Lecture 24: Superlearners & Text as Data

Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

November 4, 2021

Agenda

1. Announcements
2. Recap: Bagging, Forests, and Boosting
3. Superlearners
4. Text as Data
   - Tokenization
5. Review & Next Steps
6. Further Readings
7. Appendix: Text as Data Demo
   - Tokenization Demo
   - Text Regression: Example

Announcements

- Problem Set 4: Next Friday presentations 3:30
- Thursday you need to submit a .csv it at 8:00 pm.
  - Please upload it to your repo don't forget to follow the instructions, if you have questions ask before hand, **not at 7:30pm before submission!**
  - The lowest the better the score (smaller loss)
  - If you forget to send me the number of parameters I'll assign $100,000$
  - If I can't grab you predictions file from your repo with `grep` you won't get credit for the problem set.
  - It should be in the `stores` folder
- I've uploaded the final presentation schedule

Forests

- We can improve performance a lot using either bootstrap aggregation (bagging), random forests, or boosting.
- Bagging & Random Forests:
  - Repeatedly draw bootstrap samples $(X_i^b, Y_i^b)_{i=1}^N$ from the observed sample.
  - For each bootstrap sample, fit a regression tree $\hat{f}^b(x)$
    - Bagging: full sample
    - Random Forests: subset of predictors $\sqrt(p)$ (breaks high correlation)
  - Average across bootstrap samples to get the predictor

$$\boxed{\hat{f}_{bag} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)} \tag{1}$$

  - Basically we are smoothing predictions.

Boosting Trees

- Learning tree structure is much harder than traditional optimization problem where you can simply take the gradient.
- It is intractable to learn all the trees at once.
- Instead, we use an additive strategy: fix what we have learned, and add one new tree at a time. We write the prediction value at step m as $\hat{y}_i^m$.
- Then we have

$$
\begin{aligned}
\hat{y}_i^0 &= 0 \\
\hat{y}_i^1 &= \hat{y}_i^0 + f_1(x_i) \\
\hat{y}_i^2 &= \hat{y}_i^1 + f_2(x_i) \\
\dots \\
\hat{y}_i^M &= \sum_{m=1}^{M} f_m(x_i) = \hat{y}_i^{m-1} + f_m(x_i)
\end{aligned} \tag{2}
$$

XGBoost is a Boosting Tree

- Which tree do we want at each step?
- Add the one that optimizes our objective.

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(f_k) \tag{3}$$

- $L(.)$ is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i.
- The second term $\Omega(f)$ penalizes the complexity of the model, where

prediction

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda||\omega||_2 \tag{4}$$

Superlearners

Superlearnes: Motivation

- Superlearning is a technique for prediction that involves combining many individual statistical algorithms to create a new, single prediction algorithm that is expected to perform at least as well as any of the individual algorithms.
- The inovation?

Superlearnes: Motivation

- Superlearning is a technique for prediction that involves combining many individual statistical algorithms to create a new, single prediction algorithm that is expected to perform at least as well as any of the individual algorithms.
- The inovation?
- The superlearner algorithm “decides” how to combine, or weight, the individual algorithms based upon how well each one minimizes a specified loss function
- The motivation for this type of “ensembling” is that a mix of multiple algorithms may be more optimal for a given data set than any single algorithm.
- For example, a tree based model averaged with a linear model (e.g. random forests and LASSO) could smooth some of the model’s edges to improve predictive performance.

Superlearnes: Algorithm

- We have some data (y_i, X_i)
- The goal here is to solve something which looks like

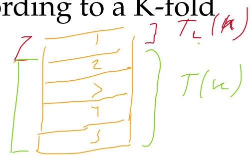
$$f^\star = \underset{f \in \mathcal{F}}{\text{argmin}} \left\{ \sum_{i=1}^{n} L(y_i, f(X_i)) \right\} \tag{5}$$

- for some loss function L, which is more often than not the squared error loss, L2 : $(y_i - f X_i)^2$
- For a given problem, a library of prediction algorithms can be proposed.
- A library is simply a collection of algorithms.

Superlearnes: Algorithm

Denote the library $\mathcal{L}$ and its cardinality as $V(n)$.

1. Fit each algorithm in $\mathcal{L}$ on the entire data set $X = \{X_i : i = 1, ..., n\}$ to estimate $\hat{f}_v(X)$ with $v = 1, \ldots, V(n)$
2. Split the data set X into a training and validation sample, according to a K-fold cross-validation scheme:

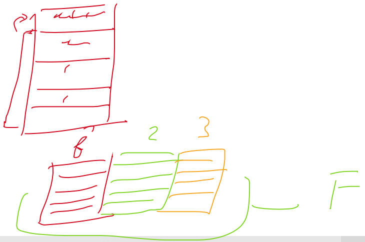


   - Splits the ordered n observations into K-equal size groups,
     - let the k-th group be the validation sample,
     - and the remaining group the training sample
   - Define $T(k)$ to be the kth training data split and $Te(k)$ to be the corresponding validation data split. $T(k) = X/V(k), k = 1, ..., K$.

Superlearnes: Algorithm

3. For the kth fold, fit each algorithm in $\mathcal{L}$ on $T(k)$ and save the predictions on the corresponding test , $\hat{f}_{k,T}(X_i)$ with $X \in Te(k)$
4. Bind the predictions from each algorithm together to create a n by V matrix

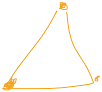
Algoritmo 1 2 3



Superlearnes: Algorithm

5 Propose a family of weighted combinations of the candidate estimators indexed by weight-vector α:

$$m(z|\alpha) = \sum_{v=1}^{V} \alpha_v \hat{f}(X_i)_v \tag{6}$$



$$\alpha_v \geq 0 \, \forall v$$

$$\sum_{v=1}^{V} \alpha_v = 1$$

$y_i = \partial F + u$

discrete

Superlearnes: Algorithm

6 Determine the α that minimizes the cross-validated risk of the candidate estimator $\sum_{v=1}^{V} \alpha_v \hat{f}(X_i)$ over all allowed α-combinations:

$$\hat{\alpha} = \underset{\alpha}{argmin} \sum_{i=1}^{n} (y_i - m(z|\alpha))^2 \tag{7}$$

7 Combine $\hat{\alpha}_v$ with $\hat{f}_v(X_i)$ according to the weights found, and create the final super learner fit

ATE = Σ θ CATE

$$\hat{f}_{SL}(X) = \sum_{v=1}^{V} \hat{\alpha}_v \hat{f}_v(X_i) \tag{8}$$

$= \frac{1}{B} \Sigma \hat{f}$

Superlearnes: Algorithm

$y = \beta F + u$

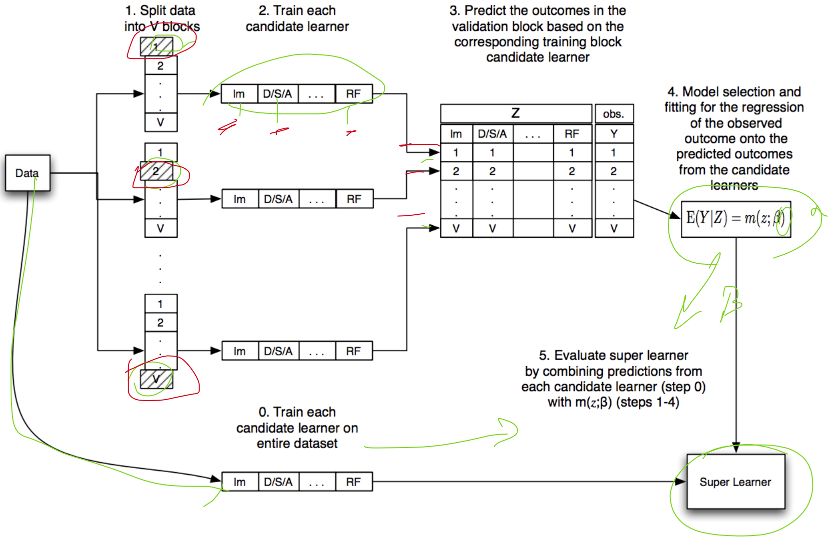
Some considerations:

- The super learner theory does not place any restrictions on the family of weighted combinations used for ensembling the algorithms in the library.
- The restriction of the parameter space for α to be the convex combination of the algorithms in the library provides greater stability of the final super learner prediction.

  $\alpha \geq 0 \quad \sum \alpha = 1$
- Restricting to the convex combination implies that if each algorithm in the library is bounded, the convex combination will also be bounded.

Superlearnes: Summary

V e k '



Source: Polley, Eric. Learning: Causal Inference for Observational and Experimental Data

Superlearnes: Available Learners

Table 1: Library of prediction algorithms for the simulations and citation for the corresponding R package.

Algorithm	Description	Author
`glm`	linear model	R Development Core Team [2010]
`interaction`	polynomial linear model	R Development Core Team [2010]
`randomForest`	random Forest	Liaw and Wiener [2002] Breiman [2001]
`bagging`	bootstrap aggregation of trees	Peters and Hothorn [2009] Breiman [1996a]
`gam`	generalized additive models	Hastie [1992] Hastie and Tibshirani [1990]
`gbm`	gradient boosting	Ridgeway [2007] Friedman [2001]
`nnet`	neural network	Venables and Ripley [2002]
`polymars`	polynomial spline regression	Kooperberg [2009] Friedman [1991]
`bart`	Bayesian additive regression trees	Chipman and McCulloch [2009] Chipman et al. [2010]
`loess`	local polynomial regression	Cleveland et al. [1992]

Source: Polley, Eric. Learning: Causal Inference for Observational and Experimental Data

Superlearnes: Performance

Algorithm	MSE	se(MSE)	R^2	se(R^2)
SuperLearner	1.193	0.200	0.759	0.040
discrete SL	1.036	0.035	0.791	0.007
SL.glm	5.040	0.106	−0.017	0.021
SL.interaction	5.057	0.126	−0.021	0.026
SL.randomForest	2.645	0.523	0.466	0.106
SL.bagging(0.01)	4.414	0.351	0.109	0.071
SL.bagging(0.1)	4.734	0.200	0.044	0.040
SL.bagging(0.0)	4.416	0.343	0.109	0.069
SL.bagging(ms5)	2.650	0.543	0.465	0.110
SL.gam(2)	5.033	0.113	−0.016	0.023
SL.gam(3)	5.061	0.131	−0.022	0.027
SL.gam(4)	5.089	0.160	−0.027	0.032
SL.gbm	4.580	0.282	0.075	0.057
SL.nnet(2)	5.067	0.447	−0.023	0.090
SL.nnet(3)	4.922	0.627	0.006	0.127
SL.nnet(4)	4.769	0.528	0.037	0.107
SL.nnet(5)	4.816	0.928	0.028	0.187
SL.polymars	4.996	0.309	−0.008	0.062

Source: Polley, Eric. Learning: Causal Inference for Observational and Experimental Data

$y = x_1 \; x_2 \; *$

$y = x_1 * x_2 *$

$y - x_1 \; x_2 \; x_3 \; *$

Text as Data

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).

Text as Data: Story Time

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: Story Time

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

United States of America

PROCEEDINGS AND DEBATES OF THE 95th CONGRESS, SECOND SESSION

WASHINGTON, MONDAY, OCTOBER 2, 1978

Vol. 124

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

  $\Rightarrow$ "death tax" is republican

$$Ideology = f(\mathbf{X}_{\text{text}}) + u \tag{9}$$

  where

$$ideology \approx g(Y_{Bush}) \tag{10}$$

- Gentzkow and Shapiro apply this logic to build an index of slant that sums across a speaker's term usage weighted by the direction of slant for each term.

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.

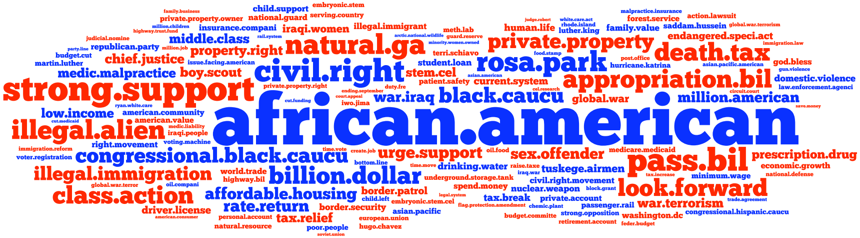
The n-gram language model

- 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.

Text as Data: Wordle

- Often best to just count words.
- For example, occurrences by party for some partisan terms

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



Text Regression

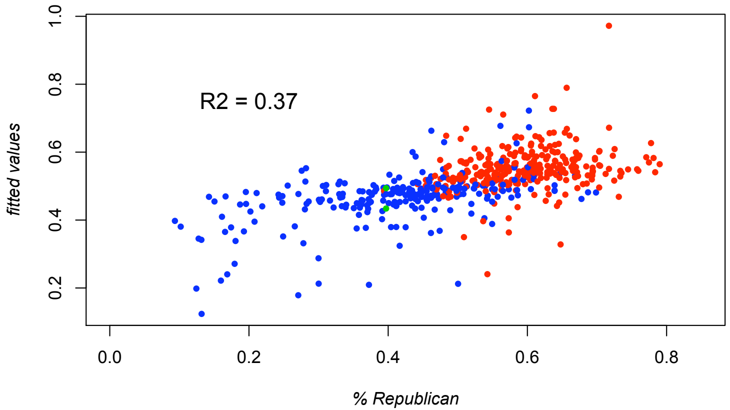
- Once you have text in a numeric format, we can use all the tools we learned so far

$$y = f(word\,counts) + u \tag{11}$$

- where you can use lasso, PCA, etc. to do dimentionality reduction

Text Regression

Slant measure for speakers in the 109th Congress

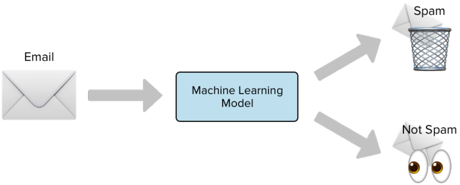


Democrats get low z_{slant} and Republicans get high z_{slant}.

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

Text Regression

- Another example: Classify emails into spam



$$\text{logit}\,[\texttt{spam}] = \alpha + f\beta \qquad (12)$$

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

Review & Next Steps

- Superlearners
- Text as Data: Intro
- Next class: More on text as data, dimention reduction, and lots of Linear Algebra!!!
- Questions? Questions about software?

Further Readings

- MJ Van der Laan, EC Polley, AE Hubbard, Super Learner, Statistical applications in genetics and molecular, 2007
- Polley, Eric. Learning: Causal Inference for Observational and Experimental Data, by M. J. van der. Laan and Sherri Rose, Springer, 2011.
- Polley E, LeDell E, Kennedy C, van der Laan M. Super Learner: Super Learner Prediction. 2016 URL https://CRAN.R-project.org/package=SuperLearner. R package version 2.0-22.
- Hoffman, K.. Become a Superlearner! An Illustrated Guide to Superlearning. https://www.khstats.com/blog/sl/superlearning/
- Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.

Tokenization Demo

```
## 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

```
## 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)
```

```
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: Example (Gentzkow and Shapiro)

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

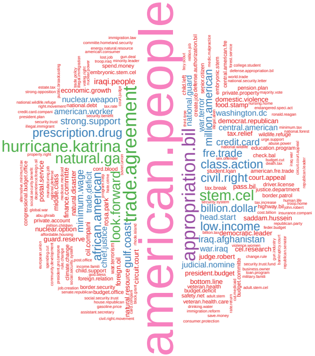
```
## 2 x 4 sparse Matrix of class "dgCMatrix"
##              stem.cel natural.ga hurricane.katrina trade.agreement
## Barack Obama        .          1                20               7
## John Boehner        .          .                14               .
```

```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)

```
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"))
```

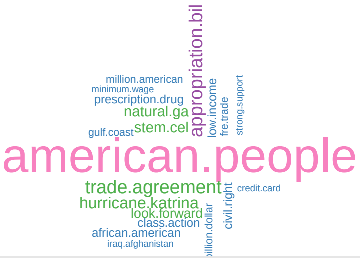


Text Regression: Wordle (Wordclouds)

```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

```
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
```