

Lecture 29:  
PCA (cont.) and Latent Dirichlet Allocation  
Big Data and Machine Learning for Applied Economics  
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

November 24, 2020

# Announcements

- ▶ Problem Set 4 is posted
- ▶ Course Perception Survey <http://bit.ly/encuesta-cursos-uniandes>
- ▶ Final Exam Date will be Friday Dec 11 8am to Sunday 13 8am
- ▶ Remember to turn in your project proposal by December 7

# Recap: Text as Data

- ▶ Text as Data: Tokenization and Bag of Words Representation
- ▶ Tokenization Demo
- ▶ Text Regression
- ▶ Text Regression: Example
- ▶ Topic Models: PCA

# Agenda

## 1 Topic Models

- PCA
  - Theory
  - Factor Computation
  - Factor Interpretation
- Latent Dirichlet Allocation
  - LDA: Example

## 2 Review & Next Steps

## 3 Further Readings

# Topic Models

- ▶ Text is super high dimensional
- ▶ Some times unsupervised 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 review 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 too loud not too 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  
## 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,

```
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

```
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

$$X = (x_1, x_2, \dots, x_K)_{N \times K} \quad (1)$$

- ▶ Factor:

$$F = X\delta \quad \delta \in K \quad (2)$$

- ▶ Idea: summarize the  $K$  variables in a single ( $F$ ).
- ▶ Vocab: the coefficients of  $\delta$  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  $\lambda$  such that  $Ax = \lambda x$  for a vector  $x_{m \times 1}$ ,
  - ▶ if  $x \neq 0$ , then  $\lambda$  is an eigenvalue of  $A$ .
  - ▶ and a vector  $x$  is an eigenvector of  $A$  corresponding to the eigenvalue  $\lambda$ .
- ▶  $A_{m \times m}$  with eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_m$ , then:

$$\text{tr}(A) = \sum_{i=1}^m \lambda_i \quad (3)$$

$$\det(A) = \prod_{i=1}^m \lambda_i \quad (4)$$

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

# Factors via main components

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

# 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

$$\max_{\delta} \delta'\Sigma\delta \tag{5}$$

► It is obvious that the solution is to bring  $\delta$  to infinity.

## Factors via main components

- ▶ Let's "fix" the problem by normalizing  $\delta$

$$\begin{aligned} \max_{\delta} \quad & \delta' \Sigma \delta \\ \text{subject to} \quad & \delta' \delta = 1 \end{aligned} \tag{6}$$

- ▶ Let us call the solution to this problem  $\delta^*$ .
- ▶  $F^* = X\delta^*$  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:

$$\max_{\delta} \delta' \Sigma \delta \quad \text{s.t.} \quad \delta' \delta = 1$$

- ▶ Setting up the Lagrangian

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

- ▶ CPO

$$\Sigma \delta = \lambda \delta \tag{7}$$

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

$$\delta \Sigma \delta = \lambda \tag{8}$$

- ▶ In order to maximize  $\delta \Sigma \delta$  we must choose  $\lambda$  equal to the maximum eigenvalue of  $\Sigma$  and  $\delta$  is the corresponding eigenvector.

# Factors is unsupervised learning

- ▶ Recall that
  - ▶ In regression we had

$$y = X\beta + u \quad (9)$$

- ▶ We minimized the MSE

$$\min \sum (y_i - \hat{y})^2 \quad (10)$$

- ▶ 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

$$X_{n \times k} \quad (11)$$

- ▶ We end with

$$F_{n \times 1}^* = X\delta^* \quad (12)$$

## q main components

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

$$\max_{\delta_2} \delta_2' \Sigma \delta_2 \quad (13)$$

subject to

$$\delta_2' \delta_2 = 1$$

and

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

- ▶  $F_2^* = X\delta_2^*$  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  $\lambda_1, \dots, \lambda_K$  be the eigenvalues of  $\Sigma = V(X)$ , ordered from highest to lowest, and  $p_1, \dots, p_K$  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  $\Sigma$ ).
- ▶ Let  $F_j = X\delta_j, j = 1, \dots, K$  be the  $j$ -th principal component. It's easy to see that

$$V(F_j) = \delta_j' \Sigma \delta_j = p_j' P \Lambda P p_j = \lambda_j \quad (14)$$

(the variance of the  $j$ -th principal component is the  $j$ -th ordered eigenvalue of  $\Sigma$ ).

## Relative importance of factors

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

$$\text{trace}(\Sigma) = \text{trace}(P\Lambda P') = \text{trace}(PP'\Lambda) = \sum_{j=1}^K \lambda_j = \sum_{j=1}^K V(F_j) \quad (15)$$

- ▶ Then

$$\frac{\lambda_k}{\sum_{j=1}^K \lambda_j} \quad (16)$$

- ▶ measures the relative importance of the  $j$ th 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:

$$\text{trace}(\Sigma) = \sum_{j=1}^K V(F_j) = K \quad (17)$$

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

$$\sum_{j=1}^K \lambda_j = K \quad (18)$$

- ▶ 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.

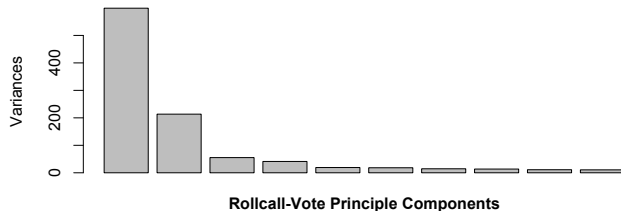
# Factor Interpretation: Example

## ► 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<sup>th</sup>)
- 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<sup>th</sup> house
- ▶ Each PC is  $F_s = X\delta_s$

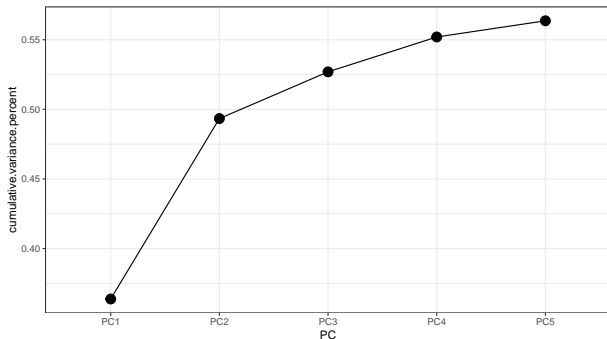


- ▶ Huge drop in variance from 1<sup>st</sup> to 2<sup>nd</sup> and 2<sup>nd</sup> to 3<sup>rd</sup> PC.
- ▶ Poli-Sci holds that PC1 is usually enough to explain congress. 2nd component has been important twice: 1860's and 1960's.



# Factor Interpretation

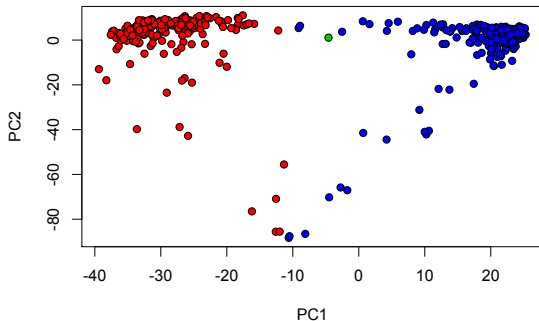
- ▶ Vote components in the 111<sup>th</sup> house
- ▶ Each PC is  $F_s = X\delta_s$



- ▶ Huge drop in variance from 1<sup>st</sup> to 2<sup>nd</sup> and 2<sup>nd</sup> to 3<sup>rd</sup> 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<sup>th</sup>** 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
```

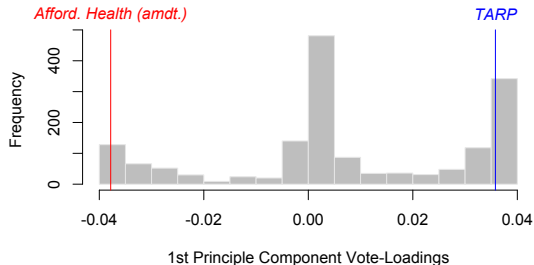
## social issues? immigration? no clear pattern

```
> 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 Republican 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  $\delta_2$  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,  $\theta_k$  with probability  $\theta_{kj}$  for each word  $j$ .
  - ▶ Then draw the word according to the probabilities encoded in  $\theta_k$ .
- ▶ After doing this over and over for each word in the document, you have proportion  $\omega_{i1}$  from topic 1,  $\omega_{i2}$  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:

$$x_i \sim MN(\omega_{i1}\theta_1 + \dots + \omega_{iK}\theta_K, m_i) \quad (19)$$

- ▶ 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:

$$E(x_i) = \delta_{i1}F_1 + \cdots + \delta_{iK}F_K \quad (20)$$

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

$$E(x_i) = \omega_{i1}\theta_1 + \cdots + \omega_{iK}\theta_K \quad (21)$$

- ▶ such that topic score  $\omega_{ik}$  is like PC score  $\delta_{ik}$  and
- ▶  $\theta_k$  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

```
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

```
# 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

```
# 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"
```

```
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"
```

# Review & Next Steps

- ▶ Topic Models
- ▶ PCA Theory
- ▶ Factor Computation Factor Interpretation
- ▶ Latent Dirichlet Allocation LDA: Example
  
- ▶ Next class: More on text as data
  
- ▶ Questions? Questions about software?

## Further Readings

- ▶ Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.
- ▶ James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.
- ▶ Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.