================================
set.seed(123456789)
D_w2v <- word2vec(x = D_txt,
                  type = "cbow",
                  window = 5,
                  dim = 50,
                  iter = 50,
                  hs = FALSE,
                  negative = 15)

# ========================================
# Investigate how closest neighbors vary
# or are similar for a set of terms
# ========================================

# Identify terms frequent in both corpora
R_dtm <- dfm(x = R_tokens)
R_freq <- colSums(R_dtm)
R_freq <- data.table(word=names(R_freq),Rfreq=R_freq)
D_dtm <- dfm(x = D_tokens)
D_freq <- colSums(D_dtm)
D_freq <- data.table(word=names(D_freq),Dfreq=D_freq)
RD_freq <- merge(x=R_freq,y=D_freq,by='word')
```

```r
RD_freq[,totfreq := Rfreq + Dfreq]
RD_freq <- RD_freq[order(Rfreq,decreasing = T)]

predict(R_w2v,c("obamacare"),type="nearest",top_n = 20)
```

```
## $obamacare
##        term1        term2 similarity rank
## 1  obamacare          aca  0.7315462    1
## 2  obamacare     mandates  0.7255138    2
## 3  obamacare      choices  0.7243416    3
## 4  obamacare      control  0.7139689    4
## 5  obamacare        taxes  0.7014903    5
## 6  obamacare       flawed  0.6908864    6
## 7  obamacare skyrocketing  0.6735579    7
## 8  obamacare      massive  0.6723880    8
## 9  obamacare     premiums  0.6710304    9
## 10 obamacare        party  0.6684552   10
## 11 obamacare         ahca  0.6651546   11
## 12 obamacare      replace  0.6627017   12
## 13 obamacare  replacement  0.6622640   13
## 14 obamacare   affordable  0.6604046   14
## 15 obamacare       market  0.6595842   15
## 16 obamacare        death  0.6580247   16
## 17 obamacare        books  0.6531382   17
## 18 obamacare      harmful  0.6510901   18
## 19 obamacare    replacing  0.6487640   19
## 20 obamacare      finally  0.6425321   20
```

```r
predict(D_w2v,c("obamacare"),type="nearest",top_n = 20)
```

```
## $obamacare
##        term1       term2 similarity rank
## 1  obamacare  affordable  0.7556255    1
## 2  obamacare   trumpcare  0.7017187    2
## 3  obamacare         aca  0.6968367    3
## 4  obamacare      repeal  0.6893328    4
## 5  obamacare       dream  0.6844210    5
## 6  obamacare      harder  0.6814033    6
## 7  obamacare replacement  0.6589020    7
## 8  obamacare        even  0.6466703    8
## 9  obamacare    congress  0.6447006    9
## 10 obamacare      waiver  0.6377623   10
## 11 obamacare     refused  0.6310932   11
## 12 obamacare         try  0.6293728   12
## 13 obamacare         rip  0.6272182   13
## 14 obamacare  republican  0.6270693   14
## 15 obamacare      happen  0.6252273   15
## 16 obamacare      choose  0.6245051   16
## 17 obamacare       idly  0.6172621   17
## 18 obamacare     nothing  0.6144113   18
## 19 obamacare      secret  0.6140355   19
## 20 obamacare       worse  0.6130987   20
```

There are indeed meaningful differences between the republicans and the democrats in so far as the most similar terms for "obamacare"; you find more negative-laiden words in the top 20 for the republicans (e.g., "flawed") compared to the democrats (e.g., "affordable").

6. **BONUS** (not obligatory). Continuing with the comparison between democrats and republicans, your next task is to construct a *sentiment dimension* for each party, and then—by projecting the rest of the words onto this dimension—explore which words that have the strongest negative association in each party. To do so, please follow these steps:

    i. Import the two text files `positive.txt` and `negative.txt`. They contain a large number of positively and negatively laiden words, respectively.

    ii. Separately for each party:

        a. Identify the rows (in the corresponding party embedding matrix) which correspond to the words in the `positive` and `negative` document, extract their vectors into two separate matrices (positive, negative) and calculate the column averages.

        b. Compute the difference between the two average vectors from a (negative – positive). This is the party-specific "negative sentiment" dimension.

        c. Use `word2vec`'s function `word2vec_similarity()` to calculate the similarity to the sentiment dimension for all words except those included in the negative file.

    iii. Using the results from ii, identify the 50 words with the strongest negative association (most similarity with the negative sentiment dimension) for both republicans and democrats. Report what you find. Are there substantive differences in the composition of words? Additionally, you shall pick a set of words which you hypothesize may be used in a more/less positive light between the parties; are they? Note: absolute differences cannot be compared across models; instead you shall report relative differences (i.e., their ranking).

```r
# ========================================
# Create & investigate sentiment dimension
# for both parties
# ========================================
pos <- fread('positive.txt',header = F)
neg <- fread('negative.txt',header = F)
check_words <- c("obamacare")

# ===========
# Republicans
# ===========
R_embedding <- as.matrix(R_w2v)
# (iii)
R_pos_vectors <- R_embedding[which(rownames(R_embedding) %in% pos$V1),]
R_neg_vectors <- R_embedding[which(rownames(R_embedding) %in% neg$V1),]
R_pos_vector <- as.matrix(apply(R_pos_vectors,2,mean))
R_neg_vector <- as.matrix(apply(R_neg_vectors,2,mean))
# (iv)
R_pos_vector <- t(R_pos_vector)
R_neg_vector <- t(R_neg_vector)
R_neg_pos_dimension <- R_neg_vector - R_pos_vector
# (v) 50 most negative
R_neg_assoc_terms <- word2vec_similarity(x = R_neg_pos_dimension,
                                         y = R_embedding,
                                         top_n = 50)
print(R_neg_assoc_terms)
```

```
##    term1         term2 similarity rank
## 1      A        crimes  0.4210336    1
## 2      A       dealing  0.3915808    2
## 3      A         fraud  0.3909482    3
## 4      A       deficit  0.3890297    4
## 5      A          tape  0.3831337    5
## 6      A          slow  0.3829209    6
## 7      A        overly  0.3821345    7
## 8      A bureaucratic  0.3787617    8
## 9      A         rules  0.3779903    9
## 10     A           bad  0.3776718   10
## 11     A       massive  0.3749900   11
## 12     A        battle  0.3707367   12
## 13     A        spread  0.3705016   13
## 14     A        terror  0.3702000   14
## 15     A         toxic  0.3693914   15
## 16     A       effects  0.3693736   16
## 17     A   devastation  0.3689857   17
## 18     A     necessity  0.3687367   18
## 19     A     excessive  0.3685800   19
## 20     A           red  0.3653285   20
## 21     A          fail  0.3649542   21
## 22     A         cases  0.3637596   22
## 23     A      wildfire  0.3637379   23
## 24     A         syria  0.3631212   24
## 25     A        storms  0.3628517   25
## 26     A        babies  0.3627899   26
## 27     A         books  0.3621374   27
## 28     A    regulators  0.3602685   28
## 29     A      attempts  0.3598765   29
## 30     A        accept  0.3596306   30
## 31     A      billions  0.3591981   31
## 32     A      criminal  0.3586470   32
## 33     A         crime  0.3583466   33
## 34     A      mandates  0.3579620   34
## 35     A      horrific  0.3577144   35
## 36     A      ignoring  0.3574017   36
## 37     A      deported  0.3560671   37
## 38     A        sexual  0.3546610   38
## 39     A      trillion  0.3543917   39
## 40     A        impose  0.3542402   40
## 41     A        damage  0.3534434   41
## 42     A      bailouts  0.3516212   42
## 43     A       certain  0.3515684   43
## 44     A        aliens  0.3510642   44
## 45     A   unnecessary  0.3496602   45
## 46     A catastrophic  0.3488730   46
## 47     A       radical  0.3487190   47
## 48     A      flooding  0.3485382   48
## 49     A      occurred  0.3472314   49
## 50     A      problems  0.3468502   50
```

```
# (vi) Compare rank for a selected term of interest
R_neg_assoc_terms2 <- word2vec_similarity(x = R_neg_pos_dimension,
```

```
                                          y = R_embedding,
                                          top_n = 10000)
print(R_neg_assoc_terms2[R_neg_assoc_terms2$term2 %in% check_words,])
```

```
##     term1     term2 similarity rank
## 461     A obamacare  0.2646011  461
```

```
# Democrats
# ===========
D_embedding <- as.matrix(D_w2v)
# (iii)
D_pos_vectors <- D_embedding[which(rownames(D_embedding) %in% pos$V1),]
D_neg_vectors <- D_embedding[which(rownames(D_embedding) %in% neg$V1),]
D_pos_vector <- as.matrix(apply(D_pos_vectors,2,mean))
D_neg_vector <- as.matrix(apply(D_neg_vectors,2,mean))
# (iv)
D_pos_vector <- t(D_pos_vector)
D_neg_vector <- t(D_neg_vector)
D_neg_pos_dimension <- D_neg_vector - D_pos_vector
# (v) 50 most negative
D_neg_assoc_terms <- word2vec_similarity(x = D_neg_pos_dimension,
                                         y = D_embedding,
                                         top_n = 50)
print(D_neg_assoc_terms)
```

```
##    term1            term2 similarity rank
## 1      A          illegal  0.4047966    1
## 2      A       despicable  0.3998492    2
## 3      A        dangerous  0.3975298    3
## 4      A        heartless  0.3974743    4
## 5      A      devastation  0.3921925    5
## 6      A        misguided  0.3908446    6
## 7      A            cruel  0.3904763    7
## 8      A           racist  0.3877698    8
## 9      A          hateful  0.3802885    9
## 10     A        proposals  0.3772642   10
## 11     A         withdraw  0.3746619   11
## 12     A            fraud  0.3731231   12
## 13     A             fake  0.3729906   13
## 14     A         attempts  0.3720219   14
## 15     A    disappointing  0.3705467   15
## 16     A           simply  0.3701539   16
## 17     A         rhetoric  0.3694327   17
## 18     A       completely  0.3638396   18
## 19     A       disastrous  0.3633486   19
## 20     A           ignore  0.3627150   20
## 21     A          extreme  0.3614656   21
## 22     A          reports  0.3613901   22
## 23     A     catastrophic  0.3601367   23
## 24     A            voter  0.3600788   24
## 25     A        disturbing  0.3568618   25
## 26     A             sham  0.3567174   26
## 27     A      disgraceful  0.3566309   27
```

```
## 28     A        provisions  0.3533413   28
## 29     A            heroin  0.3512536   29
## 30     A              hate  0.3512185   30
## 31     A            notice  0.3509179   31
## 32     A            capita  0.3502652   32
## 33     A          shocking  0.3502579   33
## 34     A              kind  0.3491014   34
## 35     A          proposal  0.3488249   35
## 36     A            meaner  0.3482925   36
## 37     A unconstitutional  0.3473482   37
## 38     A            attack  0.3471999   38
## 39     A         launching  0.3469162   39
## 40     A           harmful  0.3453651   40
## 41     A            highly  0.3451610   41
## 42     A        cigarettes  0.3445372   42
## 43     A         terrorist  0.3443696   43
## 44     A            strike  0.3435391   44
## 45     A            likely  0.3434398   45
## 46     A            terror  0.3433445   46
## 47     A           attacks  0.3426621   47
## 48     A          reckless  0.3425946   48
## 49     A           however  0.3420930   49
## 50     A             worst  0.3419595   50
```

```
# (vi) Compare rank for a selected term of interest
D_neg_assoc_terms2 <- word2vec::word2vec_similarity(x = D_neg_pos_dimension,
                                                    y = D_embedding,
                                                    top_n = 10000)
print(D_neg_assoc_terms2[D_neg_assoc_terms2$term2 %in% check_words,])
```

```
##      term1     term2 similarity rank
## 1020     A obamacare  0.1888392 1020
```

There are indeed considerable difference in what words are the most negatively associated (top 50) between the two parties. For example, "deficit" and "iranian" features high for the republicans, while "racism" and "heartless" features high for democrats. Furthermore, comparing the rank-difference for the word "obamacare", we find that it it—in line with expectation—is more negatively associated for the republicans.

7. **BONUS** (not obligatory). How can we know if our results from #6 are statistically significant / robust? The non-parametric boostrap! Repeat step 6 (the latter part, where you selected a specific set of words to compare) but this time across ×20 bootstrapped samples. Report the upper and lower bound of comparisons you made in #6.

```
# (i) Import
neg <- fread('negative.txt',header = F)
pos <- fread('positive.txt',header = F)
# (ii)
# ===========
# Republicans
# ===========
R_dt <- fb_congress[party=="Republican",]
R_dt[,doc_id := NULL]
```

```r
B <- 20 # number of bootstraps
R_obamacare_ranks <- c() # to store each bootstrap's rank
for(b in 1:B){

  # Sample documents ids with replacement (= creating the b'th bootstrap sample)
  row_ids <- 1:nrow(R_dt)
  bootstrap_b_ids <- sample(x = row_ids, size = nrow(R_dt), replace = TRUE)
  bootstrap_b_data <- R_dt[bootstrap_b_ids,]

  # Then... we use the same code as we did before, without bootstrapping
  # Just that we replace the input to the corpus.

  # Make into corpus
  posts_corpus <- corpus(x = bootstrap_b_data, # Here we insert the boostraped data for this iteration
                         text_field = "message",
                         meta = list("party"))

  # Tokenize & clean from particular types of words
  mytokens <- tokens(x = posts_corpus,
                     remove_punct = TRUE,
                     remove_numbers = TRUE,
                     remove_symbols = TRUE,
                     remove_url = TRUE,
                     padding = FALSE)
  mytokens <-   tokens_select(x = mytokens,selection = 'remove',
                              valuetype = 'glob',
                              pattern = '@',
                              padding = FALSE)

  # Make tokens lowercase
  mytokens <- tokens_tolower(x = mytokens)

  # Collapse into strings within documents
  txt <- sapply(mytokens,function(x)paste(x,collapse = ' '))
  txt <- tolower(txt)

  # Fit word embeddings
  set.seed(123456789)
  w2v <- word2vec(x = txt,
                  type = "cbow",
                  window = 5,
                  dim = 50,
                  iter = 25,
                  hs = FALSE,
                  negative = 15)

  # Extract the whole embedding matrix
  embedding <- as.matrix(w2v)

  # Create the projection:
  # 1) Extract the relevant word vectors from the "embedding" matrix
  pos_vectors <- embedding[which(rownames(embedding) %in% pos$V1),]
  neg_vectors <- embedding[which(rownames(embedding) %in% neg$V1),]
```

```r
  # 2) Compute the average across each and keep matrix format
  pos_vector <- as.matrix(apply(pos_vectors,2,mean))
  neg_vector <- as.matrix(apply(neg_vectors,2,mean))
  # 3) Transform to get 1 x K
  pos_vector <- t(pos_vector)
  neg_vector <- t(neg_vector)
  # 4) Compute the difference to get a gender dimension
  neg_pos_dimension <- neg_vector - pos_vector

  # Identify (and store) similarity rank (to sentiment dimension) for the word "fbi"
  neg_assoc_terms <- word2vec_similarity(x = neg_pos_dimension,
                                         y = embedding[-which(rownames(embedding) %in% neg$V1),],
                                         top_n = 10000) # Set this to a large value to get d to all wor
  neg_assoc_terms <- as.data.table(neg_assoc_terms)
  R_obamacare_ranks[b] <- neg_assoc_terms[term2=="obamacare"]$rank
}
```

```r
# Compute 90th interval
R_obamacare_ranks <- R_obamacare_ranks[order(R_obamacare_ranks,decreasing=F)]
print(paste0('(R) Mean:',mean(R_obamacare_ranks)))
```

```
## [1] "(R) Mean:407.95"
```

```r
print(paste0('(R) Boostrap 90% interval: [',
             R_obamacare_ranks[2], ',',
             R_obamacare_ranks[19],']'))
```

```
## [1] "(R) Boostrap 90% interval: [259,610]"
```

```r
# Republicans
# ===========
D_dt <- fb_congress[party=="Democrat",]
D_dt[,doc_id := NULL]
B <- 20 # number of bootstraps
D_obamacare_ranks <- c() # to store each bootstrap's rank
for(b in 1:B){

  # Sample documents ids with replacement (= creating the b'th bootstrap sample)
  row_ids <- 1:nrow(D_dt)
  bootstrap_b_ids <- sample(x = row_ids, size = nrow(D_dt), replace = TRUE)
  bootstrap_b_data <- D_dt[bootstrap_b_ids,]

  # Then... we use the same code as we did before, without bootstrapping
  # Just that we replace the input to the corpus.

  # Make into corpus
  posts_corpus <- corpus(x = bootstrap_b_data, # Here we insert the boostraped data for this iteration
                         text_field = "message",
                         meta = list("party"))

  # Tokenize & clean from particular types of words
  mytokens <- tokens(x = posts_corpus,
```

```
                     remove_punct = TRUE,
                     remove_numbers = TRUE,
                     remove_symbols = TRUE,
                     remove_url = TRUE,
                     padding = FALSE)
  mytokens <-   tokens_select(x = mytokens,selection = 'remove',
                              valuetype = 'glob',
                              pattern = '@',
                              padding = FALSE)

  # Make tokens lowercase
  mytokens <- tokens_tolower(x = mytokens)

  # Collapse into strings within documents
  txt <- sapply(mytokens,function(x)paste(x,collapse = ' '))
  txt <- tolower(txt)

  # Fit word embeddings
  set.seed(123456789)
  w2v <- word2vec(x = txt,
                  type = "cbow",
                  window = 5,
                  dim = 50,
                  iter = 50,
                  hs = FALSE,
                  negative = 15)

  # Extract the whole embedding matrix
  embedding <- as.matrix(w2v)

  # Create the projection:
  # 1) Extract the relevant word vectors from the "embedding" matrix
  pos_vectors <- embedding[which(rownames(embedding) %in% pos$V1),]
  neg_vectors <- embedding[which(rownames(embedding) %in% neg$V1),]
  # 2) Compute the average across each and keep matrix format
  pos_vector <- as.matrix(apply(pos_vectors,2,mean))
  neg_vector <- as.matrix(apply(neg_vectors,2,mean))
  # 3) Transform to get 1 x K
  pos_vector <- t(pos_vector)
  neg_vector <- t(neg_vector)
  # 4) Compute the difference to get a gender dimension
  neg_pos_dimension <- neg_vector - pos_vector

  # Identify (and store) similarity rank (to sentiment dimension) for the word "fbi"
  neg_assoc_terms <- word2vec_similarity(x = neg_pos_dimension,
                                         y = embedding[-which(rownames(embedding) %in% neg_terms$V1),],
                                         top_n = 10000) # Set this to a large value to get d to all wor
  neg_assoc_terms <- as.data.table(neg_assoc_terms)
  D_obamacare_ranks[b] <- neg_assoc_terms[term2=="obamacare"]$rank
}
```

```
# Compute 90th interval
D_obamacare_ranks <- D_obamacare_ranks[order(D_obamacare_ranks,decreasing=F)]
```

```r
print(paste0('(D) Mean:',mean(D_obamacare_ranks)))
```

```
## [1] "(D) Mean:1202.25"
```

```r
print(paste0('(D) Boostrap 90% interval: [',
             D_obamacare_ranks[2], ',',
             D_obamacare_ranks[19],']'))
```

```
## [1] "(D) Boostrap 90% interval: [305,3233]"
```

Although the *mean similarity rank* to the negative sentiment dimension is substantially higher (1=highest) for the republicans (408 vs. 1202), the 90% boostrap confidence intervals overlap. What to do about these large confidence intervals? One thing is to collect/use a lager data set. This provides more stable estimates, and therefore also more narrow confidence bans. Another is to focus on terms which are more frequent in the corpora(s) you do have; their embedding estimates should also be more stable.