# HW4

*Napon Jatusripitak*

*2/28/2018*

## 1 LDA

1. Import the data (hint: use DirSource), pre-process, and set up a DTM.

```
#Import
mydirectory <- file.path("~", "Downloads", "nsf")
mydocs <- VCorpus(DirSource(mydirectory), readerControl=list(language="eng"))

#Pre-process & DTM
mydocs.dtm <- mydocs %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(stemDocument) %>%
  tm_map(stripWhitespace) %>%
  tm_map(PlainTextDocument) %>%
  DocumentTermMatrix() %>%
  removeSparseTerms(sparse = 0.99)
```

2. Use LDA to assess topics in these abstracts, first with 5 topics, then with 10.

```
#https://stackoverflow.com/questions/13944252/remove-empty-documents-from-documenttermmatrix-in-r-topic

#Find the sum of words by row
rowTotals <- apply(mydocs.dtm, 1, sum)

#Remove all docs without words
mydocs.dtm   <- mydocs.dtm[rowTotals> 0, ]

#Run LDA
mod.out.5 <- LDA(mydocs.dtm, k=5, control = list(seed=6))
mod.out.10 <- LDA(mydocs.dtm, k=10, control = list(seed=6))
```

3. Report these topics in both table and visual formats.

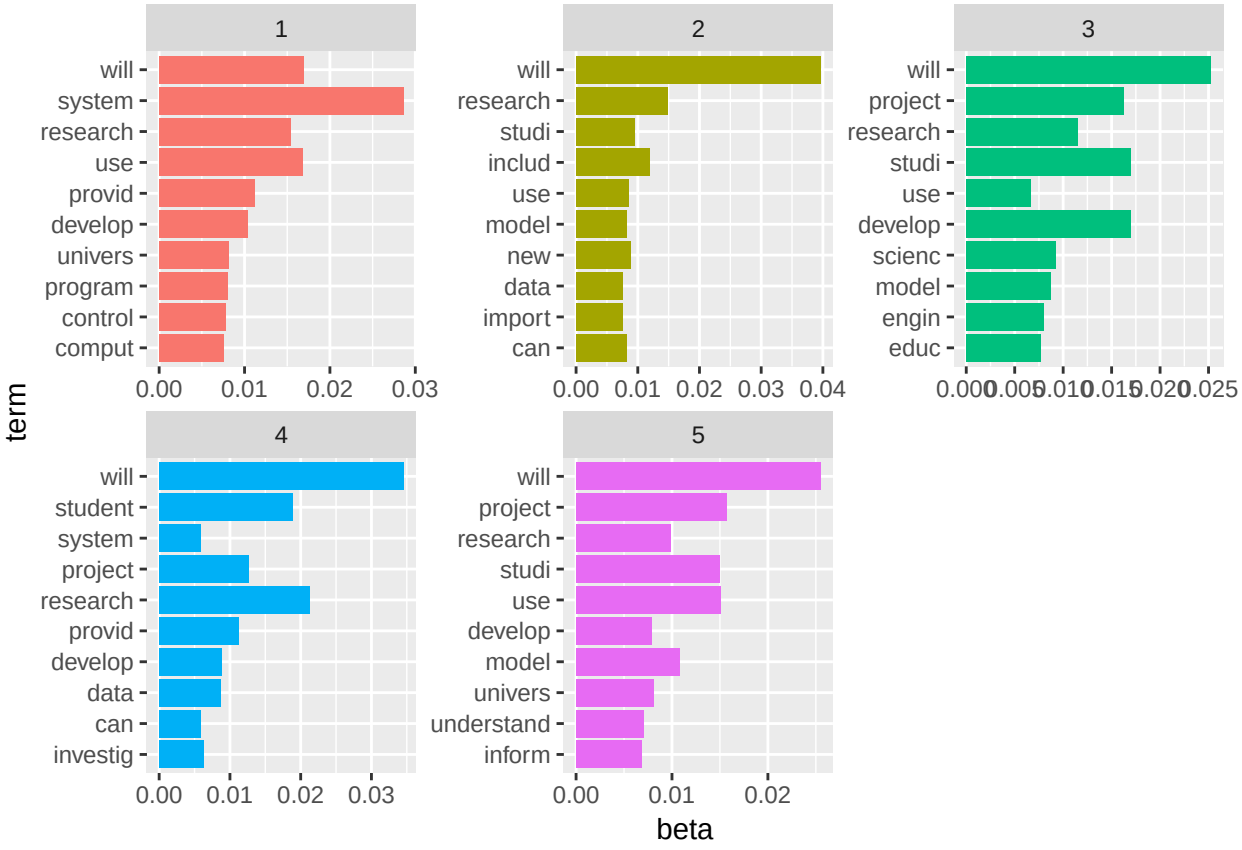
For k=5

```
tidy(mod.out.5)
```

```
## # A tibble: 8,170 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 abil  0.00160
##  2     2 abil  0.00135
##  3     3 abil  0.0000529
##  4     4 abil  0.000285
##  5     5 abil  0.000456
##  6     1 abl   0.000561
```

```
##  7     2 abl   0.000462
##  8     3 abl   0.000557
##  9     4 abl   0.000457
## 10     5 abl   0.0000940
## # ... with 8,160 more rows
```

```
top.terms <- tidy(mod.out.5) %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top.terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



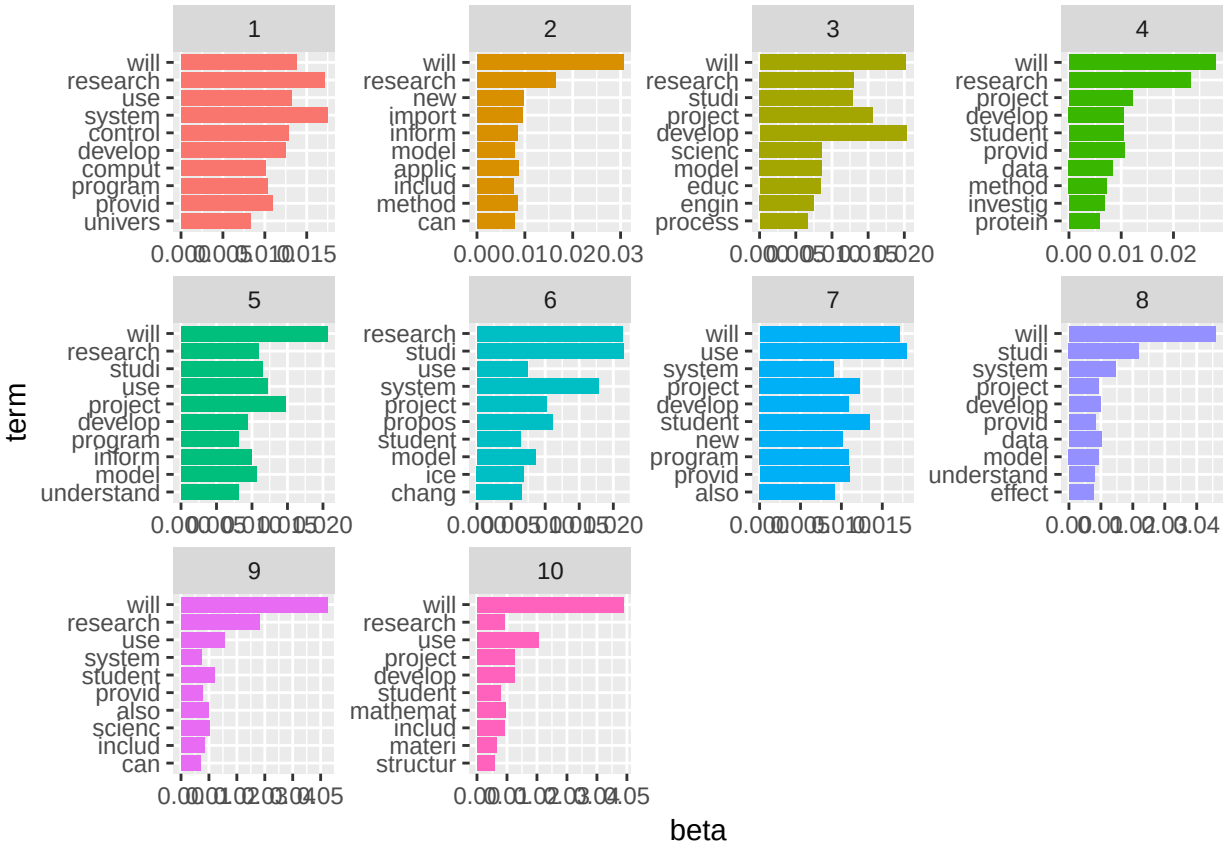

For k=10

```
tidy(mod.out.10)
```

```
## # A tibble: 16,340 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 abil   0.00115
##  2     2 abil   0.000941
##  3     3 abil   0.0000381
```

```
##  4     4 abil  0.000200
##  5     5 abil  0.000323
##  6     6 abil  0.000912
##  7     7 abil  0.00108
##  8     8 abil  0.00121
##  9     9 abil  0.000886
## 10    10 abil  0.000751
## # ... with 16,330 more rows
```

```
top.terms <- tidy(mod.out.10) %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top.terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```




4. Compare your results: how does the 10-topic model differ from the 5-topic model?

When we compare the 10-topic model to the 5-topic model, the 10-topic model provides a more specific assessment of the terms that are associated with each topic. For instance, in the 5-topic model, many of the topics share common terms (ie. “will”, “project”, “research”). It is difficult to tell the topics apart. However, in the 10-topic model, the terms that are associated with each topic are not necessarily shared but they are

more informative of what the topic is about (ie. “protein”, “mathemat”).

## 2 Structural Topic Models

For this problem, use the data on TED talks, which are posted on Canvas. 1. Import the data, pre-process the transcripts, and create a DTM.

```
# Import
ted_talks <- read.csv("ted_talks.csv", stringsAsFactors = F)

#Pre-process & DTM
ted_talks.dtm <- VectorSource(ted_talks$transcript) %>%
Corpus() %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stop_words$word) %>%
  tm_map(stemDocument) %>%
  DocumentTermMatrix()
```

2. Set up the DTM to be correctly formatted for use with the stm package. Use the documentation for the package to assist with this.

```
out <- stm::readCorpus(ted_talks.dtm, type = 'slam')
```

3. Use stm to fit a structural topic model with 9 topics, conditioning on ted_type, the venue of each of these TED talks.

```
ted.out <- stm(documents = out$documents,
               vocab = out$vocab,
               K = 9,
               prevalence = ~ted_type,
               data = ted_talks)
```

```
## Beginning Spectral Initialization
##   Calculating the gram matrix...
##   Using only 10000 most frequent terms during initialization...
##   Finding anchor words...
##      .........
##   Recovering initialization...
##      ..........................................................................................
## Initialization complete.
## ..........................................................................................
## Completed E-Step (2 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -8.692)
## ..........................................................................................
## Completed E-Step (1 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -8.164, relative change = 6.070e-02)
## ..........................................................................................
## Completed E-Step (1 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -8.122, relative change = 5.111e-03)
```

```
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 96 (approx. per word bound = -8.041, relative change = 1.085e-05)
## ..........................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 97 (approx. per word bound = -8.041, relative change = 1.110e-05)
## ..........................................................................................
## Completed E-Step (1 seconds).
## Completed M-Step.
## Completing Iteration 98 (approx. per word bound = -8.041, relative change = 1.113e-05)
## ..........................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 99 (approx. per word bound = -8.041, relative change = 1.038e-05)
## ..........................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Model Converged
```

```
class(ted.out) #STM
```

```
## [1] "STM"
```

4. Label the topics with labelTopics.

```
labelTopics(ted.out, 1:9)
```

```
## Topic 1 Top Words:
##       Highest Prob: music, sound, play, hear, song, laughter, listen
##       FREX: orchestra, music, song, heh, sing, piano, hum
##       Lift: endsapplausethank, musicapplaus, banjolittl, chinesemusicchineseoutsid, darlin, girlappla
##       Score: music, song, dcima, derek, orchestra, melodi, soundscap
## Topic 2 Top Words:
##       Highest Prob: brain, robot, comput, time, technolog, human, machin
##       FREX: electrod, robot, neuron, protocel, devic, neocortex, circuit
##       Lift: hammett, sexton, polyurethan, avaz, pavna, biometr, humanlevel
##       Score: robot, neuron, brain, protocel, sensor, selfassembl, connectom
## Topic 3 Top Words:
##       Highest Prob: peopl, dont, human, time, univers, world, question
##       FREX: higg, chimp, galaxi, particl, theori, lhc, bee
##       Lift: yesno, snobberi, defriend, siena, fishbowl, blackawton, cheval
##       Score: particl, higg, telescop, galaxi, lhc, quantum, pollen
## Topic 4 Top Words:
##       Highest Prob: peopl, time, women, life, love, feel, stori
##       FREX: gay, gender, ritual, women, shame, feminist, compass
##       Lift: abouttransl, aicha, elwafi, familieswhen, moussaoui, phylli, rodriguez
##       Score: women, gay, vagina, feminist, rape, doaa, refuge
## Topic 5 Top Words:
##       Highest Prob: peopl, world, countri, govern, percent, time, chang
##       FREX: encrypt, democraci, economi, elect, govern, incom, sector
##       Lift: afghanistanth, custer, darkth, daw, dreamwith, enda, graffiticov
##       Score: democraci, economi, countri, encrypt, polit, refuge, bitcoin
## Topic 6 Top Words:
##       Highest Prob: peopl, cancer, diseas, cell, patient, time, health
##       FREX: antibiot, slime, cancer, insulin, bioluminesc, virus, diseas
```

```
##       Lift: anecdotei, auenbrugg, auscult, axilla, axillari, breastsbut, burntisland
##       Score: cancer, tumor, diseas, antibiot, bacteria, gene, hiv
## Topic 7 Top Words:
##       Highest Prob: water, time, citi, build, world, peopl, planet
##       FREX: spacecraft, ocean, seawat, solar, atmospher, forest, carbon
##       Lift: midocean, fourdegre, beck, medina, handclean, salicornia, sitopia
##       Score: coral, citi, carbon, climat, reef, alga, dioxid
## Topic 8 Top Words:
##       Highest Prob: peopl, time, design, start, dont, your, idea
##       FREX: password, gamer, web, onlin, paint, text, blog
##       Lift: chatrou, coolso, html, javascript, parisian, firefox, firstmov
##       Score: password, design, twitter, data, blog, teszler, moma
## Topic 9 Top Words:
##       Highest Prob: peopl, school, kid, dont, children, time, start
##       FREX: classroom, school, teacher, educ, kid, grade, prison
##       Lift: mustachethem, sista, unhygien, greataunt, aakash, edx, selfpac
##       Score: teacher, school, prosecutor, women, movemb, educ, traffick
```

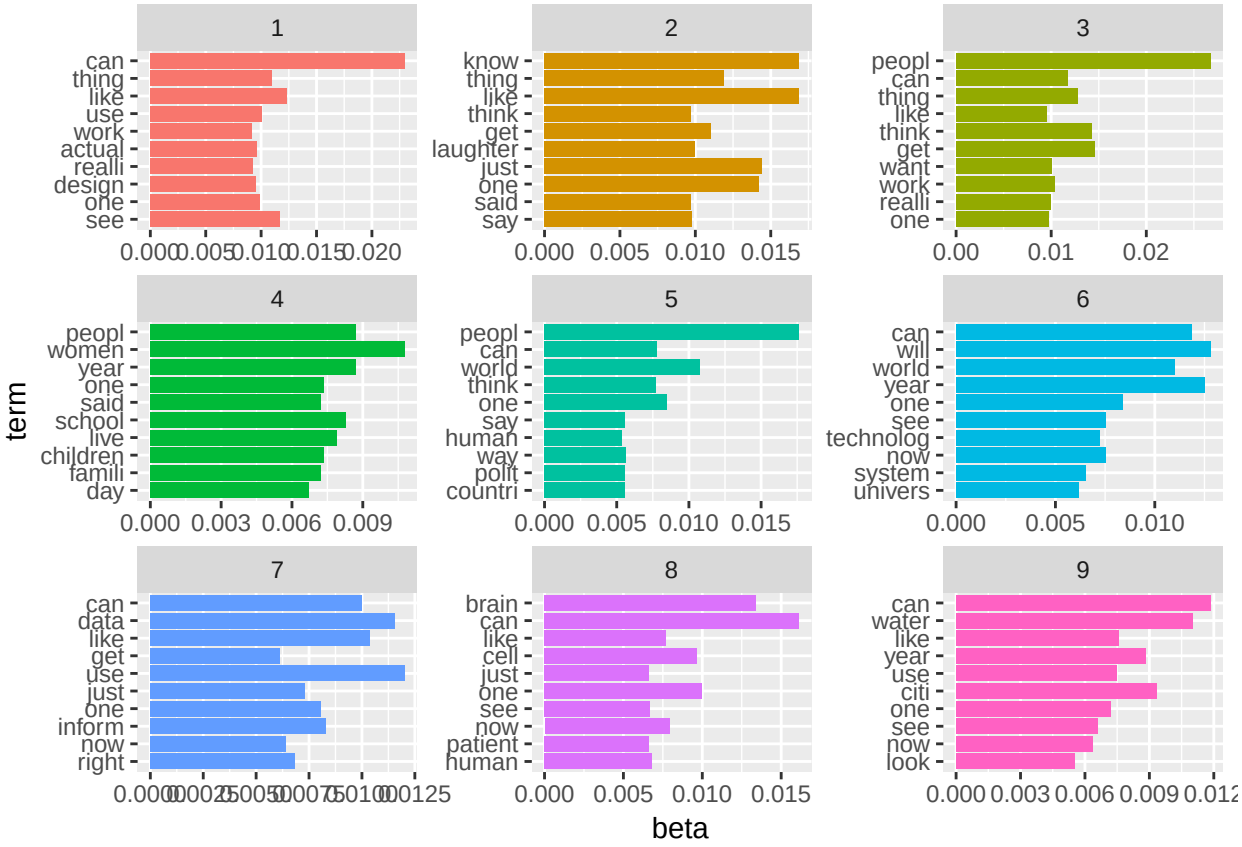
5. Using the originally cleaned and pre-processed data, fit a standard "vanilla" LDA model with 9 topics.

```
ted <- ted_talks %>%
  mutate(doc_id = seq.int(nrow(ted_talks)), text = transcript) %>%
  select(doc_id, text, -transcript, everything())

ted <- VCorpus(DataframeSource(ted), readerControl=list(language="eng"))

ted <- ted %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(stemDocument) %>%
  tm_map(stripWhitespace) %>%
  tm_map(PlainTextDocument) %>%
  DocumentTermMatrix() %>%
  removeSparseTerms(sparse = 0.99)

##


#Find the sum of words by row
rowTotals <- apply(ted, 1, sum)

#Remove all docs without words
ted   <- ted[rowTotals> 0, ]

#Run LDA
mod.out.9 <- LDA(ted, k=9, control = list(seed=6))

#Plot
class(mod.out.9) <- "LDA"

tidy(mod.out.9)
```

```
## # A tibble: 43,218 x 3
```

```
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 abandon 0.00000270
##  2     2 abandon 0.00000855
##  3     3 abandon 0.0000192
##  4     4 abandon 0.000299
##  5     5 abandon 0.0000257
##  6     6 abandon 0.0000583
##  7     7 abandon 0.0000125
##  8     8 abandon 0.0000170
##  9     9 abandon 0.000142
## 10     1 abil    0.000683
## # ... with 43,208 more rows
```

```
top.terms <- tidy(mod.out.9) %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top.terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```




6. Compare your results. How do the topics you find when conditioning on venue differ from those you

found using standard LDA?

On average, STM produces a much more informative topic labels regarding what each topic is about. For instance, in topic 1, we observe terms like “music, sound, play, hear, song, laughter, listen” which provide sufficient information for us to be confident that the topic is something related to musical performance. On the other hand, stnadard LDA produces topic labels that are quite generic (ie. “can”, “peopl”, “like”) which do not help us much to grasp wht the differences between the topics are.