Text as Data: Homework 5

Jeff Ziegler

August 23, 2017

1 Using the `Structural Topic Model` in `R`

Using the preprocessed data based on the NYT Json file and the subsequent document term matrix for analysis in `stm`.

a) Download the `stm` package for `R` from `CRAN`

b) Convert the document-term matrix to the appropriate format. To do this, create a list in `R` where each component of the list corresponds to an individual document. Store in each component of the list a two rom matrix. The number of columns corresponds to the number of non-zero entries for the document in the document-term matrix. The first row will describe the words used in the document (the columns with the non-zero entry). The second row will correspond to a count of each of the words in the document (they should all be non-zero)

c) Following the help file in `STM` fit a model with 8 topics that conditions on the `desk` of origin for topic prevalence

d) Use `labelTopics` to label each of the topics

e) Compare the 8 topic proportions for each document to the 8 topic proportions without conditioning on `desk` (in vanilla LDA). How do the results differ?

```
# Problem 1

# load libraries and .csv files
library(rjson); library(stm); library(tm); library(stringr)
NYTjson <- fromJSON(file="~/Documents/Git/WUSTL_textAnalysis/nyt_ac.json")

# fit a model with 8 topics that conditions on the desk of origin
# prep documents for STM
stmData <- as.data.frame(cbind(sapply(lapply(NYTjson, `[[`, c('meta', 'dsk')),
    paste0, collapse=""),
                                sapply(lapply(NYTjson, `[[`, c('body', 'body_
    text')), paste0, collapse="")))
```

```r
names(stmData) <- c("desk", "documents"); stmData$documents <- as.character(stmData$documents)
# create a function to only take first 25 words to preview docs
# visualize labels
makeShortDoc <- function(x) {
  ul = unlist(strsplit(x, split = "\\s+"))[1:25]
  paste(ul,collapse=" ")
}
stmData$shortdoc <- unlist(lapply(stmData$documents, makeShortDoc))

#stmData$desk <- as.factor(stmData$desk)
processedSTM <- textProcessor(stmData$documents, metadata = stmData)
outSTM <- prepDocuments(processedSTM$documents, processedSTM$vocab, processedSTM$meta)

# fit STM w/ 8 topics
# using the default settings found in the help package
stmModel <- stm(documents = outSTM$documents, vocab = outSTM$vocab, K = 8,
     prevalence =~desk, max.em.its = 75, data = outSTM$meta,
     init.type = "Spectral")

# d) label each topic
labelTopics(stmModel, c(1:8))

# create visualization of topics by previewing docs
pdf("~/Documents/Git/WUSTL_textAnalysis/HW5topicReview.pdf")
par(mfrow = c(2, 4), mar = c(.5, .5, 1, .5))
for(i in 1:8){
  plotQuote(findThoughts(stmModel, texts = stmData$shortdoc, n = 2, topics = i)$docs[[1]],
            width = 30, main = paste0("Topic", i, sep=" "))
}
dev.off()

# e) run vanilla LDA
# using the default settings found in the help package
ldaModel <- stm(documents = outSTM$documents, vocab = outSTM$vocab, K = 8,
                max.em.its = 75, data = outSTM$meta, init.type = "Spectral")
# plot LDA and STM topic proportions
pdf("~/Documents/Git/WUSTL_textAnalysis/HW5topicProportionsSTM.pdf")
plot(stmModel, type = "hist", xlim = c(0, .5), labeltype="frex")
dev.off()

pdf("~/Documents/Git/WUSTL_textAnalysis/HW5topicProportionsLDA.pdf")
plot(ldaModel, type = "hist", xlim = c(0, .5), labeltype="frex")
dev.off()
# not much difference between the two estimated topic proportions
```

Figure 1: Example documents highly associated with topics.

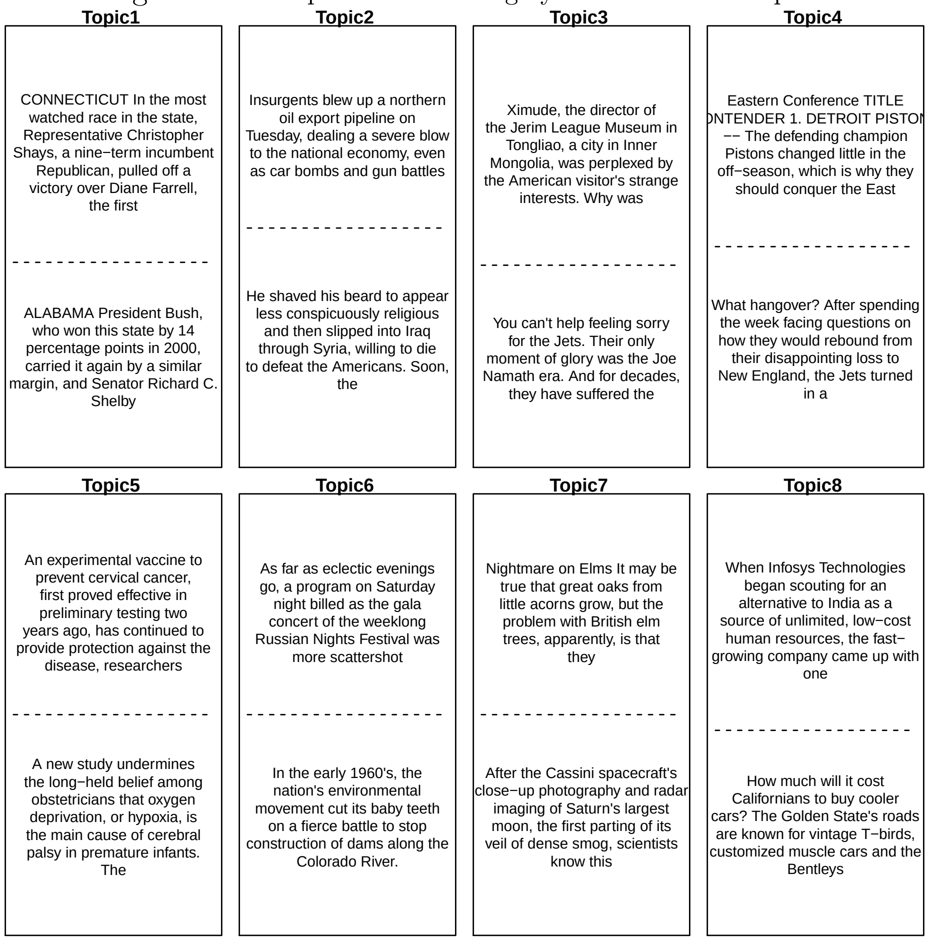
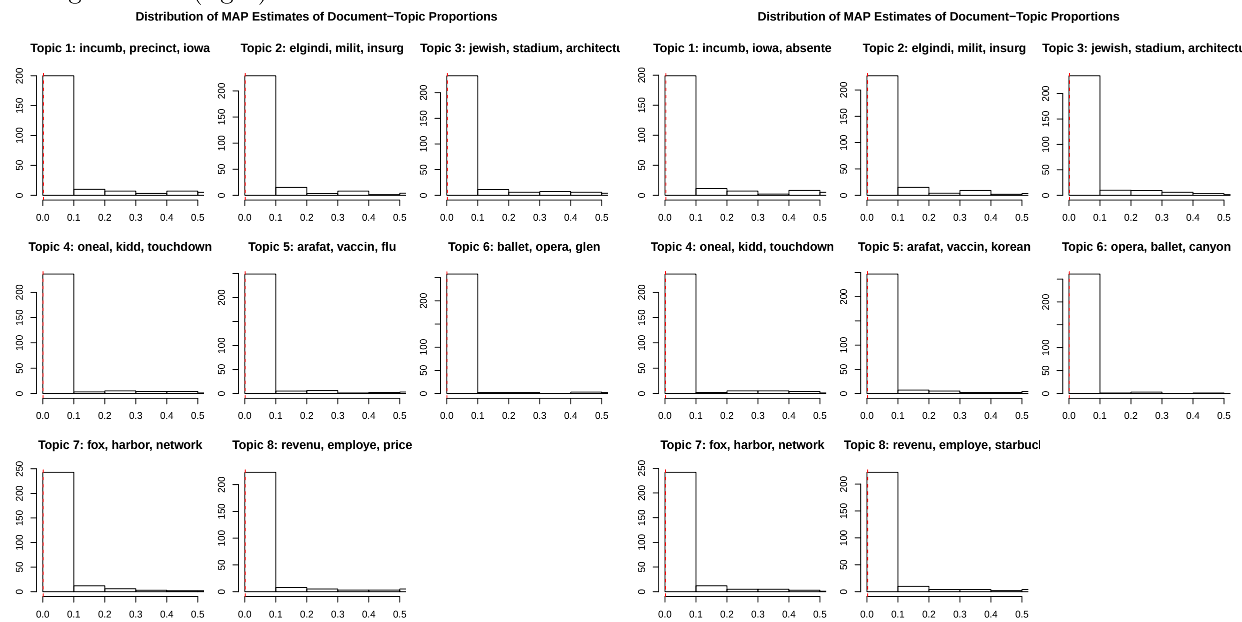


Figure 2: Expected distribution of topic proportions across the documents, vanilla LDA (left) and conditioning on `desk` (right).



2 Machiavelli's Prince

In this part of the assignment we will analyze Machiavelli's *The Prince*. Download `Mach.tar` from the course website and expand the compressed folder. (This is relevant `http://xkcd.com/1168/`).

Each file represents a subset of the manuscript. We will analyze its contents using principal components, multidimensional scaling, and clustering methods.

Create a Document-Term Matrix

Using the sections from the Machiavelli text, create a document term matrix.

- Discard punctuation, capitalization
- Apply the porter stemmer to the documents
- Identify the 500 most common unigrams
- Create a $N \times 500$ document term matrix $\boldsymbol{X}$, where the columns count the unigrams and the rows are the documents

We will work with a normalized version of the term document matrix. That is we will divide each row by the total number of words in the top 500 unigrams used:

$$\boldsymbol{x}_i^* = \frac{\boldsymbol{x}_i}{\sum_{j=1}^{500} x_{ij}}$$

$$\boldsymbol{X}^* = \begin{pmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \\ \vdots \\ \boldsymbol{x}_N^* \end{pmatrix}$$

```r
# Problem 2

# we'll make our DTM using the tm package primarily
# create a corpus and transform data
# read in files
# make corpus
machText <- Corpus(DirSource("~/Documents/Git/WUSTL_textAnalysis/MachText"),
                   readerControl = list(language = "en"))
# remove punctuation and capitalization
machText <- tm_map(machText, removePunctuation)
machText <- tm_map(machText, content_transformer(tolower))
# we'll also remove stop words
machText <- tm_map(machText, removeWords, stopwords("english"))
# apply porter stemmer
machText <- tm_map(machText, stemDocument)
# create DTM of machTexts
machDTM <- DocumentTermMatrix(machText)
# reduce DTM to  top 500 unigrams
machDTM <- machDTM[,-c(501:2367)]
# normalize by term frequaency - i.e. divide count of each word
# in document by total number of words in document
machDTM <- normalize(machDTM$j)

# reduce DTM to  top 500 unigrams
machDTM <- machDTM[,-c(501:2367)]
# show top 500 unigrams
# machDTM$dimnames$Terms
# see what DTM looks like
machDTM <- machDTM/rowSums(as.matrix(machDTM))
inspect(machDTM)
```

```
<<DocumentTermMatrix (documents: 188, terms: 500)>>
Non-/sparse entries: 6279/87721
Sparsity           : 93\%
Maximal term length: 14
Weighting          : term frequency (tf)
Sample             :
```

```
             Terms
Docs                 alway        men        one      peopl      power      ruler
  Mach_1.txt   0.00000000 0.01315789 0.00000000 0.00000000 0.00000000 0.01315789
  Mach_101.txt 0.00000000 0.01086957 0.00000000 0.00000000 0.00000000 0.01086957
  Mach_106.txt 0.01639344 0.00000000 0.01639344 0.00000000 0.00000000 0.01639344
  Mach_110.txt 0.00000000 0.00000000 0.18000000 0.00000000 0.04000000 0.00000000
  Mach_115.txt 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.05128205
  Mach_126.txt 0.04761905 0.00000000 0.04761905 0.04761905 0.02380952 0.04761905
  Mach_153.txt 0.03333333 0.06666667 0.00000000 0.00000000 0.03333333 0.06666667
  Mach_168.txt 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.02500000
  Mach_56.txt  0.00000000 0.00000000 0.02040816 0.02040816 0.00000000 0.02040816
  Mach_76.txt  0.03921569 0.01960784 0.00000000 0.01960784 0.01960784 0.05882353
```

Low Dimensional Embeddings with Principal Components

1) Wise Will (WW), your friend with a weird name, notices you looking at the slides about principal component analysis (PCA). WW casually remarks that the variance of the eigenvalues of the variance-covariance matrix is a useful heuristic for knowing if PCA can be fruitfully applied to some document-term matrix. WW, completely unsolicited, explains that as the variance of the eigenvalues goes up, the more useful PCA will be. He then laughs and leaves your office. WW is kind of a jerk.

   Let's formalize WW's suggestion. Suppose document-term matrix $\boldsymbol{X}$ has variance-covariance matrix $\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^{'}\boldsymbol{X}}{N}$. And suppose that $\boldsymbol{\Sigma}$ has eigenvalues $\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0$. Then we calculate the variance of the eigenvalues as

$$\sigma^2 \;=\; \frac{1}{d}\sum_{j=1}^{d}(\lambda_j - \bar{\lambda})^2$$

   where $\bar{\lambda}$ is $\frac{1}{d}\sum_{i=1}^{d}\lambda_i$. WW is saying that as σ^2 gets bigger, a low-dimensional embedding via PCA will provide a better summary of our data.

   Does WW have a good point? Why or why not? (Hint: what do the eigenvalues represent?)

Reducing the variance of the eigenvalues may or may not help improve PCA's ability to better summarize data, but reducing the sum of ?remaining? eigenvalues reduces the error because the total variance explained = (sum of included eigenvalues)/(sum of all eigenvalues).

2) Apply the function `prcomp` to $\boldsymbol{X}^*$. Be sure to set use a scaled version of the data, by setting `scale = T`, which will ensure that each column has unit variance.

a) Create a plot of variance explained by each additional principal component. What does this plot suggest about the number of components to include?

b) Plot the two-dimensional embedding of the text documents. Label the texts with their number. (Each file is `Mach_XX.txt`, where `XX` is the chunk number)

c) Label the two largest principal components. What does this embedding suggest about the primary variation this representation of the Prince? (Hint: if your `embed` is your object with principal components, examine `embed$rotation`)

```r
# Problem 3

# 2) apply the function prcomp
# scale data to ensure each column has unit variance
machPCA <- prcomp(machDTM, scale = T)

# a ) reate a plot of variance explained by each additional principal component
pdf("~/Documents/Git/WUSTL_textAnalysis/HW5screePlot.pdf")
plot(machPCA, type = "l")
dev.off()
# the "elbow rule" doesn't really apply, and we don't want to introduce
# more error for better fit at a certain point
# so we'll include 10 components

pdf("~/Documents/Git/WUSTL_textAnalysis/HW5pcaEmbedding.pdf")
plot(machPCA$rotation, pch='',
     xlab="1st Principal Component",ylab="2nd Principal Component")
text(machPCA$rotation, labels=str_extract(machDTM$dimnames$Docs, "[[:digit:]]+"),cex= 0.7)
dev.off()

# c) looking at figure 4, it appears that the first two components
# are orthogonal to each other and that most documents tend
# toward zero along both dimensions
# find extreme docs (13, 112, 134; and 76, 85)
# first component seems to be advice to the ruler
# w/ examples that use Romans frequently
# second compotent is discussing protecting a ruler's state
```

Figure 3: Scree plot of the variance explained by the addition of each component.

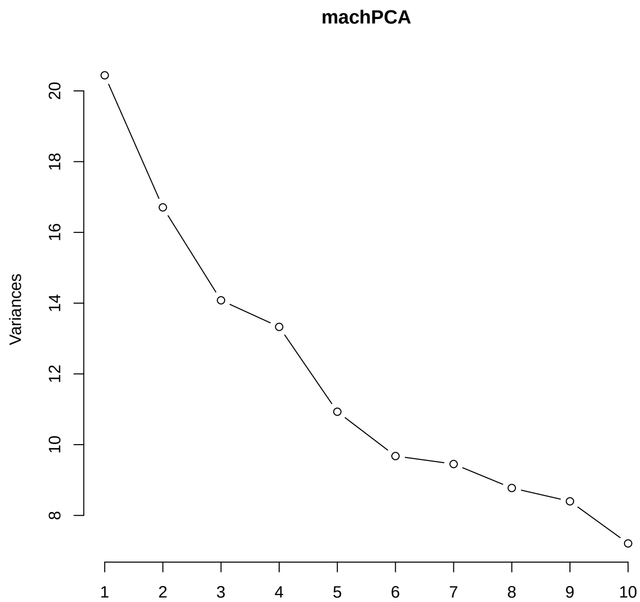
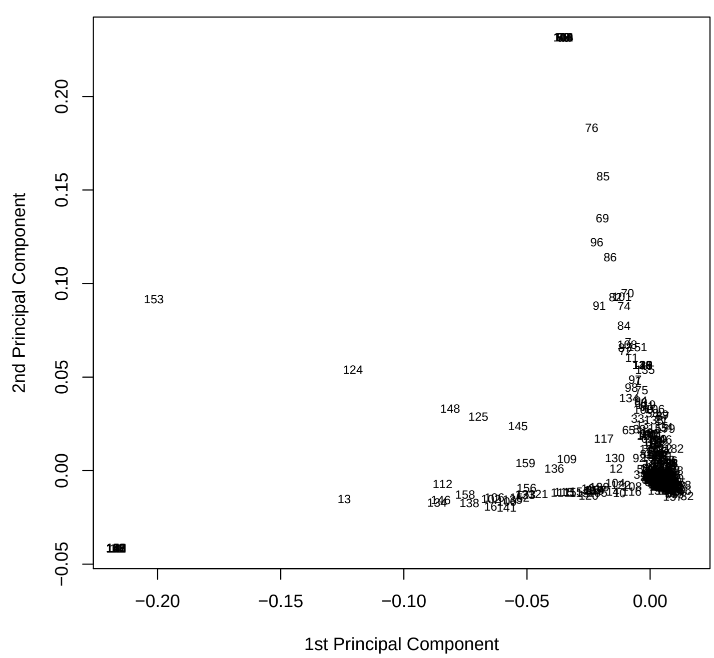


Figure 4: Projection of documents on the first two principal components.



3) An alternative method—discussed at the end of the seventh lecture—is multidimensional scaling (MDS). Classic MDS attempts to preserve distances between objects in a low dimensional scaling.

   a) Calculate the Euclidean distance between each document using $\boldsymbol{X}^*$. Call this matrix $\boldsymbol{D}(\boldsymbol{X}^*)$ (Hint: use `R`'s built in function `dist`)

   b) Apply the classic MDS to $\boldsymbol{D}(\boldsymbol{X}^*)$ using the `R` function `cmdscale`. That is, execute the code
   `mds_scale<- cmdscale(DISTANCE_MATRIX, k = 2)`

   c) Apply PCA to $\boldsymbol{X}^*$, but this time do not use `prcomp`'s scaling option. That is, use `prcomp` with `scale = F`.

   d) Compare the first dimension of the output from classic MDS to the first dimension of the embedding from principal components. What is the correlation between the embeddings?

   d) Now use `dist` to create a distance matrix using the `manhattan` metric, apply Classic multidimensional scaling to the distance matrix based on manhattan distance, and compare the first dimension of this embedding to the first dimension from PCA. What is the correlation?

   e) What do you conclude about the relationship between PCA and MDS?

```r
# c) looking at figure 4, it appears that the first two components
# are orthogonal to each other and that most documents tend
# toward zero along both dimensions
# find extreme docs (13, 112, 134; and 76, 85)
# first component seems to be advice to the ruler
# w/ examples that use Romans frequently
# second compotent is discussing protecting a ruler's state

# 3 a) calculate the Euclidean distance of machDTM
euclideanMachDTM <- as.matrix(dist(machDTM[,-1], method = "euclidean"))
# b) apply the classic MDS
classicMDS <- cmdscale(euclideanMachDTM, k = 2)
# c) re-run PCA w/o scaling
machPCAunscaled <- prcomp(machDTM, scale = F)
# d) check correlation between embeddings
cor(classicMDS[,1], machPCAunscaled$x[,1])
# cor = 0.99
# e) create distance matrix using the manhattan metric
manhattanMachDTM <- as.matrix(dist(machDTM[,-1], method = "manhattan"))
# apply classic multidimensional scaling
manhattanMDS <- cmdscale(manhattanMachDTM, k = 2)
cor(manhattanMDS[,1], machPCAunscaled$x[,1])
# cor = 0.95

# when PCA minimizes dimensions, tries to preserve covariance of data
```

```
# when MDS minimizes dimensions, tries to preserve distance between data
    points
# so if covariance in data = distance (euclidean or manhattan) between data
    points they should be the same
```