Homework 2 DS-GA 1015

Alexandre Vives, N14948495

4/11/2022

Please list everyone you collaborated with on this assignment

---

```
library(ggplot2)
library(dplyr)
library(quanteda)
library(topicmodels)
library(stm)
```

Question 1

a) The variational inference approach is the best method for this case (a lot of data and not much time) because it is the quickest due to the initial distribution assumption but it could also be risky if that assumption is not correct.

b) One method could be to fit several models with different amount of topics and compare their goodness-of-fit along with its run-time and select the one with the best goodness-of-fit with a decent runtime. Another method is to take advantage of the Dirichlet distribution and use the hierarchical Dirichlet process, which can find the optimal amount of topics dinamically. Lastly, we could formulate the topics using the words of each document and approximate the number of total topics.

c1) The most plausible answer is that the parameter alpha is pretty small (less than 1). Alpha can be seen as the "weight" of each topic, where the higher it is, the less documents contain that topic, therefore if it is small, our prior will be that documents tend to be focused on specific topics.

c2) In this case, beta (or eta in slides) being high is probably the answer. Beta sets our a priori beliefs about the probability of each word per topic, where a high beta evens out the probabilities and a low beta assigns big probabilities to specific words.

d) The EM approach is the best method for this case due to its high accuracy (even though it takes a long time to run because of all the required integrations).

e) The words given a topic are not independent, because the probability of seeing a word of topic a changes if we have seen a word of topic a previously in that document.

P(w|theta) = p(theta|w)p(w) != p(w) P(w|beta,z) = P(beta,theta,z,w) / (P(beta), P(theta), P(z|theta))

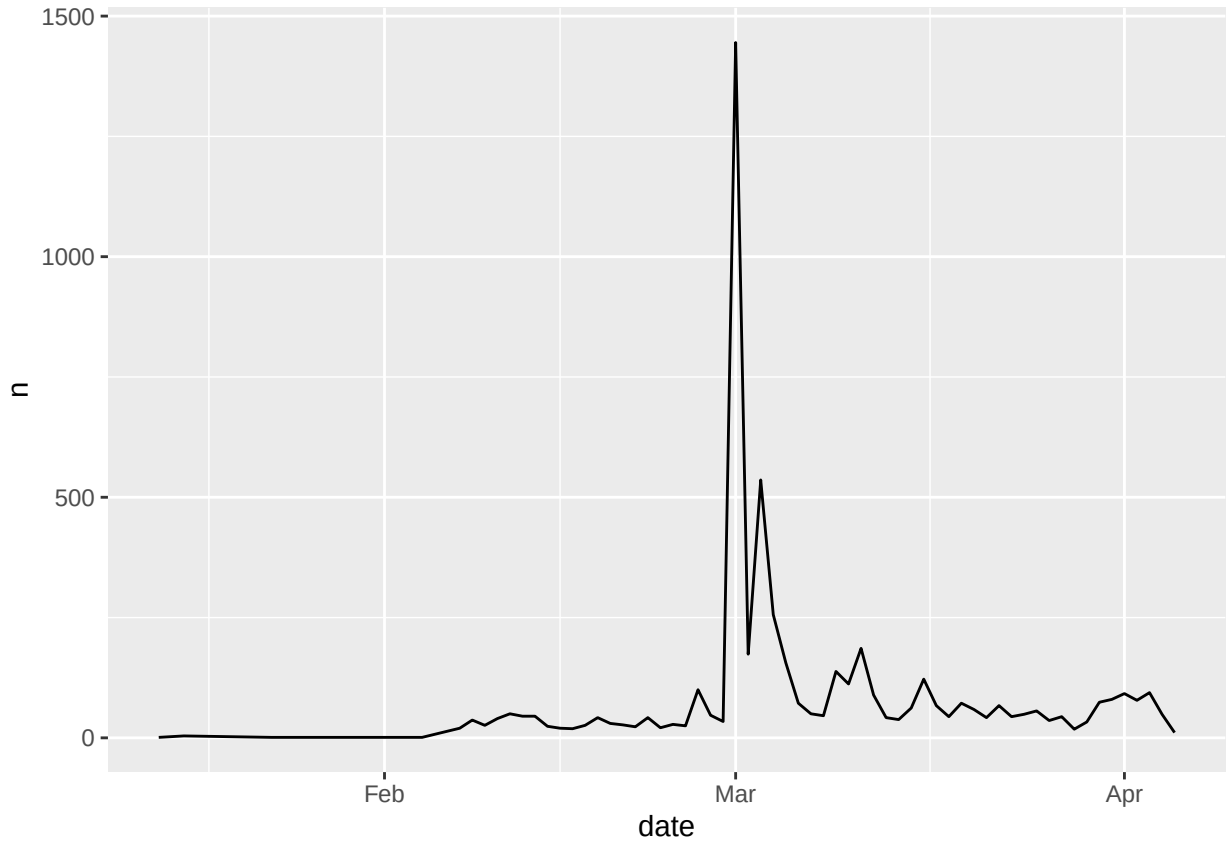
QUESTION 2

a)

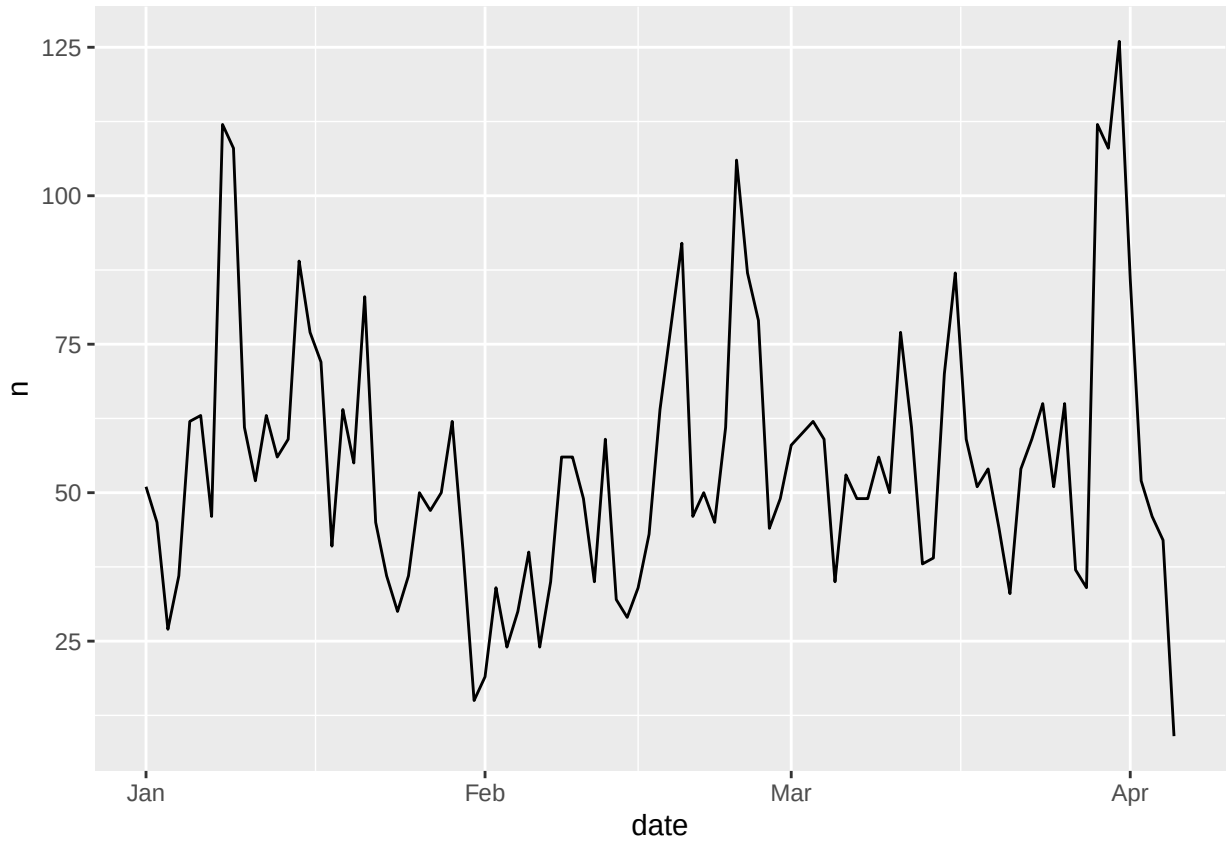
```
df <- read.csv("vaccinationtweets.csv")
df$date <- as.Date(df$date , format = "%m/%d/%Y")
df <- df[format(df$date, format = "%Y")>=2021 & df$hashtags %in% c('PfizerBioNTech', 'Covaxin'),]
```

```
df_temp <- df %>% count(hashtags, date)
covaxin <- df_temp[df_temp$hashtags == "Covaxin", c('date', 'n')]
pfizer <- df_temp[df_temp$hashtags == "PfizerBioNTech", c('date', 'n')]
```

```
ggplot(data = covaxin, aes(date, n)) + geom_line()
```




```
ggplot(data = pfizer, aes( date, n )) + geom_line()
```



b) Removing very rare words could be advantageous because the rarest of words in a corpus of tweets are probably misspelled words (lowest 10% frequency). On the other hand, rare words (between top 50% and lowest 30%) could be very useful to identify the topic of the tweet as they are probable to belong to a specific topic instead of several topics.

c) We should remove punctuation, stop-words, stem (they provide no meaningful information) and the lowest 10% frequency words (hopefully removing misspelled words).

```
df$text <- gsub("[^[:alnum:] ]", "", df$text)
df$text <- stringi::stri_trans_general(df$text, "latin-ascii")

corpus_df <- textProcessor(df$text,lowercase = TRUE, removestopwords = TRUE,
                           removenumbers = TRUE, removepunctuation = TRUE, stem = TRUE, onlycharacter =
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Removing numbers...
## Stemming...
## Creating Output...
```

```
stm_input <- prepDocuments(documents=corpus_df$documents, vocab=corpus_df$vocab,
                           meta = NULL, lower.thresh = 30)
```

```
## Removing 22748 of 23176 terms (42923 of 104868 tokens) due to frequency
## Removing 61 Documents with No Words
## Your corpus now has 10575 documents, 428 terms and 61945 tokens.
```

```
# Match teh size in case some documents were removed
binary_col <- df$hashtags[-stm_input[["docs.removed"]]]
date_col <- df$date[-stm_input[["docs.removed"]]]
```

d)

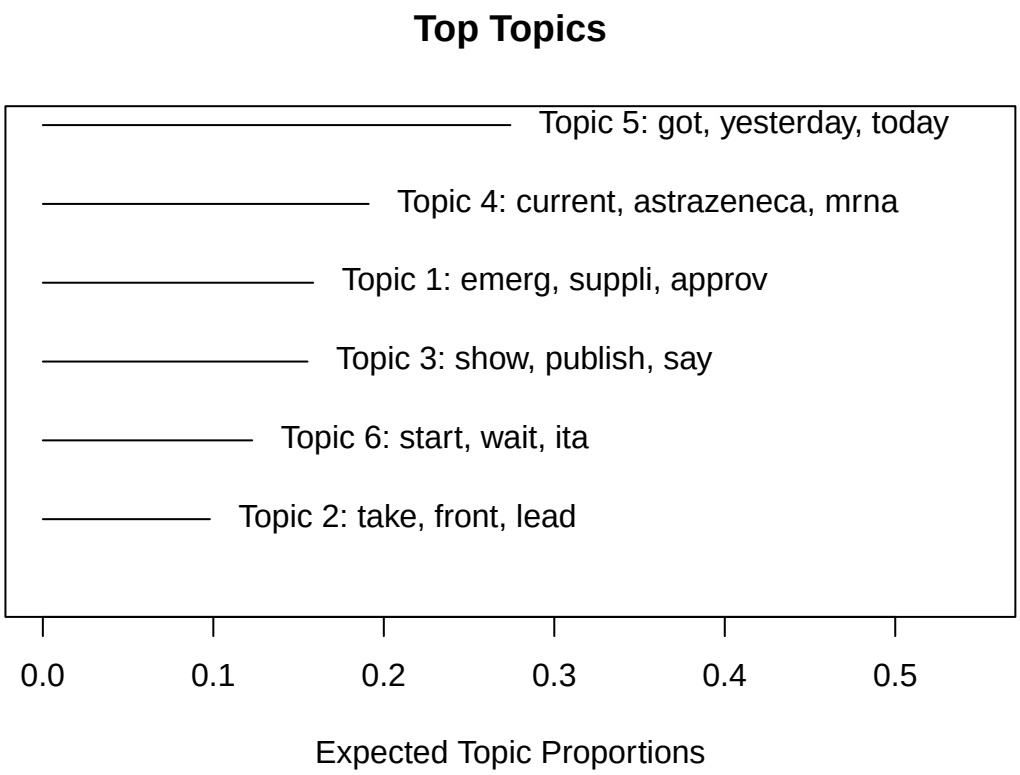
```
df$hashtags<- as.numeric(plyr::mapvalues(df$hashtags,
                                         from=c("PfizerBioNTech","Covaxin"), to=c(0,1)))
```

```
model <- stm(stm_input$documents, stm_input$vocab, K=6, content=binary_col,
             prevalence=~binary_col + s(date_col))
```

It took 65 iterations to converge.

e)

```
plot(model, type = "summary")
```

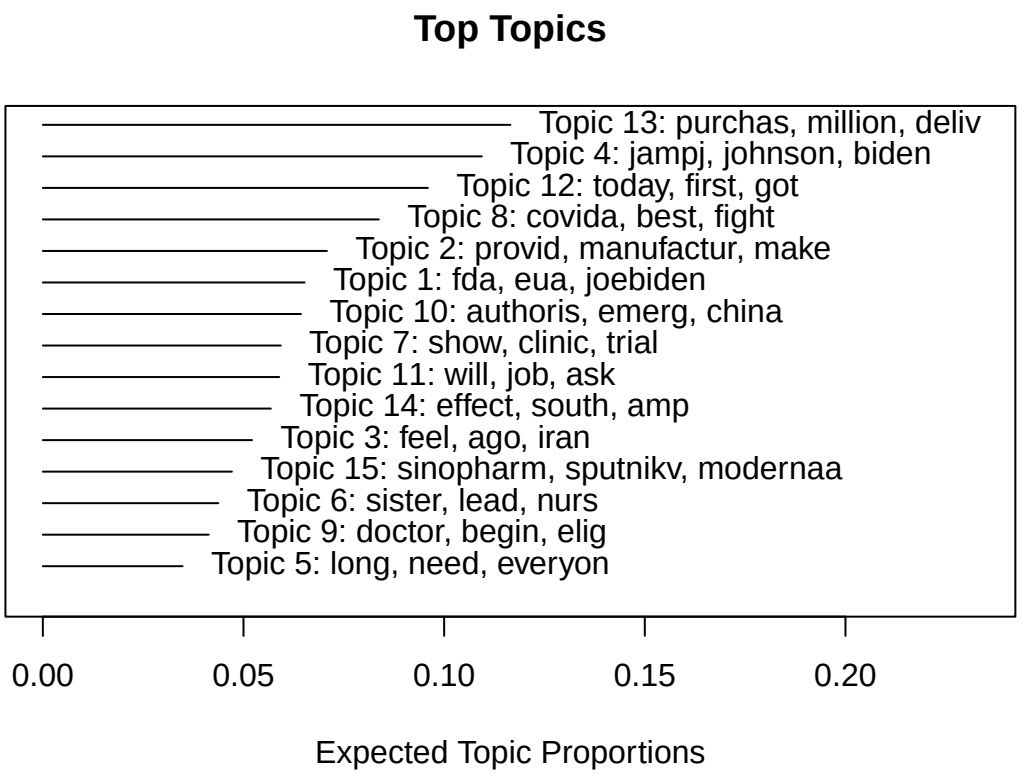


f)

```
model2 <- stm(stm_input$documents, stm_input$vocab, K=15, content=binary_col,
              prevalence=~binary_col + s(date_col))
```

It took _____ iterations to converge.

```
plot(model2, type = "summary")
```



g) When the pfizer vaccine appears, the words associated with that topic appear to be more positive, whereas fori covaxine the connotation is more negative. The graph of the prevalence over time per topic shows how the majority of topics remain constant but a small amount increase/decrease

QUESTION 3

a) As the question mentions, dictionary-based methods do not usually include an exhaustive list of terms and that is due to the high dimensionality it could reach. Pre-trained word embeddings could be used to combine words that are very similar to each other and greatly reduce that dimensionality without losing its prediciton power.

b) It depends on the balance of classes of our target variable. If the balance was similar I would use accuracy, but it was unbalanced I would use F-1 score. Additionally, if I cared more about a specific class I would use either recall or precision (the one corresponding to the correctly classifier samples of that class).

c) Word embeddings are used to assign a vector to each word that encodes its meaning, therefore allowing you to much more accurately determine which words are similar to each other. This allows you to combine many words in the bag-of-words matrix and therefore greatly reduce dimensionality.

d) Upon receiving this request, it is obvious that it is much more practical to use an off-the shelf embedding rather than training your own (specially considering my boss' impatience). But, that would only be a good idea if the off-the-shelf embedding I plan to use was trained on similar data to the one I have.

e) Word embeddings are mostly a black box and therefore it is possible that they contain some kind of bias (gender or racial bias for example). If you use a pretrained word embedding you might be introducing that bias into your predictions and, depending on the reach your model has, it could end up perpetuating that bias.