Counting words in Social Science

Matt Taddy, University of Chicago Booth School of Business

faculty.chicagobooth.edu/matt.taddy/research

What are ‘words’?

A self-contained meaningful token...

- Actual words: ‘all’, ‘world’, ‘stage’, ‘:-)’, ‘`#textdata`’.
- n-grams: ‘merely players’ (bi), ‘men and women’ (tri)
- complicated clauses: parts of speech, act-of-god.
- user selections on a website, domain ids in browser history

First step of analysis is to *parse* raw text into these tokens.

Then all we’ll do is count them.

The bag of words

Treat tokens for each doc as an i.i.d. sample.

Like drawing lottery balls (with replacement).

Document i is summarized by counts c_{ij} for tokens $j = 1...D$.

Implied probability model is a multinomial: $\mathbf{c}_i \sim \mathrm{MN}(\mathbf{q}_i, m_i)$

This is the state of the art.

Dumb but works: extra rules aren't worth their complexity.

Outline

Useful text statistics in social science.

- Brief and biased history
- Massive multinomial models
- Projection for prediction and inference.

Many examples and pictures, as little math as possible.

History: Text as data

Modern statistical treatment begins (?) with 1960s work, on author identification in the Federalist papers.

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 302 JUNE, 1963 *Volume 58*

INFERENCE IN AN AUTHORSHIP PROBLEM[1,2]

A comparative study of discrimination methods applied to the authorship of the disputed *Federalist* papers

FREDERICK MOSTELLER
Harvard University
and
Center for Advanced Study in the Behavioral Sciences
AND
DAVID L. WALLACE
University of Chicago

History: Text as data

M+W count words in papers by Hamilton and Madison,

TABLE 4.2. RATES PER THOUSAND FOR *also*, *an*, AND *because*

Word	Hamilton rate	Madison rate
also	.25	.50
an	6.00	4.50
because	.45	.50

then fit models for counts|author (essentially what I use today!), and use Bayes rule to predict authors|counts on disputed work.

$$\mathrm{p}(\text{Hamilton} \mid \text{text}) \approx \frac{\mathrm{p}(\text{text} \mid \text{Hamilton})}{\mathrm{p}(\text{text} \mid \text{Madison}) + \mathrm{p}(\text{text} \mid \text{Hamilton})}$$

Text as data

At the same time, statistical learning enters natural language processing. (same tools: counts, Bayesian discrimination).

1959 PROCEEDINGS OF THE EASTERN JOINT COMPUTER CONFERENCE 225

Pattern Recognition and Reading by Machine

W. W. BLEDSOE† AND I. BROWNING†

And since the 80s text-as-data in NLP has risen with stat ML.

- Author identification (spam filters!), sentiment prediction.
- Connecting queries and content: web search, ad targeting.
- Speech recognition, translation, stuff your phone can do.

Text as data in Social Science

There's been an explosion of interest from social scientists.

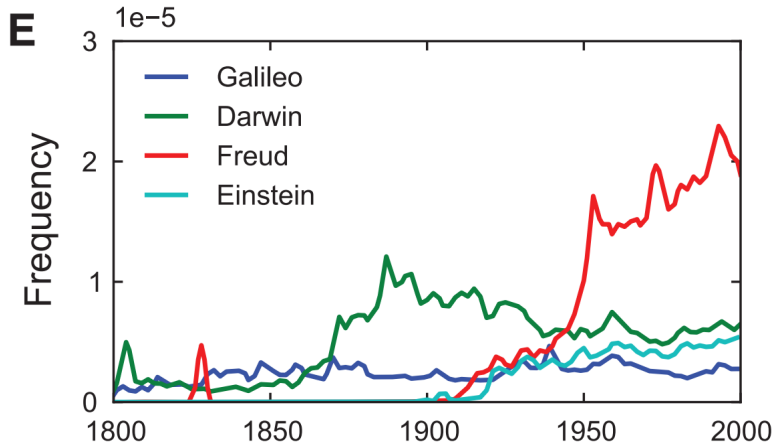
emperical evidence, quantification of social concepts...

Text as data in Social Science

There's been an explosion of interest from social scientists.
emperical evidence, quantification of social concepts...

Until very recently, one used pre-defined dictionaries.

Picking words: culturomics, Michel et al, Science 2011.



Psychosocial dictionaries, such as Harvard GI in *Tetlock 2007, Giving Content to Investor Sentiment* and others:

able, abundant, accept vs abandon, abrupt, absurd.

Text as data in Social Science

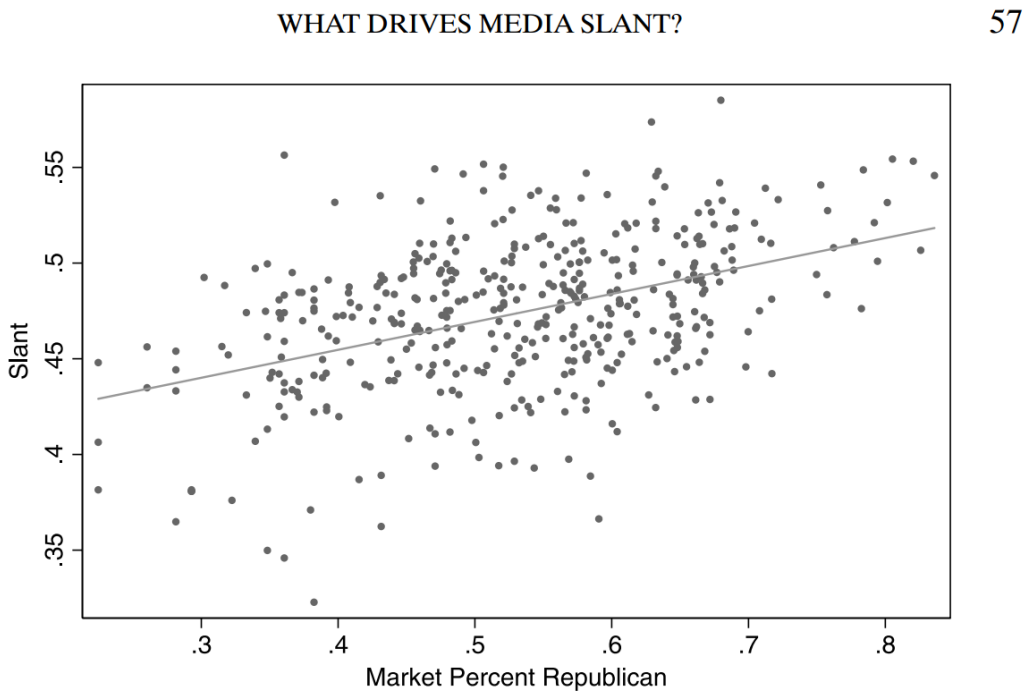
Techniques from stats and ML are beginning to filter through and researchers are estimating relationships *from the data*.

Text as data in Social Science

Techniques from stats and ML are beginning to filter through and researchers are estimating relationships *from the data*.

Gentzkow/Shapiro, Econometrica 2010: correlate speaker party with word counts, interpret fit and use for prediction.

$$\text{slant(speaker)} = \sum_w \text{count}_w(\text{speaker}) \times \text{cor}(\text{count}_w, \text{gop})$$

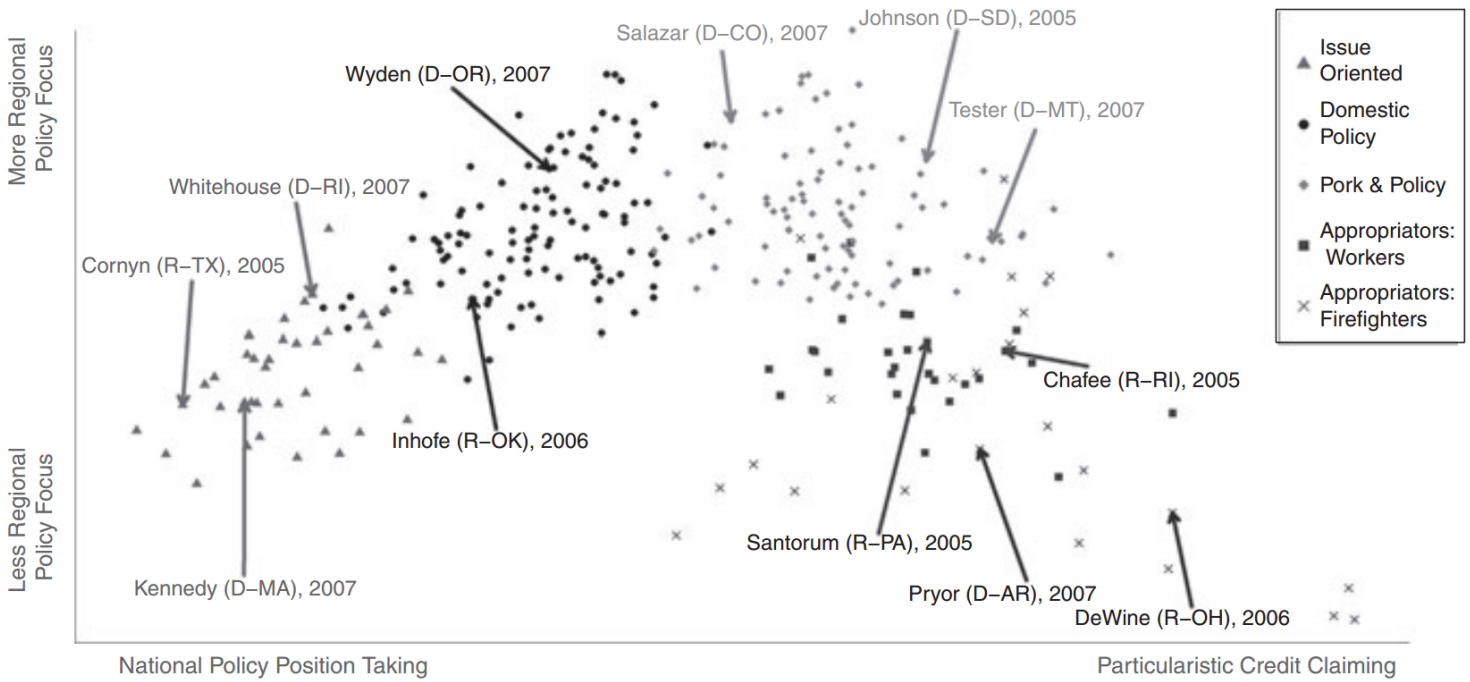


Text as data in Social Science

Techniques from stats and ML are beginning to filter through and researchers are estimating relationships *from the data*.

Grimmer AJPS 2013: fit latent topics in press releases (e.g., 'Iraq', 'Infastructure') then investigate who uses what topic.

FIGURE 1 A Typology of Home Styles in the U.S. Senate



Massive Multinomials

Much of what we want to do with text fits in familiar model.

Large response logistic regressions:

$\mathbf{c}_i \sim \mathrm{MN}(\mathbf{q}_i, m_i)$ with $q_{ij} = e^{\eta_{ij}} / \sum_l e^{\eta_{il}}$

$\eta_{ij} = \alpha_j + \mathbf{v}_i' \varphi_j$ is a 'log intensity'

You can think of it as expectation for $\log(c_{ij}/m_i)$.

$\mathbf{v}_i = \mathtt{gop}_i$ is related to Gentzkow/Shapiro slant.

Latent $\mathbf{v}_i$ is similar to the topic models of Blei, Grimmer.

I avoid latent $\mathbf{v}_i$, and instead use many fixed/random effects.

Example: Yelp

Public dataset of 220k reviews on 12k restaurants by 44k users.

C : Parse on whitespace, strip common suffixes (e.g., 's','ing'), and remove a small set of stopwords (e.g., 'and','the').

We end up with $D = 14$k tokens that occur in $> 0.01\%$ reviews.

V : Review, user, and business attributes ($K > 400$).

- # stars, # votes funny/useful/cool (f/u/c).
- age: t and t^2 in days since posting.
- vote/age interaction $[1, t, t^2] \times [f, u, c]$.
- user's review count, star average, vote totals.
- business location: 61 city effect dummies.
- business taxonomy: 333 nonexclusive classes.

yelp

Logistic regression model has $n \times D \approx 3$ billion log intensities,

$$\eta_{ij} = \alpha_j + \mathbf{v}_i' \boldsymbol{\varphi}_j = \alpha_j + \sum_k v_{ik} \varphi_{jk}$$

involving $D \times K \approx 6$ million regression coefficients.

yelp

Logistic regression model has $n \times D \approx 3$ billion log intensities,

$$\eta_{ij} = \alpha_j + \mathbf{v}_i' \boldsymbol{\varphi}_j = \alpha_j + \sum_k v_{ik} \varphi_{jk}$$

involving $D \times K \approx 6$ million regression coefficients.

Two key tools help us estimate a system of this size.

- Distribution: break the data into pieces and estimate on each simultaneously.
- Regularization: index and select from a range of simple $\rightarrow$ complex candidate models.

Distribution

With Big data, working in parallel is not enough.
You need to be distributed.

I approximate the MN likelihood with *independent* Poissons:

$$c_{ij} \sim \text{Po}(\ m_i e^{\eta_{ij}}\ )$$

$\Rightarrow$ once you have document totals m_i,
you can estimate $[\hat{\alpha}_j, \hat{\varphi}_j]$ for each token *independently*.
(distributed multinomial regression)

Counts for different tokens can live on different computers!
Massively scalable: build out not up.

Distributed Computing

Consider raw documents stored in a distributed file system (i.e., many different machines) such as HDFS or Amazon S3.

A MapReduce Algorithm

Map: parse documents to output lines `token id|count`.

Sort: All lines for given token 'j' go to the same machine.

Reduce: Do the Poisson regression for 'j' and output $[\hat{\alpha}_j, \hat{\boldsymbol{\varphi}}_j]$.

Distributed Computing

Consider raw documents stored in a distributed file system (i.e., many different machines) such as HDFS or Amazon S3.

A MapReduce Algorithm

Map: parse documents to output lines `token id|count`.

Sort: All lines for given token 'j' go to the same machine.

Reduce: Do the Poisson regression for 'j' and output $[\hat{\alpha}_j, \hat{\varphi}_j]$.

Reduce (RCC version):

- just write from each reducer to a file on scratch, then fire up midway and crunch through each file.

Regularization

To understand contemporary statistics, you have to understand regularization: depart from optimality to stabilize a system.
Common in engineering: I wouldn't fly on an optimal plane.

Consider an individual token regression $c \sim \alpha + \mathbf{v}'\boldsymbol{\varphi}$.

We minimize deviance

$$\min -\frac{1}{n} \log \mathsf{LHD}(\alpha, \boldsymbol{\varphi})$$

Regularization

To understand contemporary statistics, you have to understand regularization: depart from optimality to stabilize a system.
Common in engineering: I wouldn't fly on an optimal plane.

Consider an individual token regression $c \sim \alpha + \mathbf{v}'\boldsymbol{\varphi}$.

We minimize deviance plus a cost on the size of φ_k.

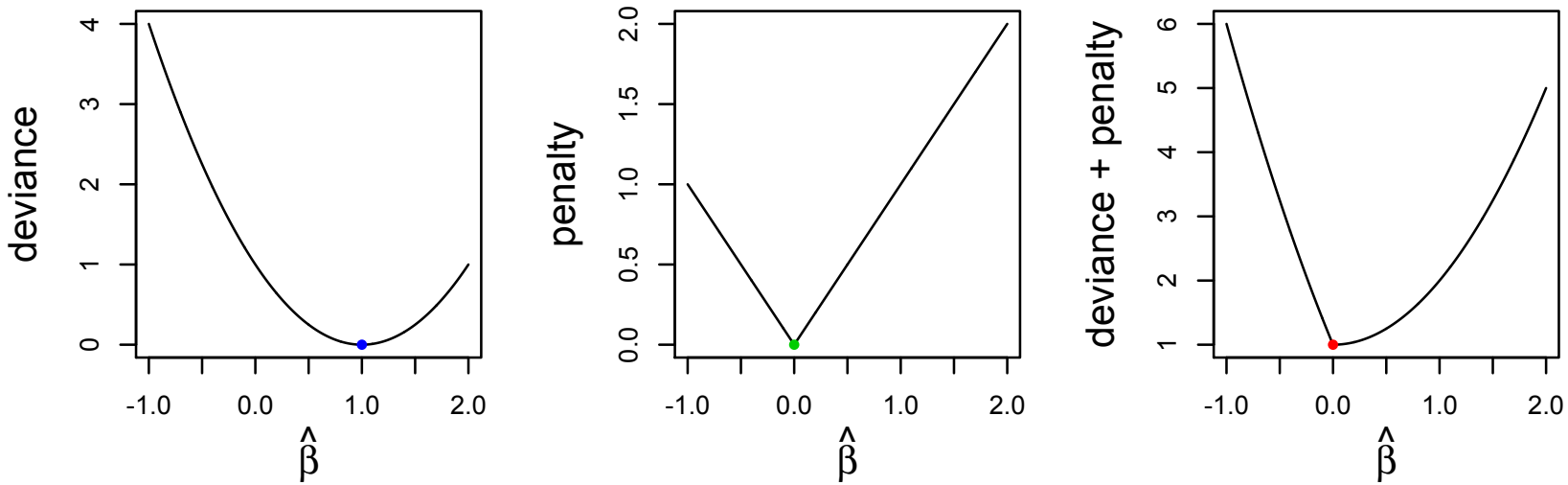
$$\min -\frac{1}{n}\log \mathrm{LHD}(\alpha, \boldsymbol{\varphi}) + \lambda \sum_k \omega_k |\varphi_k|$$

Fixed $\omega_k > 0$ is Tibshrani's 'lasso': the new least squares.

I work on versions where ω_k *adapts* to φ_k (gamma lasso).

Regularization

Since the penalty has a sharp spike at zero, we can get $\hat{\varphi}_k = 0$.



You are getting variable selection, without testing.

Regularization paths

Think of $\lambda > 0$ as a signal-to-noise filter: like squelch on a radio.

Path algorithms: start with really big penalty, so that $\hat{\varphi} = 0$, then gradually decrease λ and update estimates as you go.

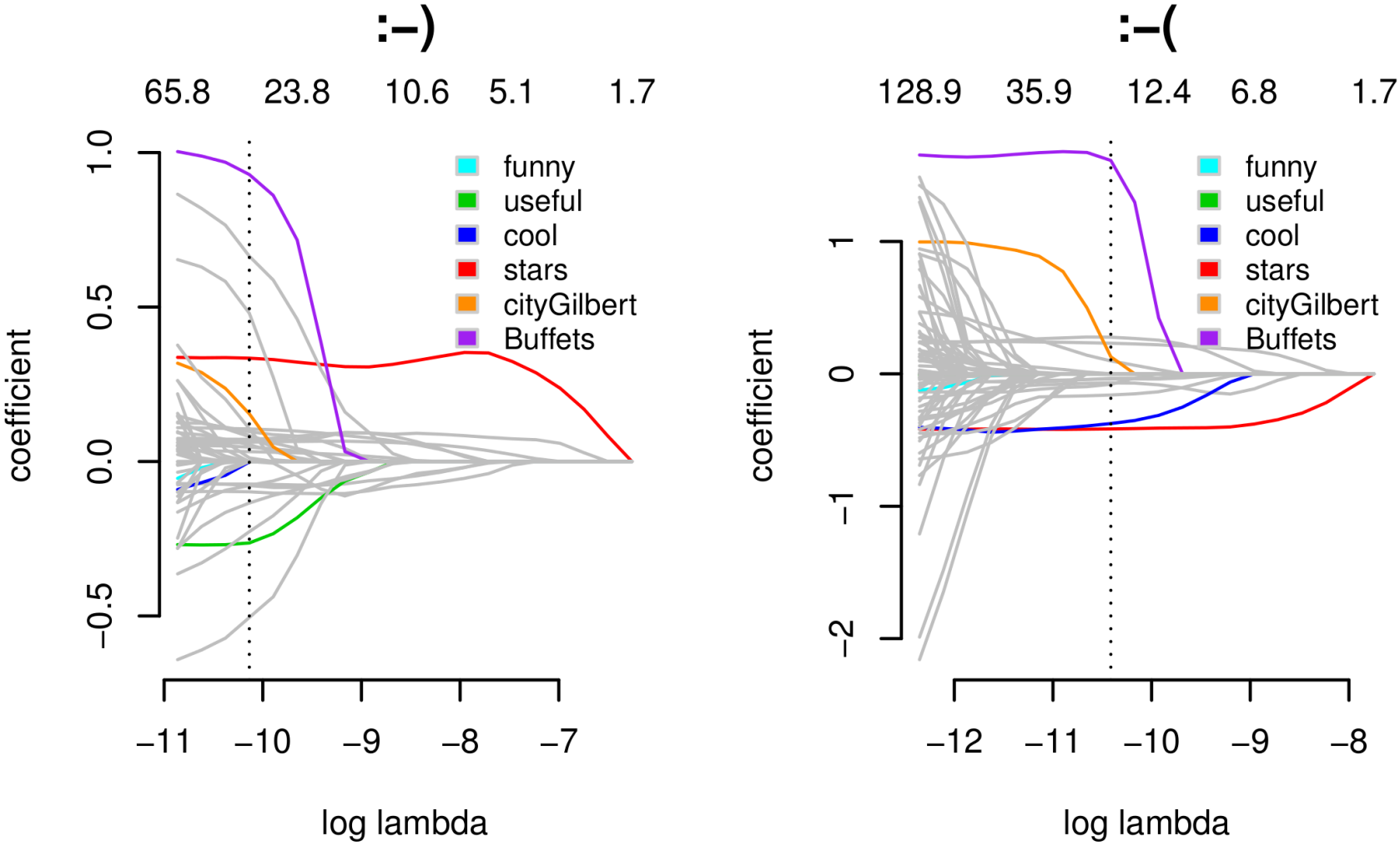
If the coefficients change little under small changes to λ, then a full path can take far less time then a single OLS fit.

Choose $[\hat{\alpha}, \hat{\varphi}]$ from the path using your preferred selection tool:

- CV: see empirically what predicts best out-of-sample.
- IC: approximate analytically what would predict best.

yelp: big multinomial regression

Paths of φ_{kj} for a couple tokens '`j`'. AICc selection is marked.



Only chosen $\hat{\varphi}_j$ are sent to head node, and $\approx$ 5% are nonzero.

yelp effects $e^{\varphi_{jk}}$

Biggest odds increase factor from an extra star:

awesome	amaz	excellent	fantastic	favorite	perfect
1.58	1.56	1.53	1.53	1.50	1.48
wonderful	love	deliciou	yum	yummy	alway
1.47	1.44	1.42	1.40	1.37	1.35

Biggest odds decrease factor from an extra star:

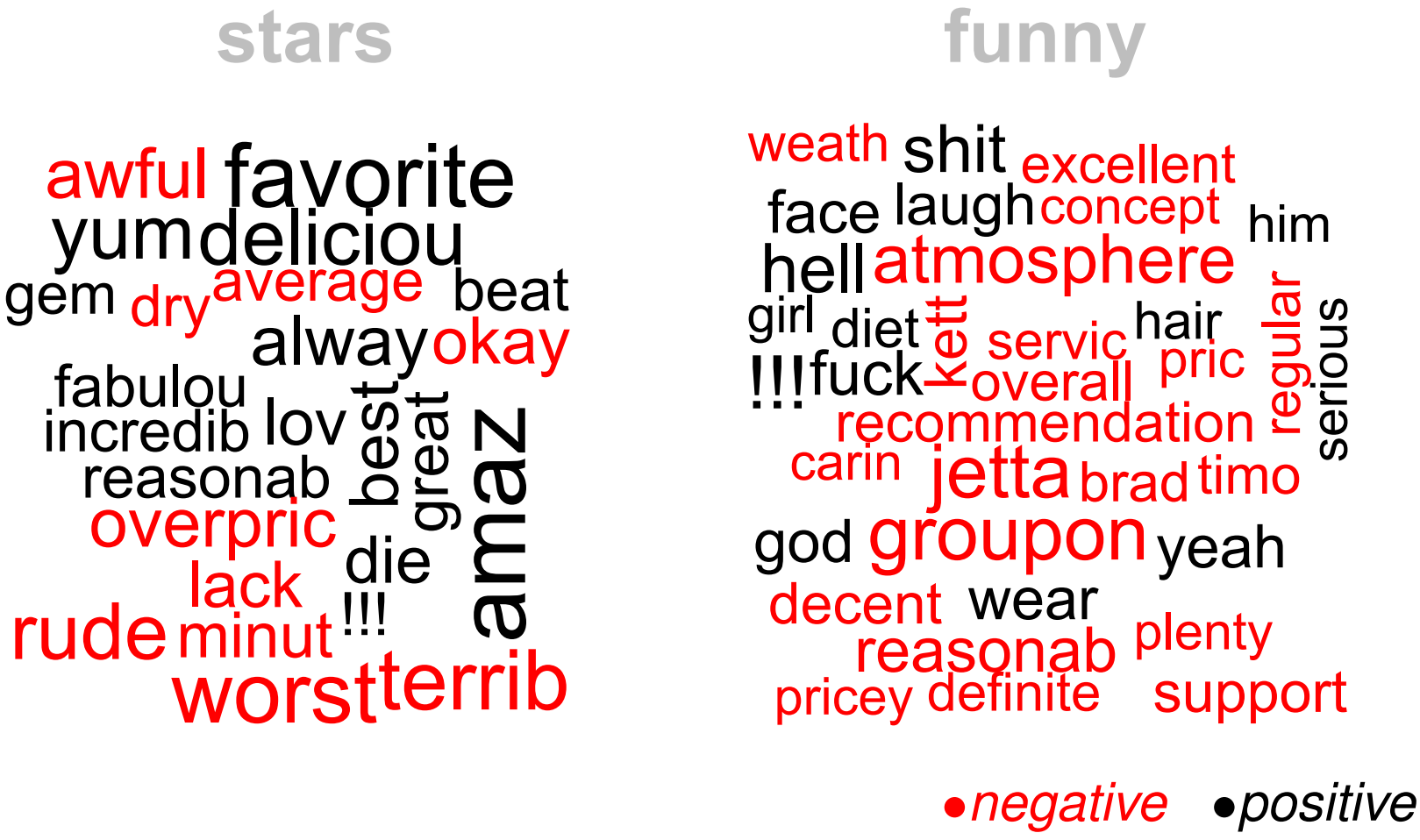
bland	worst	terrib	horrib	rude	mediocre
0.67	0.68	0.68	0.69	0.70	0.73
awful	overpric	poor	okay	lack	minut
0.74	0.74	0.75	0.76	0.76	0.76

Biggest odds increase from an extra funny vote:

hipst	!!!	hell	shit	fuck	yeah
1.21	1.14	1.13	1.12	1.11	1.11
mom	god	face	wear	laugh	diet
1.11	1.11	1.11	1.11	1.10	1.10

wordles

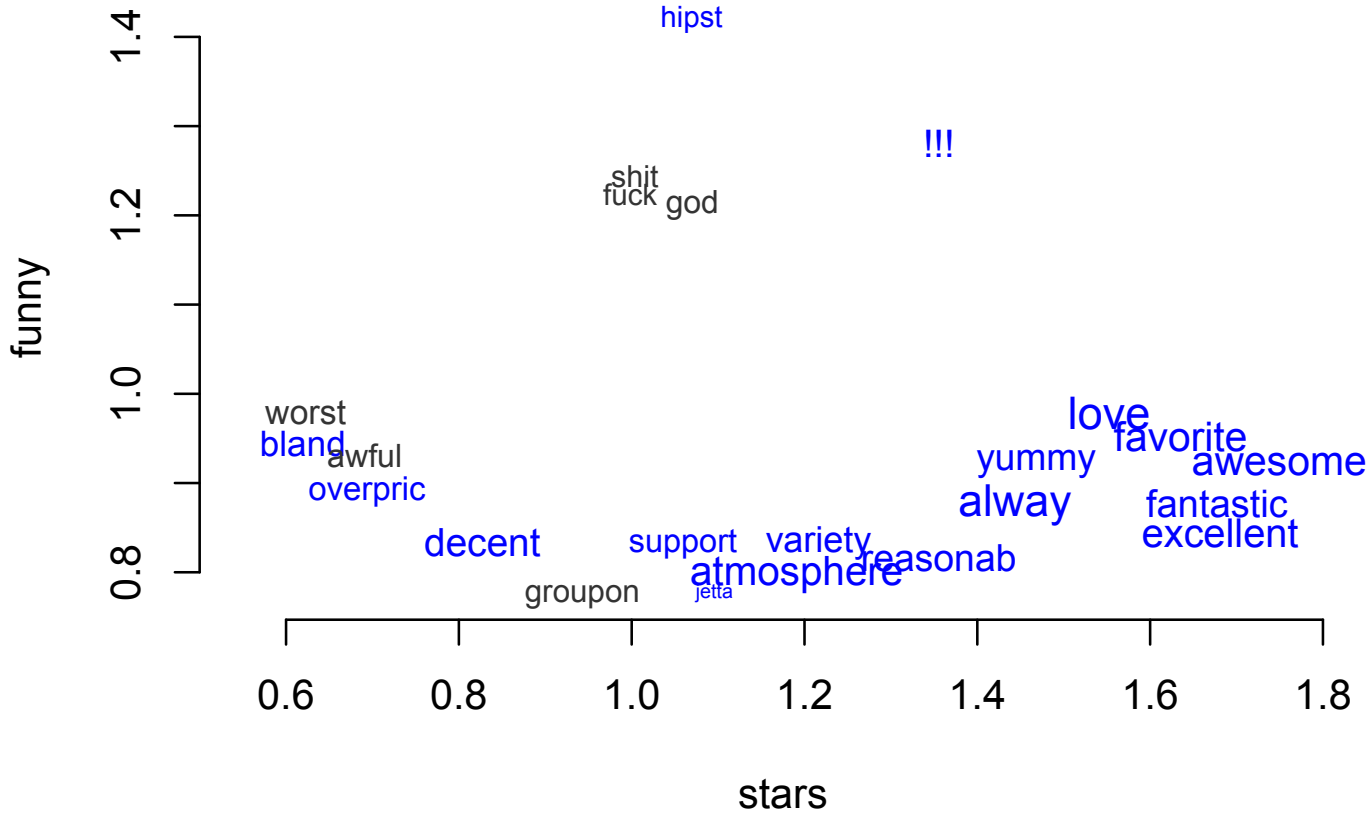
The simple printout is fine, but perhaps you prefer wordles?



Seen as a bit frivolous in CS: ‘what’s the spatial dimension?’

word scatters

Something like this is better in principle...



●*cool* ●*not cool* (size by freq)

But it takes work to make such plots legible.

Why?

We've fit a big model for how text depends on doc attributes.

Why?

We've fit a big model for how text depends on doc attributes.

Now we'll project through the model into summaries of interest.

- Prediction: project onto $\hat{y}$.
- Treatment effects: project onto $\hat{y}$ and treatment $\hat{d}$.
- Inference *about* language: project onto potential causes.

Sufficient Projection

Suppose a subset of **v** are special.

Prediction: v_k is better known as 'y', which you want to predict.

Treatment effects: **v** includes both response y and treatment d.

Call everything else '**u**', so that $\eta_{ij} = \alpha_j + \mathbf{v}_i'\boldsymbol{\varphi}_j + \mathbf{u}_i'\boldsymbol{\theta}_j$.

Sufficient Projection

Suppose a subset of $\mathbf{v}$ are special.

Prediction: v_k is better known as 'y', which you want to predict.

Treatment effects: $\mathbf{v}$ includes both response y and treatment d.

Call everything else '$\mathbf{u}$', so that $\eta_{ij} = \alpha_j + \mathbf{v}_i'\varphi_j + \mathbf{u}_i'\theta_j$.

It turns out that:

$$\mathbf{z}_i = [\varphi_1 \cdots \varphi_D]\mathbf{c}_i/m_i \;\Rightarrow\; \mathbf{v} \perp\!\!\!\perp \mathbf{c} \mid \mathbf{z}, \mathbf{u}, m$$

Such that $\mathbf{z}$ contains all information in $\mathbf{c}$ relevant to $\mathbf{v}$.

We call this a sufficient projection (SP).

Motivated by 'Inverse Regression' ideas; e.g. Cook 2007.

Prediction

We've modelled text | attributes, and we invert for prediction.

Consider predicting the number of f/u/c votes for a yelp review, to promote likely-to-please reviews without waiting for votes.

$$y = v_{\mathtt{u}} + v_{\mathtt{f}} + v_{\mathtt{c}}, \quad \text{with 3D SP } \mathbf{z} = [z_{\mathtt{u}}, z_{\mathtt{f}}, z_{\mathtt{c}}]$$

Fit forward regression $y \sim [\mathbf{u}, \mathbf{z}, m]$ via OLS or a fancy ML tool.
$y \sim [\mathbf{u}, \mathbf{c}]$ has $15k$ inputs, while $y \sim [\mathbf{u}, \mathbf{z}, m]$ has only 425.

Prediction

We've modelled text | attributes, and we invert for prediction.

Consider predicting the number of f/u/c votes for a yelp review, to promote likely-to-please reviews without waiting for votes.

$$y = v_{\mathtt{u}} + v_{\mathtt{f}} + v_{\mathtt{c}}, \quad \text{with 3D SP } \mathbf{z} = [z_{\mathtt{u}}, z_{\mathtt{f}}, z_{\mathtt{c}}]$$

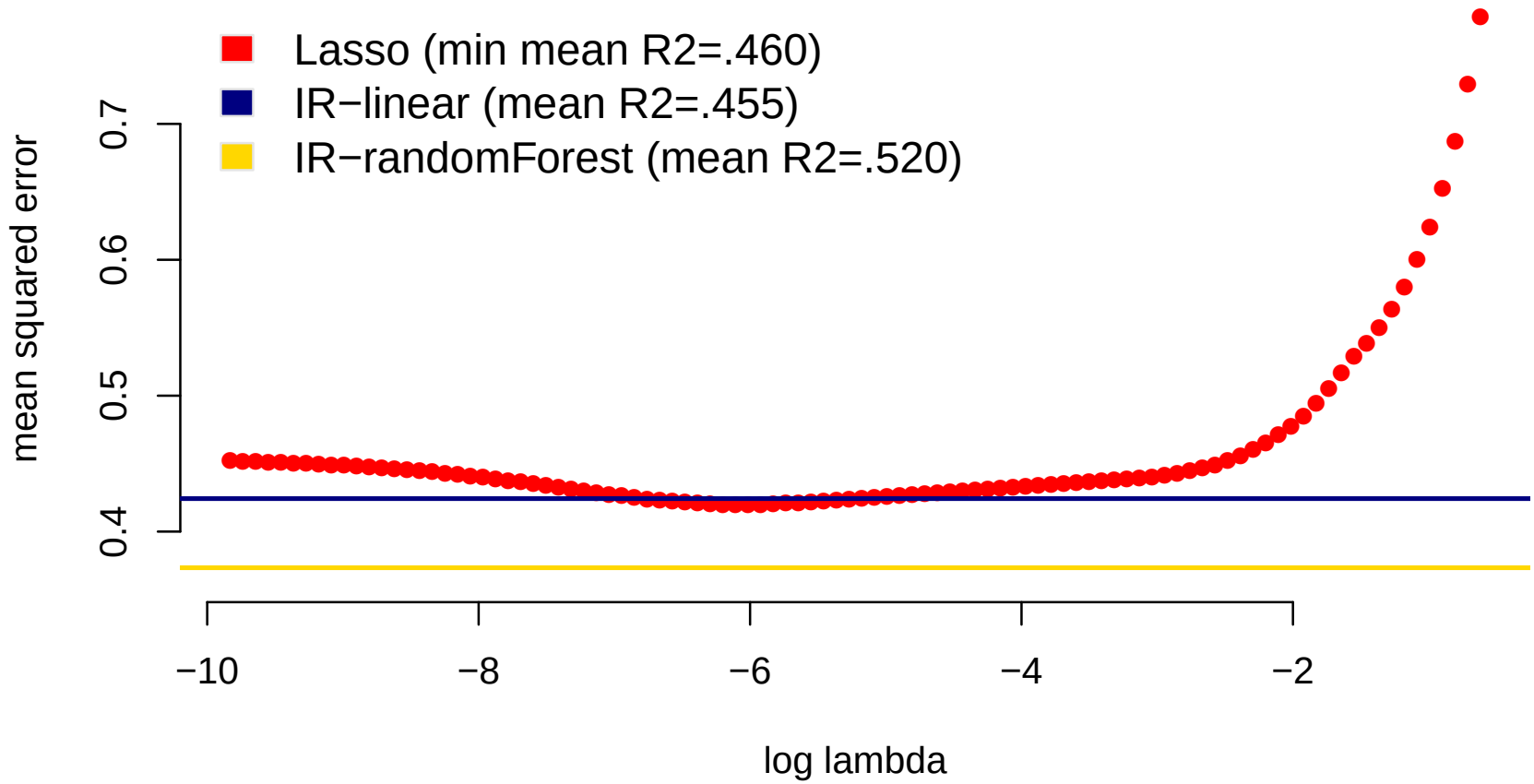
Fit forward regression $y \sim [\mathbf{u}, \mathbf{z}, m]$ via OLS or a fancy ML tool.
$y \sim [\mathbf{u}, \mathbf{c}]$ has $15k$ inputs, while $y \sim [\mathbf{u}, \mathbf{z}, m]$ has only 425.

Computational and statistical efficiency:

SP happens in distribution, $\mathrm{var}(\mathbf{z})$ drops with *amount of speech*,
nonparametric learners are fast and stable after SP,
fit misspecification in low-D, etc...
MNIR paper + rejoinder.

yelp vote prediction

20 fold CV: full 15k input lasso vs IR and forward regression.



IR routines used AIC to select λ penalties.

treatment effects

Prediction is cool, but we want to know *why* a review is popular.

e.g., maybe # of reviews (`usr.count`) builds reputation?

Just add z_d, the SP for d $(= v_{\texttt{usr.count}})$ to forward regression.

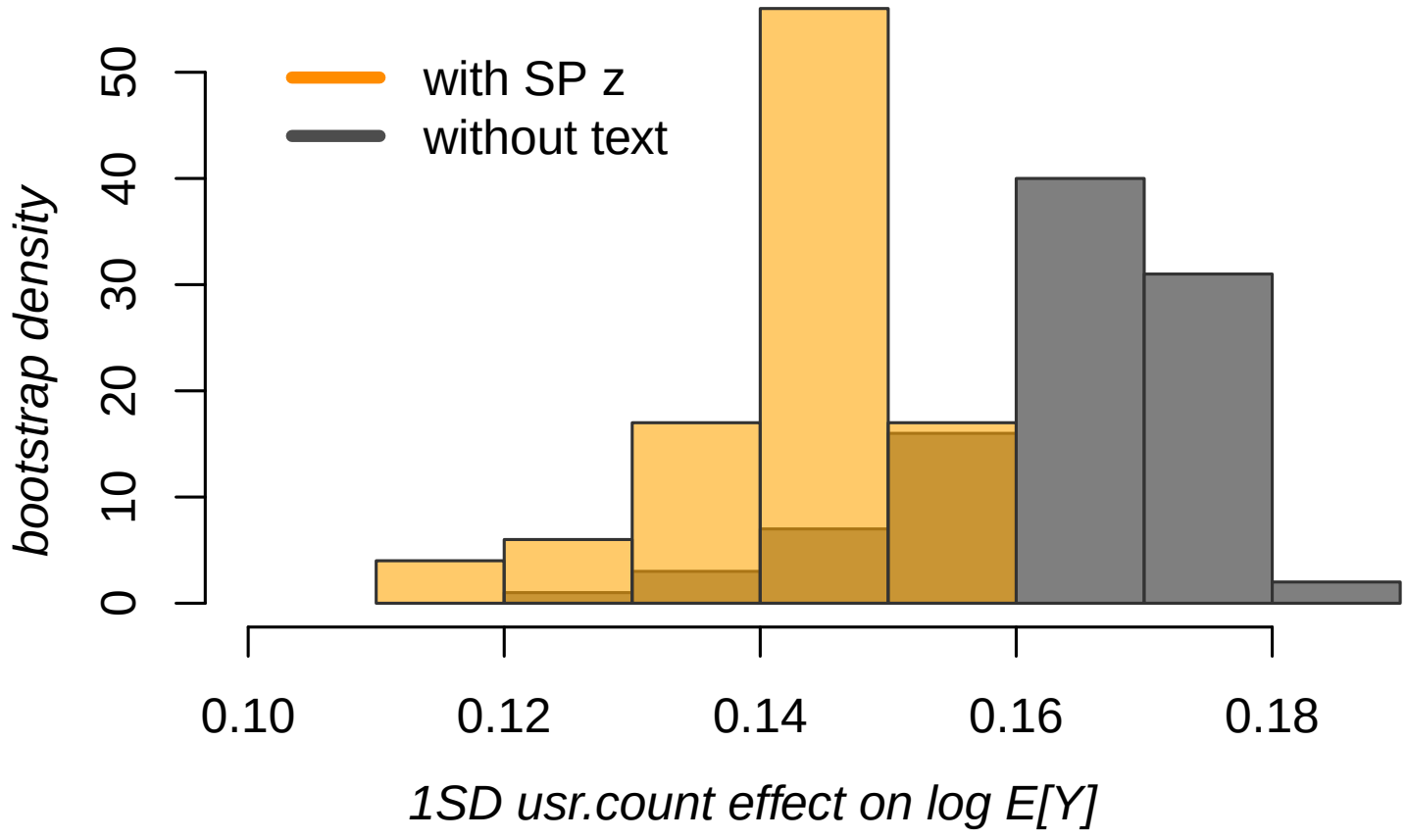
Then $\mathbf{z}$ is all info in $\mathbf{c}$ relevant to *either* 'treatment' or 'response'

$$[y, d] \perp\!\!\!\perp \mathbf{c} \mid \mathbf{z}, \mathbf{u}, m$$

and the estimated effect of d on y after controlling for $\mathbf{z}$
is free from any confounding information from word counts.

yelp review count and popularity

e.g., what is the 'veteran' reviewer effect on number of votes, after removing the influence of review content (word counts)?



This is treatment effect estimation under super-HD controls!

Inference about language

So far we've used HD models, but for low-D inference ($\hat{y}$, β_d).

To *causally* interpret coefficients on individual words, you need simultaneous inference on 10-100+ thousands of effects.

I've argued that its impossible without making big assumptions.

Inference about language

So far we've used HD models, but for low-D inference ($\hat{y}$, β_d).

To *causally* interpret coefficients on individual words, you need simultaneous inference on 10-100+ thousands of effects.

I've argued that its impossible without making big assumptions.

However, it is also a task very in-demand in social science, so I'm going to make some big assumptions and give it a try.

politext: partisanship of political rhetoric

Matt Gentzkow and Jesse Shapiro brought me the entire congressional record since 1873, with a simple question:

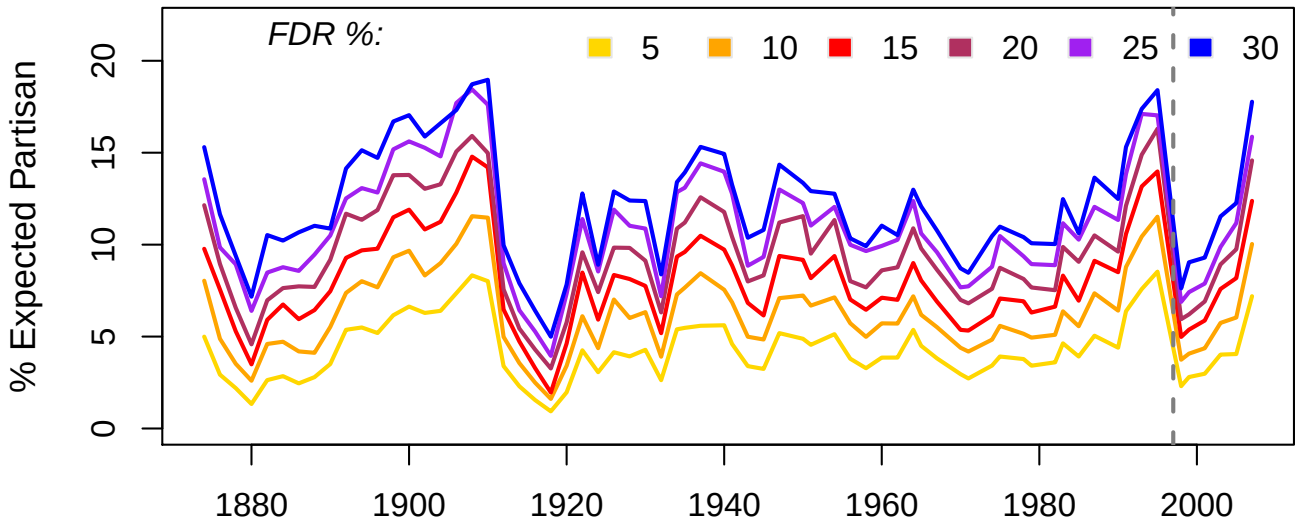
"How has the partisanship of rhetoric changed over time?"

politext: partisanship of political rhetoric

Matt Gentzkow and Jesse Shapiro brought me the entire congressional record since 1873, with a simple question:

"How has the partisanship of rhetoric changed over time?"

e.g., try to count 'significantly partisan words'.



What's driving it? Geographic diversity? Procedural language? Significance is driven by amount of speech: just plotting power?

Politext

Write down an MN-logit generative model for the text.

Each ‘document’ is a speaker in a specific congress.

$$\eta_{ij} = \alpha_j + \mathbf{v}_i'\boldsymbol{\theta}_j + \mathtt{gop}_i\varphi_j(t_i)$$

$\mathtt{gop}_i$ is a republican dummy.

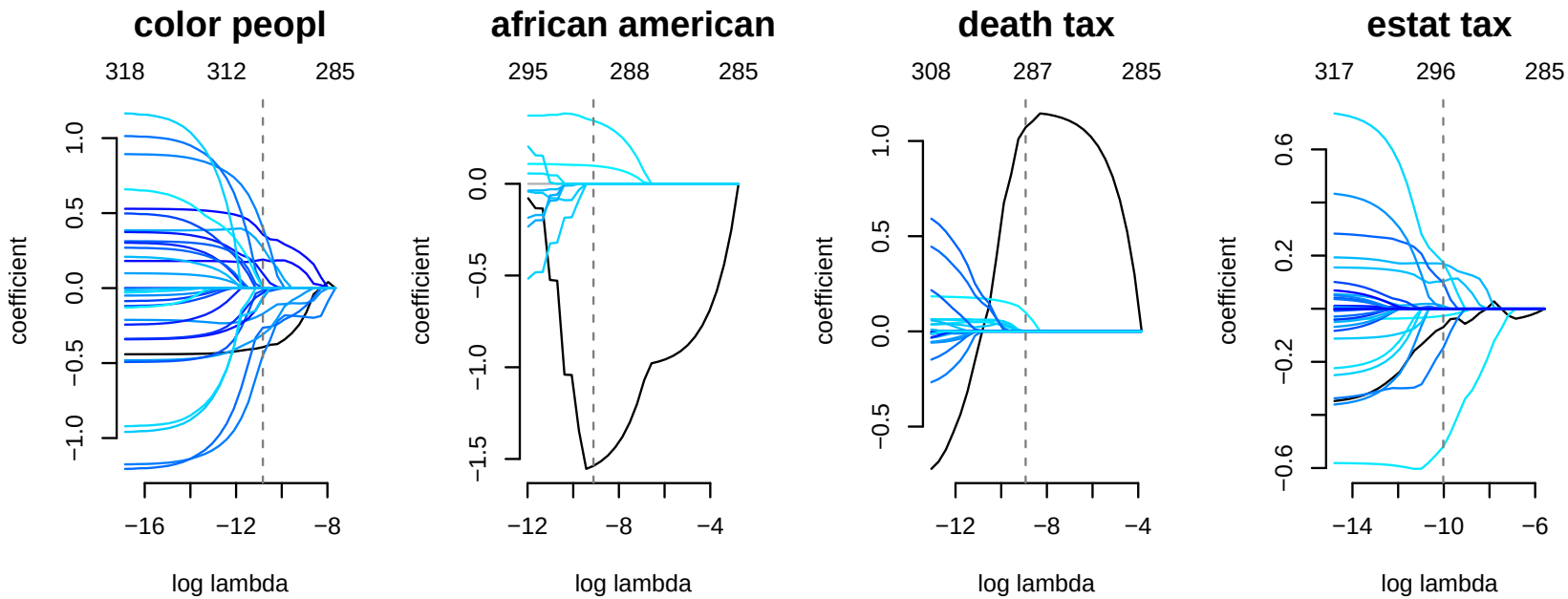
$\varphi_j(t_i)$ is a shrunk-towards-constant ‘partisan effect’
(change in time is heavily regularized).

v includes indicators for state, chamber, gender,
session, minor party, and majority membership.

Only $\boldsymbol{\Phi}$ is penalized in estimation, so all **v** effect is controlled for.

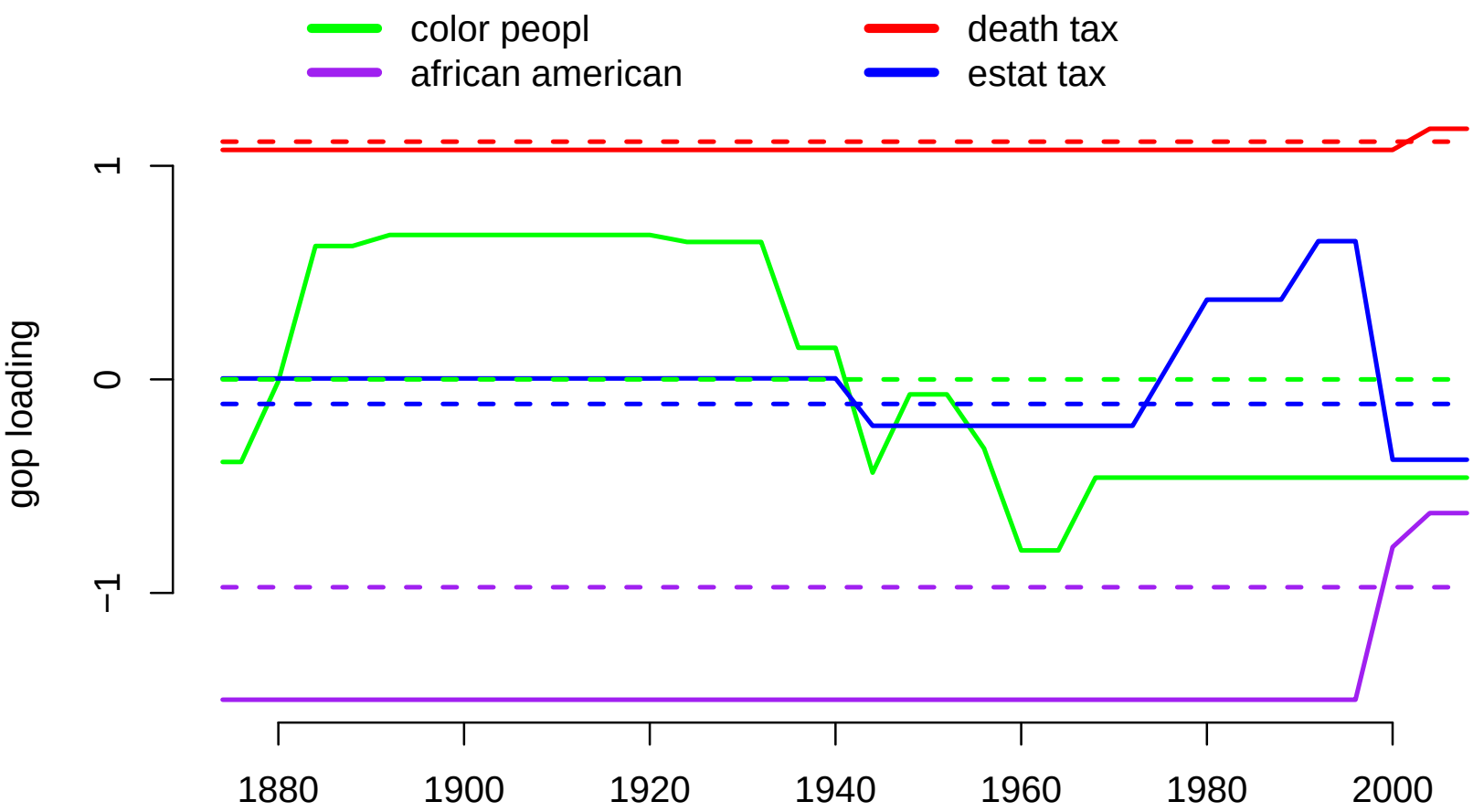
Politext

$n = 35k$, $D = 500k$ bi-grams used by at least 75 people (1%).



black φ^0 baseline, colored spline loadings ($\varphi(t)$ dynamics).

Politext



These are analogous to our yelp word list/doodle/scatters, but now the loadings ('partisan meaning') change in time.

Politext

We're ready to give an answer to the 'simple' question.

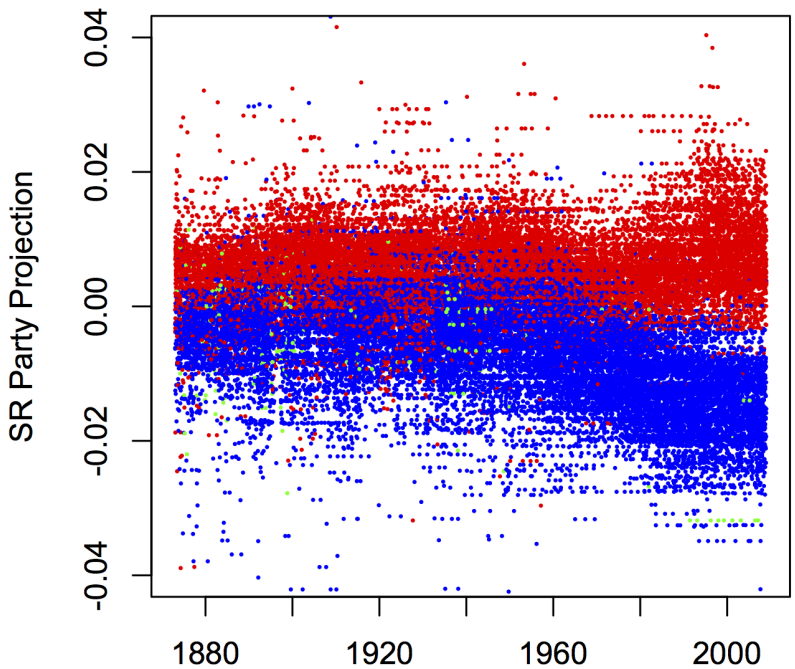
$z = \mathbf{c}'\varphi(t_i)/m_i$ contains the info in speech related to party *after* removing the effects of geography, time, gender, etc...

Politext

We're ready to give an answer to the 'simple' question.

$z = \mathbf{c}'\varphi(t_i)/m_i$ contains the info in speech related to party *after* removing the effects of geography, time, gender, etc...

We can plot the index for individual speakers in time.

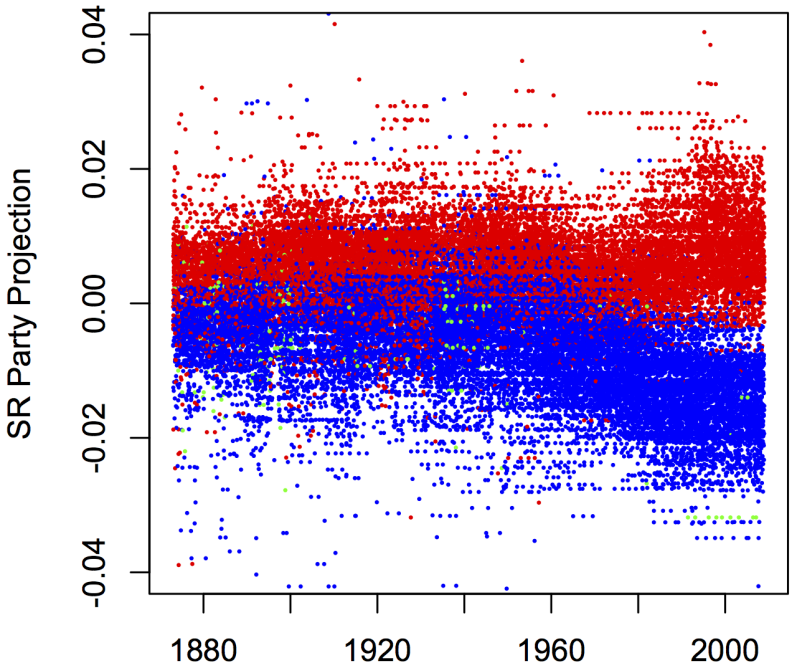


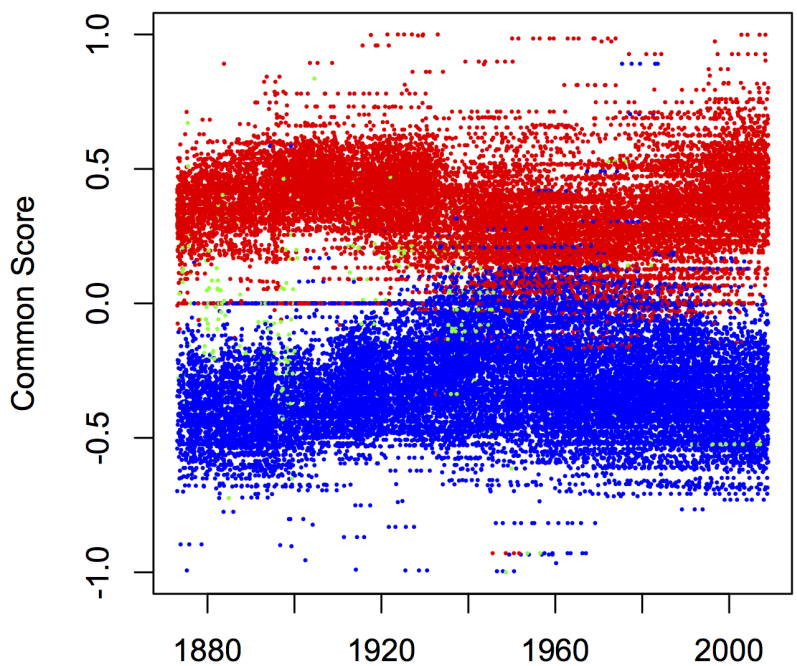
Politext

We're ready to give an answer to the 'simple' question.

$z = \mathbf{c}'\varphi(t_i)/m_i$ contains the info in speech related to party *after* removing the effects of geography, time, gender, etc...

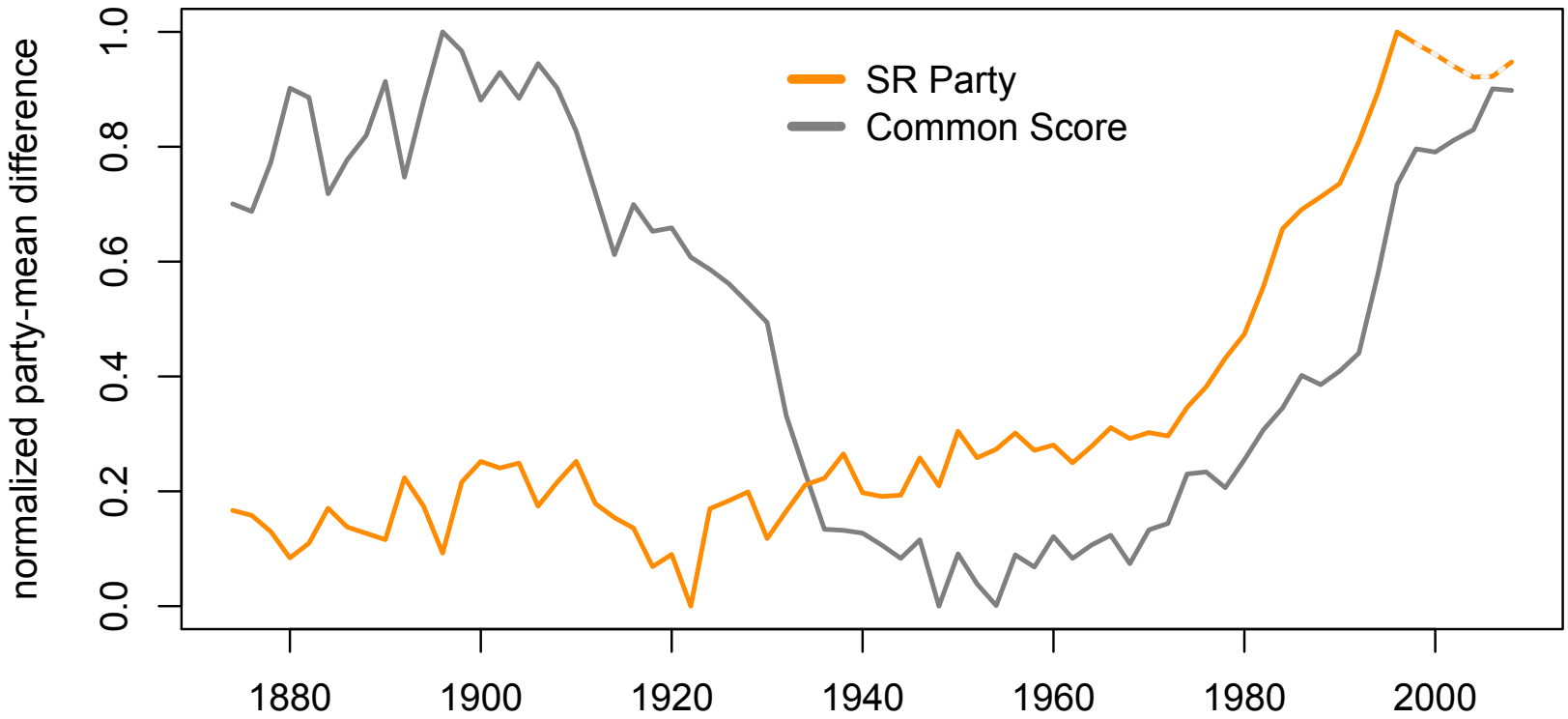
Compare to 'common score' ($\approx$ 1st factor of rollcall votes).





Politext

And finally track the spread between parties



Yes: partisanship of rhetoric has exploded since 1970.

(results also hold under constant `gop` phrase loadings).

The End

We've got the ability to estimate massive multinomial logits.

These yield an array of projections rich in text info.

Think carefully about what you want to measure.

Thanks!