Measuring Political Bias in Text with R

John Myles White

April 29, 2011

Text A

Hey, does anybody notice this crazy thing that we're on the road to socialism? I'm just saying. Wow. We got — we got the SCHIPs thing going for us. That's great.

Text B

How about that McDonalds two blocks from Ground Zero? That's killed more people than the nineteen hijackers.

Exploiting Artificial Artificial Intelligence

1. Text A was pro-Democrat and Text B was pro-Republican
2. Text A was pro-Republican and Text A was pro-Democrat

Artificial Intelligence / Machine Learning

- People can infer political views from text
- Can computers learn to do the same?

Desiderata

- Continuous measure of political viewpoint
- Ability to analyze arbitrary text

The Ideal Points Model

- Treats politics as one dimensional
- Goes from left wing to right wing

The Ideal Points Model

- Fit to Congressional roll call voting records
- Predicts unseen votes with $> 90\%$ accuracy

The Ideal Points Model

- Senators have fixed positions on the scale
- Bills have fixed positions on the scale
- Senators vote "yay" for bills that are near their own position

The Ideal Points Model

- Essentially logistic regression with hidden variables
- Easy to fit using Bayesian MCMC methods
- Use JAGS through the 'rjags' package

Senate Roll Call Voting Records

- Available as ORD files from voteview.com
- Read in to R using 'readKH()' from 'pscl' package

Senate Roll Call Voting Records

- Construct roll call matrix
  - Rows are senators
  - Columns are bills
  - Entries are yay / nay votes
- Analyze data from 111th Congress

Senate Roll Call Voting Records

```
roll.calls <- readKH(file.path('data',
                               'roll_calls_111.ord'))

votes <- roll.calls$votes

binary.votes <- apply(votes, c(1, 2), yes.no.vote)
```

Example Roll Call Matrix

Senator	Bill 1	Bill 2	Bill 3
Byrd (D WV)	1	1	1
Chambliss (R GA)	NA	0	0

Using JAGS for Bayesian Inference

- Bayesian modeling gives us flexibility
- JAGS (BUGS) is a general modeling language
- JAGS lets us describe model mathematically
- JAGS compiles model into code
- Code runs MCMC and estimates model parameters

Using 'rjags' to Call JAGS from R

```
jags <- jags.model('jags/ideal_points.bug',
                   data = list('votes' = binary.votes,
                               'M' = nrow(binary.votes),
                               'N' = ncol(binary.votes),
                               'a' = a),
                   n.chains = 4,
                   n.adapt = 500)

j.samples <- jags.samples(jags,
                          c('a', 'b', 'g'),
                          250)
```

The JAGS Model

```
for (i in 1:M)
{
  for (j in 1:N)
  {
    votes[i, j] ~ dbern(p[i, j])
    logit(p[i, j]) <- g[j] * (a[i] - b[j])
  }
}
```

The JAGS Model

```
for (i in 1:M)
{
  a[i] ~ dnorm(0, tau.a)
}

tau.a <- pow(sigma.a, -2)
sigma.a ~ dunif(0, 100)
```

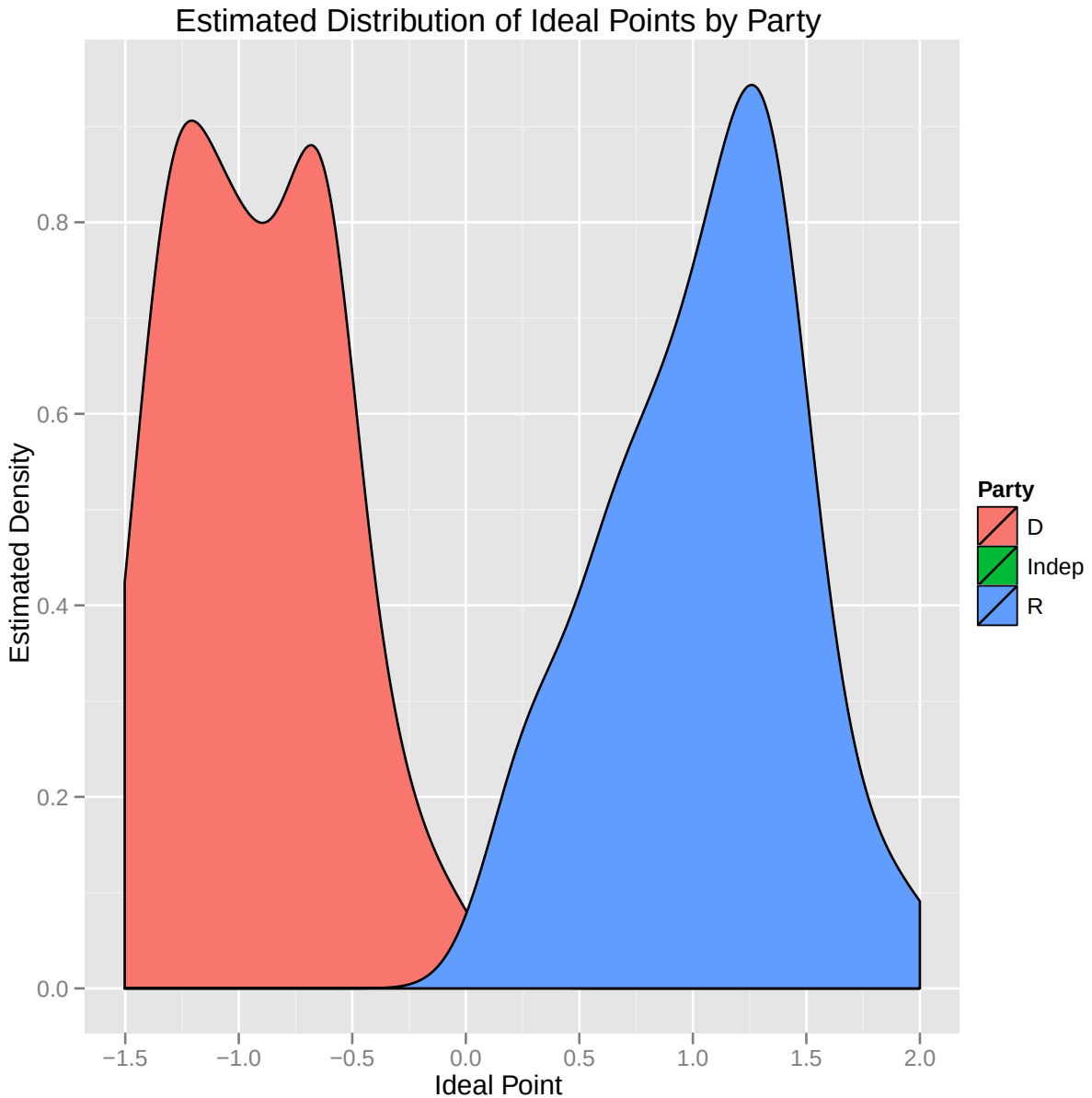
The JAGS Model

```
for (j in 1:N)
{
  b[j] ~ dnorm(mu.b, tau.b)
  g[j] ~ dnorm(0, tau.g)
}

mu.b ~ dnorm(0, 0.0001)

tau.b <- pow(sigma.b, -2)
sigma.b ~ dunif(0, 100)

tau.g <- pow(sigma.g, -2)
sigma.g ~ dunif(0, 100)
```

Example Ideal Points for 111th Congress

Senator	Ideal Point
Demint (R SC)	1.79
Snowe (R SC)	0.23
Bayh (D IN)	-0.15
Durbin (D IL)	-1.50

Ideal Points by Party



Ideal Points: Measure of Political Bias

- Ideal points are our continuous scale for measuring bias
- How can we relate them to text?

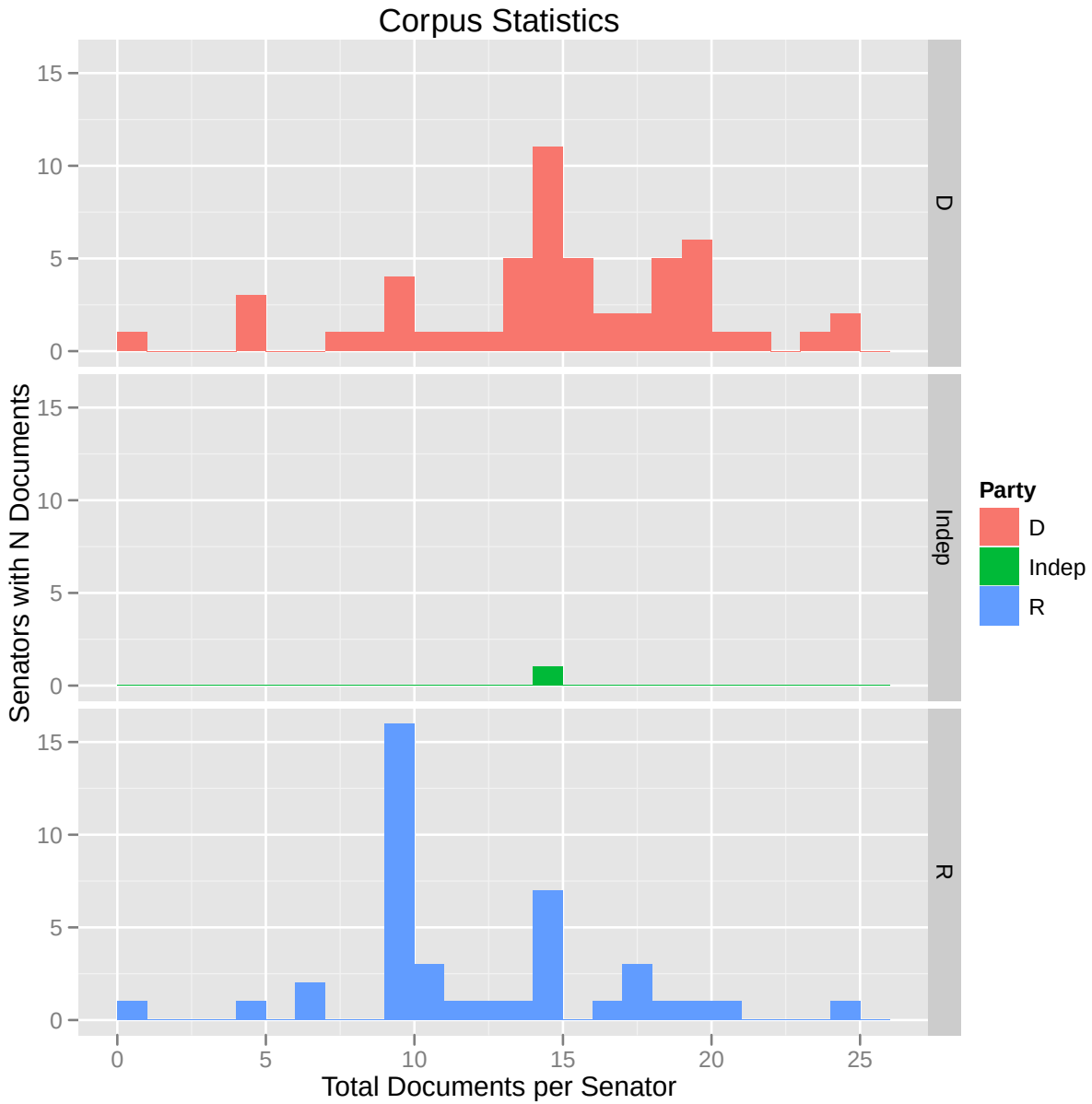
Harvest Congressional Text and Match with Ideal Points

- Manually gather text from each senator
  - Floor speeches
  - Press releases
  - Op-eds
- Assign senator's ideal point to text written by them
- Predict ideal points using text

Corpus Statistics

- 1,408 unique documents
- 20,521 unique words

Corpus Statistics



Importing Text into R

- Use the ‘tm’ package
- Store all documents in one large CSV file
- Use ‘read.csv’ to turn CSV file into a data.frame
- Create a ‘tm’ corpus from the data.frame
- Build a document-term matrix

Example Document A from the Corpus

i want to talk about jobs lately it seems that everyone says they want to talk about jobs and that we'll get around to tackling jobs next week or the week after

Example Document B from the Corpus

there was a major legislative accomplishment in washington last week and it's getting less attention than it deserves because it isn't national health care reform

Example Document-Term Matrix

Document	I	Want	Talk	Jobs	Week
A	1	2	2	3	2
B	0	0	0	0	1

Creating a Corpus with 'tm'

```
corpus <- Corpus(DataframeSource(documents))

corpus <- tm_map(corpus,
                 tolower)

corpus <- tm_map(corpus,
                 removeWords,
                 stopwords('english'))
```

Creating a Document Term Matrix with 'tm'

```
document.term.matrix <- DocumentTermMatrix(corpus)

x <- as.matrix(document.term.matrix)

y <- authors$IdealPoint
```

Text Regression

- Turn text analysis into a standard regression problem
- Outcomes: senators' ideal points
- Predictors: document words counts

OLS Regression: A Review

- Predict continuous outcome variable
- Try to minimize sum of squared errors
- Need more rows than columns
- Use ‘lm’ or maybe ‘glm’

OLS Regression in R

```
x <- 1:10
y <- 3 * x + rnorm(10, 0, 1)

fit <- lm(y ~ x)

coef(fit)
```

OLS Regression in R

```
(Intercept)           x
 -0.1896415   2.9648153
```

OLS Regression in R

```
summary(fit)
```

OLS Regression in R

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.01532 -0.29611 -0.06683  0.73038  1.37964

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1896     0.7174  -0.264    0.798
x             2.9648     0.1156  25.644 5.73e-09 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.05 on 8 degrees of freedom
Multiple R-squared: 0.988, Adjusted R-squared: 0.9865
F-statistic: 657.6 on 1 and 8 DF,  p-value: 5.734e-09
```

Underdetermined Problems

- Fewer rows than columns
- OLS finds infinitely many solutions, all with zero error
- Need regularization to find useful solutions

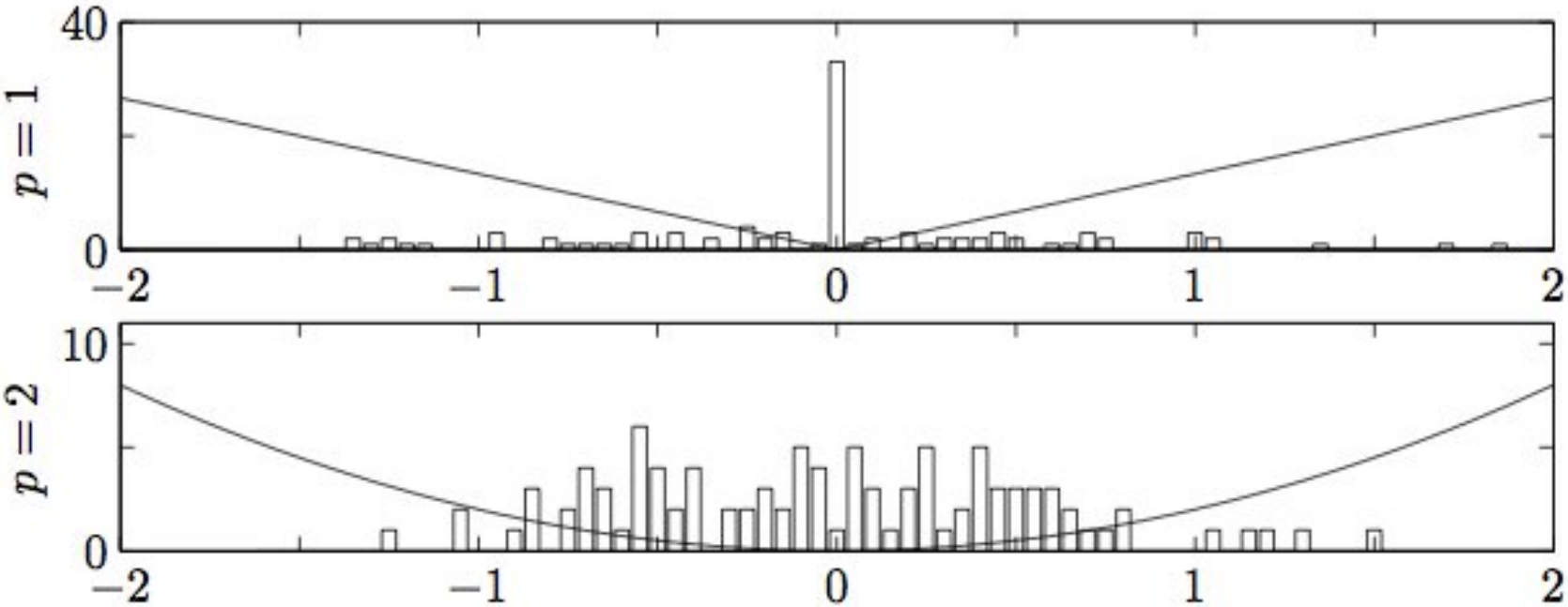
Regularized Regression

- Standard regression uses prediction error to find coefficients
  - OLS regression
  - LAD regression
- Regularization adds a penalty for large coefficients
  - Ridge regression (L2)
  - LASSO regression (L1)
  - Elastic Net regression (L2 + L1)

L_p Penalities

- L2 penalty is harsh on large values
- L1 penalty is harsh on any non-zero value
- L1 induces sparse solutions
- Very few columns get non-zero coefficients
- Most columns get exactly zero coefficients

Typical Distribution of Coefficients



Regularized Regression in R

- Use the 'glmnet' package
- Fits any form of Elastic Net
- 'alpha' shifts between LASSO and ridge
- 'alpha' $= 1$ by default, which gives LASSO
- 'lambda' controls the prediction error / regularization tradeoff
- 'glmnet' fits many values of 'lambda' automatically

Simple glmnet Example

```
x1 <- c(1, 2, 3)
x2 <- c(1, 1, 3)
x3 <- c(1, 4, 3)
x <- cbind(x1, x2, x3)

a <- 1
b <- 2
c <- 3

y <- a * x1 + b * x2 + c * x3 + rnorm(3, 0, 1)
```

Simple glmnet Example

```
library('glmnet')

fit <- glmnet(x, y)

fit
```

Simple glmnet Example

```
Call:  glmnet(x = x, y = y)

      Df   %Dev Lambda
 [1,]  0 0.0000 4.3300
 [2,]  1 0.1550 3.9460
...
[38,]  2 0.9989 0.1385
[39,]  2 0.9991 0.1262
```

Out of Sample Model Validation

- Don't want to overfit data
- Assess final model on data not used during model fitting
- Split data into two parts: training set and test set
- 2 / 3 of data goes into training set
- 1 / 3 of data goes into test set

Training Set / Test Set Split

```
training.size <- round(length(y) * (2 / 3))
test.size <- round(length(y) * (1 / 3))

training.indices <- sample(1:length(y), training.size)

training.x <- x[training.indices, ]
training.y <- y[training.indices]
test.x <- x[!test.indices, ]
test.y <- y[!test.indices]
```

Back to 'lambda'

- If 'lambda' $= 0$, get OLS regression results
- Zero prediction error, but model is meaningless
- If 'lambda' $= \infty$, coefficients go to zero
- Model predicts the same outcome for all rows
- Need to find sweet spot

Hyperparameter Tuning

- Split training set into fitting subset and assessment subset
- Repeat splitting process many times
- Calculate error for many ‘lambda’ values
- Pick ‘lambda’ with lowest average out-of-sample error

Example Model Performance Table

Iteration	Lambda	RMSE
1	1	1.03
2	1	1.07
1	0.5	0.98
2	0.5	0.99

Hyperparameter Tuning

```
mean.rmse <- ddply(performance, 'Lambda', mean)

min.rmse <- with(mean.rmse, min(RMSE))

optimal.lambda <- with(subset(mean.rmse,
                              RMSE == min.rmse),
                       Lambda)
```

The ..ply functions from 'plyr'

- Split, apply, combine strategy
- Specify initial data.frame
- Specify columns to use to split data
- Specify function to apply to subsets
- Returns combined results from all subsets

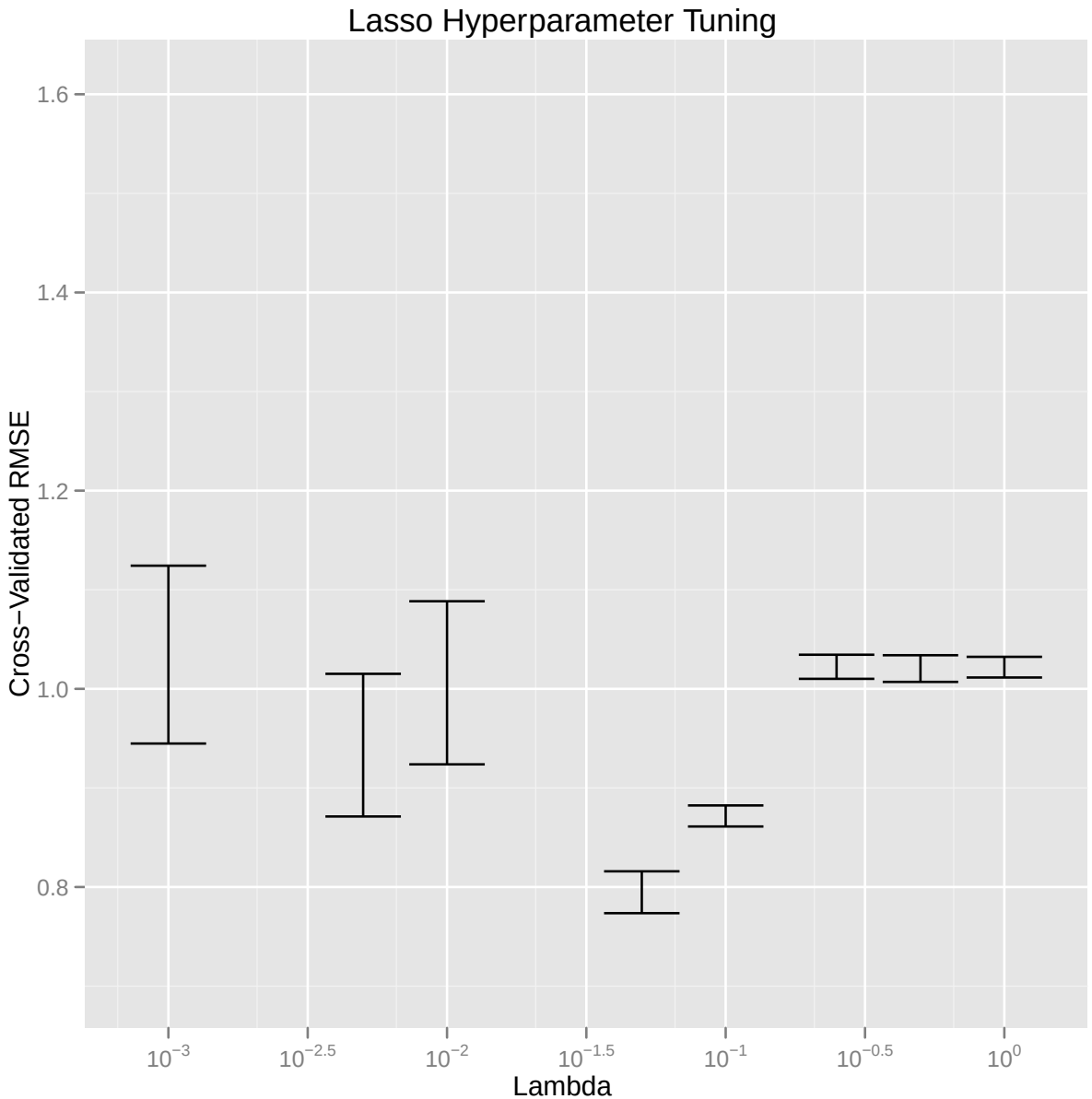
ddply Example

```
df <- data.frame(Class = c(1, 1, 2, 2),
                 Score = c(10, 15, 20, 25))

ddply(df, 'Class', function (df) {with(df, mean(Score))})
```

ddply Example

```
  Class   V1
1     1 12.5
2     2 22.5
```

Hyperparameter Tuning Results



Fit Final Model

```
fit <- glmnet(training.x, training.y)
```

Find Most Biased Terms

```
term.weights <- coef(fit, s = optimal.lambda)

sorted.terms <- sort(term.weights[,1])

n <- length(sorted.terms)

most.democratic.terms <- sorted.terms[1:10]
most.republican.terms <- sorted.terms[(n - 9):n]
```

Top 10 Most Republican Terms

Term	Value
okla	1.23
bailey	0.647
johnny	0.588
administering	0.561
neb	0.556
sam	0.542
986	0.532
texans	0.493
patriotism	0.466
demint	0.417

Top 10 Most Democratic Terms

Term	Value
sherrod	-0.367
sheldon	-0.249
dec	-0.196
possess	-0.168
salaries	-0.158
tom	-0.152
debbie	-0.151
dark	-0.148
lautenberg	-0.133
fought	-0.106

Debugging Our Results

- Too many names of senators in our list
- Strip out all the names from corpus
- Run analysis from scratch on clean corpus

Top 10 Most Republican Terms excluding Names

Term	Value
okla	1.13
neb	0.726
bailey	0.674
2415	0.638
986	0.578
kansans	0.543
administering	0.516
texans	0.467
profoundly	0.459
patriotism	0.430

Top 10 Most Democratic Terms excluding Names

Term	Value
cedar	-0.224
chaired	-0.197
dec	-0.158
dark	-0.146
blocked	-0.138
reverses	-0.134
1960s	-0.125
insurers	-0.0958
fought	-0.0926
possess	-0.0923

Assessing Our Predictive Power

```
predicted.y <- predict(fit,
                       newx = test.x,
                       s = optimal.lambda)

predicted.y <- as.numeric(predicted.y[,1])

predictions <- data.frame(Predicted = predicted.y,
                          Empirical = test.y,
                          Residual = predicted.y - test.y)
```

Assessing Our Predictive Power

```
RMSE <- with(predictions, sqrt(mean(Residual ^ 2)))
```

Final Model Comparison Results

Model	RMSE
Pure Intercept Regression	1.02
Lasso Text Regression excluding Senators' Names	0.879
Lasso Text Regression including Senators' Names	0.805

CRAN Packages Used

- pscl
- rjags
- plyr
- ggplot2
- glmnet
- ProjectTemplate

Links

- Senate Analyses on GitHub
- Adam Bonica's Blog
- Simon Jackman's Bayes Page
- Noah Smith's CMU Page
- David Blei's Princeton Page
- My Blog