Lecture 06:
Texto como Datos

Aprendizaje y Minería de Datos para los Negocios

Ignacio Sarmiento-Barbieri

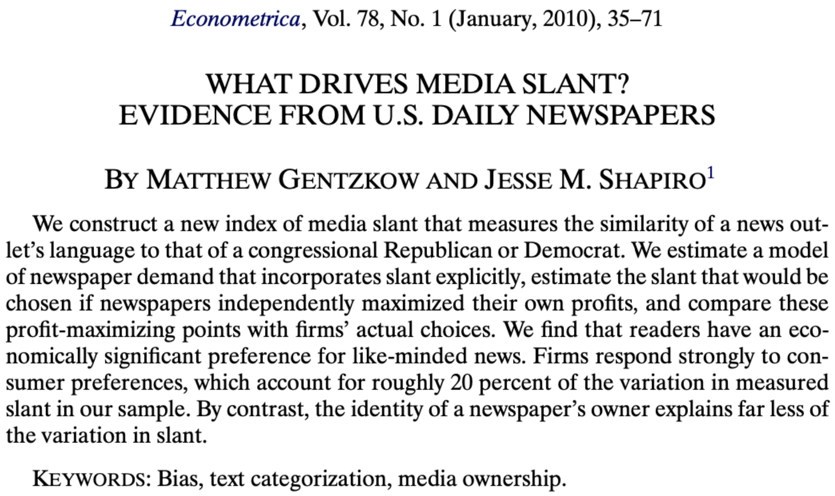
Universidad de los Andes

November 1, 2021

Agenda

Text as Data: Motivation

We'll start with a story: Slant in Partisan Speech



Econometrica, Vol. 78, No. 1 (January, 2010), 35–71

WHAT DRIVES MEDIA SLANT?
EVIDENCE FROM U.S. DAILY NEWSPAPERS

BY MATTHEW GENTZKOW AND JESSE M. SHAPIRO[1]

We construct a new index of media slant that measures the similarity of a news outlet's language to that of a congressional Republican or Democrat. We estimate a model of newspaper demand that incorporates slant explicitly, estimate the slant that would be chosen if newspapers independently maximized their own profits, and compare these profit-maximizing points with firms' actual choices. We find that readers have an economically significant preference for like-minded news. Firms respond strongly to consumer preferences, which account for roughly 20 percent of the variation in measured slant in our sample. By contrast, the identity of a newspaper's owner explains far less of the variation in slant.

KEYWORDS: Bias, text categorization, media ownership.

Text as Data: Motivation

Gentzkow and Shapiro: What drives media slant? Evidence from U.S. daily newspapers (*Econometrica*, 2010)

- Build an economic model for newspaper demand that incorporates political partisanship (Republican vs Democrat)
  - What would be independent profit-maximizing "slant"?
  - Compare this to slant estimated from newspaper text.

Congressional Record

PROCEEDINGS AND DEBATES OF THE 95th CONGRESS, SECOND SESSION

United States of America

Vol. 124

WASHINGTON, MONDAY, OCTOBER 2, 1978

TRIBUTE TO THE PHILOMATHEAN SOCIETY

HON. MICHAEL O. MYERS

OF PENNSYLVANIA

IN THE HOUSE OF REPRESENTATIVES

Text as Data: Motivation

- Jerry Moran, R-KS, says “death tax” relatively often and his district (Kansas 1st) voted 73% for George W. Bush in 2004.

$$\mathbf{X}_{\text{text}} = f(\text{ideology}) \approx g(Y_{Bush})$$

$$\Rightarrow \text{“death tax” is republican}$$

$$\Rightarrow \text{the Wall Street Journal is slanted right.}$$

- William Jefferson, D-LA, says “estate tax” relatively often and his district voted 24% for George W. Bush in 2004.

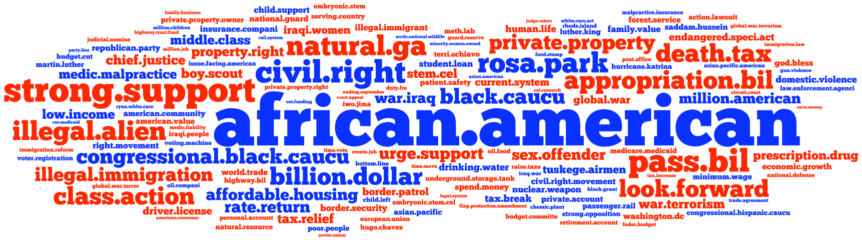
Text as Data: What is slant?

- Text: phrase-counts by speaker in 109^{th} US Congress (05-06)
- Sentiment: two-party constituent vote-share for Bush in 2004.
- Use covariance between phrase frequencies (f_{ij}) and 'Bush' sentiment (y_i) to build an index of partisanship for text.

$$z_i^{slant} = \sum_j \text{cov}(f_j, y) f_{ij}$$

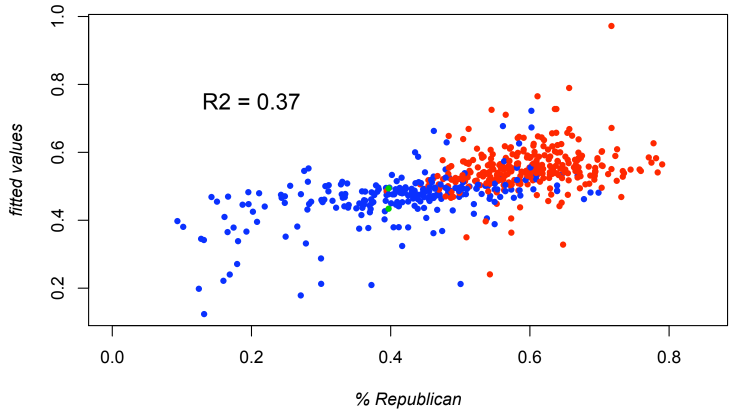
- For example, if phrase j forms a high proportion of what you say, and usage of phrase
- j is correlated with Bush vote-share, then this contributes a positive amount to your slant score.
- This is a type of *marginal regression*.

Text as Data: Wordle



- Colored by sign (positive, negative)
  Size proportional to loading $\text{cov}(f_j, y)$.
- Since y is Republican vote-share,
  - Big positive is a right term and
  - Big negative is a left term.

Slant measure for speakers in the 109th Congress



Democrats get low z_{slant} and Republicans get high z_{slant}.

Do this for newspaper text and you'll get a similar picture

Text as Data: The Big Picture

- **Text is a vast source of data for business**
- It comes connected to interesting "author" variables
  - What you buy, what you watch, your reviews
  - Group membership, who you represent, who you email
  - Market behavior, macro trends, the weather
- Opinion, subjectivity, etc.
- Sentiment is *very* loosely defined: Observables linked to the variables motivating language choice

Text as Data: The Big Picture

- **Text is also super high dimensional**
- And it gets higher dimensional as you observe more speech.
- Analysis of phrase counts is the state of the art (hard to beat).
- For example, occurrences by party for some partisan terms

Congress	State	Party	America	Death Tax	Estate Tax	$\cdots$
63	NM	dem	108	30	140	
		gop	100	220	12	

- Basic units of data
  - doc-term count matrix $\mathbf{X}$ with rows $\mathbf{x}_i$.
  - doc totals $\mathbf{m} = \texttt{rowSums}(\mathbf{X}) = [m_1 \cdots m_n]$.
  - frequency matrix $\mathbf{F} = \mathbf{X}/\mathbf{m}$ with rows $\mathbf{f}_i$.

Review & Next Steps

- XGBOOST
- XGBOOST demo
- Text as data
- Next class: More on text as data
- Questions? Questions about software?

Further Readings

- Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
- Chen, T., He, T., & Benesty, M. (2018). XGBoost Documentation.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.
- Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.

Information Retrieval and Tokenization

- A passage in '*As You Like It*' from Shakepeare:

  All the world's a stage,
  and all the men and women merely players:
  they have their exits and their entrances;
  and one man in his time plays many parts...

- What the econometrian sees:

```
world stage men women play exit entrance time
    1     1   2     1    2    1        1    1
```

- This is the Bag-of-Words representation of text.

Possible tokenization steps

- Remove words that are super rare (in say $< \frac{1}{2}\%$, or $< 15\%$ of docs; this is application specific). For example, if Argentine occurs only once, it's useless for comparing documents.
- Stemming: 'tax' $\leftarrow$ taxing, taxes, taxation, taxable, ...
  A stemmer cuts words to their root with a mix of rules and estimation.'Porter' is standard for English.
- Remove a list of stop words containing irrelevant tokens.
  If, and, but, who, what, the, they, their, a, or, ...
  Be careful: one person's stopword is another's key term.
- Convert to lowercase, drop numbers, punctuation, etc ...
  Always application specific: e.g., don't drop `:-)` from tweets.

The n-gram language model

- An n-gram language model is one that describes a dialect through transition probabilities on n consecutive words.
- An n-gram tokenization counts length-n sequences of words.
  A unigram is a word, bigrams are transitions between words.
  e.g., `world.stage`, `stage.men`, `men.women`, `women.play`, ...
- This can give you rich language data, but be careful: n-gram token vocabularies are very high dimensional (p^n)
- More generally, you may have domain specific ‘clauses’ that you wish to tokenize.
- There is always a trade-off between complexity and generality.
- Often best to just count words.

Tokenization Demo

```r
## the tm library (and related plugins) is R's ecosystem for text mining.
## for an intro see http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf
library(tm)
notes<-readPDF(control = list(text = "-layout -enc UTF-8"
  ))(elem=list(uri="~/Papers/Beauty_Hamermesh.pdf"), id=fname,
  language='en')
writeLines(content(notes)[1])
```

```
    ARTICLE IN PRESS
                                   Economics of Education Review 24 (2005) 369{376
                                                                                       www.elsevier.com/locate/econedurev
Beauty in the classroom: instructors’ pulchritude and putative
                                         pedagogical productivity
                                     Daniel S. Hamermesh, Amy Parker
                              Department of Economics, University of Texas, Austin, TX 78712-1173, USA
                                            Received 14 June 2004; accepted 21 July 2004
Abstract
   Adjusted for many other determinants, beauty affects earnings; but does it lead directly to the differences in
productivity that we believe generate earnings differences? We take a large sample of student instructional ratings for a
group of university teachers and acquire six independent measures of their beauty, and a number of other descriptors of
them and their classes. Instructors who are viewed as better looking receive higher instructional ratings, with the impact
of a move from the 10th to the 90th percentile of beauty being substantial. This impact exists within university
departments and even within particular courses, and is larger for male than for female instructors. Disentangling
whether this outcome represents productivity or discrimination is, as with the issue generally, probably impossible.
```

Tokenization Demo

```r
content(notes) <-iconv(content(notes), from="UTF-8", to="ASCII", sub="")

docs <- Corpus(VectorSource(notes))

names(docs) <- names(notes)

## you can then do some cleaning here
## tm_map just maps some function to every document in the corpus
docs <- tm_map(docs, content_transformer(tolower)) ## make everything lowercase
docs <- tm_map(docs, content_transformer(removeNumbers)) ## remove numbers
docs <- tm_map(docs, content_transformer(removePunctuation)) ## remove punctuation
## remove stopword.
##be careful with this: one's stopwords are anothers keywords.
# you could also do stemming; I don't bother here.
docs <- tm_map(docs, content_transformer(removeWords), stopwords("SMART"))

docs <- tm_map(docs, content_transformer(stripWhitespace)) ## remove excess white-space
```

Tokenization Demo

```r
## create a doc-term-matrix
dtm <- DocumentTermMatrix(docs)
dtm
```

```
## <<DocumentTermMatrix (documents: 8, terms: 913)>>
## Non-/sparse entries: 1555/5749
## Sparsity           : 79%
## Maximal term length: 30
## Weighting          : term frequency (tf)
```

```r
dtm <- removeSparseTerms(dtm, 0.75)
dtm
```

```
## <<DocumentTermMatrix (documents: 8, terms: 156)>>
## Non-/sparse entries: 650/598
## Sparsity           : 48%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

Tokenization Demo

```r
## You can inspect them:
inspect(dtm[1:5,1:8])
```

```
## <<DocumentTermMatrix (documents: 5, terms: 8)>>
## Non-/sparse entries: 26/14
## Sparsity           : 35%
## ...
## Docs academic article beauty becker behavior biddle class classes
##    1        1       1      9      1        1      2     1       1
##    2        2       1      7      0        1      0     5       5
##    3        0       1      6      0        0      0     0       1
```

```r
## find words with greater than a min count
findFreqTerms(dtm,50)
```

```
## [1] "beauty"  "ratings"
```

```r
## or grab words whose count correlates with given words
findAssocs(dtm, "beauty", .7)
```

```
## $beauty
##  equation    effect     basic  positive     table perceived   results potential
##      0.86      0.83      0.79      0.77      0.77      0.77      0.73      0.72
##   problem   effects   instruc
##      0.72      0.71      0.70
```

Text Regression

- Once you have text in a numeric format, we can use all the tools we learned so far
- For example: Classify emails into spam

$$\text{logit}\left[\texttt{spam}\right] = \alpha + f\beta \tag{1}$$

- where $f_i = \frac{x_i}{\sum_j x_{ij}}$ are the normalized text counts

Text Regression: Example (Gentzkow and Shapiro)

```r
#load packages
library(textir)
#load data
data(congress109)
congress109Counts[c("Barack Obama","John Boehner"),995:998]
```

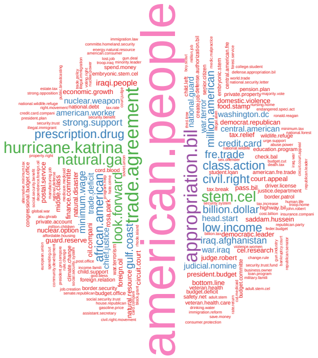
```
## 2 x 4 sparse Matrix of class "dgCMatrix"
##              stem.cel natural.ga hurricane.katrina trade.agreement
## Barack Obama        .          1                20               7
## John Boehner        .          .                14               .
```

```r
congress109Ideology[1:4,1:5]
```

```
##                              name party state chamber  repshare
## Chris Cannon         Chris Cannon     R    UT       H 0.7900621
## Michael Conaway Michael Conaway     R    TX       H 0.7836028
## Spencer Bachus   Spencer Bachus     R    AL       H 0.7812933
## Mac Thornberry   Mac Thornberry     R    TX       H 0.7776520
```

Text Regression: Example (Gentzkow and Shapiro)

```r
require("wordcloud")
wordcloud(words = colnames(congress109Counts),
          freq = colSums(congress109Counts),
          min.freq = 100,
          scale = c(3, 0.1), max.words=200,
          random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Set1"))
```

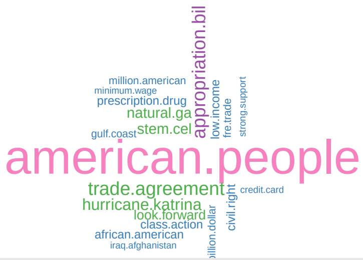


Text Regression: Wordle (Wordclouds)

```r
tail(colSums(congress109Counts))
```

```
##          stem.cel        natural.ga hurricane.katrina   trade.agreement
##              1699              1792              2020              2329
## appropriation.bil   american.people
##              2357              6256
```

```r
wordcloud(words = colnames(congress109Counts),
          freq = colSums(congress109Counts),
          min.freq = 1000,
          scale = c(3, 0.1), max.words=30,
          random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Set1"))
```



Text Regression

- We can use LASSO

```r
f <- congress109Counts
y <- congress109Ideology$repshare
# lasso
lassoslant <- cv.gamlr(congress109Counts>0, y)
B <- coef(lassoslant$gamlr)[-1,]
head(sort(round(B[B!=0],4)),10)
```

```
##    congressional.black.caucu                  family.value 
##                      -0.0839                       -0.0443 
##        issue.facing.american            voter.registration 
##                      -0.0324                       -0.0298 
##       minority.owned.business             strong.opposition 
##                      -0.0284                       -0.0264 
##                  civil.right        universal.health.care 
##                      -0.0259                       -0.0254 
## congressional.hispanic.caucu            ohio.electoral.vote 
##                      -0.0187                       -0.0183
```

Text Regression

```r
tail(sort(round(B[B!=0],4)),10)
```

```
##          illegal.alien       percent.growth  illegal.immigration
##                 0.0079               0.0083               0.0087
##             global.war         look.forward           war.terror
##                 0.0098               0.0099               0.0114
##       private.property       action.lawsuit         human.embryo
##                 0.0133               0.0142               0.0226
## million.illegal.alien
##                 0.0328
```

Topic Models

- Text is super high dimensional
- there is often abundant *unlabeled* text
- Some times unsupervized factor model is a popular and useful strategy with text data
- You can first fit a factor model to a giant corpus and use these factors for supervised learning on a subset of labeled documents.
- The unsupervised dimension reduction facilitates the supervised learning

Topic Models: Example

- We have 6166 reviews, with an average length of 90 words per review, `we8there.com`.
- A useful feature of these reviews is that they contain both text and a multidimensional rating on overall experience, atmosphere, food, service, and value.
- For example, one user submitted a glowing review for Waffle House #1258 in Bossier City, Louisiana: *I normally would not revue a Waffle House but this one deserves it. The workers, Amanda, Amy, Cherry, James and J.D. were the most pleasant crew I have seen. While it was only lunch, B.L.T. and chili, it was great. The best thing was the 50′ s rock and roll music, not to loud not to soft. This is a rare exception to what you all think a Waffle House is. Keep up the good work.*
  Overall: 5, Atmosphere: 5, Food: 5, Service: 5, Value: 5.

Topic Models: Example

- After cleaning and Porter stemming, we are left with a vocabulary of 2640 bigrams.
- For example, the first review in the document-term matrix has nonzero counts on bigrams indicating a pleasant meal at a rib joint:

```
#load packages
library(textir)
#load data
data(we8there)
x <- we8thereCounts
x[1,x[1,]!=0]
```

```
## even though larg portion  mouth water     red sauc     babi back     back rib chocol mouss
##           1             1            1            1             1            1            1
## veri satisfi
##            1
```

Topic Models: Example

- We can apply PCA to get a factor representation of the review text.
- PC1 looks like it will be big and positive for positive reviews,

```r
pca <- prcomp(x, scale=TRUE) # can take a long time

tail(sort(pca$rotation[,1]))
```

```
##      food great      staff veri      excel food high recommend      great food
##     0.007386860     0.007593374     0.007629771     0.007821171     0.008503594
##      food excel
##     0.008736181
```

- while PC4 will be big and negative

```r
tail(sort(pca$rotation[,4]))
```

```
##    order got after minut  never came   ask check readi order drink order
##   0.05918712   0.05958572  0.06099509  0.06184512  0.06776281  0.07980788
```

Principal Component Analysis

- Dimensionality via main components

$$X = (x_1, x_2, \ldots, x_n)_{N\times K} \tag{2}$$

- Factor:

$$F = X\delta \;\; \delta \in K \tag{3}$$

- Idea: summarize the K variables in a single (F).
- Vocab: the coefficients of δ are the loadings: how much 'matters' each x s in the factor.
- Dimensionality: summarize the original K variables in a few $q < K$ factors.

Algebra Review

- Let $A_{m\times m}$. It exists
  - a scalar λ such that $Ax = \lambda x$ for a vector $x_{m\times 1}$,
  - if $x \neq 0$, then λ is an eigenvalue of A.
  - and a vector x is an eigenvector of A corresponding to the eigenvalue λ.
- $A_{m\times m}$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$, then:

$$tr(A) = \sum_{i=1}^{m} \lambda_i \tag{4}$$

$$det(A) = \Pi_{i=1}^{m} \lambda_i \tag{5}$$

- If $A_{m\times m}$ has m different eigenvalues, then the associated eigenvectors are all linearly independent.
- Spectral decomposition: $A = P\Lambda P$, where $\Lambda = diag(\lambda_1, \ldots \lambda_n)$ and P is the matrix whose columns are the corresponding eigenvectors.

Factors via main components

- $x_1, x_2, \ldots, x_K$, K vectors of N observations each.
- Factor: $F = X\delta$
- What is the 'best' linear combination of $x_1, x_2, \ldots, x_K$?
- Best? Maximum variance. Why? The one that best reproduces variability original of all xs

Factors via main components

- Let
  - $X = (x_1, \dots, x_K)_{N\times K}$,
  - $\Sigma V(X)$
  - $\delta \in K$
- $F = X\delta$ is a linear combination of X, with $V(X\delta) = \delta'\Sigma\delta$.
- Let's set up the problem as

$$\underset{\delta}{max}\ \delta'\Sigma\delta \tag{6}$$

  - It is obvious that the solution is to bring δ to infinity.

Factors via main components

- Let's "fix" the problem by normalizing δ

$$\begin{aligned} \max_{\delta} \delta' \Sigma \delta \\ \text{subject to} \\ \delta' \delta = 1 \end{aligned} \tag{7}$$

- Let us call the solution to this problem δ^*.
- $F^* = X\delta^*$ is the 'best' linear combination of X.

Factors via main components

- Result: δ^* is the eigenvector corresponding to the largest eigenvalue of $\Sigma = V(X)$.
- $F^* = X\delta^*$ is the first principal component of X.
- Intuition: X has K columns and $Y = X\delta$ has only one. The factor built with the first principal component is the best way to represent the K variables of X using a single single variable.

Topic Models

- Text is super high dimensional
- Some times unsupervized factor model is a popular and useful strategy with text data
- You can first fit a factor model to a giant corpus and use these factors for supervised learning on a subset of labeled documents.
- The unsupervised dimension reduction facilitates the supervised learning

Topic Models: Example

- We have 6166 reviews, with an average length of 90 words per review, we8there.com.
- A useful feature of these reviews is that they contain both text and a multidimensional rating on overall experience, atmosphere, food, service, and value.
- For example, one user submitted a glowing review for Waffle House #1258 in Bossier City, Louisiana: *I normally would not revue a Waffle House but this one deserves it. The workers, Amanda, Amy, Cherry, James and J.D. were the most pleasant crew I have seen. While it was only lunch, B.L.T. and chili, it was great. The best thing was the 50′ s rock and roll music, not to loud not to soft. This is a rare exception to what you all think a Waffle House is. Keep up the good work.*
  Overall: 5, Atmosphere: 5, Food: 5, Service: 5, Value: 5.

Topic Models: Example

- After cleaning and Porter stemming, we are left with a vocabulary of 2640 bigrams.
- For example, the first review in the document-term matrix has nonzero counts on bigrams indicating a pleasant meal at a rib joint:

```r
#load packages
library(textir)
#load data
data(we8there)
x <- we8thereCounts
x[1,x[1,]!=0]
```

```
## even though larg portion  mouth water      red sauc     babi back     back rib chocol mouss
##           1            1            1            1            1            1            1
## veri satisfi
##            1
```

Topic Models: Example

- We can apply PCA to get a factor representation of the review text.
- PC1 looks like it will be big and positive for positive reviews,

```r
pca <- prcomp(x, scale=TRUE) # can take a long time

tail(sort(pca$rotation[,1]))
```

```
##      food great       staff veri      excel food high recommend      great food
##     0.007386860     0.007593374     0.007629771     0.007821171     0.008503594
##      food excel
##     0.008736181
```

- while PC4 will be big and negative

```r
tail(sort(pca$rotation[,4]))
```

```
##    order got after minut   never came    ask check readi order drink order
##   0.05918712   0.05958572   0.06099509   0.06184512   0.06776281   0.07980788
```

Principal Component Analysis

- Dimensionality via main components

$$X = (x_1, x_2, \dots, x_K)_{N\times K} \tag{8}$$

- Factor:

$$F = X\delta \;\; \delta \in K \tag{9}$$

- Idea: summarize the K variables in a single (F).
- Vocab: the coefficients of δ are the loadings: how much 'matters' each x s in the factor.
- Dimensionality: summarize the original K variables in a few $q < K$ factors.

Algebra Review

- Let $A_{m\times m}$. It exists
  - a scalar λ such that $Ax = \lambda x$ for a vector $x_{m\times 1}$,
  - if $x \neq 0$, then λ is an eigenvalue of A.
  - and a vector x is an eigenvector of A corresponding to the eigenvalue λ.
- $A_{m\times m}$ with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_m$, then:

$$tr(A) = \sum_{i=1}^{m} \lambda_i \tag{10}$$

$$det(A) = \Pi_{i=1}^{m} \lambda_i \tag{11}$$

- If $A_{m\times m}$ has m different eigenvalues, then the associated eigenvectors are all linearly independent.
- Spectral decomposition: $A = P\Lambda P$, where $\Lambda = diag(\lambda_1, \dots \lambda_n)$ and P is the matrix whose columns are the corresponding eigenvectors.

Factors via main components

- $x_1, x_2, \ldots, x_K$, K vectors of N observations each.
- Factor: $F = X\delta$
- What is the 'best' linear combination of $x_1, x_2, \ldots, x_K$?
- Best? Maximum variance. Why? The one that best reproduces variability original of all xs

Factors via main components

- Let
  - $X = (x_1, \ldots, x_K)_{N \times K}$,
  - $\Sigma = V(X)$
  - $\delta \in K$
- $F = X\delta$ is a linear combination of X, with $V(X\delta) = \delta' \Sigma \delta$.
- Let's set up the problem as

$$\max_{\delta} \; \delta' \Sigma \delta \tag{12}$$

  - It is obvious that the solution is to bring δ to infinity.

Factors via main components

- Let's "fix" the problem by normalizing δ

$$\begin{aligned} \max_{\delta} \ \delta'\Sigma\delta \\ \text{subject to} \\ \delta'\delta = 1 \end{aligned} \tag{13}$$

- Let us call the solution to this problem δ^*.
- $F^* = X\delta^*$ is the 'best' linear combination of X.
- Intuition: X has K columns and $Y = X\delta$ has only one. The factor built with the first principal component is the best way to represent the K variables of X using a single single variable.

Factors via main components

- Solution to the problem of the first principal component
- Problem:

$$\underset{\delta}{max}\ \delta'\Sigma\delta \ \text{ s.t. } \delta'\delta = 1$$

- Seting up the Lagrangian

$$\mathcal{L}(\delta, \lambda) = \delta'\Sigma\delta + \lambda(1 - \delta'\delta)$$

- CPO

$$\Sigma\delta = \lambda\delta \tag{14}$$

- At the optimum, δ is the eigenvector corresponding to the eigenvalue λ of Σ.
- Premultiplying by δ and remembering that $\delta'\delta = 1$:

$$\delta\Sigma\delta = \lambda \tag{15}$$

- In order to maximize $\delta\Sigma\delta$ we must choose λ equal to the maximum eigenvalue of Σ and δ is the corresponding eigenvalue.

Factors is unsupervised learning

- Recall that
  - In regression we had

$$y = X\beta + u \tag{16}$$

  - We minimized the MSE

$$min \sum (y_i - \hat{y})^2 \tag{17}$$

  - Learning is supervised: the difference between y and $\hat{y}$ "guides" the learning.
- The factor construction problem is unsupervised: we construct an index (the factor) without ever seeing it.
  - We start with

$$X_{n\times k} \tag{18}$$

  - We end with

$$F^*_{n\times 1} = X\delta^* \tag{19}$$

q main components

- The first main component? Are there others?
- Let's consider the following problem:

$$\max_{\delta_2} \delta_2' \Sigma \delta_2 \tag{20}$$

$$\text{subject to}$$

$$\delta_2'\delta_2 = 1$$

$$and$$

$$Cov(\delta_2' X, \delta^{*'} X) = 0$$

- $F_2^* = X\delta_2^*$ is the second principal component : the best linear combination which is orthogonal to the best initial linear combination.
- Recursively, using this logic you can form q main components.
- Note that algebraically we could construct $q = K$ factors.

q main components

- Let $\lambda_1, \ldots, \lambda_K$ be the eigenvalues of $\Sigma = V(X)$, ordered from highest to lowest, and $p_1, \ldots, p_K$ the corresponding eigenvectors. Let us call P the matrix of eigenvectors.
- Result: $\delta_j = p_j \,, \forall j$ ('loadings' of the principal components =ordered eigenvectors of Σ).
- Let $F_j = X\delta_j \,, j = 1, \ldots, K$ be the j-th principal component. It's easy to see that

$$V(F_j) = \delta_j' \Sigma \delta_j = p_j P \Lambda P p_j = \lambda_j \tag{21}$$

(the variance of the j-th principal component is the j-th ordered eigenvalue of Σ).

Relative importance of factors

- The total variance of X is the sum of the variances of x_j , $j = 1, ..., K$, that is $trace(\Sigma)$
- It is easy to show that:

$$trace(\Sigma) = trace(P\Lambda P') = trace(PP'\Lambda) = \sum_{j=1}^{K} \lambda_j = \sum_{j=1}^{K} V(F_j) \tag{22}$$

- Then

$$\frac{\lambda_k}{\sum_{j=1}^{K} \lambda_j} \tag{23}$$

- measures the relative importance of the jth principal component.

Selection of factors

- Look at the importance of the first principal components. If the first one explains a lot, there is really only one dimension (one dimension explains almost everything).
- The coefficients of the eigenvectors are weights. See how each of the variables ’contributes’ in each factor.
- Beware of differences in scale. Always standardize

Selection of factors

- Let the columns of X be standardized, so that each variable has unit variance.
- In this case:

$$trace(\Sigma) = \sum_{j=1}^{K} V(F_j) = K \tag{24}$$

- and recall $\sum_{j=1}^{K} \lambda_j = \sum_{j=1}^{K} V(F_j)$ then

$$\sum_{j=1}^{K} \lambda_j = K \tag{25}$$

- On average, each factor contributes one unit. When $\lambda_j > 1$, that factor it explains the total variance more than the average. $\rightarrow$ Retain the factors with $\lambda_j > 1$

Useful Tips: Factor Computation

- As a practical aside, note that `prcomp` converts x here from sparse to dense matrix storage.
- For really big text DTMs, which will be very sparse, this will cause you to run out of memory.
- A big data strategy for PCA is to first calculate the covariance matrix for x and then obtain PC rotations as the eigenvalues of this covariance matrix.
- The first step can be done using sparse matrix algebra.
- The rotations are then available as

```
## eigen( xvar, symmetric = TRUE)$vec.
```

- There are also approximate PCA algorithms available for fast factorization on big data. See, for example, the `irlba` package for R.

Factor Interpretation

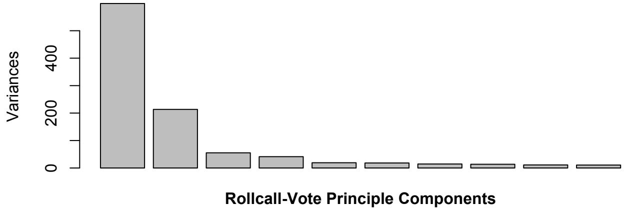
- $F_s = X\delta_s$: 'loadings' often suggest that a factor works as a 'index' of a group of variables.
- Idea: look at the 'loadings'
- Caution: factors via principal components are orthogonal recursively.

Factor Interpretation: Example

- **Congress and Roll Call Voting**
  - Votes in which names and positions are recorded are called ‘roll calls’.
  - The site `voteview.com` archives vote records and the R package `pscl` has tools for this data.
  - 445 members in the last US House (the 111^{th})
  - 1647 votes: nea = -1, yea=+1, missing = 0.
  - This leads to a large matrix of observations that can probably be reduced to simple factors (party).

Factor Interpretation

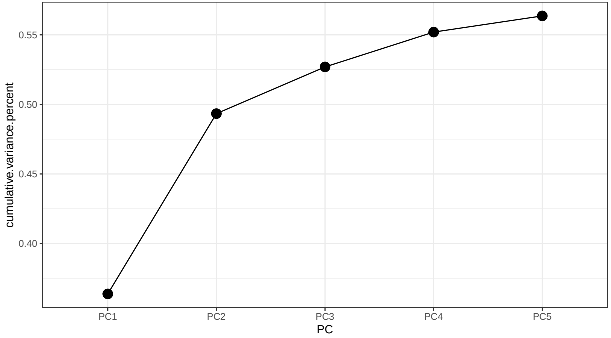
- Vote components in the **111**th house
- Each PC is $F_s = X\delta_s$



- Huge drop in variance from 1st to 2nd and 2nd to 3rd PC.
- Poli-Sci holds that PC1 is usually enough to explain congress.
  2nd component has been important twice: 1860's and 1960's.

Factor Interpretation

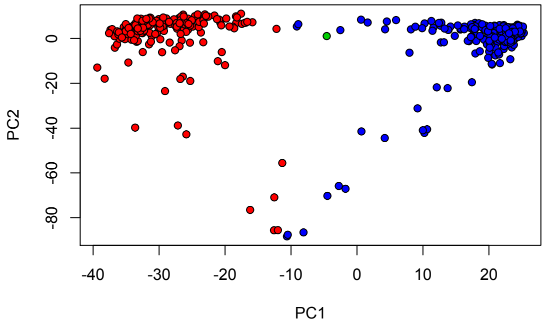
- Vote components in the **111**th house
- Each PC is $F_s = X\delta_s$



- Huge drop in variance from 1st to 2nd and 2nd to 3rd PC.
- Poli-Sci holds that PC1 is usually enough to explain congress.
  2nd component has been important twice: 1860's and 1960's.

Factor Interpretation

- Top two PC directions in the **111**th house



- Republicans in red and Democrats in blue:
  - Clear separation on the first principal component.
  - The second component looks orthogonal to party.

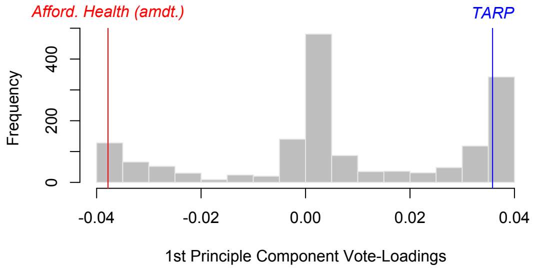
Factor Interpretation

```
## Far right (very conservative)
> sort(votepc[,1])
    BROUN (R GA-10)       FLAKE (R AZ-6)   HENSARLIN (R TX-5)
        -39.3739409          -38.2506713          -37.5870597

## Far left (very liberal)
> sort(votepc[,1], decreasing=TRUE)
   EDWARDS (D MD-4)   PRICE (D NC-4)    MATSUI (D CA-5)
         25.2915083       25.1591151         25.1248117

## social issues?  immigration?  no clear pattern
> sort(votepc[,2])
    SOLIS (D CA-32) GILLIBRAND (D NY-20)      PELOSI (D CA-8)
       -88.31350926         -87.58871687         -86.53585568
   STUTZMAN (R IN-3)       REED (R NY-29)      GRAVES (R GA-9)
       -85.59217310         -85.53636319         -76.49658108
```

- PC1 is easy to read, PC2 is ambiguous (is it even meaningful?)

Factor Interpretation

- **High PC1-loading votes are ideological battles.**
- These tend to have informative voting across party lines.



- A vote for Repulican amendments to 'Affordable Health Care for America' strongly indicates a negative PC1 (more conservative), while
  a vote for Troubled Asset Relief Program (TARP) indicates a positive PC1 (more progressive).

Factor Interpretation

- Look at the largest loadings in δ_2 to discern an interpretation.

```
> loadings[order(abs(loadings[,2]), decreasing=TRUE)[1:5],2]
 Vote.1146   Vote.658  Vote.1090  Vote.1104  Vote.1149
0.05605862 0.05461947 0.05300806 0.05168382 0.05155729
```

- These votes all correspond to near-unanimous symbolic action.
- For example, 429 legislators voted for resolution 1146:
  'Supporting the goals and ideals of a Cold War Veterans Day'
  If you didn't vote for this, you weren't in the house.
- Mystery Solved: the second PC is just attendance!

```
> sort(rowSums(votes==0), decreasing=TRUE)
    SOLIS (D CA-32) GILLIBRAND (D NY-20)       REED (R NY-29)
               1628                 1619                 1562
  STUTZMAN (R IN-3)       PELOSI (D CA-8)      GRAVES (R GA-9)
               1557                 1541                 1340
```

Principal Component Regression

- The concept is very simple: instead of regressing onto X, use a lower dimension set of principal components F as covariates.
- This works well for a few reasons:
  - PCA reduces dimension, which is always good.
  - Higher variance covariates are good in regression, and we choose the top PCs to have highest variance.
  - The PCs are independent: no multicollinearity.
- The 2-stage algorithm is straightforward. For example,

```
mypca = prcomp(X, scale=TRUE)
z = predict(mypca)[,1:K]
reg = glm(y˜., data=as.data.frame(z))
```

Latent Dirichlet Allocation

- The approach of using PCA to factorize text was common before the 2000s.
- Versions of this algorithm were referred to under the label latent semantic analysis.
- However, this changed with the introduction of topic modeling, also known as Latent Dirichlet Allocation (LDA), by Blei et al. in 2003.
- These authors pointed out that the squared error loss (i.e., Gaussian model) implied by PCA is inappropriate for analysis of sparse word-count data.
- Instead, they proposed you take the bag-of-words representation seriously and model token counts as realizations from a multinomial distribution.

Latent Dirichlet Allocation

- That is, they proposed topic models as a multinomial factor model.
- Topic models are built on a simple document generation process:
  - For each word, pick a “topic” k. This topic is defined through a probability vector over words, say, θ_k with probability θ_{kj} for each word j.
  - Then draw the word according to the probabilities encoded in θ_k .
- After doing this over and over for each word in the document, you have proportion ω_{i1} from topic 1, ω_{i2} from topic 2, and so on.

Latent Dirichlet Allocation

- This basic generation process implies that the full vector of word counts, x_i, has a multinomial distribution:

$$x_i \sim MN(\omega_{i1}\theta_1 + \cdots + \omega_{iK}\theta_K, m_i) \tag{26}$$

- where $m_i = \sum_j x_{ij}$ is the total document length and, for example,
- the probability of word j in document i will be $\sum_k \omega_{ik}\theta_{kj}$

Latent Dirichlet Allocation vs PCA

- Recall our PC model:

$$E(x_i) = \delta_{i1}F_1 + \cdots + \delta_{iK}F_K \tag{27}$$

- The analogous topic model representation, implied by the above equation, is

$$E(x_i) = \omega_{i1}\theta_1 + \cdots + \omega_{iK}\theta_K \tag{28}$$

- such that topic score ω_{ik} is like PC score δ_{ik} and
- θ_k topic probabilities are like rotations F_k.
- The distinction is that the multinomial in implies a different loss function (from a multinomial) rather than the sums of squared errors that PCA minimizes.
- Note that we condition on document length here so that topics are driven by relative rather than absolute term usage.

LDA: Example

```r
library(textir)

library(maptpx) # for the topics function

data(we8there)

# you need to convert from a Matrix to a `slam' simple_triplet_matrix
x <- as.simple_triplet_matrix(we8thereCounts)

# to fit, just give it the counts, number of `topics' K, and any other args
tpc <- topics(x,K=10)
```

```
##
## Estimating on a 6166 document collection.
## Fitting the 10 topic model.
## log posterior increase: 4441.8, 461.4, 101.5, 57.4, 51, 19.2, 26.2, 15.3, 15.4, 11.7, 6.7, 12.2, 8, 10.1,
4.8, 5.3, 3.2, 6.6, 2.8, 7, 3.6, 3.9, 6.7, 5.5, 8.6, 5, 11, 10.3, 12, 7.9, 12.1, 9, 8.8, 13.9, 8.6, 7.3, 6.1,
4.9, 4.3, 12, 11.1, 8.7, 3.2, 2.8, 5.1, 1.9, 2.6, 2.4, 4.9, 2.9, 1.5, 2.5, 4.7, 1.7, 0.9, 1.4, 0.7, 2.5, 2.2,
1.7, 1, 1.3, 1.5, 2, 0.8, 1.7, 0.5, 0.2, 0.5, 0.6, 0.9, 3.9, 0.5, 0.6, 0.4, 0.2, 0.8, 0.2, 1.4, 0.3, 0.5, 0.6, done.
```

LDA: Example

- Choosing the number of topics

```
# If you supply a vector of topic sizes, it uses a Bayes factor to choose
# (BF is like exp(-BIC), so you choose the biggest BF)
# the algo stops if BF drops twice in a row
tpcs <- topics(x,K=5*(1:5), verb=1) # it chooses 10 topics
```

```
##
## Estimating on a 6166 document collection.
## Fit and Bayes Factor Estimation for K = 5 ... 25
## log posterior increase: 2853.9, 327.1, 85.3, 36.7, 25.9, 19.9, 13.8, 11.6, 9.6, 11.4, 20.3, 7.1, ..., done.
## log BF( 5 ) = 79521.94
## log posterior increase: 4626.7, 197.4, 53, 24.9, 19, 9.3, 7.4, 4.6, 5.2, 3.4, 2.3, 1.7, 0.8, ..., done.
## log BF( 10 ) = 87157.28
## log posterior increase: 3445, 170.2, 49.8, 23.6, 14.1, 31.4, 16.2, 4.8, 6.6, 5.5, 1.9, 5.9, ..., done.
## log BF( 15 ) = 3334.33
## log posterior increase: 2327.1, 139.8, 39.5, 16.7, 20.1, 5.3, 4.5, 3, 3.4, 2.9, 4.4, 1.8, ..., done.
## log BF( 20 ) = -66254.44
```

Topic Models: Example

- Interpretation

```r
# summary prints the top `n' words for each topic,
# under ordering by `topic over aggregate' lift:
#    the topic word prob over marginal word prob.
summary(tpcs, n=10)
```

```
##
## Top 10 phrases by topic-over-null term lift (and usage %):
##
## [1] 'food great', 'great food', 'great servic', 'veri good', 'food veri', ... (14.6)
## [2] 'high recommend', 'italian food', 'best italian', 'mexican food', ...  (11.6)
## [3] 'over minut', 'never go', 'go back', 'flag down', 'anoth minut', ...  (10.4)
## [4] 'enough share', 'open daili', 'highlight menu', 'until pm', ...  (10.4)
## [5] 'never return', 'one worst', 'don wast', 'wast time', ...  (9.4)
## [6] 'good work', 'best kept', 'out world', 'great experi', ... (9.1)
## [7] 'thai food', 'veri pleasant', 'ice cream', 'breakfast lunch', ...  (9)
## [8] 'take out', 'best bbq', 'can get', 'pork sandwich', 'home cook', ...  (9)
## [9] 'food good', 'food place', 'chees steak', 'good select', 'food pretti',...  (8.7)
## [10] 'wasn whole', 'came chip', 'got littl', 'over drink', 'took seat',...  (7.8)
##
## Log Bayes factor and estimated dispersion, by number of topics:
##
##               5        10       15        20
## logBF 79521.94 87157.28 3334.33 -66254.44
## Disp      7.09     4.96    3.95      3.33
##
## Selected the K = 10 topic model
```

Topic Models: Example

```r
# alternatively, you can look at words ordered by simple in-topic prob
## the topic-term probability matrix is called 'theta',
## and each column is a topic
## we can use these to rank terms by probability within topics
rownames(tpcs$theta)[order(tpcs$theta[,1], decreasing=TRUE)[1:10]]
```

```
##  [1] "veri good"    "great food"   "food great"   "great place"  "veri friend"
##  [6] "veri nice"    "good food"    "great servic" "food excel"   "food servic"
```

```r
rownames(tpcs$theta)[order(tpcs$theta[,2], decreasing=TRUE)[1:10]]
```

```
##  [1] "dine experi"     "high recommend"  "wait staff"      "wine list"
##  [5] "mexican food"    "italian food"    "italian restaur" "fine dine"
##  [9] "staff friend"    "make feel"
```

Volvemos en 15 mins con R

R para ML

photo from https://www.dailydot.com/parsec/batman-1966-labels-tumblr-twitter-vine/