SMU
SINGAPORE MANAGEMENT UNIVERSITY

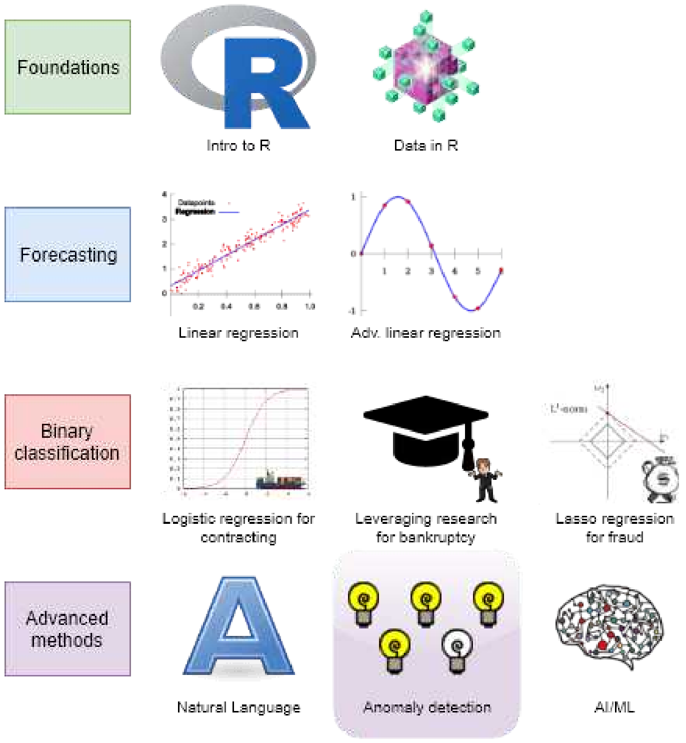
Forecasting and Forensic Analytics

Session 9: Topic Modeling and Anomaly Detection

Dr. Wang Jiwei

Preface

Learning objectives



- **Theory:**
  - NLP
  - Anomaly detection
- **Application:**
  - Understand the distribution of readability
  - Examine the *content* of annual reports (*topic modeling*)
  - Group firms on content
  - Fill in missing data
- **Methodology:**
  - ML/AI (LDA)
  - ML/AI (k-means, t-SNE)
  - More ML/AI (KNN)

Sets of documents (corpus)

Importing corpus

- package:readtext
  - Importing 6,000 U.S. 2014 annual reports
  - readtext() function
  - for .txt files: by default creating a dataframe with two columns: "doc_id" (file name) and "text" (text contained in the .txt file)
- package:quanteda's corpus() function
  - creating a summarized corpus as shown next page
- Other options include using
  - package:tm and VCorpus()
  - package:textreadr and read_dir()

```r
library(readtext)
library(quanteda)
# Needs ~1.5GB to read 6,000 files in one year
# Can read .txt, .json, .pdf, .csv, .doc, .xml, .zip, etc
# The following code reads all files with .txt extension in the folder
# Convert into a corpus using corpus() from package:quanteda
corp <- corpus(readtext("*.txt"))
```

Corpus summary

```
summary(corp)
# "Text": the file name as an identifier
# "Types": the number of distinct tokens/words
# "Tokens": the number of words/tokens
# "Sentences": the number of sentences in the file
```

```
##                         Text Types Tokens Sentences
## 1  0000002178-14-000010.txt  2929  22450       798
## 2  0000003499-14-000005.txt  2710  23907       769
## 3  0000003570-14-000031.txt  3866  55142      1541
## 4  0000004187-14-000020.txt  2902  26959       934
## 5  0000004457-14-000036.txt  3050  23941       883
## 6  0000004904-14-000019.txt  3408  30358      1119
## 7  0000004904-14-000029.txt   370   1308        40
## 8  0000004904-14-000031.txt   362   1302        45
## 9  0000004904-14-000034.txt   358   1201        42
## 10 0000004904-14-000037.txt   367   1269        45
```

Readability across the corpus

```r
# Uses ~20GB of RAM...  Break corp into chunks if RAM constrained
# textstat_readability() has 47 readability measures
# https://rdrr.io/cran/quanteda/man/textstat_readability.html
corp_FOG <- textstat_readability(corp, "FOG")
corp_FOG %>%  head() %>%  html_df()
```

document	FOG
0000002178-14-000010.txt	21.03917
0000003499-14-000005.txt	20.36549
0000003570-14-000031.txt	22.24386
0000004187-14-000020.txt	18.75720
0000004457-14-000036.txt	19.22683
0000004904-14-000019.txt	20.51594

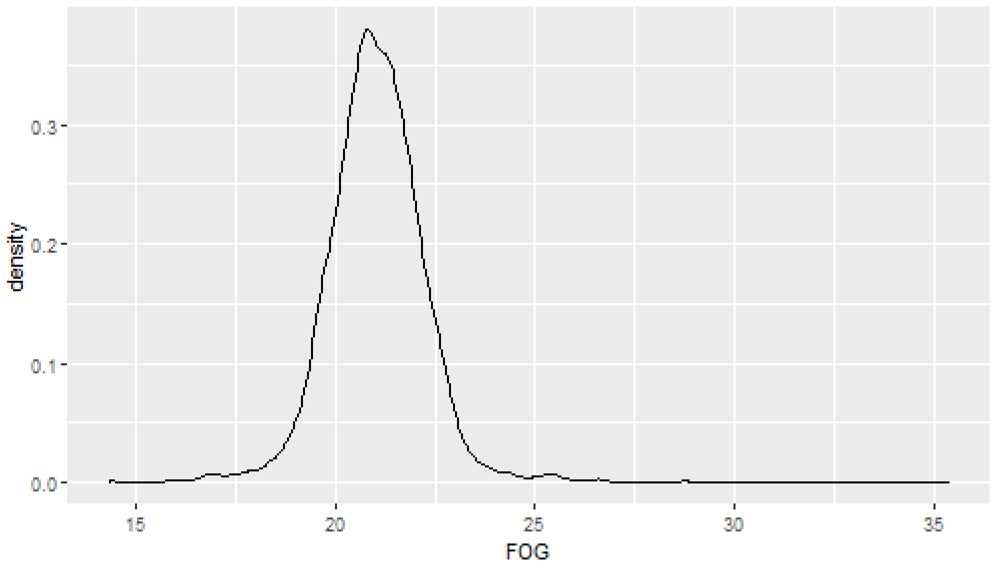
> Recall that lower FOG index is more readable, and DBS's annual report had a Fog index of 21.32

Readability across documents

```
summary(corp_FOG$FOG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.33   20.32   21.01   21.05   21.75   35.37
```

```
ggplot(corp_FOG, aes(x=FOG)) + geom_density()
```



Are certain industries' filings more readable?

- We'll use Standard Industrial Classification (SIC)
- SIC classification has TEN broad industries (division): Agriculture, Mining, Construction, Manufacturing, Utilities, Wholesale Trade, Retail Trade, Finance, Services, Public Admin
- SIC codes are 4 digits
  - The first two digits represent the major group (more like a sector)
  - The third digit represents the industry group
  - The fourth digit represents the specialization
- Example: Citigroup is SIC 6021
  - 60: Depository institution
  - 602: Commercial bank
  - 6021: National commercial bank

Are certain industries' filings more readable?

- Merge in SIC code by industry division

```r
# the csv file contains filing info to SEC Edgar, from WRDS
# accession: file name used for SEC filing; regsic: SIC industry code
# case_when(): multiple if_else, same as SQL CASE WHEN
df_SIC <- read.csv('../../Data/Session_9-Filings2014.csv') %>%
  select(accession, regsic) %>%
  mutate(accession = paste0(accession, ".txt")) %>%
  rename(document = accession) %>%
  mutate(industry = case_when(
    regsic >=0100 & regsic <= 0999 ~ "Agriculture",
    regsic >=1000 & regsic <= 1499 ~ "Mining",
    regsic >=1500 & regsic <= 1799 ~ "Construction",
    regsic >=2000 & regsic <= 3999 ~ "Manufacturing",
    regsic >=4000 & regsic <= 4999 ~ "Utilities",
    regsic >=5000 & regsic <= 5199 ~ "Wholesale Trade",
    regsic >=5200 & regsic <= 5999 ~ "Retail Trade",
    regsic >=6000 & regsic <= 6799 ~ "Finance",
    regsic >=7000 & regsic <= 8999 ~ "Services",
    regsic >=9100 & regsic <= 9999 ~ "Public Admin")) %>%
  group_by(document) %>% slice(1) %>% ungroup()
corp_FOG <- corp_FOG %>% left_join(df_SIC)
```

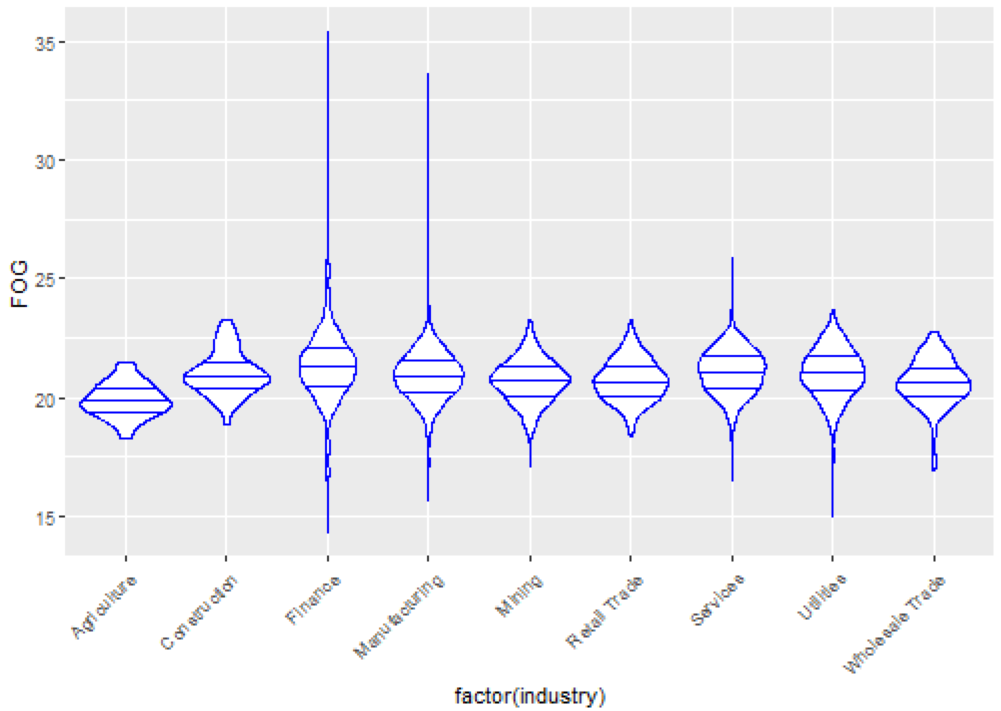
A sample of the data

```
corp_FOG %>%  head() %>%  html_df()
```

document	FOG	regsic	industry
0000002178-14-000010.txt	21.03917	5172	Wholesale Trade
0000003499-14-000005.txt	20.36549	6798	Finance
0000003570-14-000031.txt	22.24386	4924	Utilities
0000004187-14-000020.txt	18.75720	4950	Utilities
0000004457-14-000036.txt	19.22683	7510	Services
0000004904-14-000019.txt	20.51594	4911	Utilities

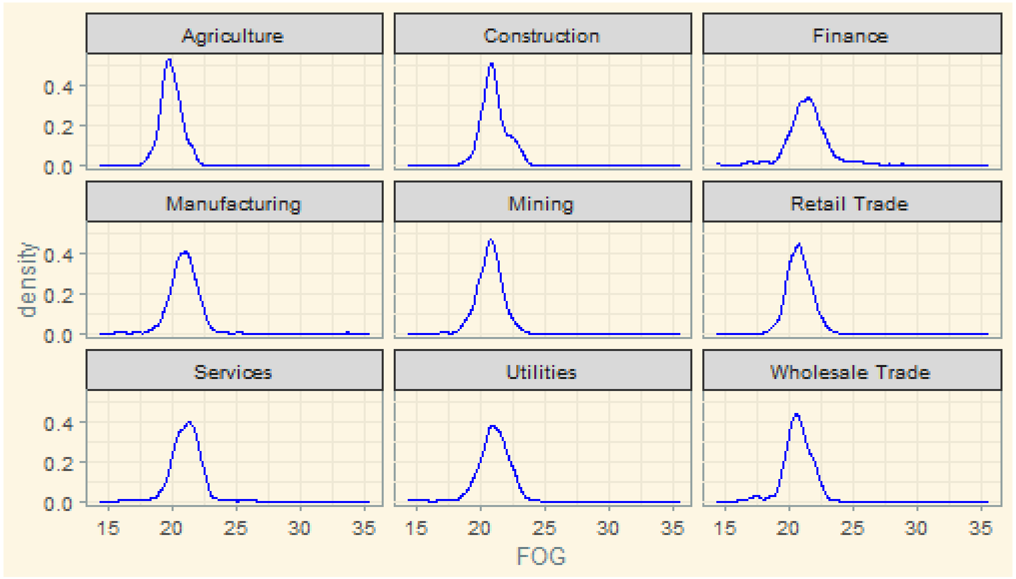
Are certain industries' filings more readable?

```
ggplot(corp_FOG[!is.na(corp_FOG$industry), ], aes(x = factor(industry),
                                                   y = FOG)) +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75), color = "blue") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



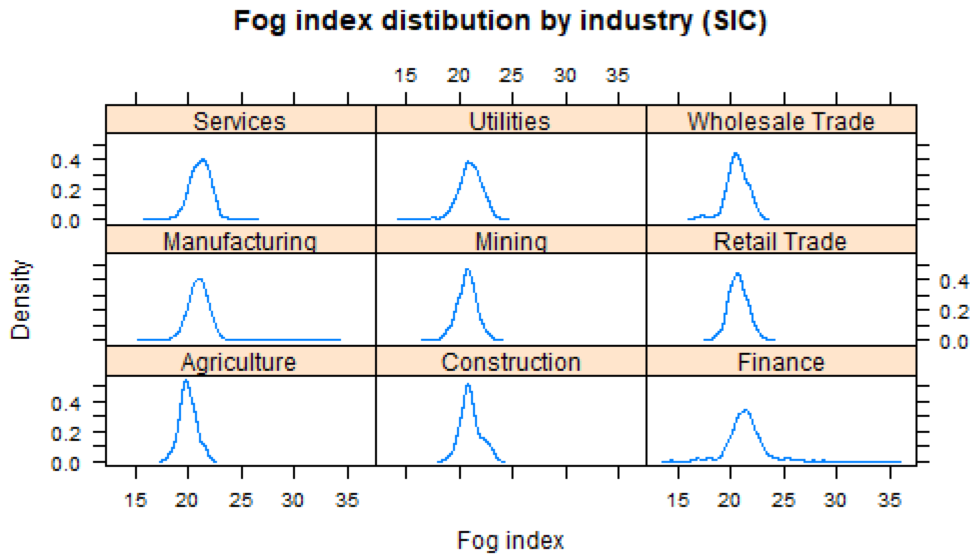
Are certain industries' filings more readable?

```r
library(ggthemes)
ggplot(corp_FOG[!is.na(corp_FOG$industry), ], aes(x = FOG)) +
  geom_density(color = "blue") + facet_wrap(~industry) + theme_solarized()
```



Are certain industries' filings more readable?

```r
library(lattice)
densityplot(~FOG | industry,
            data = corp_FOG,
            plot.points = F,
            main = "Fog index distibution by industry (SIC)",
            xlab = "Fog index",
            layout = c(3, 3))
```



Finding references across text

```r
# kwic() is a function in quanteda to locate a phrase
kwic(corp, phrase("global warming")) %>%
  mutate(text = paste(pre, keyword, post)) %>%
  select(docname, text) %>% datatable(options = list(pageLength = 3),
                                      rownames = F)
```

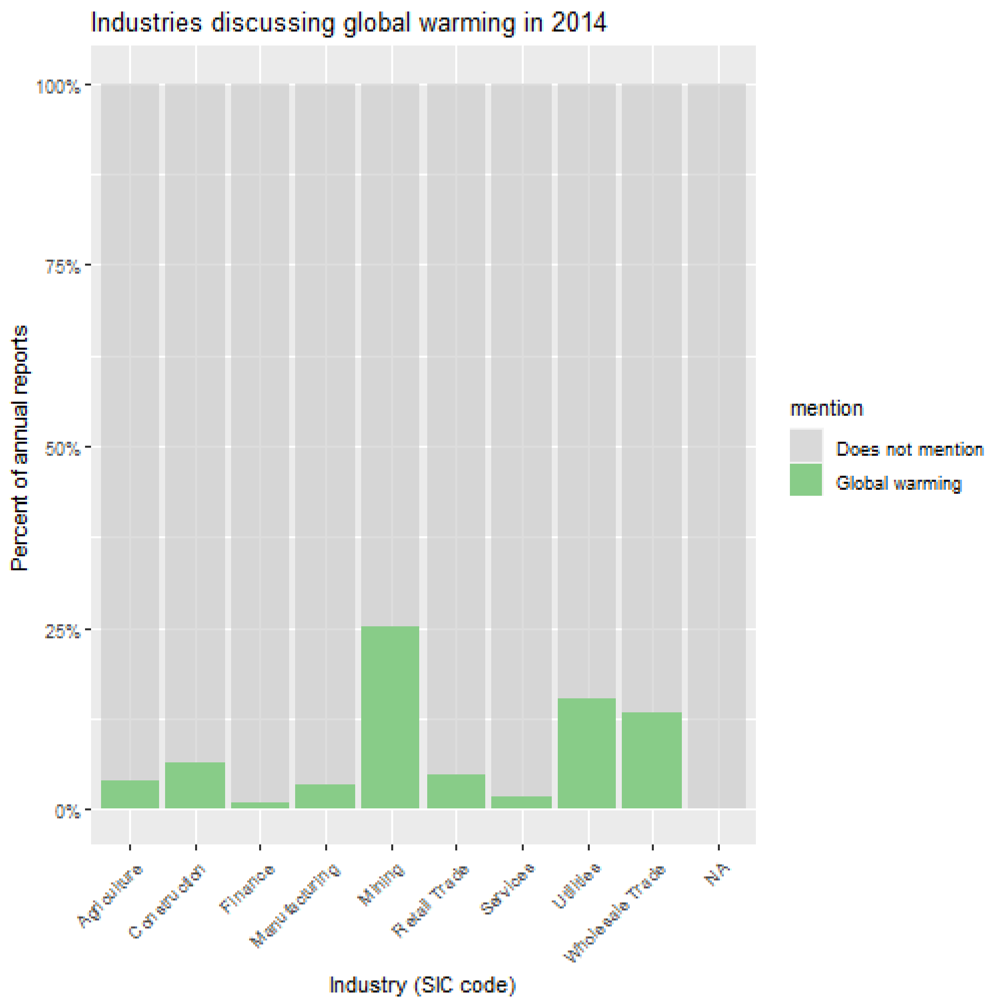
Show 3 entries Search:

docname	text
0000003499-14-000005.txt	. Potentially adverse consequences of global warming could similarly have an impact
0000004904-14-000019.txt	nuisance due to impacts of global warming and climate change . The
0000008947-14-000068.txt	timing or impact from potential global warming and other natural disasters ,

Showing 1 to 3 of 310 entries

Previous 1 2 3 4 5 … 104 Next

Mentions by industry



Topic modeling

Going beyond simple text measures

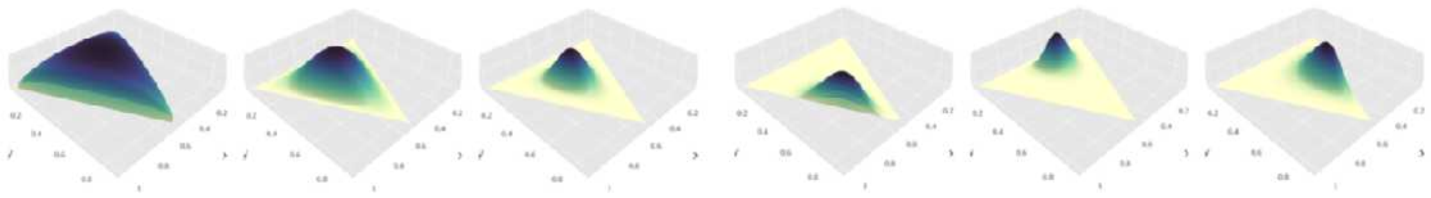
- This is a hard question to answer -- our sample has 6000 annual reports and 104,690,796 words in it!
  - 69.8 hours for the "worlds fastest reader", per this source
  - 103.86 days for a standard speed reader (700wpm)
  - 290.8 days for an average reader (250wpm)
- We could read a small sample of them
- Complemented by computer to read all of them!

> Recall the topic variable to detect fraud

- Topic was a set of variables indicating *how much* a given topic was discussed
- This measure was created by making a machine read every annual report
  - The computer then used a technique called LDA to process these reports' content into topics

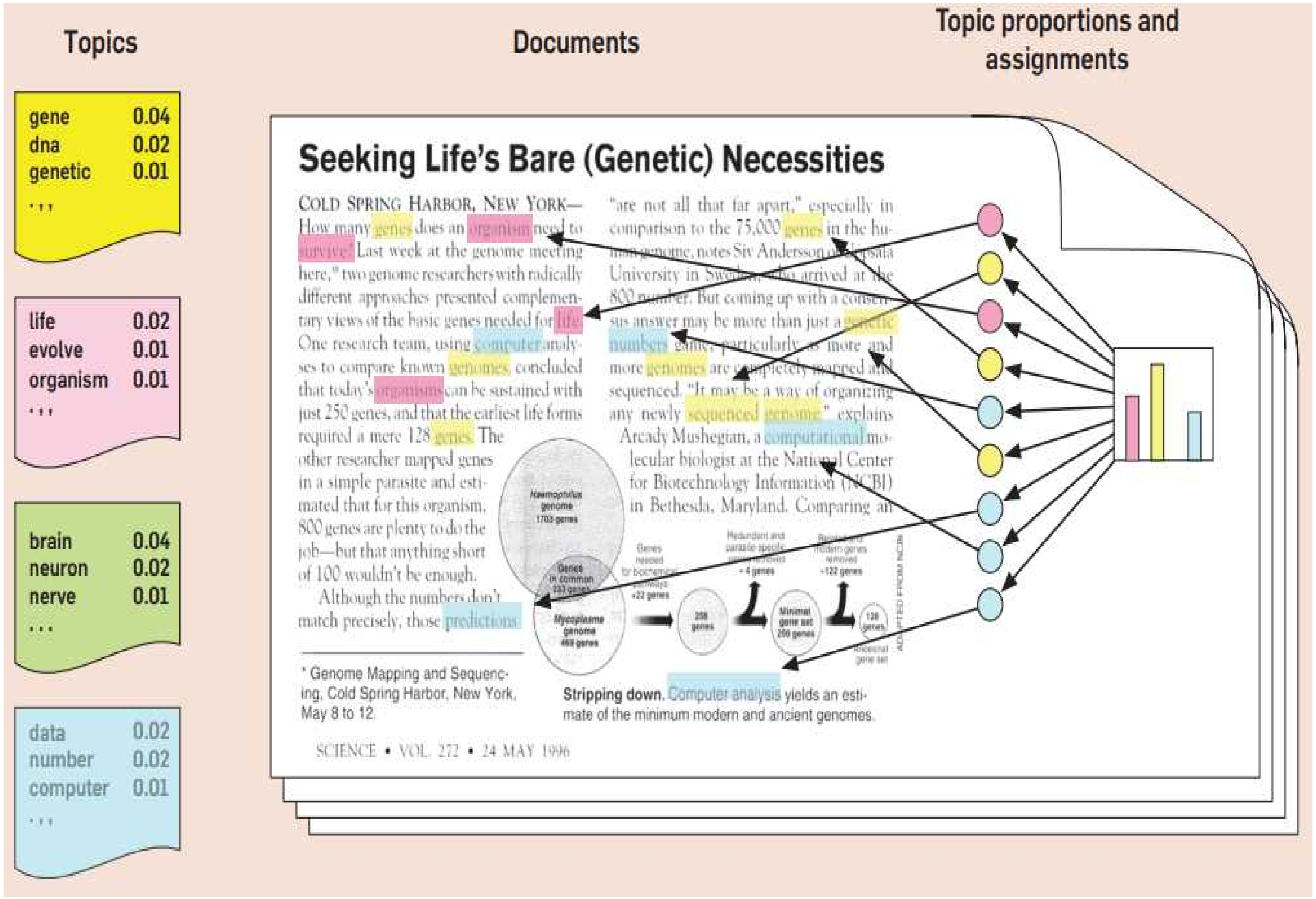
What is LDA?

- *L*atent *D*irichlet *A*llocation
- One of the most popular methods under the field of *topic modeling*
- LDA is a Bayesian method of assessing the content of a document
- LDA assumes there are a set of topics in each document, and that this set follows a *Dirichlet* prior distribution for each document
  - Words within topics also have a *Dirichlet* prior distribution
  - *Dirichlet* distribution is the most popular prior distribution in Bayesian statistics



More details from the creator

How does it work



How does it work?

1. Assume a set of topics following Dirichlet distribution and each topic has a set of words (prior distribution)
   - But these topics are not observed, we only observe words and documents, thus topics are *latent*.
   - To use the observed documents/words to infer the hidden topic structure
2. Read all the documents, count each word within the document, and allocate to topics according to the likelyhood of distribution (posterior distribution).
   - By using a Gibbs sampler to simulate the underlying distributions, which is an Markov Chain Monte Carlo (MCMC) method

> It's quite complicated in the background

Implementations in R

- There are four main implementations of LDA in R (maybe more)
  1. `package:lda`: A somewhat rigid package with difficult setup syntax, but it plays nicely with the great `package:LDAvis` package for visualizing models. Supported by `package:quanteda`.
  2. `package:topicmodels`: An extensible topic modeling framework that plays nicely with `package:quanteda`
  3. `package:stm`: A bit of a tweak on the usual LDA model that plays nicely with `package:quanteda`
  4. `package:mallet`: An R package to interface with the venerable MALLET Java package, capable of more advanced topic modeling

Term document matrices (TDM)

- Before we begin, we'll need a matrix of word counts per document
- Term document matrix (TDM) describes the frequency of terms that occur in a collection of documents
- We'll create something called a *sparse matrix* for this
- A sparse matrix is a matrix in which many of the elements are zero
- The transpose of TDM (ie, DTM) is equivalent, and we don't differentiate them
- "term" may also refer to "feature", such as "document feature matrix"

> Think about the structure of a matrix where rows are document names and columns are individual words (ie, each cell will be the frequency of each word to appear in each document). How many elements in this matrix will be 0s?

Making a TDM

- In `package:quanteda`, use `dfm()` function
  - `stem = TRUE`, Code similar words as the same
    - Ex.: *code*, *cod*ing, and *cod*er would all become *cod*
      - Helps with the curse of dimensionality
  - `remove = c(...)`, You can supply a list of stop words to remove
    - You can use `remove = stopwords()` for a simple list
    - The `stopwords()` from the `package:stopwords` package, supports over 50 languages
    - We can use SMART like last week: `remove = stopwords(source = 'smart')`
    - For other languages, use `remove=stopwords("zh", source = "stopwords-iso")`

Making a TDM

```r
# adding industry to the corpus
docs <- docnames(corp) # get document names of a corpus
# convert to a dataframe
docs <- data.frame(document = docs)
docs <- docs %>% left_join(df_SIC)
# set variables for the documents in a corpus
docvars(corp, field = "industry") <- docs$industry
```

```r
tdm <- dfm(corp) # Simplest way
# With stopwords, stemming and others -> Used in next slides
tdm <- dfm(corp,
           stem = TRUE,
           remove = stopwords(source = 'smart'),
           remove_punct = TRUE,
           remove_numbers = TRUE) %>%
  dfm_trim(min_termfreq = 10, termfreq_type = "count")
```

- refer here for the `dfm_trim()` function

What words matter by industry?

```r
topfeatures(tdm, n = 5, groups = "industry")
```

```
## $`Wholesale Trade`
## compani    oper million financi product
##   30371   20340   18085   17552   17300
##
## $Finance
## compani    loan financi  decemb million
##  438185  392164  299978  286791  274376
##
## $Utilities
##    oper million compani financi  includ
##  112038  107322  101971   79010   76604
##
## $Services
## compani    oper million financi  servic
##  222276  145506  138397  131881  120817
##
## $Manufacturing
## compani product million    oper financi
##  434805  368900  275829  240181  231687
##
## $Mining
## compani    oper     gas     oil  decemb
##   97798   92076   74150   65532   60475
##
## $Construction
## compani million    oper financi  decemb
##   15479   14885   12431   10899   10149
```

TF-IDF

- Words counts are not very informative
- Knowing the words that show up frequently in one group *but not in the others* would be much more useful
- This is called TF-IDF
  - *T*erm *F*requency-*I*nverse *D*ocument *F*requency
- Think of it roughly as:

$$\frac{\text{How many times a word is in the document}}{\text{How many documents the word is in}}$$

- We can easily calculate tf-idf using `dfm_tfidf()` from `package:quanteda`

The actual TF-IDF equation

$$\frac{f_{w,d}}{f_d} * [-\log_2\left(\frac{n_w}{N}\right)]$$

- w represents 1 word
- d represents 1 document
- $f_{w,d}$ is the number of times w appears in d
- f_d is the number of times any word appears in d
- $\frac{f_{w,d}}{f_d}$ is the term frequency
- n_w is the number of documents with w at least once
- N is the number of documents
- $\frac{n_w}{N}$ is document frequency in [0, 1], its log transformation will be non-positive
- Hence, (higher TF + lower DF) ===>>> higher TF-IDF, it can be interpretated as a weighted TF

What words matter by industry?

```r
# scheme_tf is the type of calculation
# the default is 'count', ie, the number of times
# 'prop' is the proportion
# base= is the base for log transformation, default 10
# https://quanteda.io/reference/dfm_weight.html
# note that tf-idf is not in %, but we can compare the magnitude
tfidf_mat <- dfm_tfidf(tdm, base = 2, scheme_tf = "prop")
topfeatures(tfidf_mat, n = 5, groups = "industry")
```

```
## $`Wholesale Trade`
##   graybar  grainger       oil   million     bottl
## 0.3140485 0.2899255 0.2187512 0.2184815 0.2122642
##
## $Finance
##        ab   mortgag depositor      loan      reit
##  9.863862  7.414096  6.192815  5.109854  5.046502
##
## $Utilities
##      gas      fcc  pipelin   energi aircraft
## 2.005220 1.484092 1.227766 1.164767 1.020255
##
## $Services
##     game   client   casino  million  softwar
## 2.394468 1.760647 1.635549 1.496073 1.404740
##
## $Manufacturing
##   clinic      fda    trial     drug  patient
## 7.057913 5.487707 3.949705 3.935010 3.799611
##
```

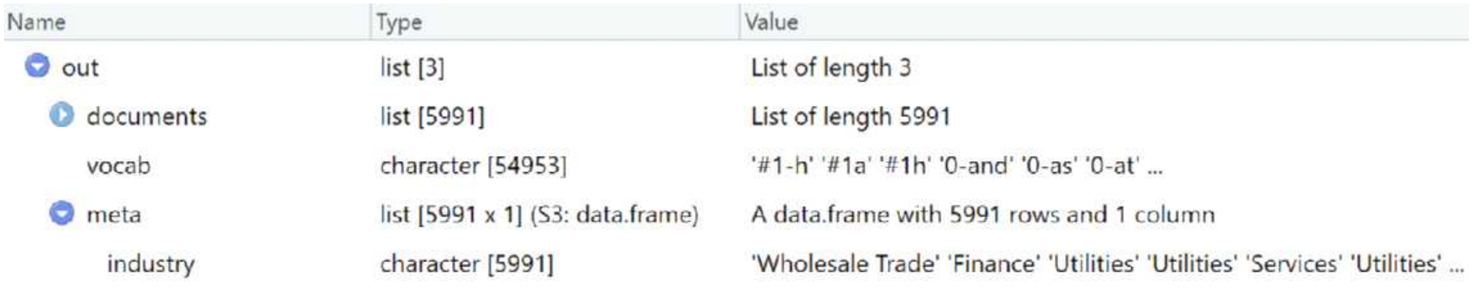
What words matter for banks?

```r
topfeatures(tfidf_mat, n = 20, groups = "industry")$Finance
```

```
##          ab     mortgag   depositor        loan        reit       trust 
##    9.863862    7.414096    6.192815    5.109854    5.046502    4.394811 
##     reinsur     truste       estat      tenant    instruct partnership 
##    3.809024    3.607591    3.188824    3.100092    2.970419    2.697215 
##        real     million        pool        fdic   residenti     bancorp 
##    2.506670    2.482285    2.287610    2.238533    2.149133    2.074819 
##     obligor        rmbs 
##    2.055811    2.055453
```

Topic modeling with STM

```r
# quanteda's conversion for input files to the stm/lda/topicmodels packages
out <- convert(tdm, to = 'stm')
# Change to 'lda' or 'topicmodels' for other packages
# Each package has its own input format
```



Name	Type	Value
out	list [3]	List of length 3
documents	list [5991]	List of length 5991
vocab	character [54953]	'#1-h' '#1a' '#1h' '0-and' '0-as' '0-at' ...
meta	list [5991 x 1] (S3: data.frame)	A data.frame with 5991 rows and 1 column
industry	character [5991]	'Wholesale Trade' 'Finance' 'Utilities' 'Utilities' 'Services' 'Utilities' ...

Topic modeling with STM

- The `convert()` function creates a list of 3 items:
  - `out$documents`: a list of index number of each word and its count for each document
  - `out$vocab`: a vector of words and their index numbers
  - `out$meta` a data frame of the other information from the corpus (like `industry`)

```
out$documents[[1]][ , 126:130]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 5170 5338 5494 5949 6044
## [2,]    7    1    1    1    9
```

```
out$vocab[c(out$documents[[1]][ , 126:130][1, ])]
```

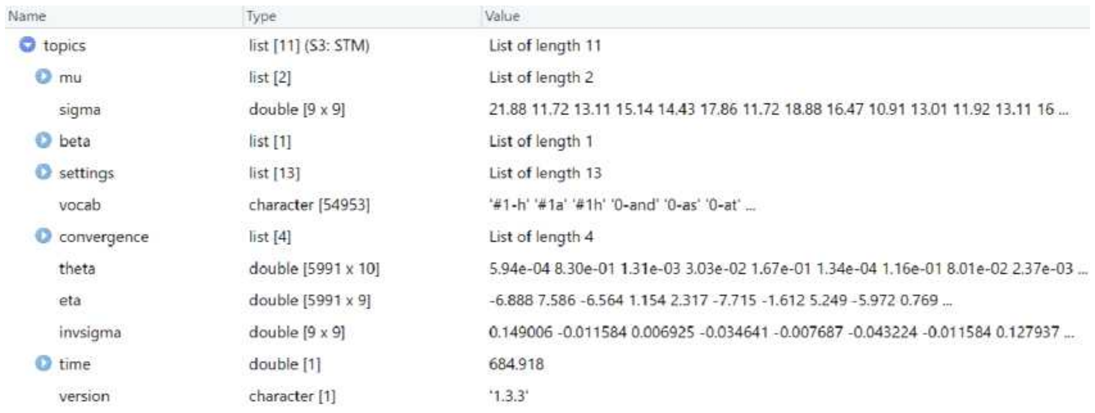
```
## [1] "benefit" "better"  "bill"    "blowout" "board"
```

Running the model

- We will use the `stm()` function from the `package:stm` package
  - It has a lot of options that you can explore to tweak the model
  - The most important is `K`, the number of topics we want. I'll use 10 for simplicity, but often we need more to neatly categorize the text
    - `K=100` is a popular choice when we are using the output as an input to another model

```r
library(stm)
topics <- stm(out$documents, out$vocab, K = 10)
```

> What this looks like while running



Name	Type	Value
topics	list [11] (S3: STM)	List of length 11
mu	list [2]	List of length 2
sigma	double [9 x 9]	21.88 11.72 13.11 15.14 14.43 17.86 11.72 18.88 16.47 10.91 13.01 11.92 13.11 16 ...
beta	list [1]	List of length 1
settings	list [13]	List of length 13
vocab	character [54953]	'#1-h' '#1a' '#1h' '0-and' '0-as' '0-at' ...
convergence	list [4]	List of length 4
theta	double [5991 x 10]	5.94e-04 8.30e-01 1.31e-03 3.03e-02 1.67e-01 1.34e-04 1.16e-01 8.01e-02 2.37e-03 ...
eta	double [5991 x 9]	-6.888 7.586 -6.564 1.154 2.317 -7.715 -1.612 5.249 -5.972 0.769 ...
invsigma	double [9 x 9]	0.149006 -0.011584 0.006925 -0.034641 -0.007687 -0.043224 -0.011584 0.127937 ...
time	double [1]	684.918
version	character [1]	'1.3.3'

LDA model by top words

- `Highest prob`: words with the highest probability of being chosen in the topic. Others are weighted measures (page 13).
- Another way is to extract example documents for each topic using `findThoughts()` function from the `package:stm`

```
labelTopics(topics, c(4, 6, 10))
```

```
## Topic 4 Top Words:
##       Highest Prob: invest, fund, manag, market, asset, trade, interest
##       FREX: uscf, nfa, unl, uga, mlai, bno, dno
##       Lift: a-1t, aion, apx-endex, bessey, bolduc, broyhil, buran
##       Score: uscf, fhlbank, rmbs, uga, invest, mlai, ung
## Topic 6 Top Words:
##       Highest Prob: loan, bank, compani, financi, decemb, million, interest
##       FREX: nonaccru, oreo, tdrs, bancorp, fdic, charge-off, alll
##       Lift: 100bp, 4-famili, acnb, acquired-impair, amerihom, ameriserv, annb
##       Score: fhlb, loan, bank, mortgag, risk-weight, tdrs, nonaccru
## Topic 10 Top Words:
##       Highest Prob: gas, oper, oil, natur, million, cost, decemb
##       FREX: ngl, ngls, oneok, mgp, permian, qep, wes
##       Lift: 12asset, 1businesscommerci, 94-mile, aivh, amargo, amopp, angell
##       Score: gas, drill, oil, ngl, crude, unithold, ngls
```

Applying topics to our data

```r
# Convert into a document-topic dataframe (5991*10 obs)
# The theta object is the posterior proportion of a topic
out$meta$industry <- factor(out$meta$industry)
out$meta$industry <- addNA(out$meta$industry) # turning NA into an extra level
doc_topics = data.frame(document = names(out$documents),
                        industry = out$meta$industry,
                        topic = 1,
                        weight = topics$theta[ , 1]) # topic proportion
for (i in 2:10) {
  temp = data.frame(document = names(out$documents),
                    industry = out$meta$industry,
                    topic = i,
                    weight = topics$theta[ , i])
  doc_topics = rbind(doc_topics, temp)}
# Proportional topics (%)
doc_topics <- doc_topics %>%
  group_by(document) %>%
  mutate(topic_prop = weight / sum(weight)) %>% ungroup()
```

```r
# Manually label topics
topic_labels = data.frame(topic = 1:10,
                          topic_name = c('Real Estate', 'Management', 'Product',
                                         'Investment', 'Services', 'Financing',
                                         'Service2', 'Insurance', 'Industrial',
                                         'Utility'))
doc_topics <- doc_topics %>% left_join(topic_labels)
```

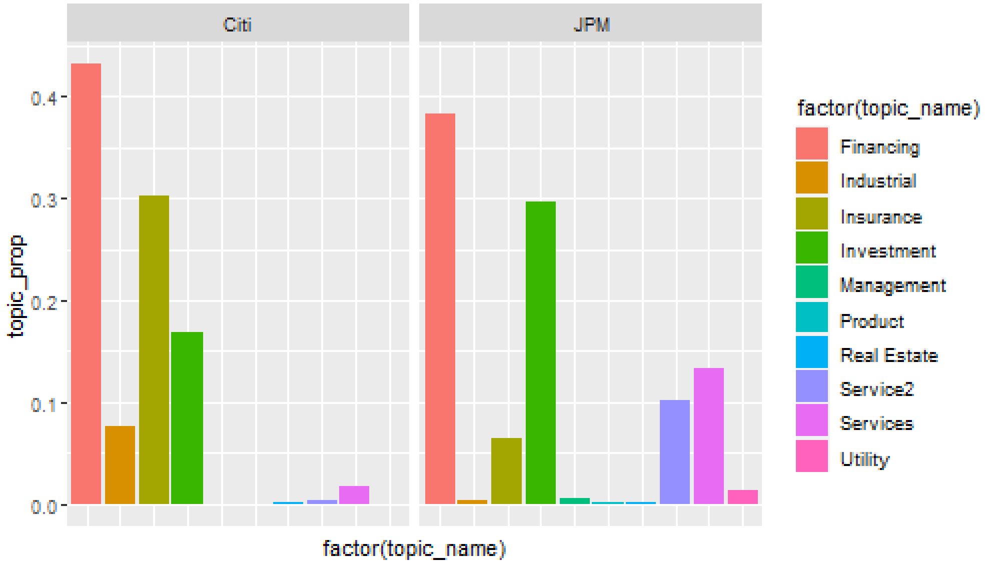
Topic content of the Citi 10-K

```
doc_topics %>% filter(document=='0001104659-14-015152.txt')
```

```
## # A tibble: 10 x 6
##    document                 industry topic    weight topic_prop topic_name
##    <chr>                    <fct>    <dbl>     <dbl>      <dbl> <chr>
##  1 0001104659-14-015152.txt Finance      1 0.000316   0.000316  Real Estate
##  2 0001104659-14-015152.txt Finance      2 0.0000594  0.0000594 Management
##  3 0001104659-14-015152.txt Finance      3 0.0000153  0.0000153 Product
##  4 0001104659-14-015152.txt Finance      4 0.168      0.168     Investment
##  5 0001104659-14-015152.txt Finance      5 0.0172     0.0172    Services
##  6 0001104659-14-015152.txt Finance      6 0.433      0.433     Financing
##  7 0001104659-14-015152.txt Finance      7 0.00332    0.00332   Service2
##  8 0001104659-14-015152.txt Finance      8 0.303      0.303     Insurance
##  9 0001104659-14-015152.txt Finance      9 0.0755     0.0755    Industrial
## 10 0001104659-14-015152.txt Finance     10 0.0000558  0.0000558 Utility
```

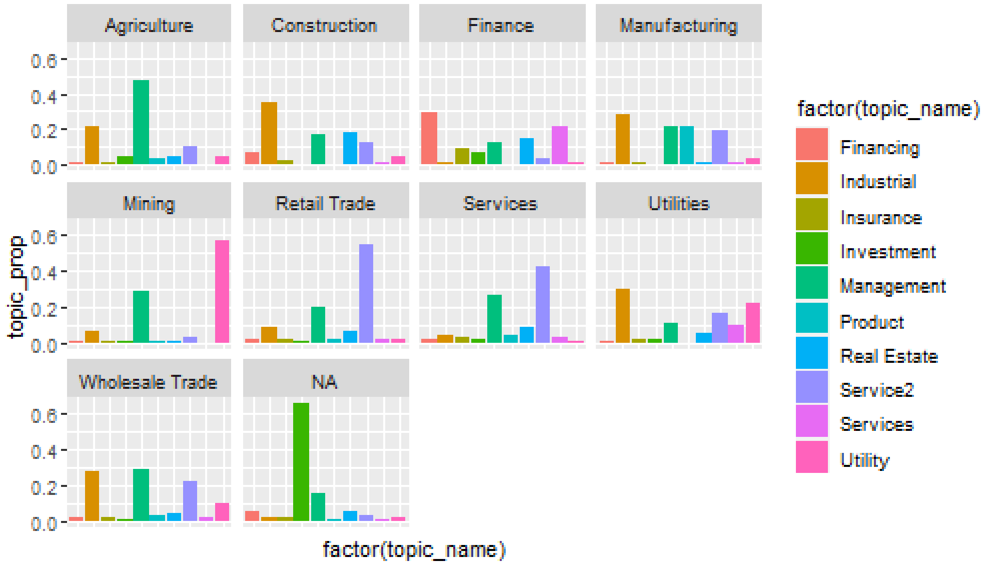
Topic content of Citi vs JPMorgan

```r
doc_topics %>% filter(document=='0001104659-14-015152.txt' |
                      document=='0000019617-14-000289.txt') %>%
  mutate(Company=ifelse(document=='0001104659-14-015152.txt', 'Citi','JPM')) %>%
  ggplot(aes(x=factor(topic_name), y = topic_prop, fill = factor(topic_name))) +
  geom_col() + facet_wrap(~Company) +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



Topic content by industry

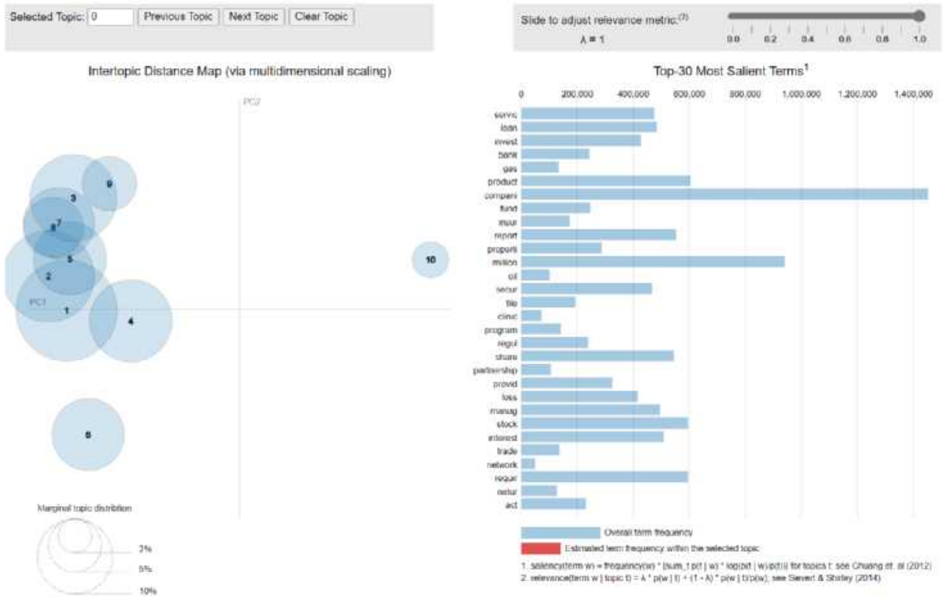
```r
doc_topics %>%  group_by(industry, topic) %>%
  mutate(topic_prop = mean(topic_prop)) %>% slice(1) %>%  ungroup() %>%
  ggplot(aes(x = factor(topic_name), y = topic_prop, fill=factor(topic_name))) +
  geom_col() + facet_wrap(~industry) +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



A nice visualization

- Using LDAvis via `package:STM`'s `toLDAvis()` function
  - Need `package:LDAvis` and `package:servr` installed to run

```r
# Code to generate LDAvis
# need to install LDAvis and servr packages
toLDAvis(topics, out$documents)
```



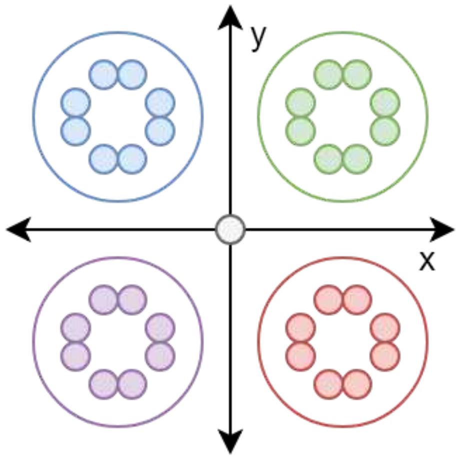
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Anomaly detection through clustering

Clustering

- One important aspect of detecting anomalies is determining groups in the data
  - We call this *clustering*
- If we find that a few elements of our data don't match the usual groups in the data, we can consider this to be an anomaly
  - Similar to the concept of outliers, but taking into account *multiple variables* simultaneously

- The grey dot is at the mean of both the x and y dimensions
  - it isn't an outlier
- But there are 4 clear clusters... and it doesn't belong to any!



One clustering approach: k-means

$$\min_{C_k} \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

- K-means clustering: unsupervised machine learning for unlabeled data (data without defined categories, ie, no response variable)
- The goal is to find groups in the data, with the number of groups represented by the variable k.
- You need to specify the number of groups you want
- How: minimizes the total sum of squared distance between points and their cluster mean (total within-cluster variation)

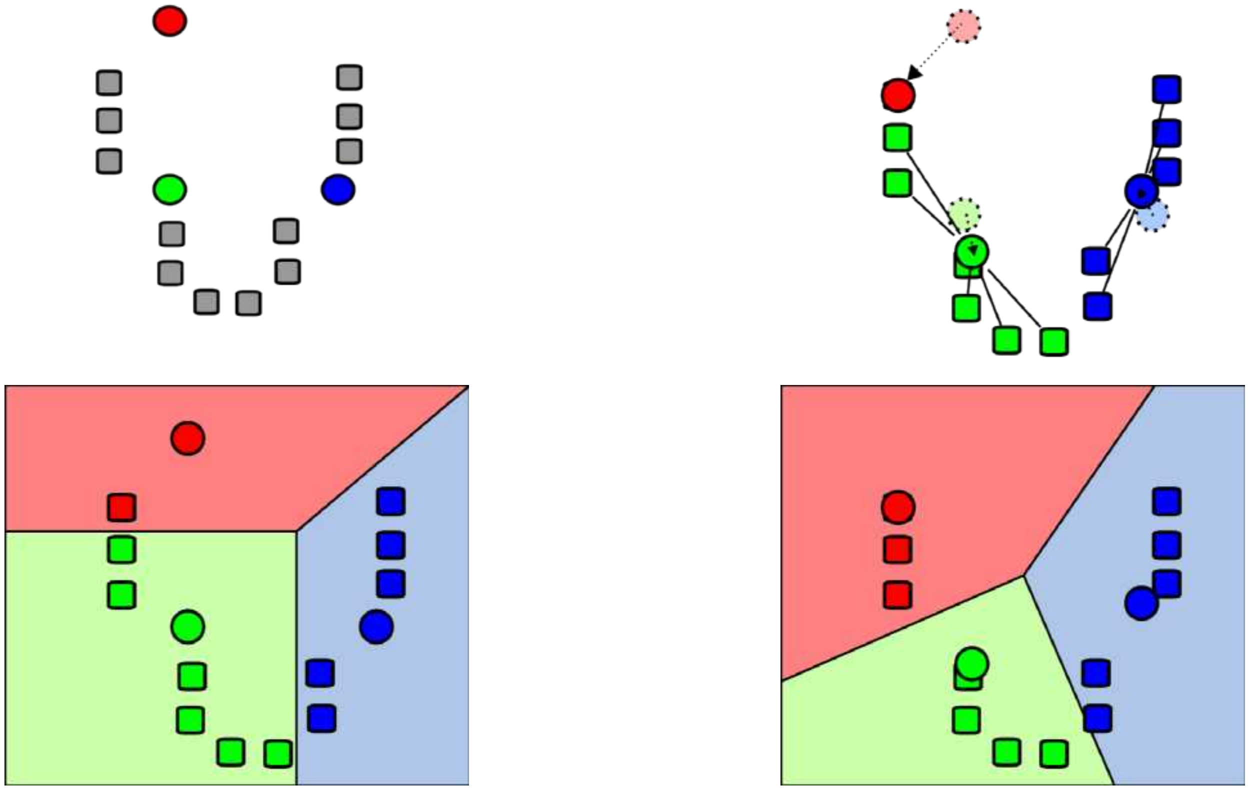
- Pros:
  - Very fast to run
  - Simple interpretation
- Cons
  - Simple algorithm
  - Need to specify k, the number of clusters

How to do?

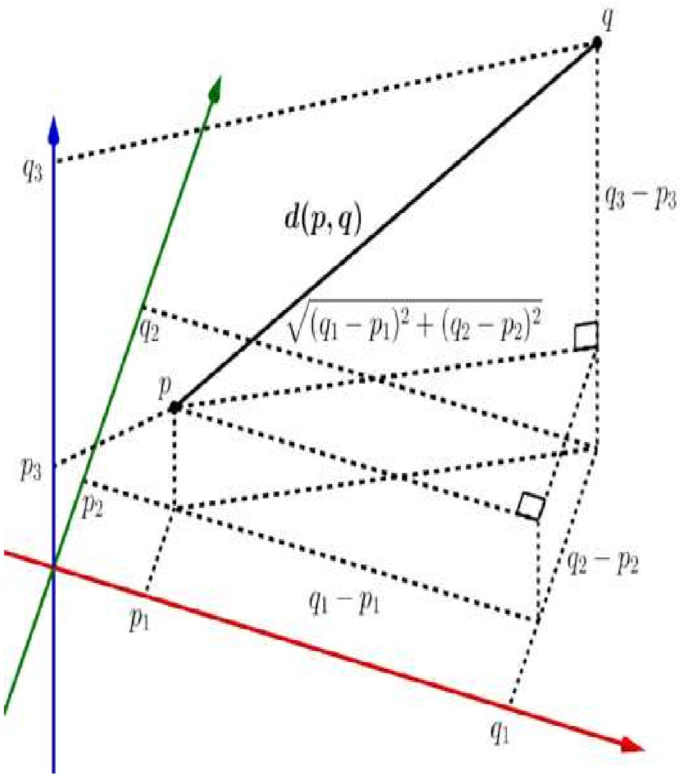
1. Specify the number of clusters (K) to be created (by the analyst)
2. Select randomly K objects from the data set as the initial cluster centers or means (the centroid)
3. Assign each observation to their closest centroid, based on the distance between the object and the centroid
4. For each of the K clusters update the cluster centroid by calculating the new mean values of all the data points in the cluster.
5. Iteratively minimize the total within sum of square (wss). That is, iterate steps 3 and 4 until the cluster assignments stop changing or the maximum number of iterations is reached. By default, R uses 10 as the default value for the maximum number of iterations.

> Read a tutorial here

A simple illustration of k-means



Eucliden distance measure



$$d_{euc}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

- Other measures such as
  - Manhattan
  - Pearson correlation
  - Spearman correlation
  - Kendall correlation

Example: Classifying companies based on disclosure

- While industry code is one classification of firms, it has a number of drawbacks:
    1. The classification system is old and perhaps misses new industries
    2. It relies on self-reporting
    3. Firms' classifications rarely change, even when firms themselves change

> We'll build a different classification system, based on what they discuss in their annual reports

Prepping data

- We will need data to be in a matrix format, with...
  - 1 row for each observation
  - 1 column for each variable we want to cluster by
- Recast data from long format to wide format using `package:tidyr`

```r
library(tidyr)
# refer previous slides for doc_topics data structure
wide_topics <- spread(doc_topics[, c(1,2,5,6)], topic_name, topic_prop)
mat <- wide_topics[, 3:12] # keep topic_name, drop document and industry
# we should scale/standardize the data if they are in different scale units
# our data is fine, so we don't need to scale
# you may use scale() and it does not change the subsequent results
mat[, 1:6] %>% head(n=3) %>% html_df()
```

Financing	Industrial	Insurance	Investment	Management	Product
0.0105862	0.1578543	0.1088631	0.0004632	0.1161191	0.0002101
0.0467173	0.0059438	0.0235389	0.0005284	0.0801189	0.0001432
0.0069105	0.0351987	0.0003661	0.0201215	0.0023672	0.0000186

Calculating k-means

```r
set.seed(678)
clusters <- kmeans(mat, 9) # k = 9
# kmeans() function returns a list of several components, including:
# cluster: integers vector (from 1:k) indicating the cluster each point is in
# centers: A matrix of cluster centers (cluster means)
# totss: The total sum of squares.
# withinss: Vector of within-cluster sum of squares, one component per cluster.
# tot.withinss: Total within-cluster sum of squares, i.e. sum(withinss).
# betweenss: The between-cluster sum of squares, i.e. $totss-tot.withinss$.
# size: The number of points in each cluster.
str(clusters[1:7]) # structure of the first seven components
```

```
## List of 7
##  $ cluster     : int [1:5991] 6 4 9 1 1 1 5 5 5 5 ...
##  $ centers     : num [1:9, 1:10] 0.016 0.082 0.8996 0.0689 0.0174 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:9] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:10] "Financing" "Industrial" "Insurance" "Investment" ...
##  $ totss       : num 2947
##  $ withinss    : num [1:9] 50.6 33.7 14.5 37.3 495 ...
##  $ tot.withinss: num 835
##  $ betweenss   : num 2111
##  $ size        : int [1:9] 792 280 536 470 1053 239 1060 1249 312
```

Recap: within and between cluster variation

$$SS_{totss} = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu)^2$$

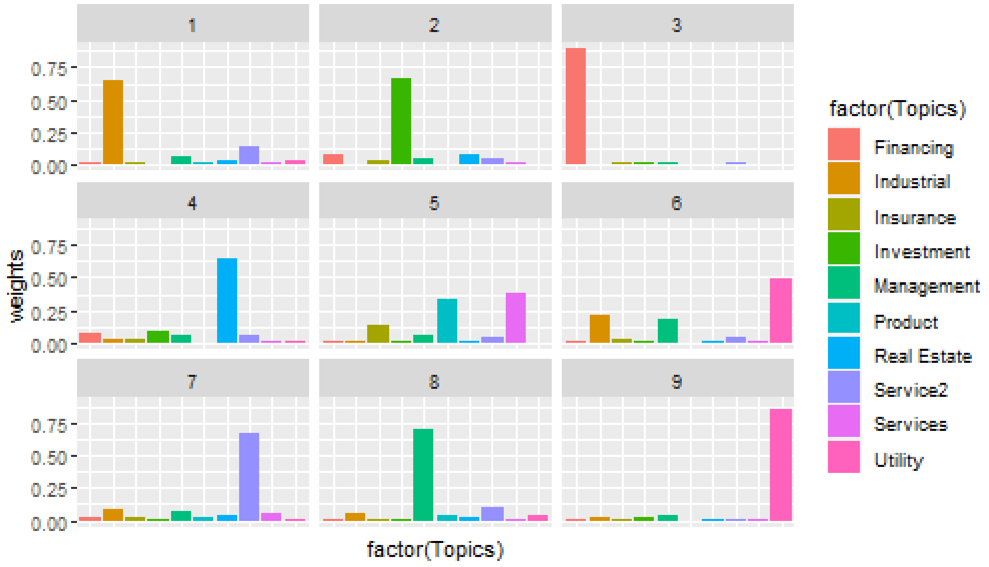
$$SS_{tot.withinss} = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

$$SS_{betweenss} = \sum_{k=1}^{K} |C_k| (\mu_k - \mu)^2$$

- $|C_k|$: number of elements in cluster k set
- μ: sample mean
- μ_k: cluster mean (centroid)

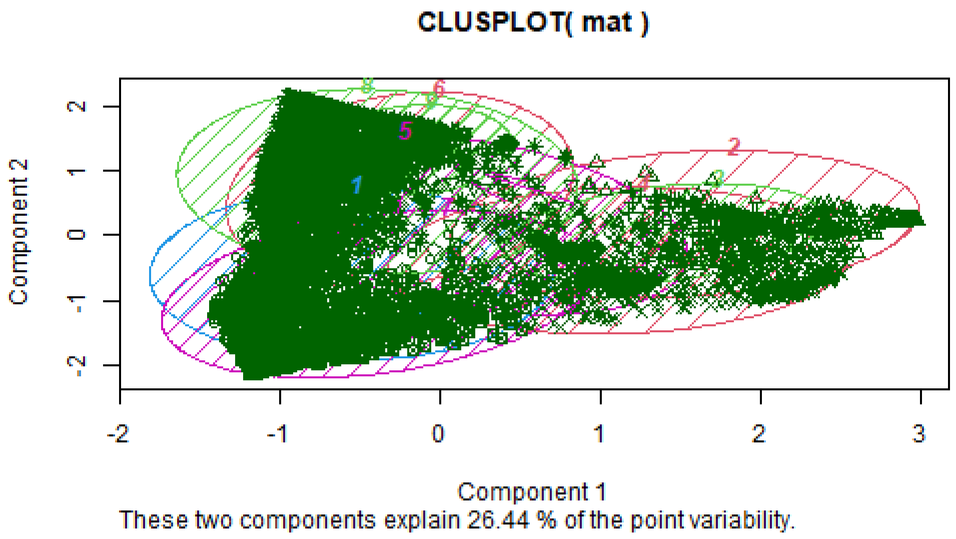
Visualizing the clusters

```r
cbind(as.data.frame(clusters$centers), data.frame(kmean=1:9)) %>% # add kmean
  gather("Topics","weights",-kmean) %>% # convert to long format
  ggplot(aes(x=factor(Topics), y=weights, fill=factor(Topics))) +
  geom_col() + facet_wrap(~kmean) +
  theme(axis.text.x=element_blank(),axis.ticks.x = element_blank())
```



Visualizing k-means with PCA

```r
# How to plot 10 features in a 2-D plot?
# For a multi-dimensional dataset, we may perform Principal Component Analysis
# and to plot data points according to the first two principal components.
library(cluster)
clusplot(mat, clusters$cluster, color=T, shade=T, labels=4)
```



- Read more on clusplot() and PCA using R

Improving our visualization

- The PCA based map is really unreadable
  - This is usually the case, unless you have only a few dimensions to the data
- *t*-distributed *S*tochastic *N*eighbor *E*mbedding (2008), t-SNE, is significantly better
  - A machine learning algorithm designed to explain machine learning algorithms
    - It maintains neighbor relationships while reducing dimensions
    - How to best represent the data using less dimensions by matching both distributions
  - It takes a much longer time to run than PCA, however
    - Recommend to use PCA before using t-SNE
  - Implemented efficiently in R in the `package:Rtsne` package

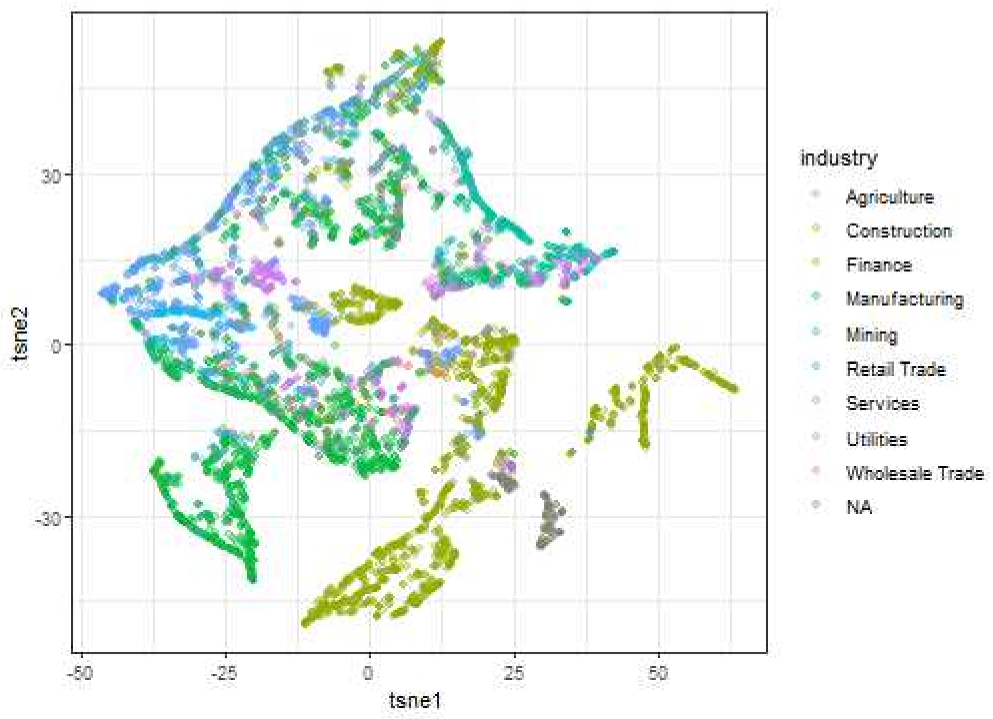
Visualizing with t-SNE

```r
library(Rtsne)
dups <- which(duplicated(mat)) # return the duplicated position
wide_nodup <- wide_topics[-dups,] # remove duplicated obs
wide_nodup$kmean <- clusters$cluster[-dups]

tsne_data <- Rtsne(mat[-dups,])

wide_nodup <- wide_nodup %>%
  mutate(tsne1 = tsne_data$Y[, 1], tsne2 = tsne_data$Y[, 2])
```

> To judge the performance of the model, we can simply look at the scatter plot generated to see if the clusters are well demarcated. However, for a more mathematical measure, we can compare the Kullback-Leibler divergences that t-SNE reports.

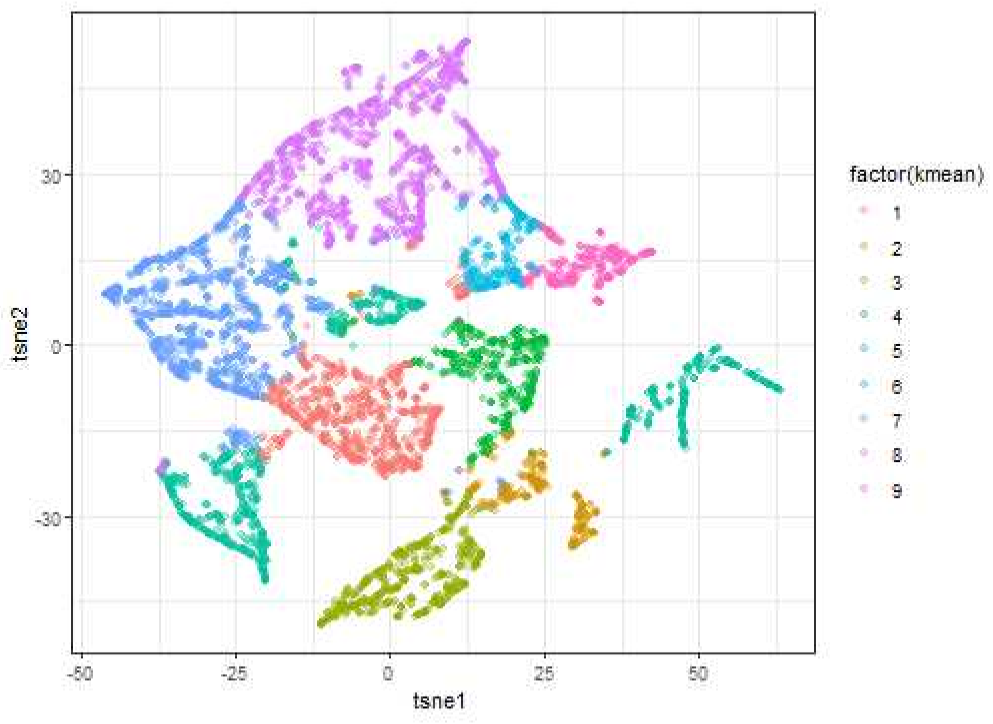
Visualizing with t-SNE: Industries

```r
ggplot(wide_nodup, aes(x = tsne1, y = tsne2, colour = industry)) +
    geom_point(alpha = 0.3) + theme_bw()
```



Visualizing with t-SNE: k-means

```
ggplot(wide_nodup, aes(x = tsne1, y = tsne2, colour = factor(kmean))) +
    geom_point(alpha = 0.3) + theme_bw()
```

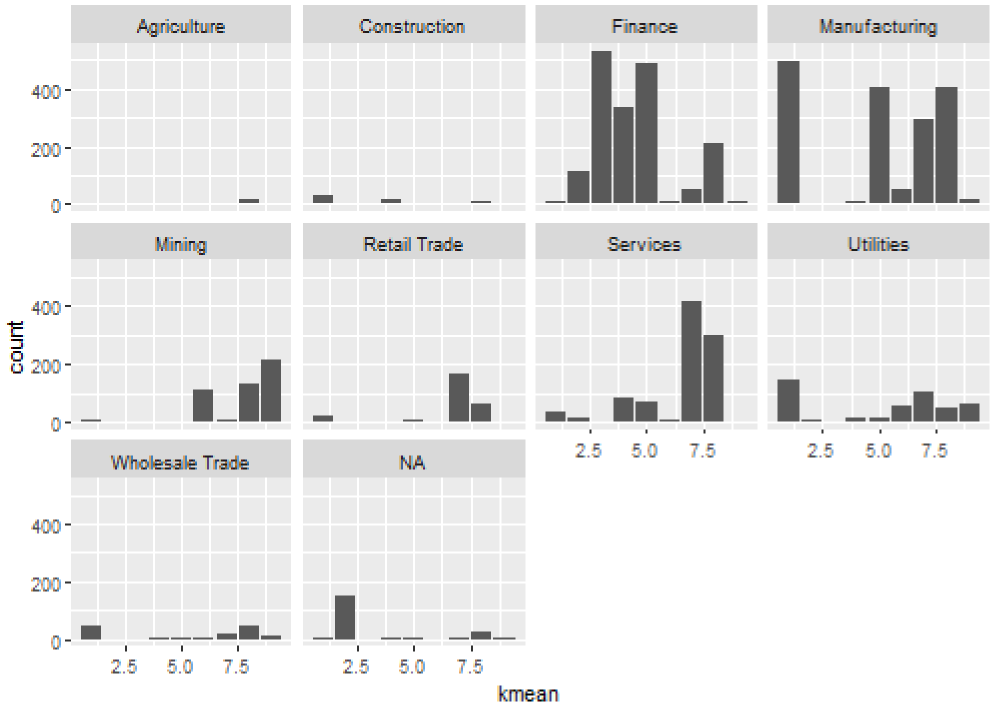


Why are these graphs different?

- Possibilities due to
  - Data: 10-K disclosure content doesn't fully capture industry inclusion
  - LDA: The measure is noisy -- needs more data
  - SIC code: The measure doesn't cleanly capture industry inclusion
    - Some firms are essentially misclassified

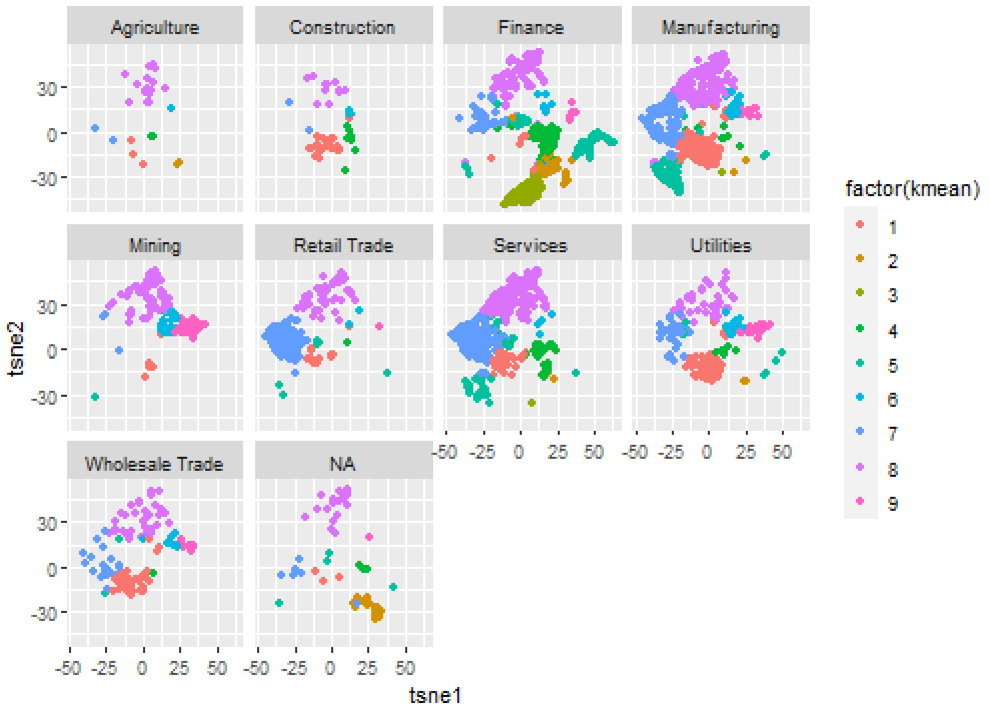
How related are clusters and industries?

```
# one bar for each cluster if they match
ggplot(wide_nodup, aes(x=kmean)) + geom_bar() + facet_wrap(~factor(industry))
```



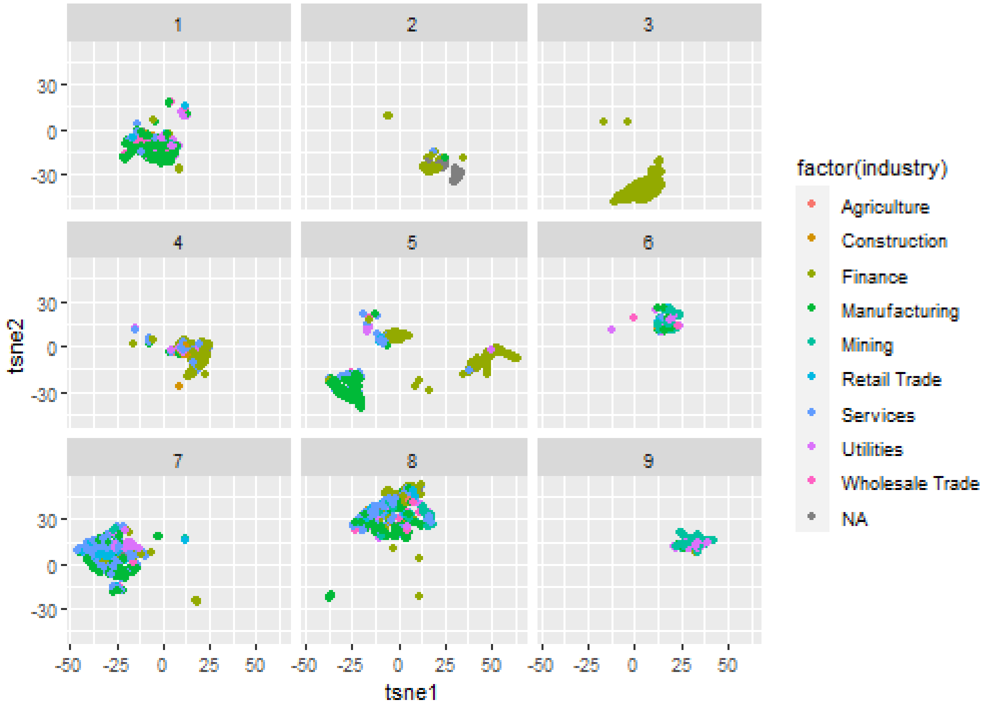
How related are clusters and industries?

```
# same color if match
ggplot(wide_nodup, aes(x=tsne1, y=tsne2, color=factor(kmean))) +
  geom_point() +  facet_wrap(~factor(industry))
```



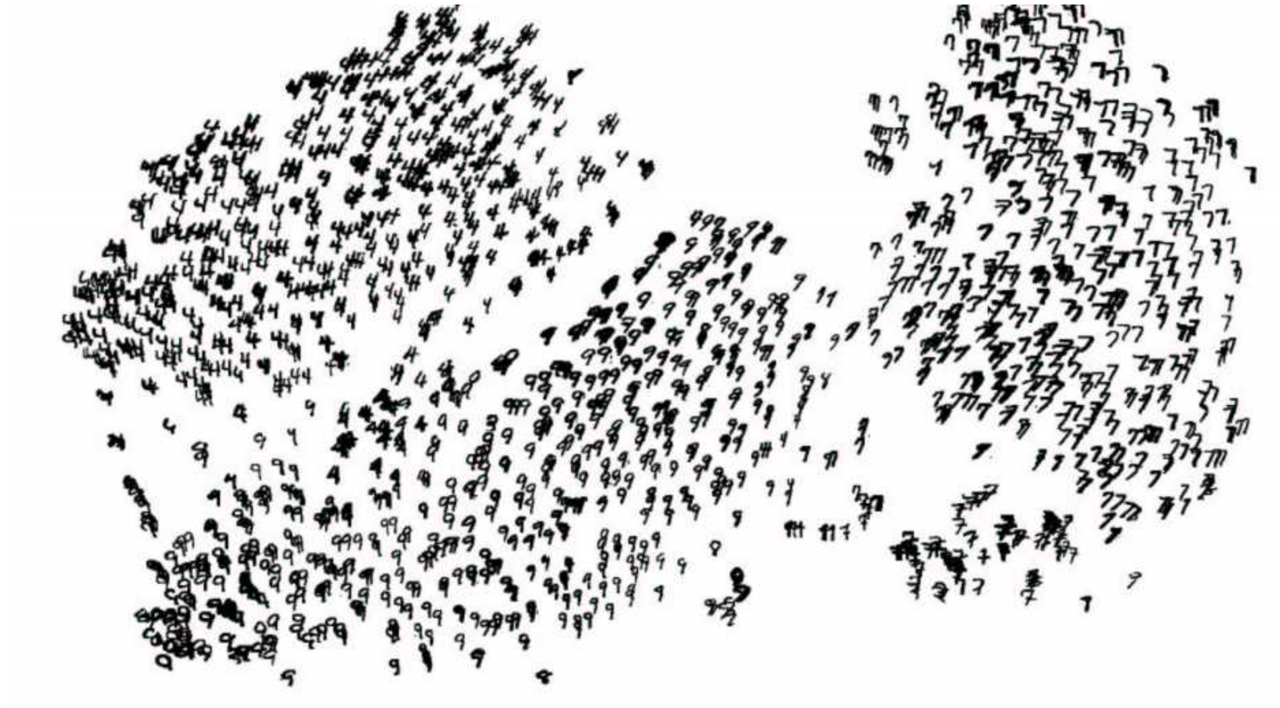
How related are clusters and industries?

```
# same color and in one group for each kmean
ggplot(wide_nodup, aes(x=tsne1, y=tsne2, color=factor(industry))) + geom_point() +
  facet_wrap(~factor(kmean))
```



Great examples of t-SNE usage

- Visualizing handwritten numbers
- Visualizing Wikipedia articles
  - The full blog post, which is a great read about visualizing high-dimensional data



Looking for anomalies

- k-means minimizes the distance from a central point
- We can look for the firms that are farthest from said point!

```r
wide_topics$dist <- sqrt(rowSums((mat - fitted(clusters))^2)) #Euclidean dist
# wide_topics$dist <- sqrt(rowSums(abs(mat - fitted(clusters))))
wide_topics[,c(1,2,3,5,13)] %>% arrange(desc(dist)) %>% slice(1:3) %>% html_df()
```

document	industry	Financing	Insurance	dist
0001171500-14-000007.txt	Finance	0.0003177	0.9972499	1.0019865
0001095073-14-000008.txt	Finance	0.0002339	0.9916079	0.9969129
0000021175-14-000021.txt	Finance	0.0036298	0.9875105	0.9935194

- These are the top 3 companies which could be different from their peers
- SIC codes lump banks and insurance together, but they are actually very different industries
  - all 3 are insurance companies
  - Platinum Underwriters; Everest Re Group; CNA Financial Corp

What we have accomplished

- We have created a classification of firms based on 10-K content
    - How similar each firm's content is to other firms' content
    - K-means clustering method is used.
    - There are some other clustering methods though
- We have used this classification to identify anomalies

> Text based industry classification using 10-Ks has been shown to be quite viable, see Hoberg and Phillips.
>
> What else could we use clustering to solve?

- Where in business to we would like to group something, but we don't know the groups?

Determining optimal clusters

How many clusters?

> We choose 9 clusters in the previous analysis becasue we intend to compare with the 9 industry classification.

- In many cases, we don't have a given number of clusters (unsupervised learning). So how to determine the optimal number of clusters?
  - Elbow method
  - Silhouette method
  - Gap statistic method

Elbow method

- The basic idea of K-means is to minimize the total within sum of square (tot.withinss)
- The elbow method is to plot the tot.withinss for various k and choose the best k by visualization
  - Apply k-means clustering for different values of k
  - For each k, calculate the tot.withinss
  - Plot the curve of tot.withinss according to the number of clusters k
  - The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.
- `fviz_nbclust()` function from the `package:factoextra` package

Silhouette method

- Silhouette method is to study the separation distance between the resulting clusters.
- It measures how close each point in one cluster is to points in the neighboring clusters.
- The measure has a range of [-1, 1] and the higher the better
  - +1 indicates the sample is far away from the neighboring clusters
  - 0 indicates the sample is on or very close to the decision boundary between two neighboring clusters
  - Negative values indicate those samples might have been assigned to the wrong cluster.
- `fviz_nbclust()` function from the `package:factoextra` package

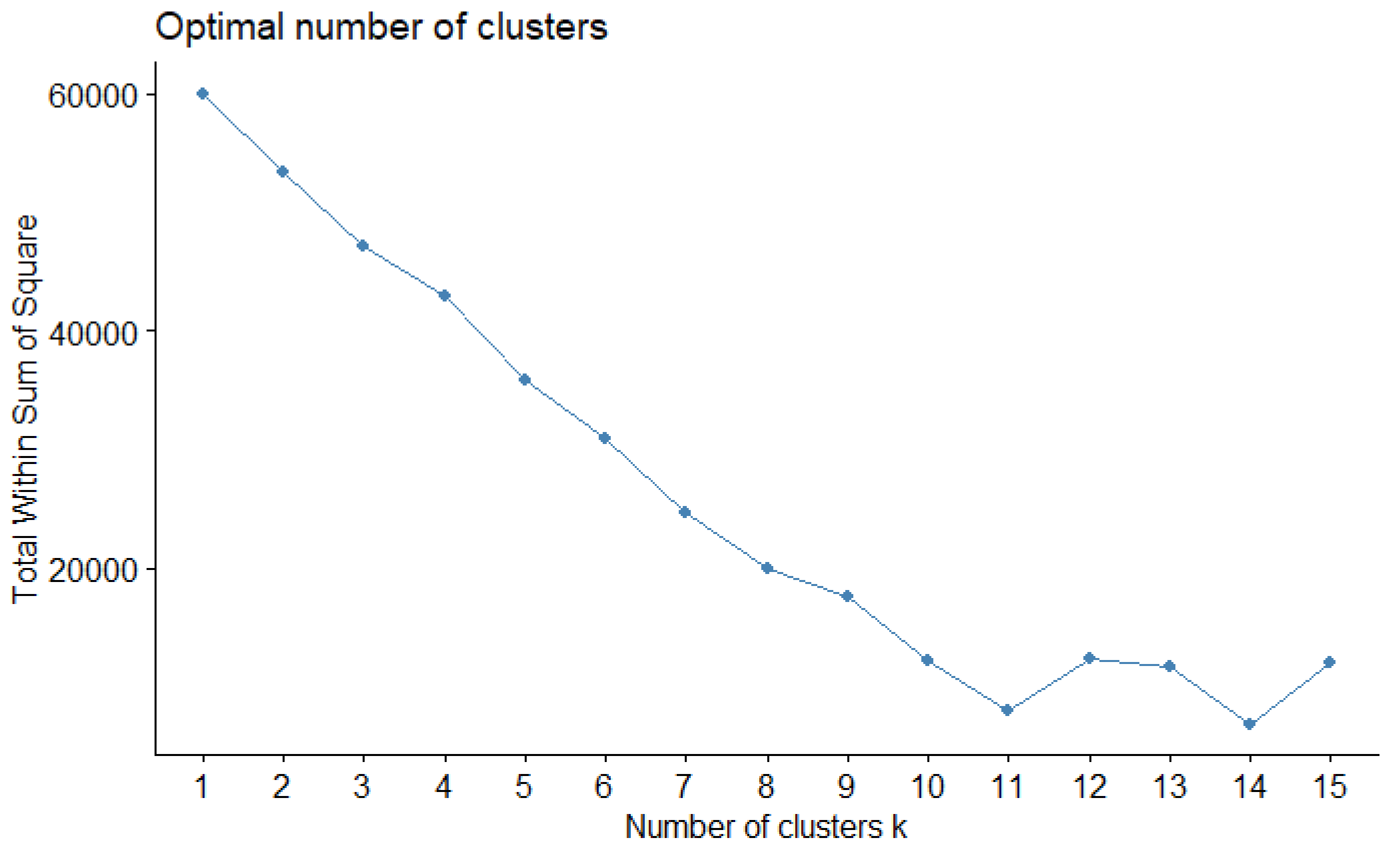
Gap statistic method

- The gap statistic compares the tot.withinss with expected values under null reference distribution of the data (i.e. a distribution with no obvious clustering).
- The reference dataset is generated using Monte Carlo simulations
- The estimate of the optimal clusters will be the value that maximizes the gap
- `fviz_nbclust()` function from the `package:factoextra` package

Computing the number of clusters using R

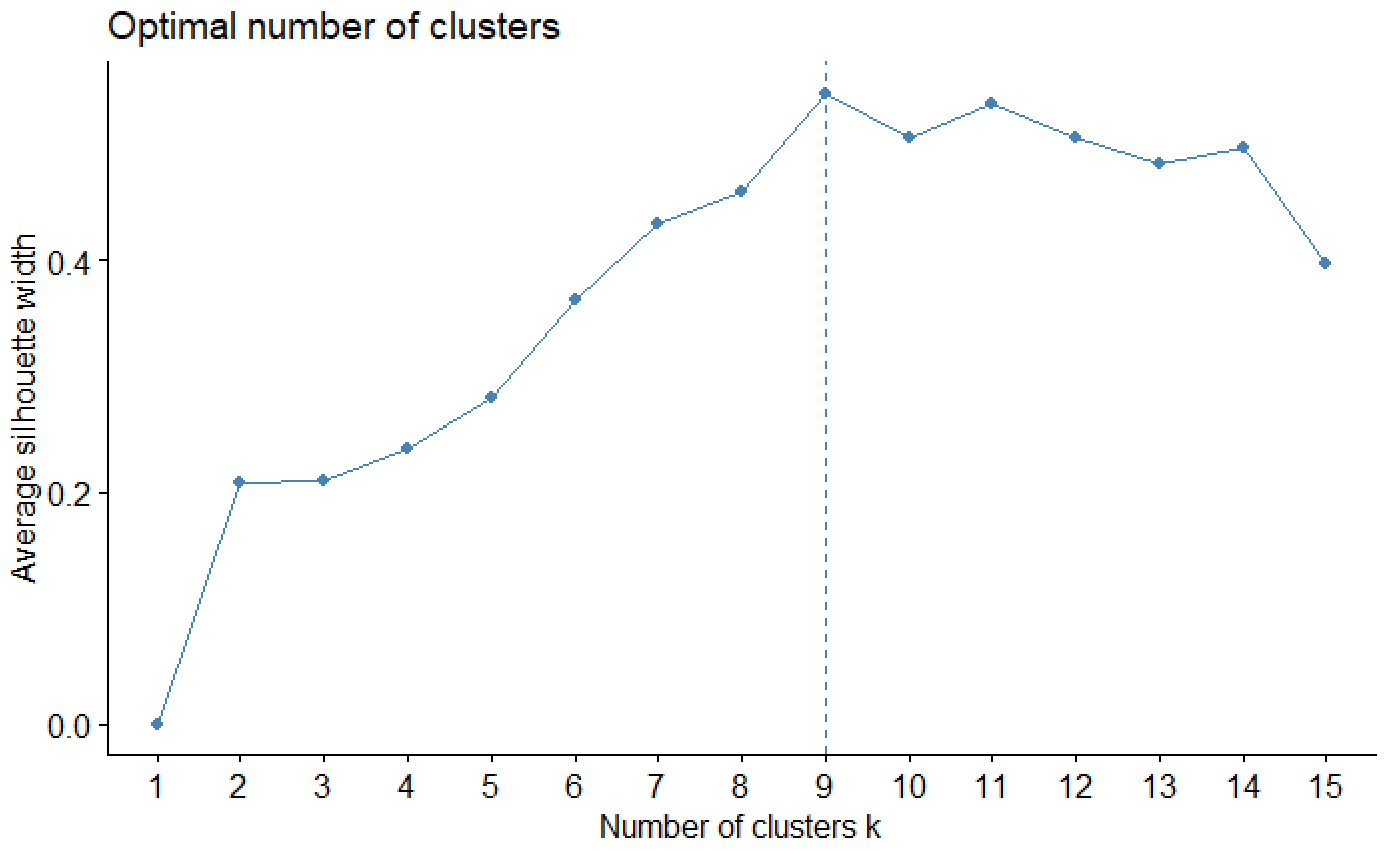
```r
# It takes very long to compute the gap statistic
library(factoextra)
set.seed(123)
fviz_nbclust(mat, kmeans, method = "wss", k.max = 15) # default k.max is 10
fviz_nbclust(mat, kmeans, method = "silhouette", k.max = 15)
fviz_nbclust(mat, kmeans, method = "gap_stat", k.max = 15)
```

Elbow method plot

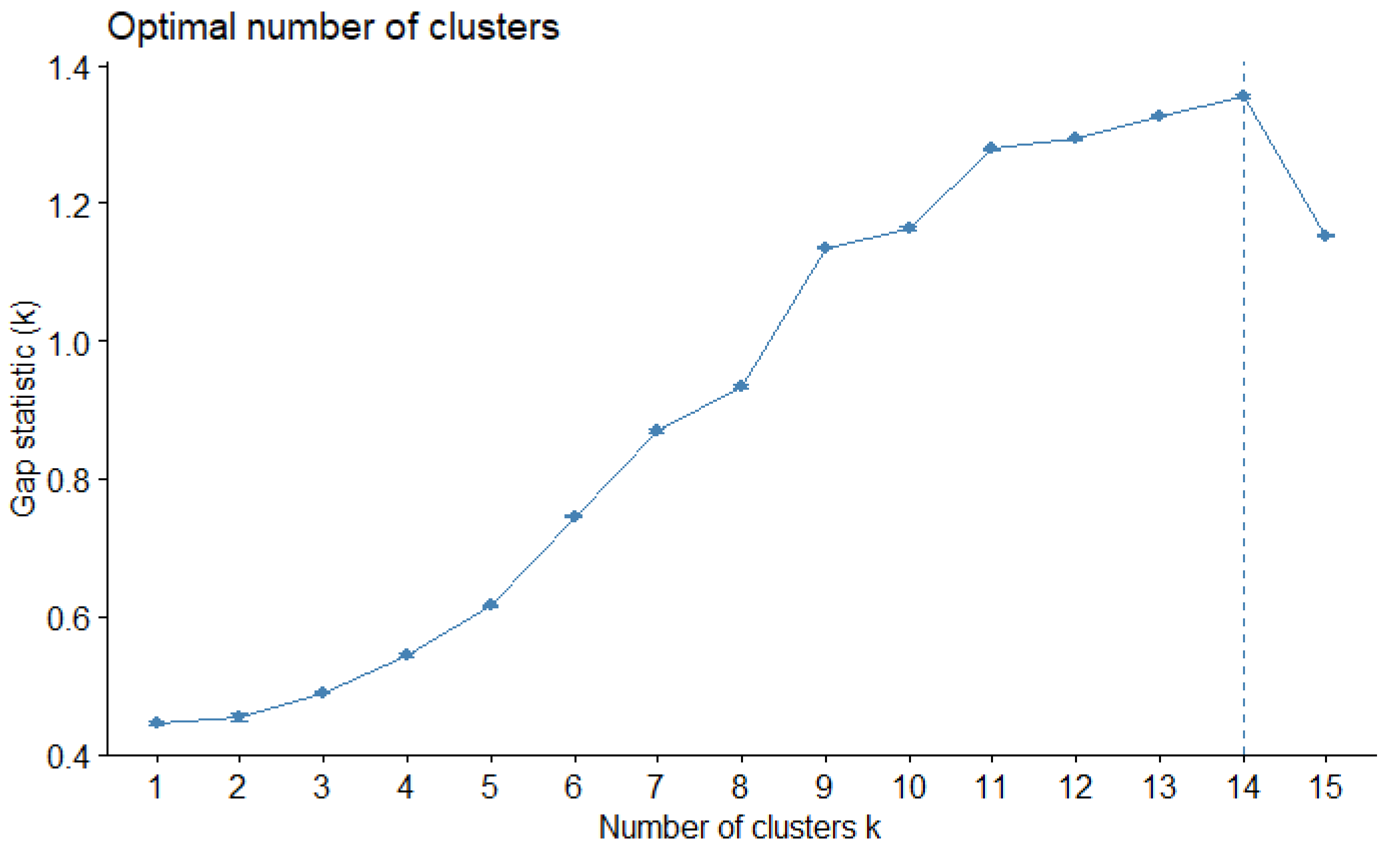


- The first bend at k = 11, through visualization only.

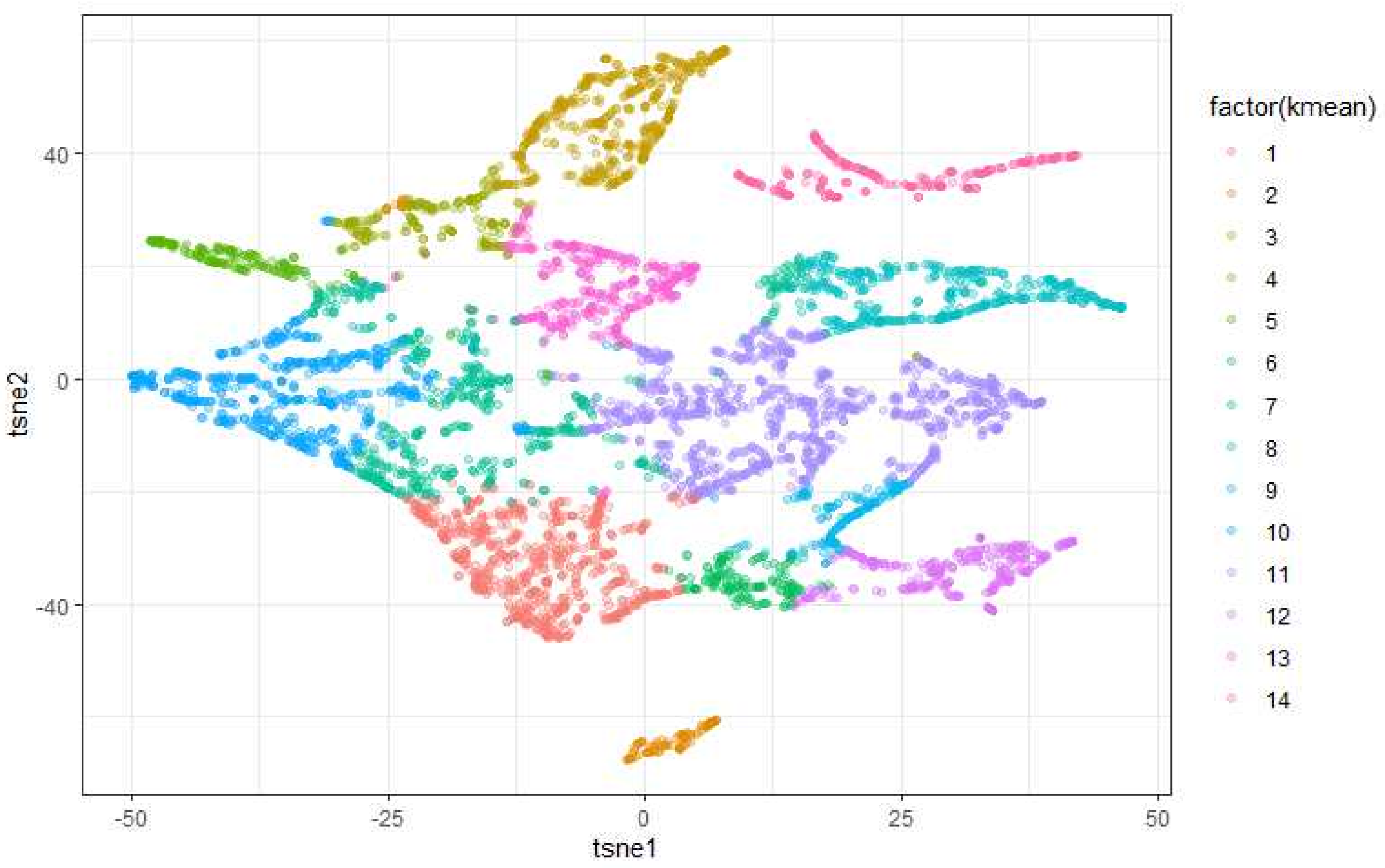
Silhouette method plot



Gap statistic method plot



Clustering in 14 groups



Filling in missing data

Problem: Missing data

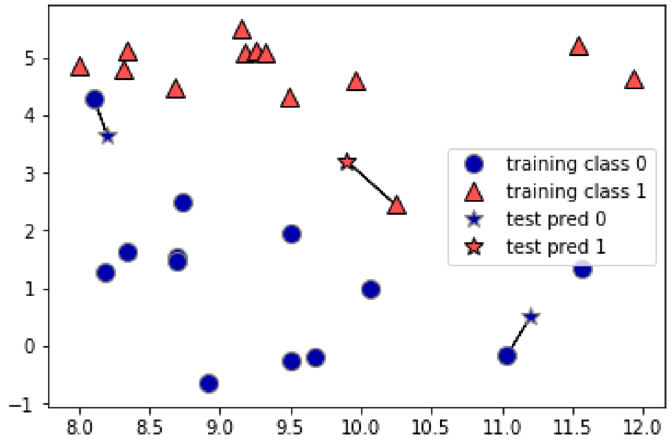
- You may have noticed that some of the `industry` measure was `NA`
- What if we want to assign an industry to these firms based on the content of their 10-K filings?

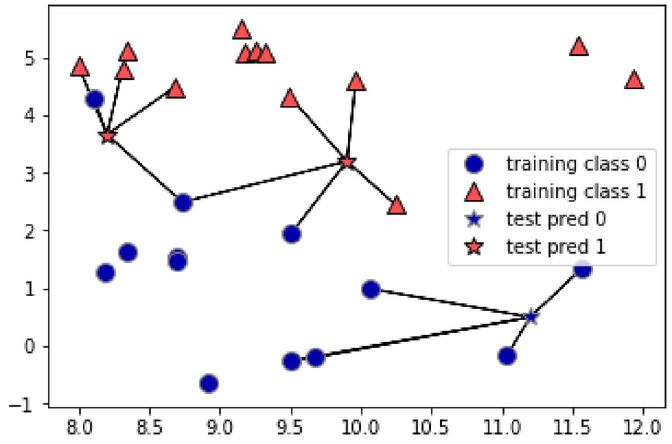
> Using k-means

- One possible approach we could use is to fill based on the category assigned by k-means
- However, as we saw, k-means and SIC code don't line up perfectly...
  - So using this classification will definitely be noisy and may be inconsistent with SIC classification

A better approach with KNN

- KNN, or *K*-*N*earest *N*eighbors is a *supervised* approach to clustering
- Since we already have industry classifications for most of our data, we can use that structure to inform our assignment of the missing industry codes
- The way the model uses the information is by letting the nearest labeled points "vote" on what the point should be
  - Points are defined by 10-K content in our case
- When considering more than one neighbor, we use voting to assign a label: the majority class among the k-nearest neighbors.





A pseudocode for the KNN algo

> to predict/classify one data point in the test data (let's call it A):

For every point in our training data:

- (1) calculate the distance between A and the current point (typically Euclidean distance, ie, the straight-line distance between two points)
- (2) sort the distances in increasing order
- (3) take k items with lowest distances to A
- (4) find the majority class among these items
- (5) return the majority class as our prediction for the class of A

Implementing KNN in R

- We'll use the `package:caret` package for this, as it will allow us to use cross validation to select a model (to find the best k)
  - The same technique we used for LASSO

```
wide_topics$industry <- factor(wide_topics$industry)
train <- wide_topics[!is.na(wide_topics$industry), ]
label <- wide_topics[is.na(wide_topics$industry), ]
head(train)
```

```
## # A tibble: 6 x 13
##   document industry Financing Industrial Insurance Investment Management Product
##   <chr>    <fct>        <dbl>      <dbl>     <dbl>      <dbl>      <dbl>   <dbl>
## 1 0000002~ Wholesa~ 0.0106       0.158     0.109      0.000463   0.116    2.10e-4
## 2 0000003~ Finance  0.0467       0.00594   0.0235     0.000528   0.0801   1.43e-4
## 3 0000003~ Utiliti~ 0.00691      0.0352    0.000366   0.0201     0.00237  1.86e-5
## 4 0000004~ Utiliti~ 0.0870       0.827     0.000326   0.000333   0.0206   4.85e-5
## 5 0000004~ Services 0.00361      0.268     0.268      0.000881   0.00264  9.49e-5
## 6 0000004~ Utiliti~ 0.0000976    0.530     0.000159   0.000753   0.000953 3.18e-5
## # ... with 5 more variables: `Real Estate` <dbl>, Service2 <dbl>,
## #   Services <dbl>, Utility <dbl>, dist <dbl>
```

Implementing KNN in R

```r
library(caret)
set.seed(123)
trControl <- trainControl(method='cv', number=10) # 10-fold cross validation
tout <- train(industry ~ ., # including all as independent varaibles
      method = 'knn', # there are 237 methods in the package
      tuneGrid = expand.grid(k = 1:20), # tune the parameter k from 1 to 20
      trControl = trControl,
      metric = "Accuracy", # overall agreement rate averaged over cv iterations
      data = select(train, -c(document, dist))) # remove "document" and "dist"
#saveRDS(tout, '../../Data/Session_9-corp_knn.rds') # save the R object
```

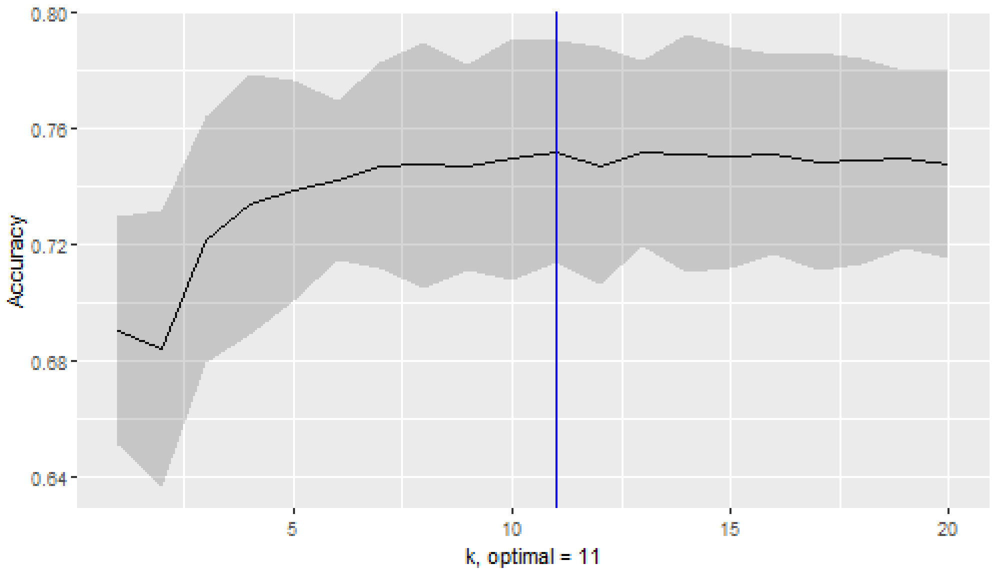
Implementing KNN in R

```
tout
```

```
## k-Nearest Neighbors
##
## 5804 samples
##   10 predictor
##    9 classes: 'Agriculture', 'Construction', 'Finance', 'Manufacturing', 'Mining', 'Retai
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5221, 5224, 5225, 5224, 5222, 5225, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.6905400  0.6015063
##    2  0.6841633  0.5931187
##    3  0.7217274  0.6392866
##    4  0.7337778  0.6535920
##    5  0.7386107  0.6597035
##    6  0.7418899  0.6634721
##    7  0.7472288  0.6706120
##    8  0.7473973  0.6704299
##    9  0.7467100  0.6694643
##   10  0.7494749  0.6728174
##   11  0.7520605  0.6759208
##   12  0.7472279  0.6694467
##   13  0.7517243  0.6753636
##   14  0.7513667  0.6748581
##   15  0.7501601  0.6730399
```

KNN performance

```r
ggplot(tout$results, aes(x=k, y=Accuracy)) +
  geom_line() +
  geom_ribbon(aes(ymin=Accuracy - AccuracySD*1.96,
                  ymax=Accuracy + AccuracySD*1.96), alpha=0.2) +
  geom_vline(xintercept=11, color="blue") +
  xlab("k, optimal = 11")
```



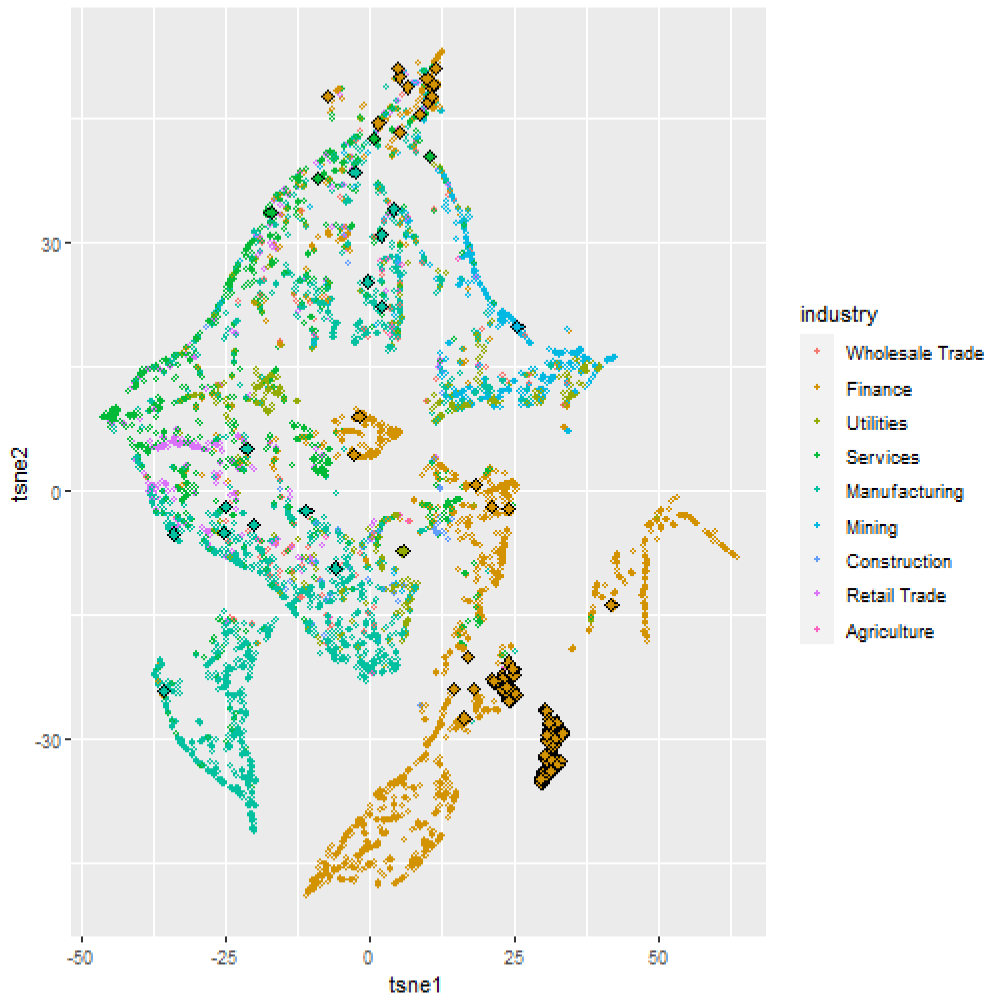
Using KNN to fill in industry

```r
label$industry_pred <- predict(tout, label)
label[ , c("document", "industry", "industry_pred")] %>% head %>% html_df
```

document	industry	industry_pred
0000817473-14-000010.txt	NA	Finance
0000820027-14-000025.txt	NA	Finance
0000837465-14-000002.txt	NA	Manufacturing
0000837919-14-000002.txt	NA	Finance
0000891092-14-000570.txt	NA	Finance
0000891092-14-002078.txt	NA	Finance

- American Capital: Asset manager and private equity
  - SIC missing in both data and filing, Finance ✓
- Ameriprise Certificate Co: Investment company
  - SIC missing in both data and filing, Finance ✓
- Callaway Golf: Golf equipment
  - SIC 3949 manufacturing ✓
- Everest Fund L P: Speculative trading of commodity futures
  - SIC missing in data but SIC = 6221 on filing, Finance ✓

t-SNE on KNN


30
0
-30
tsne2
-50
-25
0
25
50
tsne1
industry
Wholesale Trade
Finance
Utilities
Services
Manufacturing
Mining
Construction
Retail Trade
Agriculture

Summary of Session 9

Recap

Today, we:

1. Processed a set of 6,000 annual reports from 2014 to examine their readability
2. Examined the content discussed in annual reports in 2014
3. Examined the natural groupings of content across firms
   - This doesn't necessarily match up well with SIC codes
   - There are some firms that don't quite fit with others in their industry (as we algorithmically identified)
4. Filled in missing industry data using KNN, and were correct in all 6 checked entries ✓

For next week

- Try to replicate the code
- Continue your project, where is your first submission to Kaggle?
- Continue your Datacamp career track
- Fourth individual assignment
  - Submit on eLearn

Supplementary readings

- Ken Benoit - Why you should stop using other text mining packages and embrace quanteda
- twitter's AnomalyDetection for time series data

R packages used in this slide

This slide was prepared on 2021-03-04 from Session_9s.Rmd with R version 4.0.3 (2020-10-10) Bunny-Wunnies Freak Out on Windows 10 x64 build 18362 😆.

The attached packages used in this slide are:

```
##      caret      Rtsne    cluster        stm         DT    lattice   ggthemes
##   "6.0-86"     "0.15"    "2.1.0"    "1.3.5"     "0.17"  "0.20-41"    "4.2.4"
##   quanteda   readtext    forcats    stringr      dplyr      purrr      readr
##    "2.0.1"     "0.76"    "0.5.1"    "1.4.0"    "1.0.4"    "0.3.4"    "1.4.0"
##      tidyr     tibble    ggplot2  tidyverse kableExtra      knitr
##    "1.1.2"    "3.0.6"    "3.3.3"    "1.3.0"    "1.1.0"     "1.31"
```