[7] Big Data: Clustering

Matt Taddy, University of Chicago Booth School of Business

`faculty.chicagobooth.edu/matt.taddy/teaching`

Clustering via Mixture Models

You've seen lots of models for $[y \mid \mathbf{x}]$ (and $[y \mid d, \mathbf{x}]$, etc).

Today is about models for $\mathbf{x}$.

We'll work with mixtures: assume that each $\mathbf{x}_i$ is drawn from one of K different *mixture components*, $\mathrm{p}_k(\mathbf{x})$.

Even if the individual components are very simple, their *mixture* can yield all sorts of complicated distributions

$$\mathrm{p}(\mathbf{x}) = \pi_1 \mathrm{p}_1(\mathbf{x}) + \ldots \pi_K \mathrm{p}_K(\mathbf{x})$$

Two algorithms:

- K-means with Normal $\mathrm{p}_k(\mathbf{x})$
- topic models with Multinomial $\mathrm{p}_k(\mathbf{x})$

Supervision

The goal in everything we do has been *Dimension Reduction*.

DR: move from high dimensional $\mathbf{x}$ to low-D summaries.

Dimension reduction can be supervised or unsupervised.

Supervised: Regression and classification

HD $\mathbf{x}$ is projected through β into 1D $\hat{y}$

Outside info (y) supervises how you simplify $\mathbf{x}$.

Unsupervised: Mixture and Factor Models

$\mathbf{x}$ is modeled as built from a small number of components.

You're finding the simplest representation of $\mathbf{x}$ alone.

We always want the same things: low deviance, without overfit.

Clustering: unsupervised dimension reduction

Group observations into similar 'clusters',
and understand the rules behind this clustering.

- Demographic Clusters: Soccer moms, NASCAR dads.
- Consumption Clusters: Jazz listeners, classic rock fans.
- Industry Clusters: Competitor groups, supply chains.

The fundamental model of clustering is a mixture:
observations are random draws from K distributions,
each with different average characteristics.

Top down Collaborative Filtering: group individuals into clusters, and model average behavior for each.

The K-means Mixture Model

Suppose you have K possible means for each observed $\mathbf{x}_i$:

$$\mathbb{E}[\mathbf{x}_i \mid k_i] = \boldsymbol{\mu}_{k_i}, \quad \text{where} \quad k_i \in \{1 \ldots K\}$$

e.g., if $k_i = 1$, $\mathbf{x}_i$ is from cluster 1: $\mathbb{E}[x_{i1}] = \mu_{11}$ ... $\mathbb{E}[x_{ip}] = \mu_{1p}$.

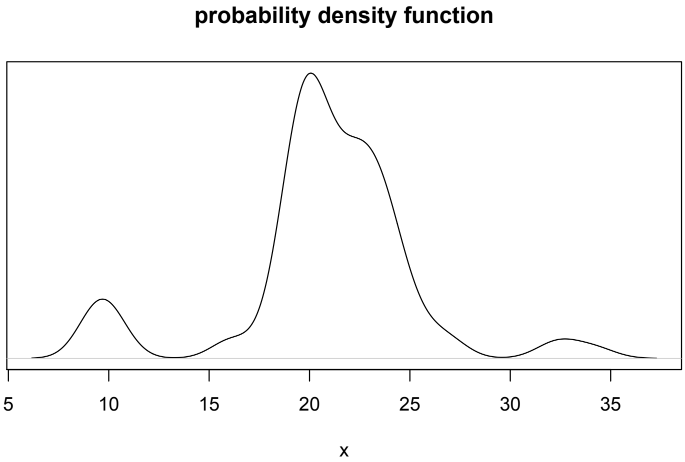
Then you have a mixture model.

For new $\mathbf{x}$ with unknown k,

$$\mathbb{E}[\mathbf{x}] = \mathrm{p}(k = 1)\boldsymbol{\mu}_1 + \ldots + \mathrm{p}(k = K)\boldsymbol{\mu}_K$$

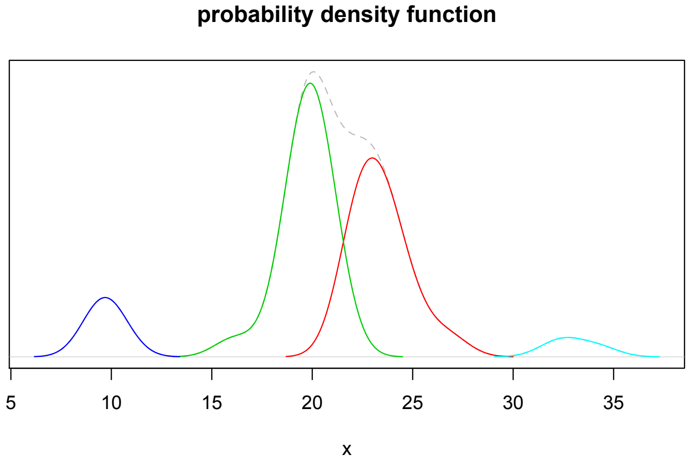
DR: Given $\boldsymbol{\mu}_k$'s, you discuss data in terms of K different types, rather than trying to imagine all possible values for each $\mathbf{x}$.

Mixture Model: without knowing membership '*k*'



The *marginal* density has multiple modes; one for each μ_k.

Mixture Model: breaking into K components



Here, we have $K = 4$ different cluster centers. Should it be 5?

K-means

K-means algorithm clusters data by fitting a mixture model.

Suppose data $\mathbf{x}_1 \dots \mathbf{x}_n$ comes from K clusters.

If you know membership k_i for each x_i, then estimate

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:k_i=k} \mathbf{x}_i$$

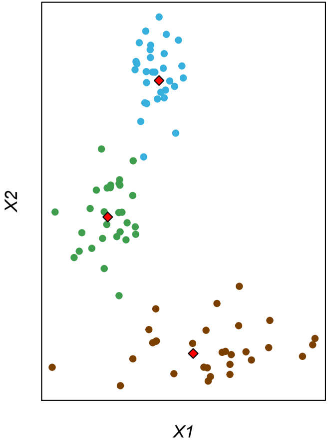
where $\{i : k_i = k\}$ are the n_k observations in group k.

K-means finds $\boldsymbol{k} = k_1 \dots k_n$ to minimize the sum-of-squares

$$\sum_{k=1} \sum_{i:k_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^2$$

Mixture deviance: total sums of squares within each cluster.

Give K-means the x_i's, and it gives you back the k_i's



This assigns a cluster to each x_i, and implies the group means $\hat{\mu}_k$.

The algorithm starts at random $\boldsymbol{k}$, and changes k_i's until the sum of squares stops improving.

Solution depends on start location. Try multiple, take the best answer.

Choosing K: For most applications, everything is descriptive. So try a few and use clusters that make sense to you.

kmeans(x, centers, nstart)

Clusters `x` (numeric!) into centers groups using `nstart` starts.

```
> grp = kmeans(x=mydata, centers=3, nstart=10)

K-means clustering with 3 clusters of sizes 28, 31, 31
Cluster means:
           x           y
1  1.0691704 -0.99099545
2 -0.2309448 -0.04499839
3  0.4987361  1.01209098

Clustering vector:
 [1] 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
[39] 1 1 3 3 3 3 3 3 3 3 3 3 2 2 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 12.332683  4.911522  3.142067
 (between_SS / total_SS =  80.5 %)
```

`grp$cluster` holds cluster assignments for each observation.

Scaling for K-means

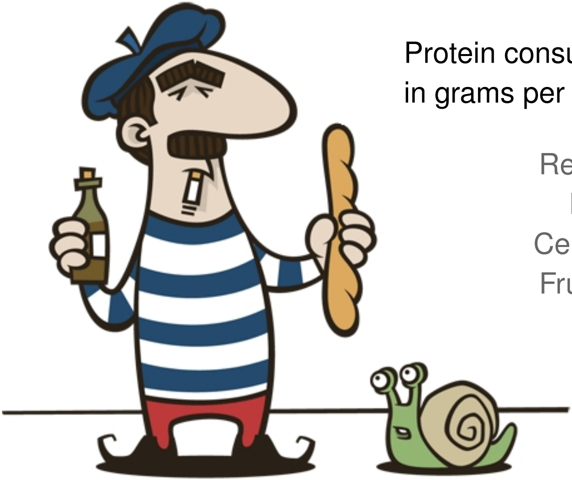
The algorithm minimizes total [squared] distance from center, *summed across all dimensions of* $\mathbf{x}$.

Scale matters: if you replace x_j with $2x_j$, that dimension counts twice as much in determining distance from center (and will have more influence on cluster membership).

Standard solution is to standardize:

$$\text{cluster on scaled } \tilde{x}_j = \frac{x_{ij} - \bar{x}_j}{\mathrm{sd}(x_j)}$$

Then the centers μ_{jk} are interpreted as *standard deviations from marginal average*.



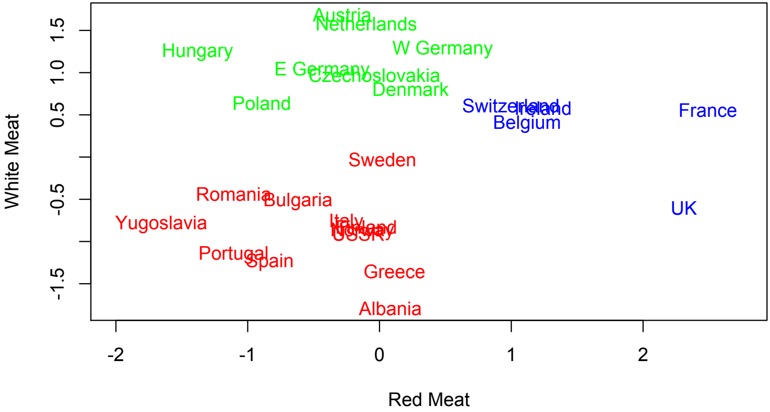
Clustering Europe by Food

Protein consumption by country, in grams per person per day for

Red and White Meat
Eggs, Milk, Fish
Cereals, Starch, Nuts
Fruit and Vegetables

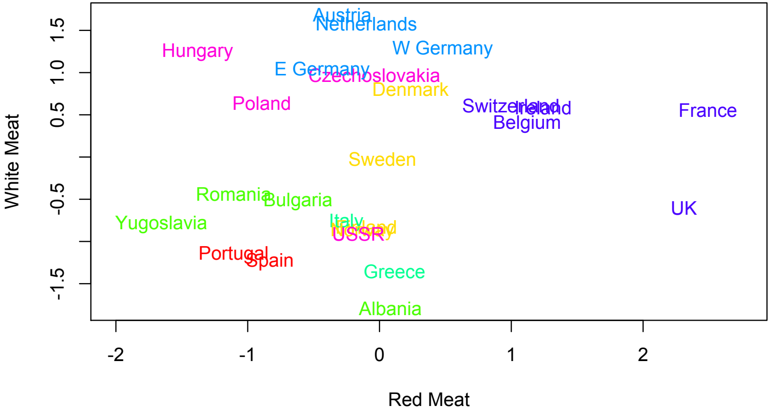
See `protein.R`.

3-means clustering on Red vs White meat consumption



Consumption is in units of standard deviation from the mean.

7-means clustering on all nine protein types



Plotting the red vs white plane, but *clustering on all variables.*

Wine Clustering

Today is all about food!

As a bigger data example, `wine.csv` contains data on 11 chemical properties of *vino verde* wine from northern Portugal.

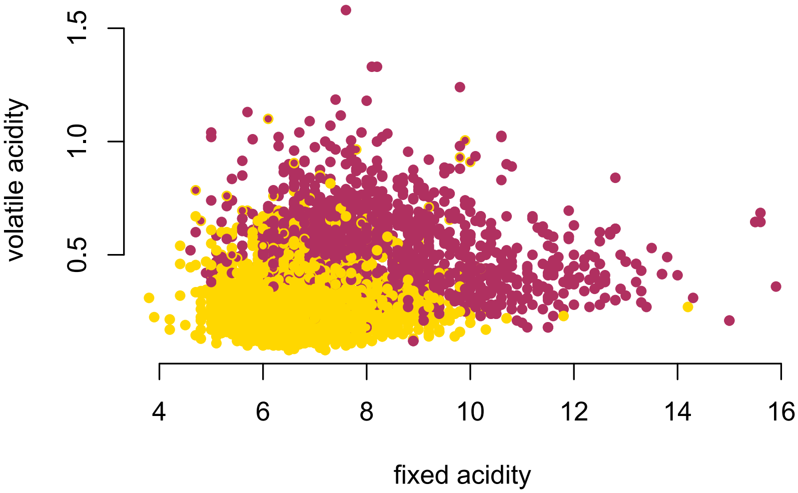
We have chemistry for 6500 bottles (1600 red, 4900 white), along with average quality rating (out of 10) by ‘expert’ tasters.

If you fit a 2-mean mixture, you get what you might expect

```
> tapply(wine$color,km$cluster,table)
      $`1`              $`2`
        red  white        red  white
         24   4830       1575     68
```

The two clusters are red vs white wine.

In 2D slices of **x**, we see clear red v white discrimination.



Point border is true color, body is cluster membership.

Choosing K

1st order advice: most often clustering is an exploration exercise, so choose the K that makes the most sense to you.

But, we can apply data-based model building here:

1. Enumerate models for $k_1 < k_2 \dots k_K$.
2. Use a selection tool to choose the best model for new $\mathbf{x}$.

Step one is easy. Step two is a tougher.

For example, for CV you'd want to have high OOS $\mathrm{p}_{k_i}(\mathbf{x}_i)$.

But you don't know k_i! This is a *latent* variable.

There's no ground truth like y to compare against.

AIC and BIC for K-means

We *can* use IC to select K.

- Deviance is (as always) D=-2logLHD,
  which for K-means is the total sum of squares (slide 8).
- *df* is the number of μ_{kj}: $K \times p$. (where p is dimension of $\mathbf{x}$)

Then our usual AICc and BIC formulas apply.

I've added these in `kIC.R` your convenience.

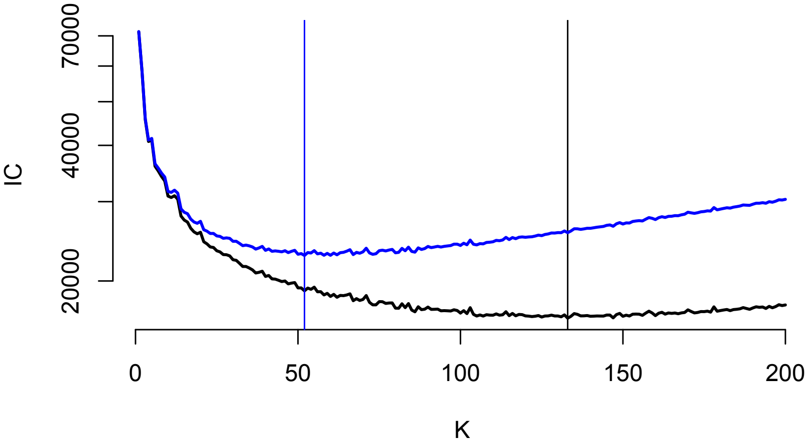
Beware: the assumptions behind both AICc and BIC calculation are only roughly true for K-means, and often not even that.

These tools are lower quality here than in regression.

I (and others) tend to trust BIC most for mixtures.

But you're often better off just using descriptive intuition.

BIC and AICc for wine K-means



BIC likes $K \approx 50$, AICc likes 130.

Both are way more complicated than is useful.

Cluster Regression

Once use of *unsupervised* clustering is
to throw the results into a *supervised* regression model.

For example, we can use the wine cluster memberships as a factor variable to predict wine quality (this is equivalent to just predicting quality with the average for each cluster).

If the dominant sources of variation in $\mathbf{x}$ are related to y, this can be a good way to work. Otherwise its not.

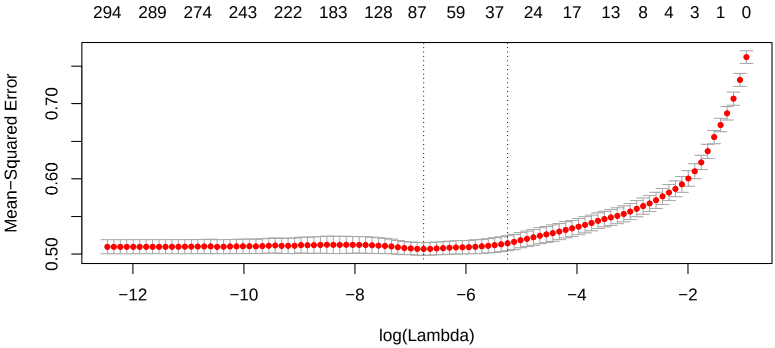
The clusters all have around the same average quality

```
> tapply(wine$quality,kfit[[k]]$cluster,mean)
 5.4 6.0 5.0 5.4 5.4 6.6 5.7 5.4 6.3 5.5
 6.1 5.8 5.7 6.3 6.3 6.1 5.2 6.5 5.9 5.5
 5.3 6.1 6.2 5.4 5.1 5.9 5.5 5.9 5.5 5.3
```

so the strategy wouldn't work here.

This isn't the same as there being no predictive power in **x**.

Regression onto 3-way chemistry interactions gets us an OOS R^2 of around 33% when predicting quality.



It's just that wine 'quality' has a weak signal, and most of the variation in **x** is driven by other factors (e.g., grape color).

Re-visiting the K-means model

In minimizing sums-of-squares, K-means targets the model

$$\mathrm{p}_k(\mathbf{x}) = \prod_j \mathrm{N}(x_j \mid \mu_{kj}, \sigma^2)$$

$\Rightarrow$ independence across dimensions (no multicollinearity) and uniform variance (same σ for all j, which is why scale matters).

Despite being a silly model, this tends to do a decent job of clustering when $\mathbf{x}$ consists of *continuous* variables.

It does a worse job for $\mathbf{x}$ made up of dummies or counts.

Restaurant reviews from we8there.com

2978 bigrams from 6260 reviews (average of 90 words) with 5-star *overall*, *atmosphere*, *food*, *service*, and *value* ratings.

Great Service: Waffle House #1258, Bossier City LA

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 we all think a Waffle House is. Keep up the good work.

[5: 5555]

Terrible Service: Sartin's Seafood, Nassau Bay TX

Had a very rude waitress and the manager wasn't nice either. [1: 1115]

Clustering Text

Often, low-D structure underlies text

- happy/sad, document purpose, topic subject.

The **x** for text are *counts* of text *tokens*.
A token can be a word, bigram (pair of words), etc.

We can try to transform **x** to look like something K-means would work with (i.e., something that looks built out of normals).

`we8there.R` fits K-means to standardized token proportions x_{ij}/m_i, where $m_i = \sum_i x_{ij}$ are the *document totals*.

Looking at the big $\hat{\mu}_{jk}$ (i.e., phrases j with big proportions in cluster k), it comes up with some decent interpretable factors.

Topic Models

Although it 'kinda works', K-means for text is far from ideal.
Instead, use a mixture of p_k that are appropriate for count data.

A multinomial mixture:

- For each word, you pick a 'topic' K.
- This topic has probability θ_{kj} on each word j.
- You draw a random word according to $\boldsymbol{\theta}_k$

After doing this over and over for each word in your document, you've got proportion ω_{i1} from topic 1, ω_{i2} from topic 2, etc.

The full vector of words for document $\mathbf{x}_i$ then has distribution

$$\mathbf{x}_i \sim \mathrm{MN}(\omega_{i1}\boldsymbol{\theta}_1 + \ldots + \omega_{iK}\boldsymbol{\theta}_K, m_i)$$

a multinomial with probabilities $\sum_k \omega_{ik}\boldsymbol{\theta}_k$ *and total* $m_i = \sum_j x_{ij}$.

Topic Models: clustering for text

Each document ($\mathbf{x}_i$) is drawn from a multinomial with probabilities that are a mixture of topics $\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K$.

$$\mathbf{x}_i \sim \mathrm{MN}(\omega_1\boldsymbol{\theta}_1 + \ldots + \omega_K\boldsymbol{\theta}_K, m)$$

These θ's are topics of phrase-probabilities: $\sum_{j=1}^{p} \theta_{kj} = 1$.
e.g., a hotdog topic has high probability on mustard, relish, ...

The doc weights $\boldsymbol{\omega}_i$ are also probabilities: $\sum_{k=1}^{K} \omega_{ik} = 1$.
e.g., a hotdog stand review has big ω_{ki} on the hotdog topic.

This is subtly different from K-means: instead of each *document i* being from a cluster, now each *word* is from a different topic and the document is a mixture of topics.

Fitting topic models in R

`maptpx` package has a `topics` function; see `?topics`.

```
tpc <- topics(x,K=10,tol=10) # 10 topic model
```

Fitting a topic model is computationally very difficult.

Just like in K-means, we need only roughly estimate $\hat{\boldsymbol{\theta}}_k$ (analogous to μ_k) to get a decent clustering.

So Big Data implementations will implement approximate (stochastic) deviance minimization.

Plenty of other packages out there; note that topic modeling is also called LDA (latent Dirichlet allocation) if you're exploring.

Selecting the number of topics

Choosing K here is same as in K-means: this is exploratory analysis, so just choose what you want for story-telling.

But, you can use BIC: if you give `topics` a vector for `K`, it incrementally grows and stops when the BIC keeps increasing. It reports `log Bayes Factor`, which is like `-BIC`.

The model returned is that for K with lowest BIC (highest BF).

```
Log Bayes factor and estimated dispersion:
             5       10      15        20
logBF 76020.73 90041.99 7651.02 -62745.37
Disp      7.19     5.06    4.02      3.43
```

Here, max BF selects $K = 10$ (don't worry about 'dispersion').

Interpreting topics

We build interpretation by looking at ‘top’ words for each topic.

You can order by lift: $\theta_{jk}/$`mean(X[,j]/m)`.

```
summary(tpcs)
Top 5 phrases by topic-over-null term lift (and usage %):
[1] food great, great food, veri good, food veri, veri nice (13.7)
[2] over minut, ask manag, flag down, speak manag, arriv after (11.6)
```

Sometimes straight word probabilities are more intuitive

```
rownames(tpcs$theta)[order(tpcs$theta[,1], decreasing=TRUE)[1:10]]
  veri good great food food great great place veri nice
  wait staff   good food    food excel   great servic place eat
rownames(tpcs$theta)[order(tpcs$theta[,2], decreasing=TRUE)[1:10]]
  go back came out tast like never go brought out
  wait minut  take order  minut later come out    drink order
```

Here, topic 1 looks ‘good’ and 2 looks ‘bad’.

Wordles!

You can use the `wordcloud` library to efficiently visualize.

```
wordcloud(row.names(theta), freq=theta[,2], min.freq=0.004)
```

Topic 1

Topic 2

In the language of wordles, `freq` controls word size.
I've made word size proportional to the in-topic probability.

Topic Regression

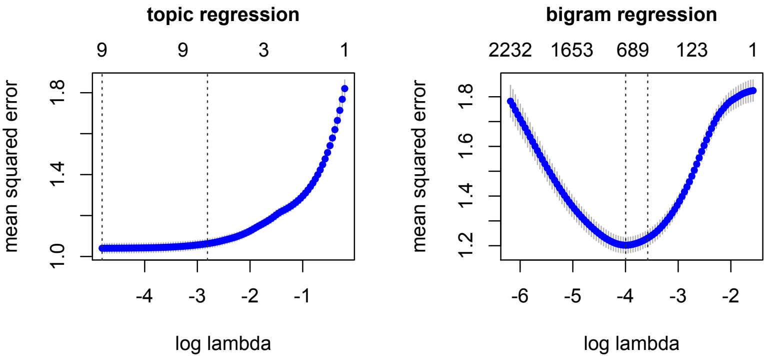
Just like with K-means, we can relate topics to other variables in a second-stage low-D regression.

Here, the topics looked motivated by quality.
Perhaps they'll be useful in predicting review rating?

```
stars <- we8thereRatings[,"Overall"]
tpcreg <- gamlr(tpcs$omega, stars)
# Effect stars from 10% increase in topic use
drop(coef(tpcreg))*0.1
     intercept       1       2       3       4
         0.414   0.075  -0.386   0.068   0.042
             5       6      7      8       9      10
         0.000   0.076  0.121  0.000  -0.134  -0.049
```

So, e.g., you drop an expected -.4 star for if an extra 10% of the review comes from topic 2 here (our negative topic from above).

Comparison regressing `stars` on bigram proportions x_{ij}/m_i.



The topic model does better than regression onto words!

Round up on unsupervised clustering

It is largely an exploratory analysis technique.

Don't be too fancy in your rules to choose K, because all that matters is that the model tells an intuitive story.

You can use clustering results as inputs for low-D regressions.

This is great if the dominant sources of variation in $\mathbf{x}$ are related to y (especially if you have more $\mathbf{x}_i$'s than y_i's).

But it's useless if y is not connected main drivers behind x,

as is common in big data!

Computational Roundup

Both K and topic modeling take a long time:

clustering uses massively high dimensional models ($K \times p$)

There are approximate distributed algorithms
(e.g., stochastic descent).

For really big data, just fit on subsamples:
since unsupervised clustering is driven by dominant
sources of variation, you shouldn't need all the data.

Speech in the 109th Congress

`textir` contains `congress109` data: counts for 1k phrases used by each of 529 members of the 109th US congress.
Load it with `data(congress109)`. See `?congress109`.

The counts are in `congress109Counts`.

We also have `congress109Ideology`, a data.frame containing some information about each speaker.

The includes some partisan metrics:

- party (Republican, Democrat, or Independent)
- `repshare`: share of constituents voting for Bush in 2004.
- Common Scores [cs1,cs2]: basically, the first two principal components of roll-call votes (next week!).

Homework due next week: congressional speech

[1] Fit K-means to speech text for K in 5,10,15,20,25.
Use BIC to choose the K and interpret the selected model.

[2] Fit a topic model for the speech counts. Use Bayes factors to choose the number of topics, and interpret your chosen model.

[3] Connect the unsupervised clusters to partisanship.

- tabulate party membership by K-means cluster.
  Are there any non-partisan topics?
- fit topic regressions for each of `party` and `repshare`.
  Compare to regression onto phrase percentages:

```
x<-100*congress109Counts/rowSums(congress109Counts)
```

No starter scipt; look at `we8there.R` and `wine.R`.