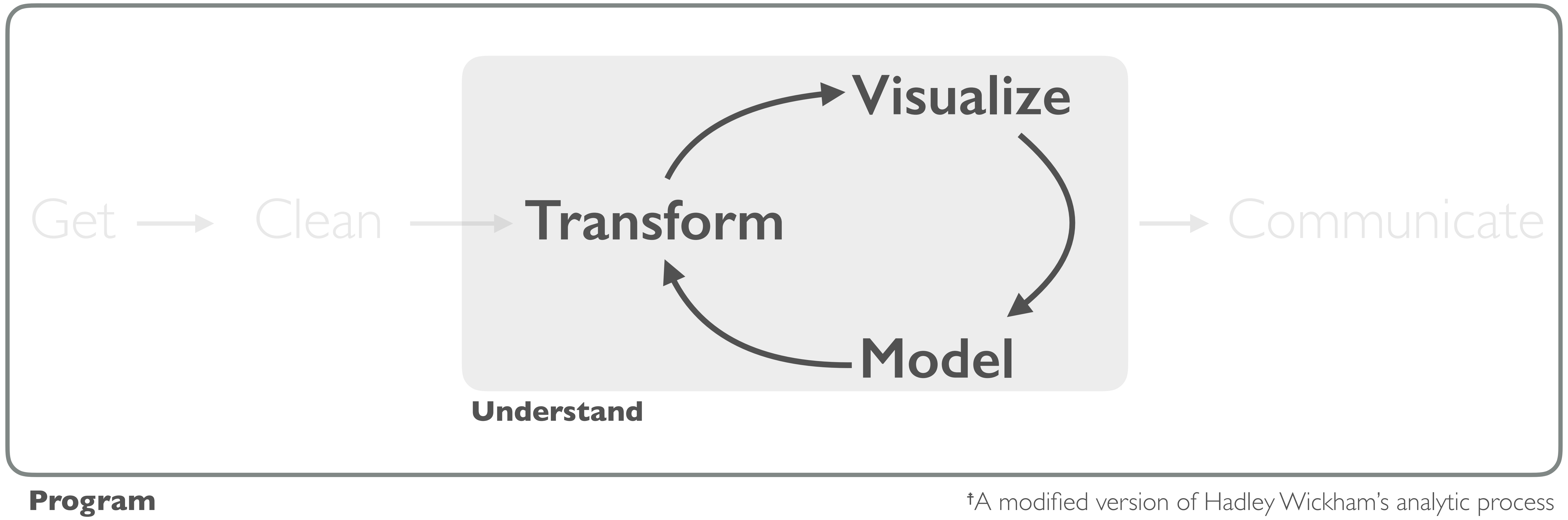
PREDICTIVE MODELING



†A modified version of Hadley Wickham's analytic process


What society thinks I do
What my friends think I do
What other computer scientists think I do
amc
What mathematicians think I do
What I think I do
Source on Save
# Load required pkgs
library(randomForest)
library(xgboost)
What I actually do

PREDICTIVE MODELING

- Text data can be used for predictive modeling much like more traditional data.
- Can be used for both classification and regression problems
- Caveats
  - Must be ***cleaned*** and ***tidied*** (document term matrix)
  - Technique must be able to handle the ***curse of dimensionality***
  - Technique must be able to handle ***sparsity***

PREREQUISITES

PACKAGE PREREQUISITE

```
library(tidyverse)          # data wrangling & plotting
library(tidytext)           # text manipulation
library(glmnet)             # elastic net modeling
library(pROC)               # ROC curve / AUC
```

DATA PREREQUISITE

```
# Click-bait news headlines
url <- "https://raw.githubusercontent.com/kwartler/text_mining/master/all_3k_headlines.csv"
headlines <- read_csv(url)

headlines
# A tibble: 3,000 x 4
   headline                             url                                       site        y
   <chr>                                <chr>                                     <chr>   <int>
 1 Mom Sentenced To 6 Years In Prison Fo… http://dailybuzzlive.com/mom-sentenced-to… dailyb…      1
 2 "The Most Shocking '\x98Jerry Springe… http://dailybuzzlive.com/the-most-shockin… dailyb…      1
 3 America''s Self-Inflicted Defense Woes http://www.activistpost.com/2016/08/ameri… activi…      1
 4 "A Man Spots A Reckless Driver, When … http://dailybuzzlive.com/a-man-spots-a-re… dailyb…      1
 5 Tim Cook Asks FBI To Withdraw Order T… http://www.buzzfeed.com/johnpaczkowski/ap… buzzfe…      1
 6 Police Union Threatens Not To Patrol … http://www.buzzfeed.com/salvadorhernandez… buzzfe…      1
 7 Ed Rendell: Clinton's Speech Was Pres… http://www.buzzfeed.com/christophermassie… buzzfe…      1
```

DATA PREP

Preparing for the modeling process

GENERALIZABILITY

- **Generalizability**: the confidence that a model will perform similarly on unobserved data.
- The difference between a descriptive vs. predictive capability

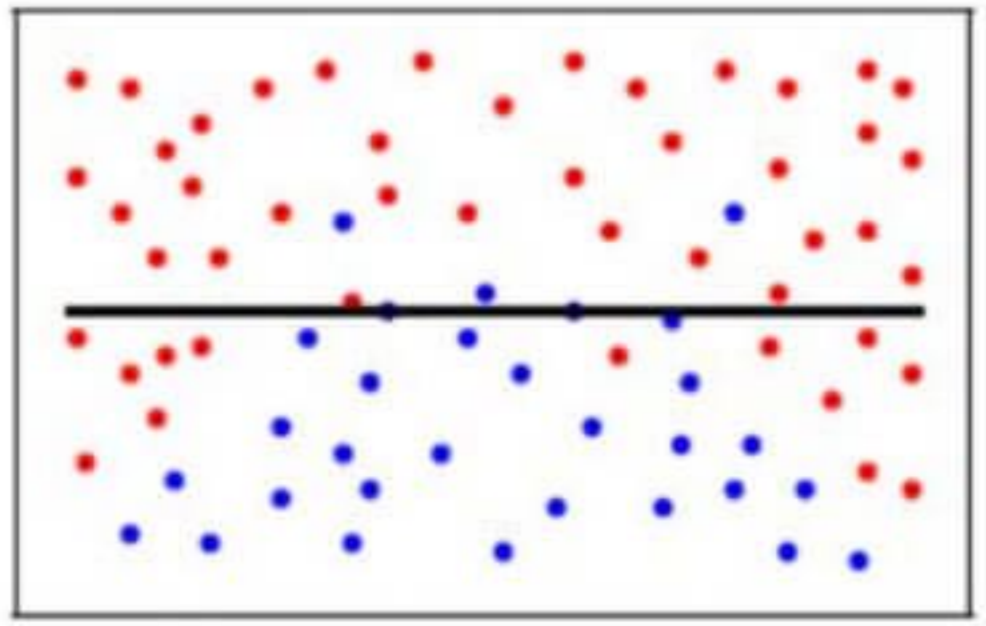
A more generalizable model gives us greater confidence in future accuracy

BIAS VS. VARIANCE

Bias

- Wrong model assumptions
- Model restrictions on hyperparameters
- Error due to model specification

Underfitting



BIAS vs. VARIANCE

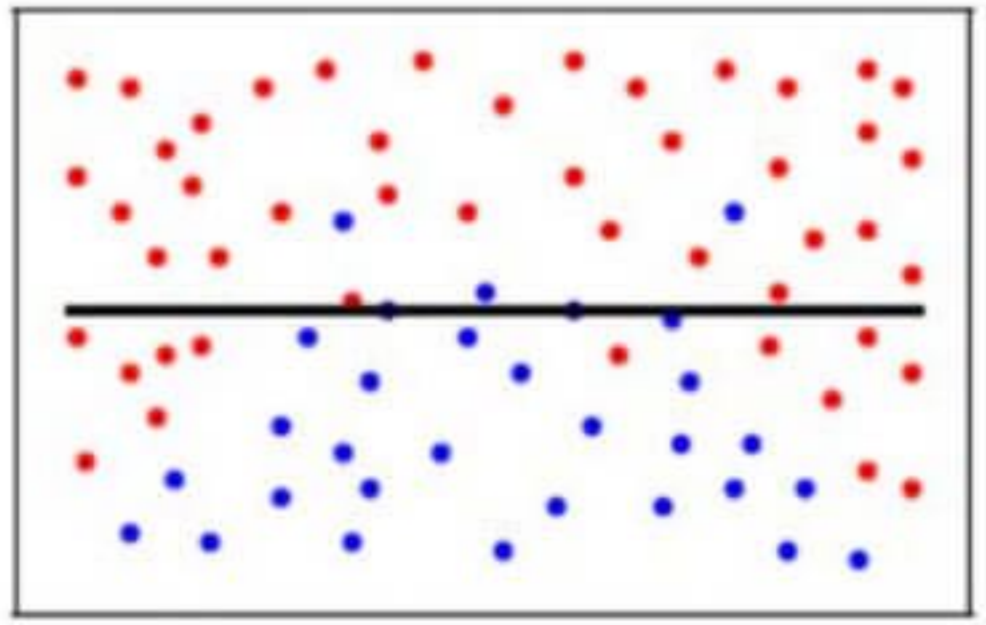
Bias

- Wrong model assumptions
- Model restrictions on hyperparameters
- Error due to model specification

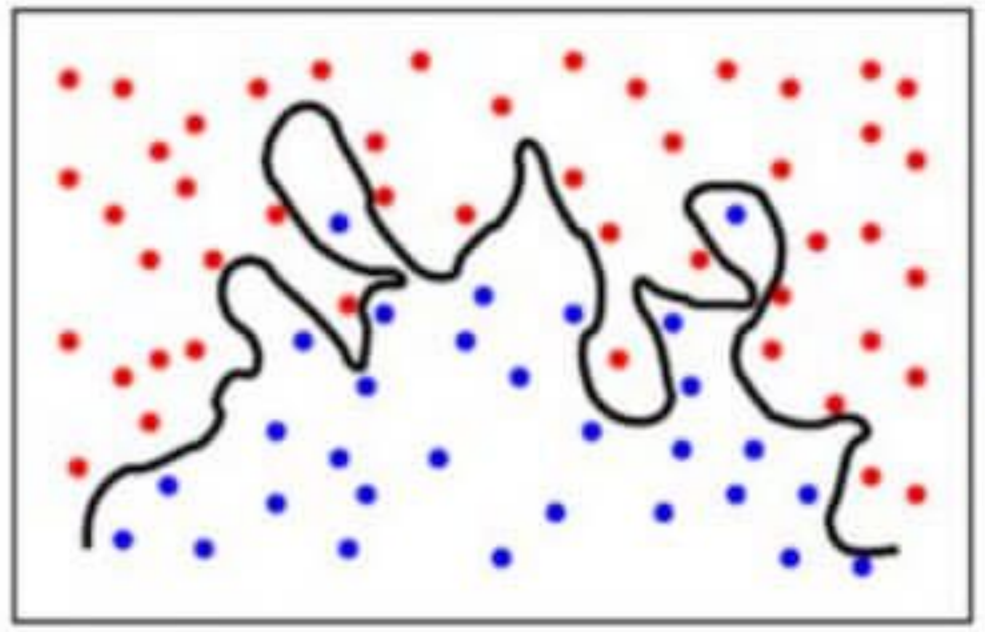
Variance

- Model over adapts to training data
- Hyperparameters overly tuned to training data
- Error due to the sampling of training set

Underfitting



Overfitting



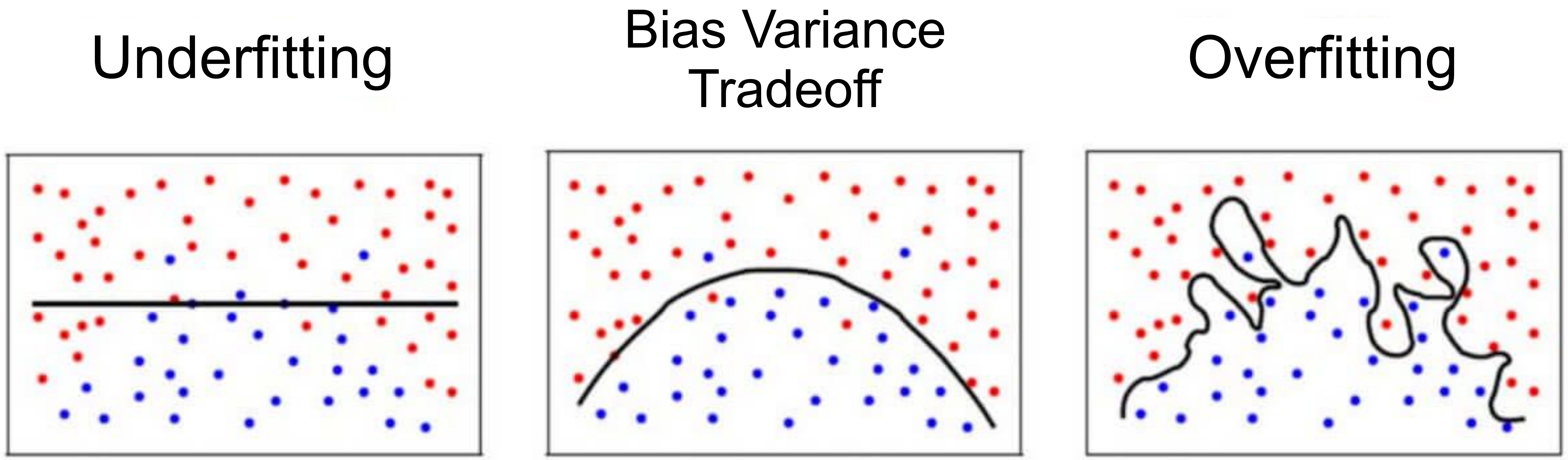
BIAS vs. VARIANCE

Bias

- Wrong model assumptions
- Model restrictions on hyperparameters
- Error due to model specification

Variance

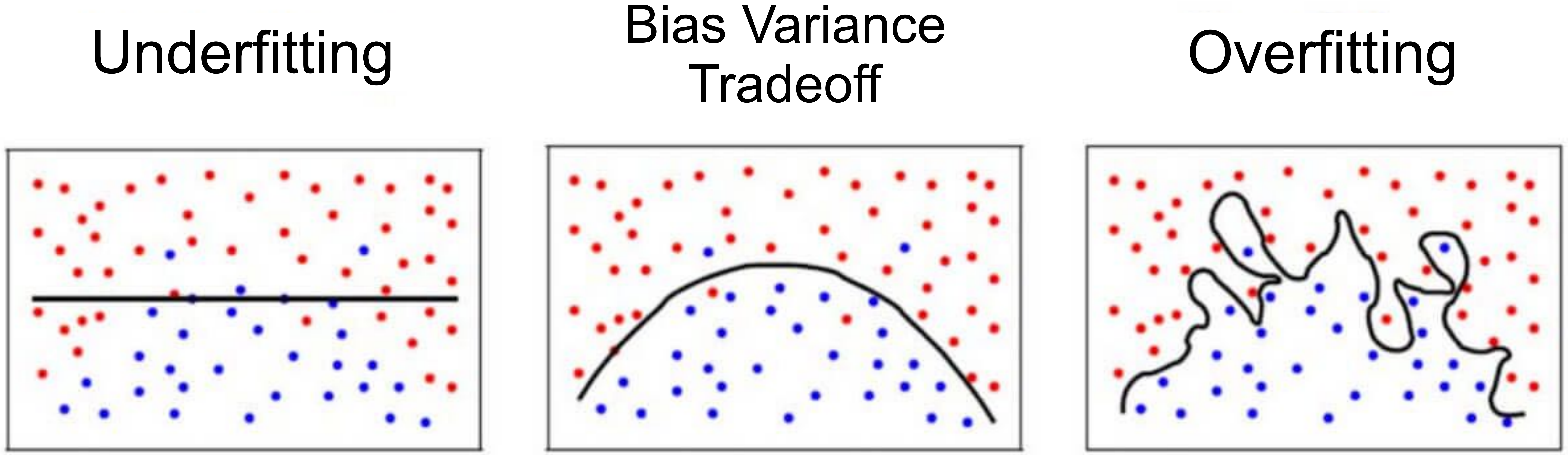
- Model over adapts to training data
- Hyperparameters overly tuned to training data
- Error due to the sampling of training set



BIAS VS. VARIANCE

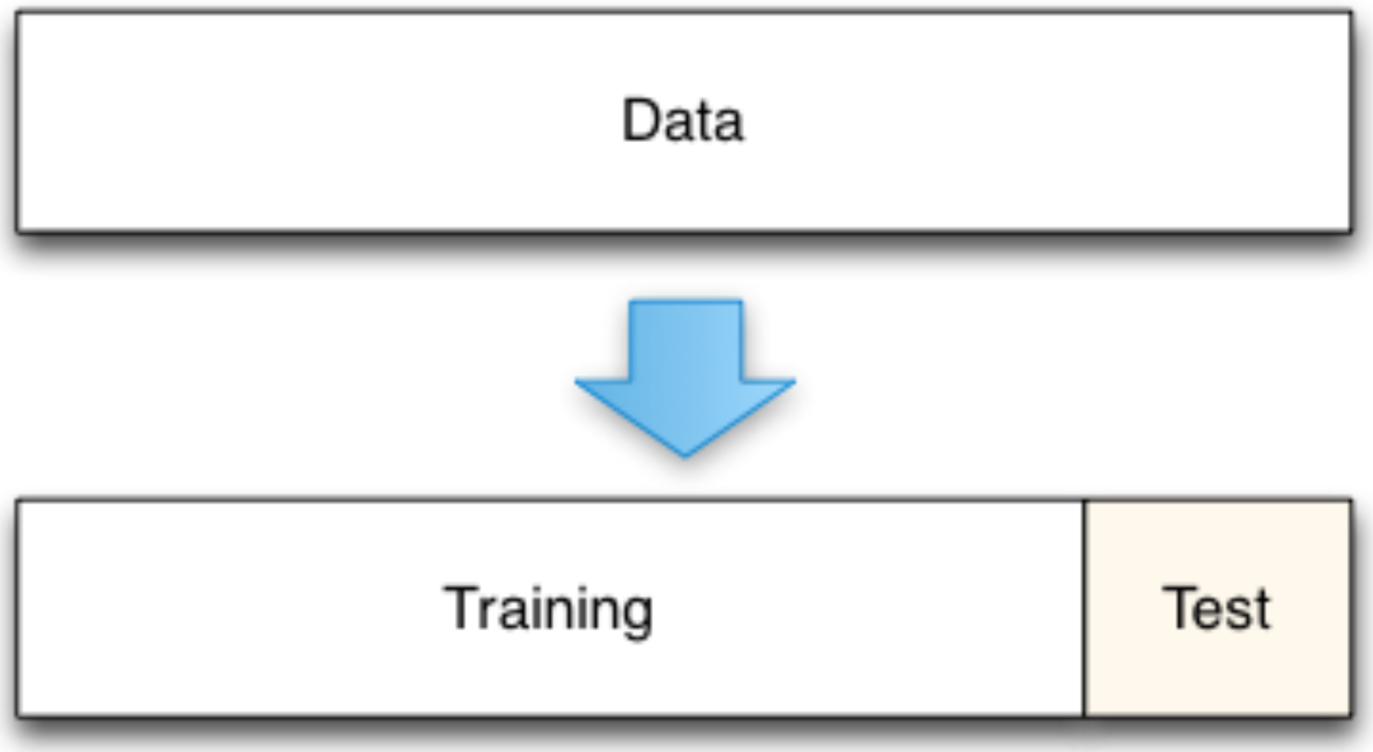
Resampling

- The more adaptable our modeling techniques become, the greater the importance to use advanced resampling methods to minimize overfitting…***aka over-confident predictions***



BASIC TRAIN/TEST SPLIT

- At a minimum, always use a training-test split to assess out-of-sample accuracy.
- Typical: 70/30
  - Dependent on size of data set



<u>Disadvantages</u>

- Model can overfit to training data
- Single test set provides only a single insight into predictive performance

ESTABLISHING TRAINING VS. TESTING SETS

```r
# assign about 70% to training and 30% to test
set.seed(123)

headlines <- headlines %>%
  mutate(partition = sample(
    x = c("Train", "Test"),
    size = 3000,
    prob = c(.7, .3),
    replace = TRUE)
   )
```

`set.seed` provides reproducibility

Next, create a `partition` variable

- sample from `x`
- 3000 times (length of data set)
- 70% from Train, 30% from Test

CLEAN AND TIDY DATA

```
# assign about 70% to training and 30% to test
headlines_tidy <- headlines %>%
  unnest_tokens(word, headline) %>%
  anti_join(stop_words) %>%
  count(url, partition, y, word) %>%
  mutate(id = paste(url, partition, sep = "_"))

headlines_tidy
# A tibble: 18,647 x 6
  url              partition      y word      n id
  <chr>            <chr>      <int> <chr> <int> <chr>
1 http://21stce… Train          1 anti      1 http://21stce…
2 http://21stce… Train          1 gove…     1 http://21stce…
3 http://21stce… Train          1 inst…     1 http://21stce…
4 http://21stce… Train          1 lead…     1 http://21stce…
5 http://21stce… Train          1 obama     1 http://21stce…
6 http://21stce… Train          1 opin…     1 http://21stce…
7 http://21stce… Train          1 party     1 http://21stce…
```

This is performing the same tidying process you've seen.

However, we add one step where we combine the url & partition:

id: http://21stcenturywire…_Train

We will use this later!

CONVERT TO A DOCUMENT TERM MATRIX

```
# convert to a DTM matrix
headlines_matrix <-  headlines_tidy %>%
  cast_dtm(id, word, n) %>%
  as.matrix()
```

To model we need to have our data in a DTM style but as a matrix object.

Now we're ready to create our training & testing splits

TRAINING SPLIT

```
train.x <- headlines_matrix %>%
  as_data_frame() %>%
  mutate(id = row.names(headlines_matrix)) %>%
  filter(str_detect(id, "_Train")) %>%
  select(-id) %>%
  as.matrix() %>%
  Matrix::Matrix(sparse = TRUE)

train.y <- headlines_tidy %>%
  filter(partition == "Train") %>%
  distinct(id, y) %>%
  .$y

train.x[1:5, 1:5]
5 x 5 sparse Matrix of class "dgCMatrix"
     anti government instructs leader obama
[1,]    1          1         1      1     1
[2,]    .          .         .      .     .
[3,]    .          .         .      .     .
```

Create training feature set, which is in the form of a sparse matrix… computationally efficient!

Create response variable vector

YOUR TURN!

You created the training split, now can you do the same thing to create the test split?

TESTING SPLIT

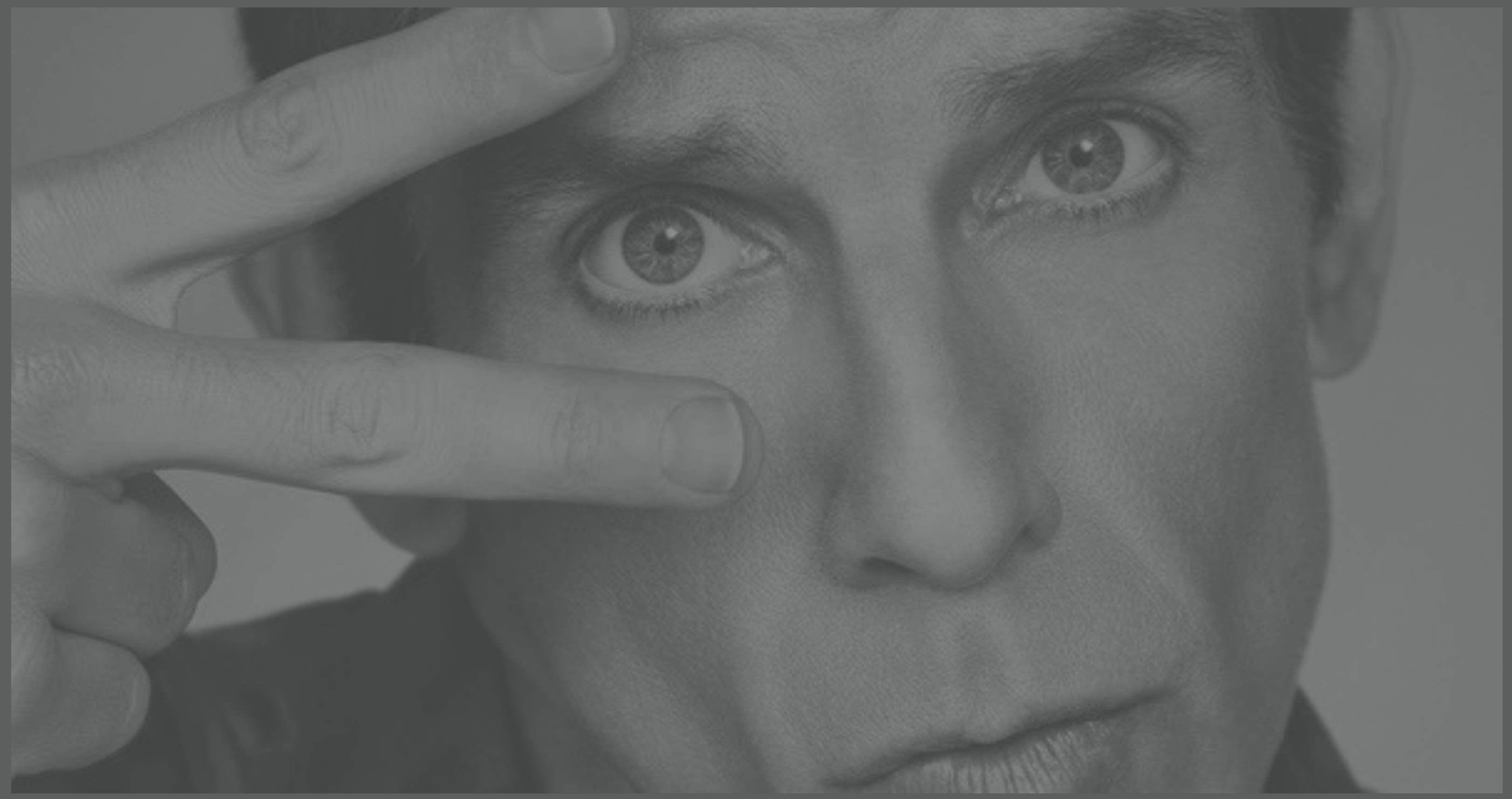
```
test.x <- headlines_matrix %>%
  as_data_frame() %>%
  mutate(id = row.names(headlines_matrix)) %>%
  filter(str_detect(id, "_Test")) %>%
  select(-id) %>%
  as.matrix() %>%
  Matrix::Matrix(sparse = TRUE)

test.y <- headlines_tidy %>%
  filter(partition == "Test") %>%
  distinct(id, y) %>%
  .$y

test.x[1:5, 1:5]
5 x 5 sparse Matrix of class "dgCMatrix"
     anti government instructs leader obama
[1,]    .          .         .      .     1
[2,]    .          .         .      .     .
[3,]    .          .         .      .     .
[4,]    .          .         .      .     .
[5,]    .          .         .      .     .

dim(test.x)
[1]  890 7149

table(test.y)
  0   1
450 440
```

Create testing feature set, which is in the form of a sparse matrix… computationally efficient!

Create response variable vector

MODELING

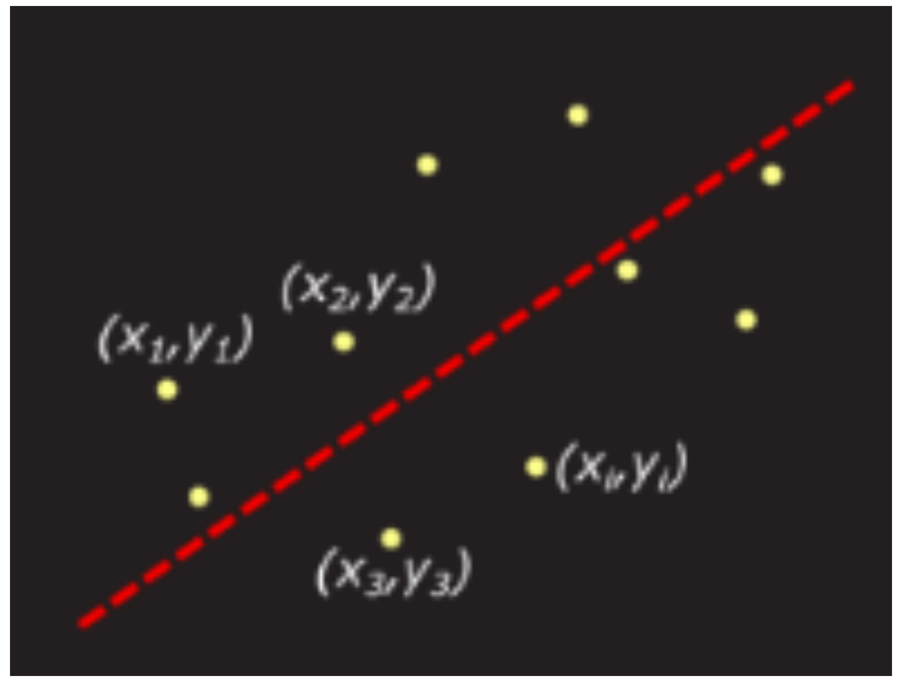
Regularized regression: Ridge, Lasso, & Elastic Nets

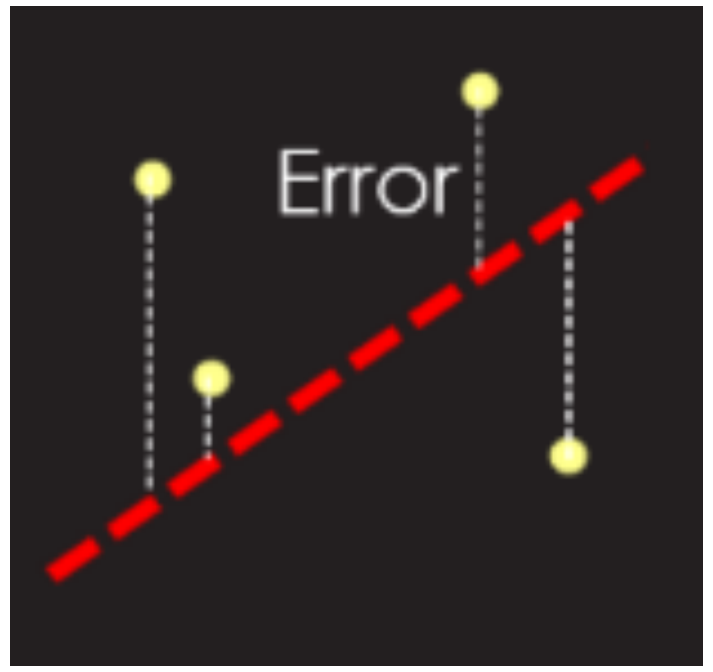


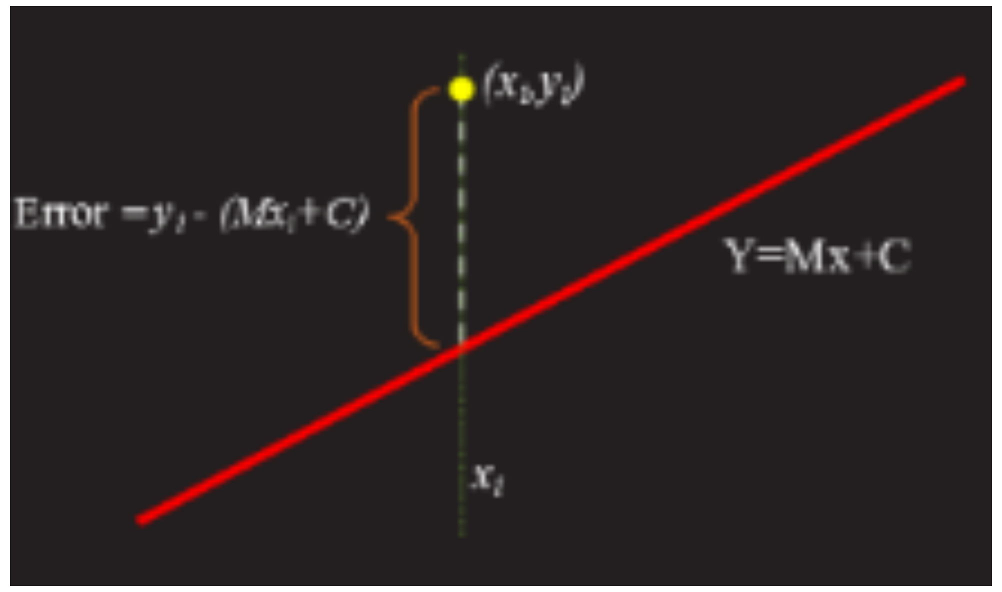
LINEAR REGRESSION

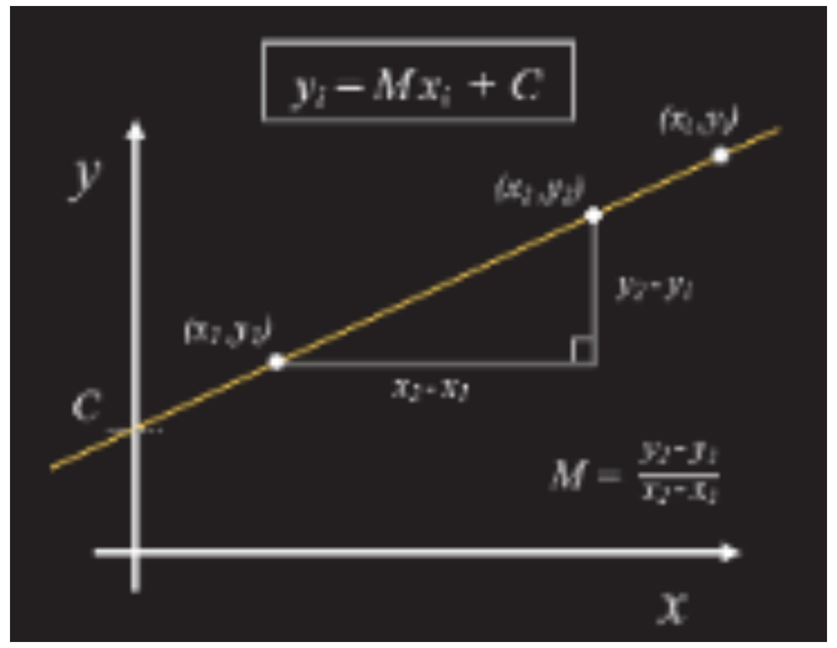
Goal of linear regression is to fit a line to the data that minimizes total squared error

Minimize RSS: $$\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$









LINEAR REGRESSION

Goal of linear regression is to fit a line to the data that minimizes total squared error

- Assumptions:
  - Linear relationship
  - Multivariate normality
  - No or little multicollinearity
  - No autocorrelation
  - Homoscedastic (constant variance in residuals)
  - $p < n$ (there is no unique solution when $p > n$

LINEAR REGRESSION

Goal of linear regression is to fit a line to the data that minimizes total squared error

- Assumptions:
  - Linear relationship
  - Multivariate normality
  - No or little multicollinearity ✔
  - No autocorrelation
  - Homoscedastic (constant variance in residuals)
  - $p < n$ (there is no unique solution when $p > n$ ✔

Two problems we run into with text
Both cause serious problems with model stability & reliability

REGULARIZED REGRESSION

- Three **shrinkage methods** have been developed to resolve these (and other) concerns: <u>*Lasso*</u>, <u>*Ridge regression*</u>, <u>*Elastic nets*</u>.
- **Constrains** or **regularizes** the coefficient estimates, or equivalently, that **shrinks** the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.

REGULARIZED REGRESSION

- Three **shrinkage methods** have been developed to resolve these (and other) concerns: *Lasso*, *Ridge regression*, *Elastic nets*.

- **Constrains** or **regularizes** the coefficient estimates, or equivalently, that **shrinks** the coefficient estimates towards zero.

  - Lasso minimizes: $\mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$

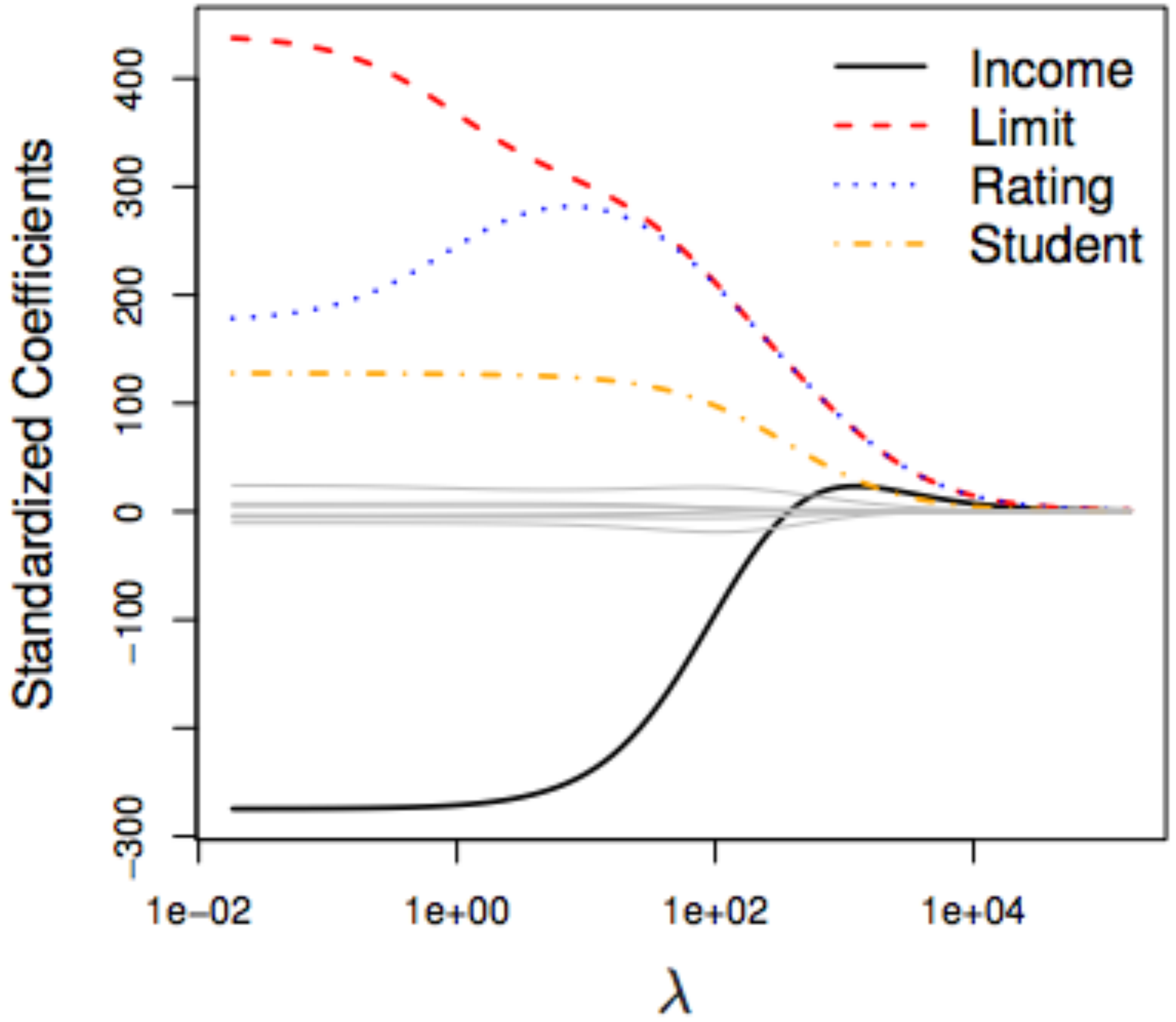
  - Ridge minimizes: $\mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2.$

lambda (λ) becomes a tuning parameter:

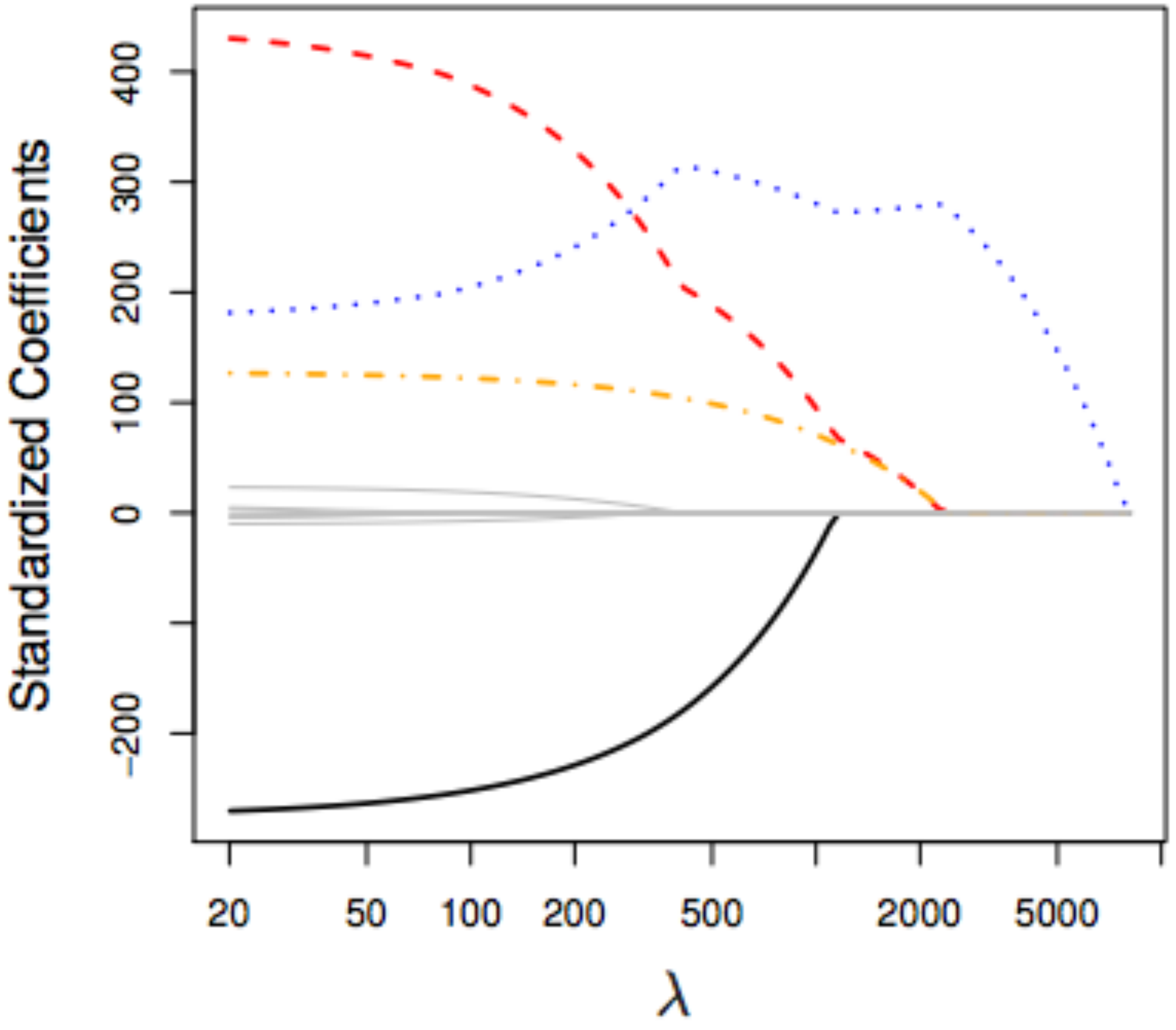
- $\lambda \cong 0$: equates to OLS regression
- $\lambda \cong \infty$: equates to the mean as all coefficients are forced to 0

REGULARIZED REGRESSION

Ridge Regression



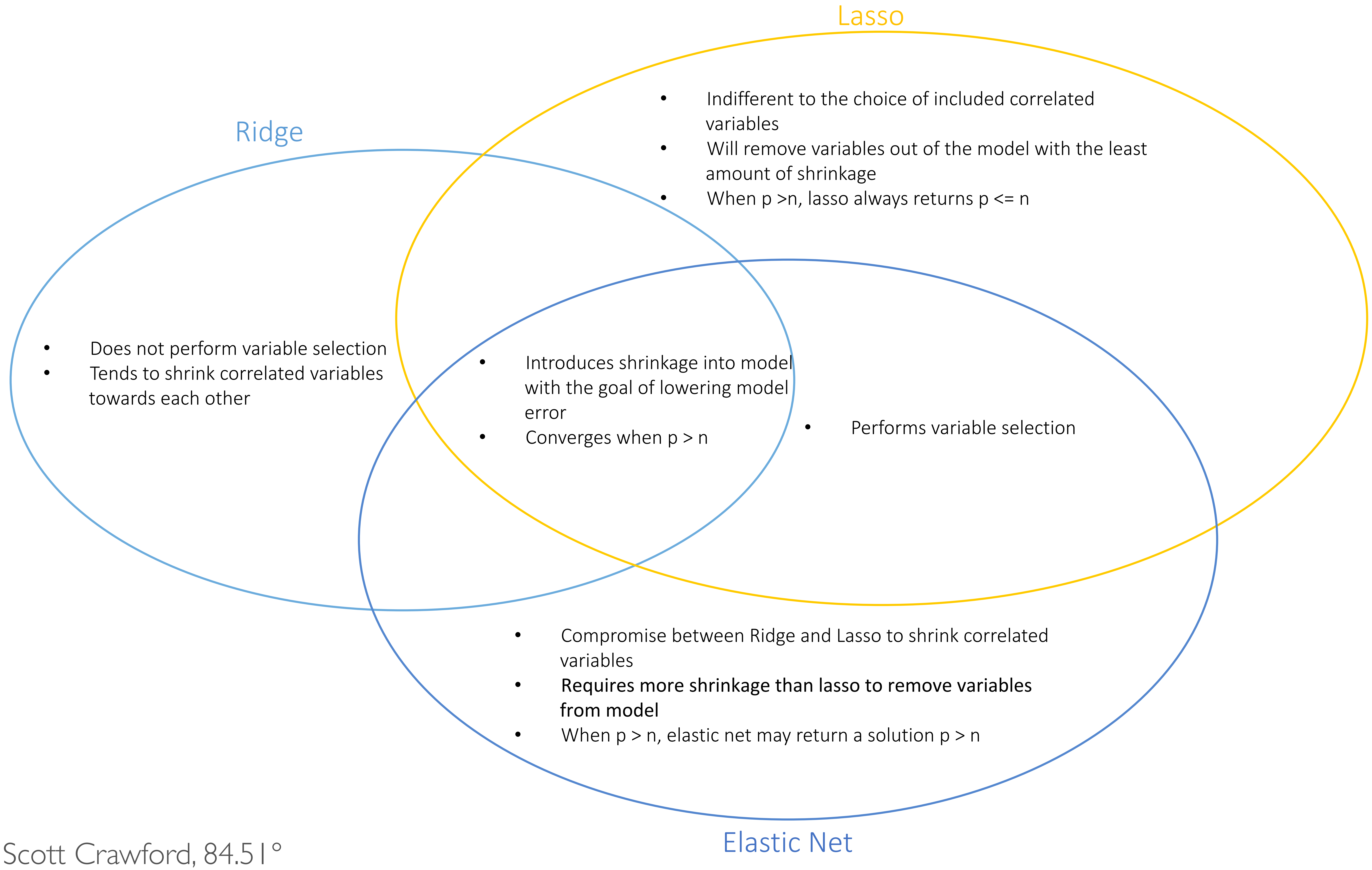
Lasso Regression



REGULARIZED REGRESSION

Elastic Net

REGULARIZED REGRESSION



Source: Mike Payne & Scott Crawford, 84.51°

REGULARIZED REGRESSION

Q: When do I use one versus the other?

When to use Ridge:

- You have reason to believe all coefficients should be kept in the model.
- You want correlated variables to be shrunk towards each other.
- Lower error rate than Lasso and Elastic Net

When to use Lasso:

- Perform variable selection - You have reason to believe a small subset of variables should be included in the final model
- p >> n
- Highly correlated variables that can be "arbitrarily" dropped
- Lower error rate than Ridge and Elastic Net

When to use Elastic net:

- A balance between the effects of Ridge and Lasso
- p >> n but okay with a solution p > n
- Lower error rate than Ridge and Lasso
- This is currently the most popular version of regularized regression

Source: Mike Payne & Scott Crawford, 84.51°

REGULARIZED REGRESSION

Q: When do I use one versus the other?

A: Most of the time *try them all*!

When to use Ridge:

- You have reason to believe all coefficients should be kept in the model.
- You want correlated variables to be shrunk towards each other.
- Lower error rate than Lasso and Elastic Net

When to use Lasso:

- Perform variable selection - You have reason to believe a small subset of variables should be included in the final model
- p >> n
- Highly correlated variables that can be "arbitrarily" dropped
- Lower error rate than Ridge and Elastic Net

When to use Elastic net:

- A balance between the effects of Ridge and Lasso
- p >> n but okay with a solution p > n
- Lower error rate than Ridge and Lasso
- This is currently the most popular version of regularized regression

Source: Mike Payne & Scott Crawford, 84.51°

IMPLEMENTATION

```r
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

`set.seed` provides reproducibility

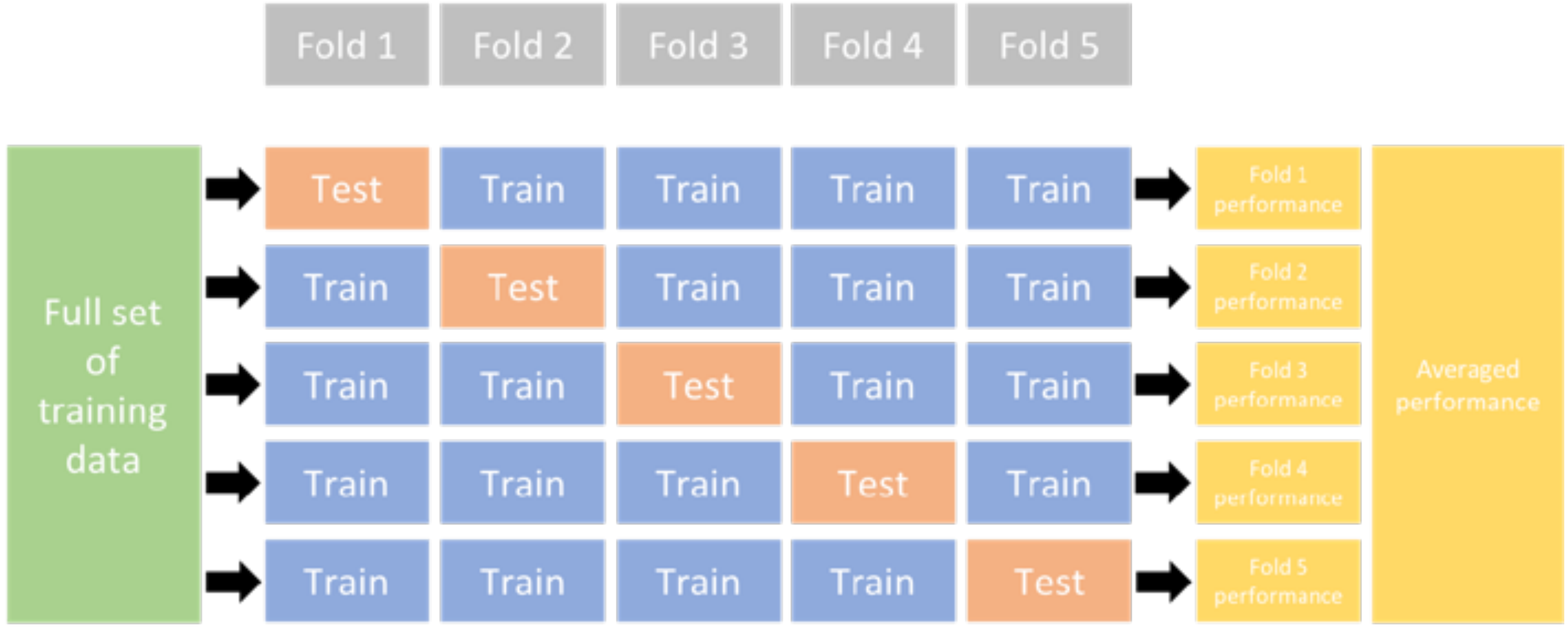
IMPLEMENTATION

```
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

`cv.glmnet` provides built-in cross validation.

There is an equivalent `glmnet` function w/o built-in cv.

We use `nfolds` to tell the model how many cv folds to apply.



Source: Brandon Greenwell, Ascend Innovations

IMPLEMENTATION

```r
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

`x` & `y` feed in our training and response data.

When performing classification we want our response to be encoded `as.factor`

IMPLEMENTATION

```
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

`alpha` is the parameter that tells it which of the regularization methods to use:

- `alpha = 1`: Lasso
- `alpha = 0`: Ridge
- `0 < alpha < 1`: Elastic net

IMPLEMENTATION

```
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

`family` describes the distribution of the y response variable.

Many to choose from but most common:

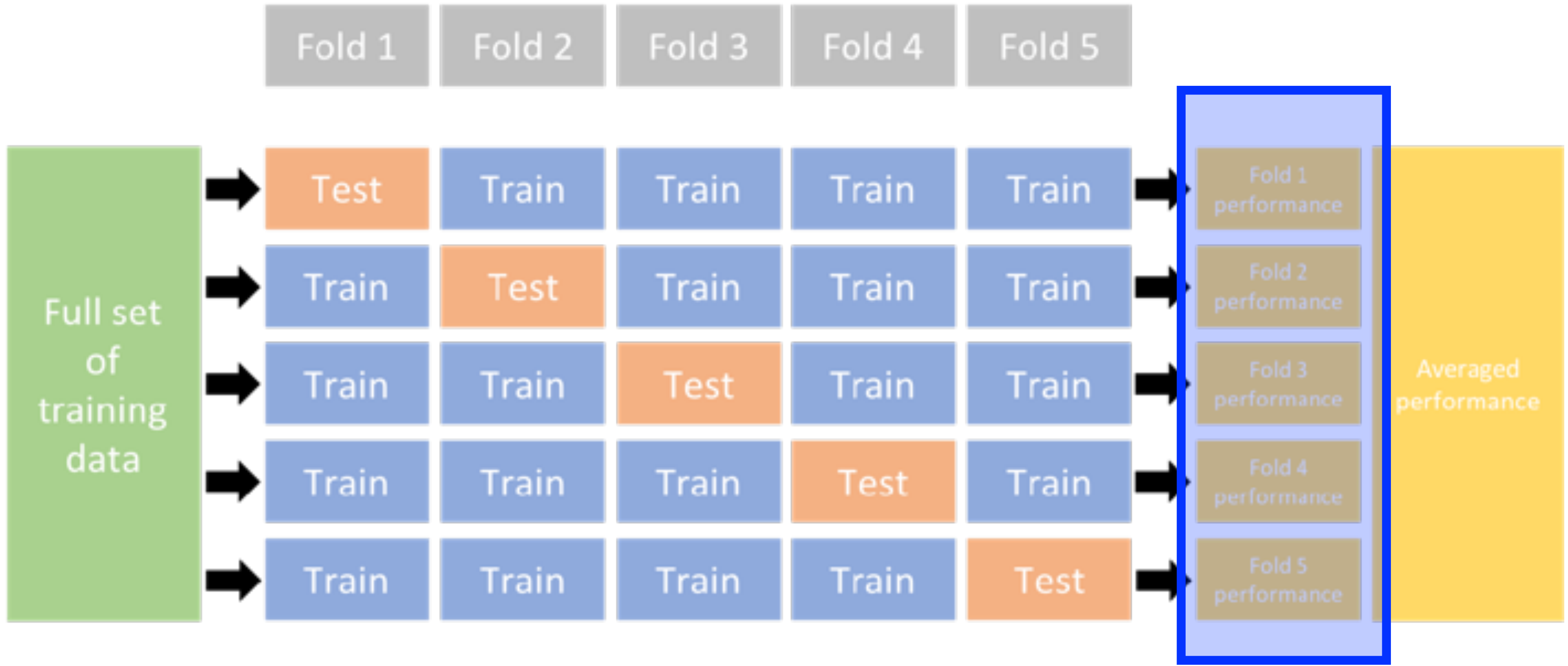
- `gaussian`: continuous
- `binomial`: binomial for two level classification

IMPLEMENTATION

```
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

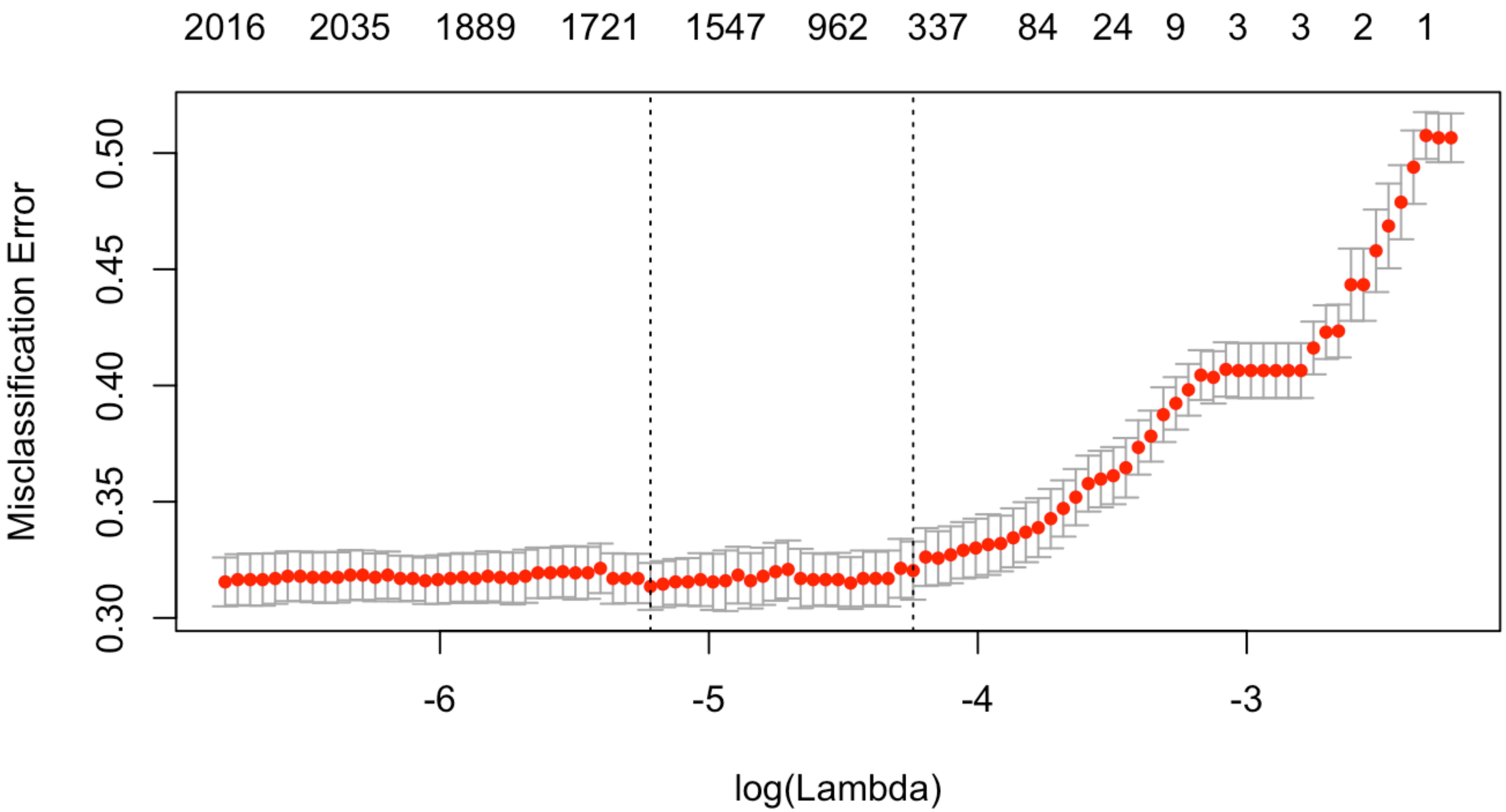
`type.measure` states the performance measure used during the cv process:

- `class`: minimize misclassification rate
- `AUC`: maximize area under the curve
- `MSE`: minimize mean squared error
- `class`: minimize absolute square error



ASSESSMENT

```r
set.seed(123)

cv.lasso <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 1,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)

plot(cv.lasso)
```

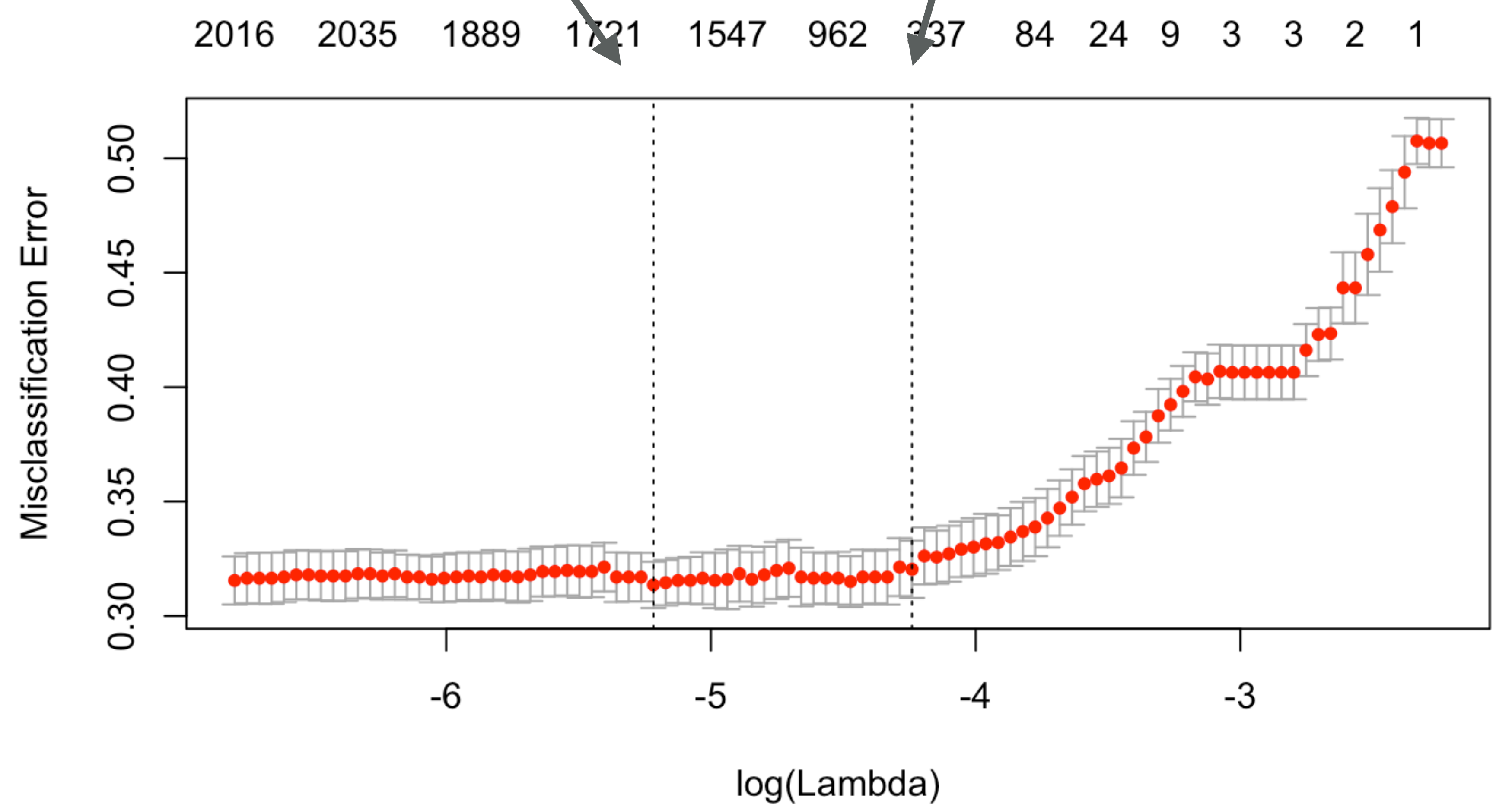


ASSESSMENT

```
# predict lasso
pred.lasso <- predict(
  cv.lasso,
  train.x,
  type = "class",
  s = cv.lasso$lambda.1se
)
```

To get predictions we feed predict:

- `cv.lass`: model
- `train.x`: new feature set
- `type`: prediction type (class or probability)
- `s`: `lambda.min` or `lambda.1se`

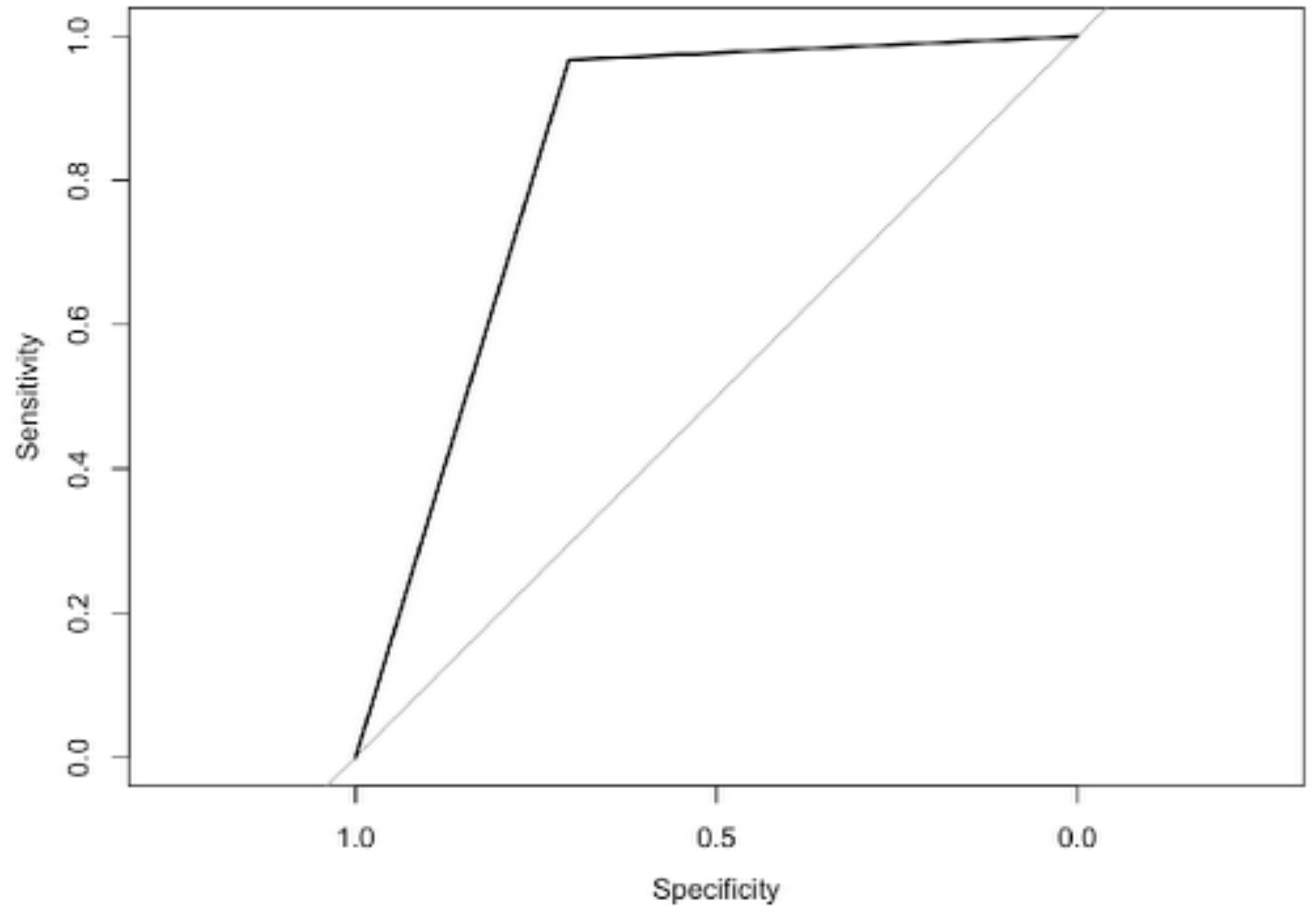


ASSESSMENT

```
# predict lasso
pred.lasso <- predict(
  cv.lasso,
  train.x,
  type = "class",
  s = cv.lasso$lambda.1se
)

auc.lasso <- roc(train.y, as.numeric(pred.lasso))
auc.lasso
Area under the curve: 0.8359

plot(auc.lasso)
```

We can assess the area under the curve with `pROC::roc`

- Provides us with how well the model balances:
  - True positives
  - True negatives



ASSESSMENT

```
# confusion matrix
confusion <- table(pred.lasso, train.y)
confusion
          train.y
pred.lasso   0   1
         0 731  33
         1 307 986

# overall accuracy
sum(diag(confusion)) / sum(confusion)
[1] 0.8347107
```

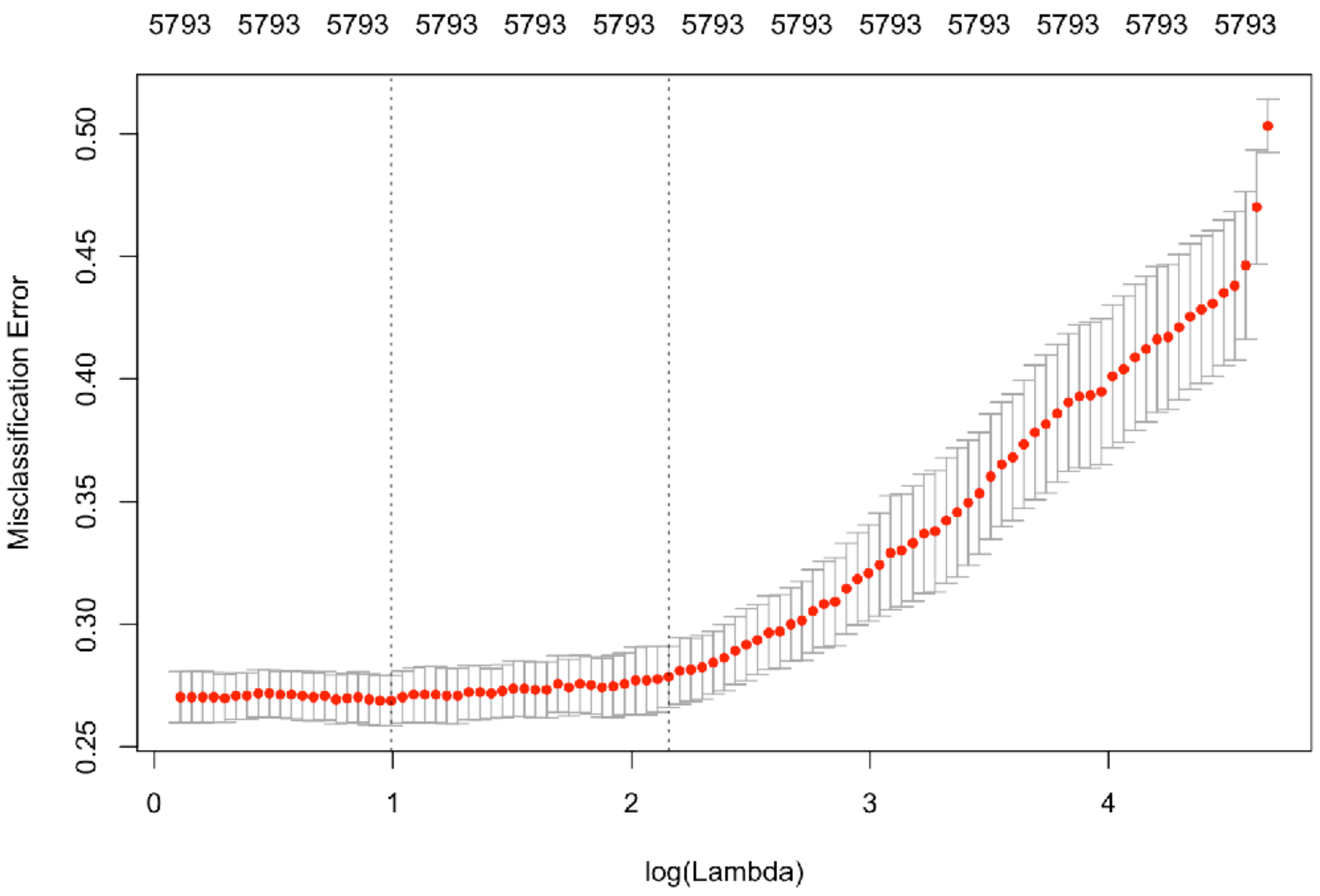
We can also the confusion matrix

YOUR TURN 1!

Apply a Ridge regression model (hint: alpha = ?) to the same training data and plot the misclassification rate. Which range of lambdas performs the best?

SOLUTION

```r
# apply Ridge model
set.seed(123)

cv.ridge <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = 0,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)

# plot misclassification error
plot(cv.ridge)
```

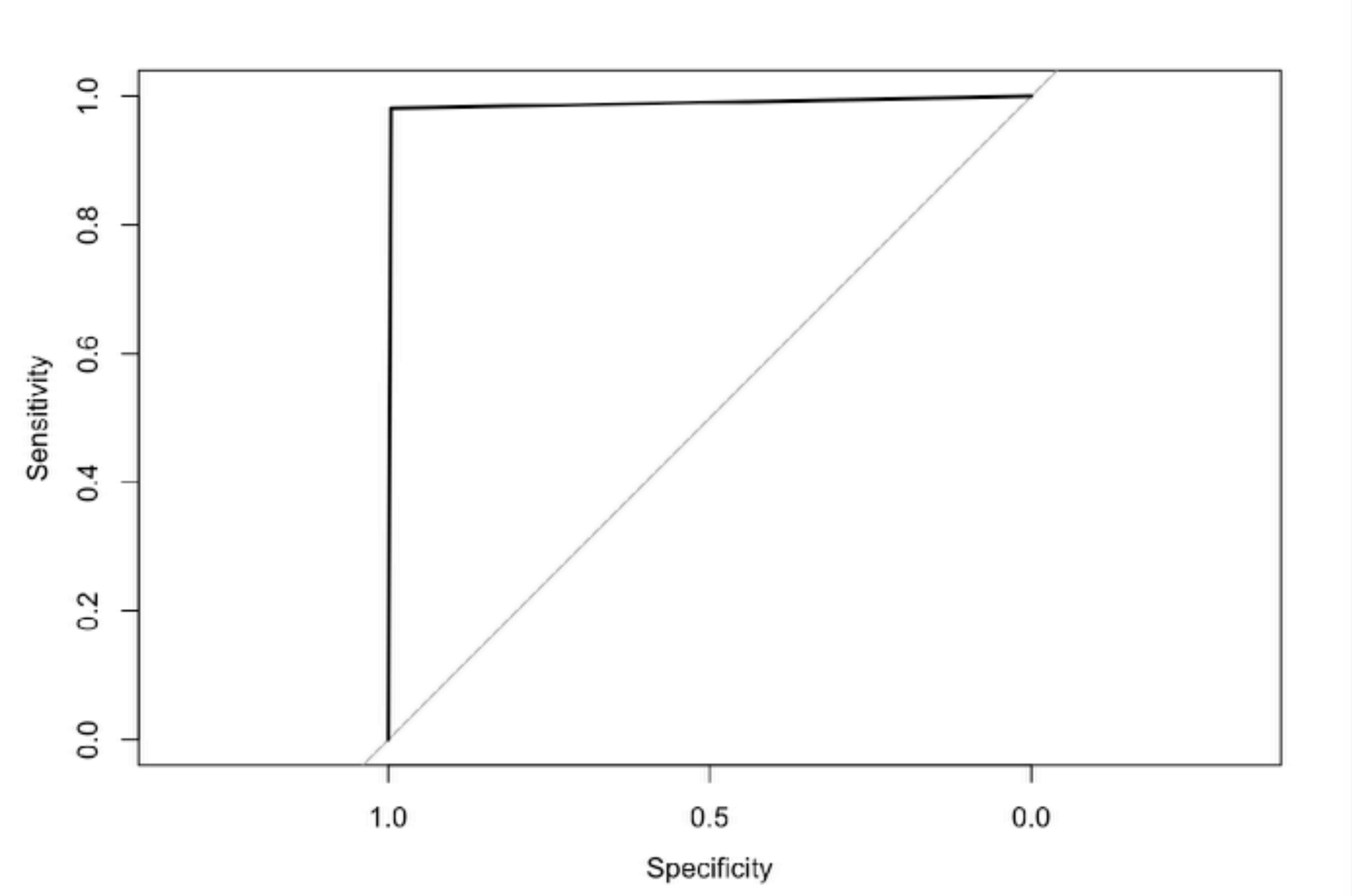


YOUR TURN 2!

Now get the predictions for the `train.x` *data using* `s = cv.ridge$lambda.1se`*. What is the area under the curve? Can you create the confusion matrix?*

SOLUTION

```
# predict ridge
pred.ridge <- predict(cv.ridge, train.x, type = "class", s = cv.ridge$lambda.1se)

# area under the curve
auc.ridge <- roc(train.y, as.numeric(pred.ridge))
auc.ridge
Data: as.numeric(pred.ridge) in 1038 controls (train.y 0) < 1019 cases (train.y 1).
Area under the curve: 0.9888

plot(auc.ridge)
```



ELASTIC NET

```
# Let's assess how an elastic net compares
```

We can apply an elastic net model by choosing `0 < alpha < 1`

But what value do we pick?

ELASTIC NET

```r
# Let's assess how an elastic net compares
tuning <- expand.grid(
  alpha    = seq(0, 1, by = .01),
  accuracy = NA
)

head(tuning)
  alpha accuracy
1  0.00       NA
2  0.01       NA
3  0.02       NA
4  0.03       NA
5  0.04       NA
6  0.05       NA
```

We can apply an elastic net model by choosing `0 < alpha < 1`

1. Let's create a tuning grid

ELASTIC NET

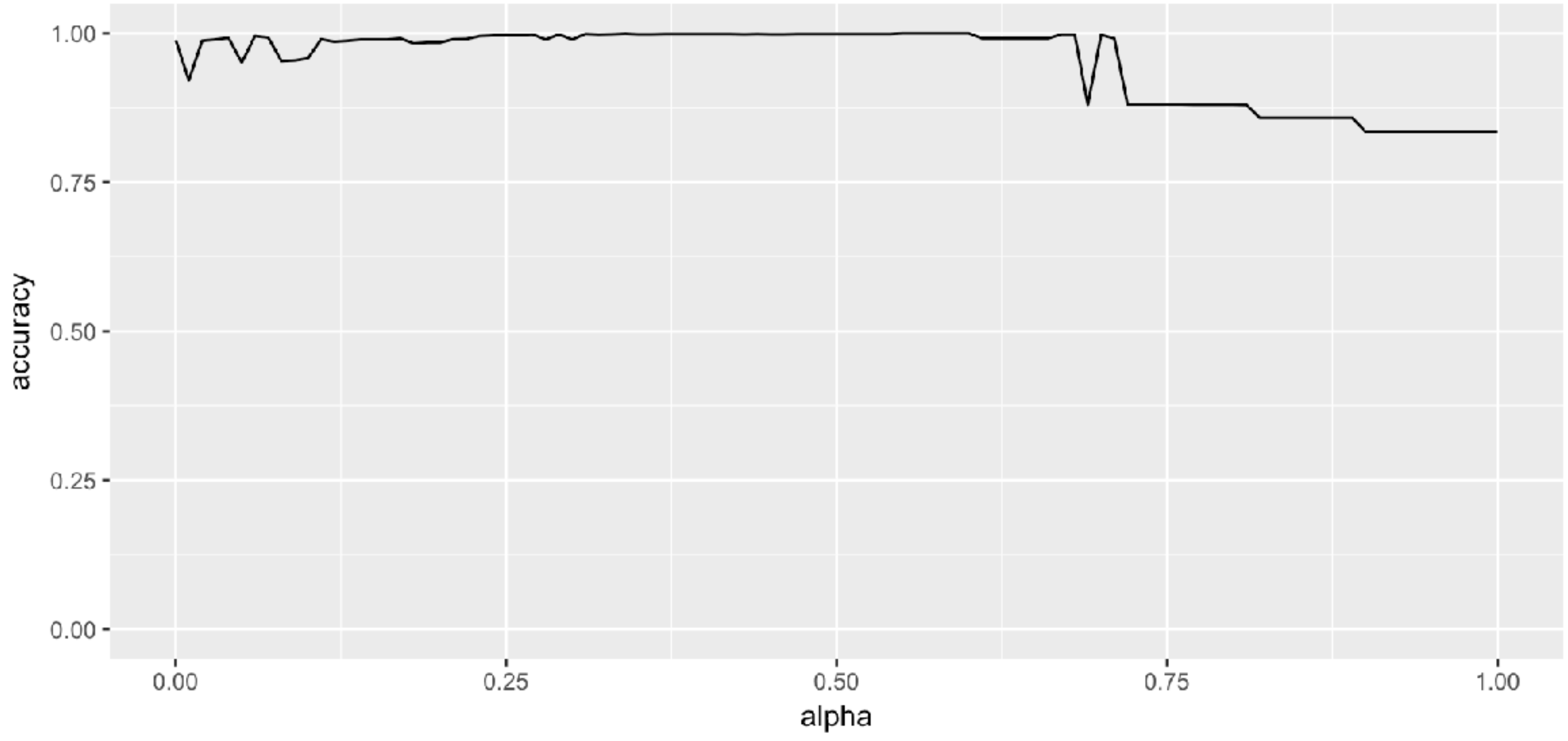
```
# Let's assess how an elastic net compares
for(i in seq_along(tuning$alpha)) {

  set.seed(123)

  cv <- cv.glmnet(
    x = train.x,
    y = as.factor(train.y),
    alpha = tuning$alpha[i],
    family = "binomial",
    nfolds = 10,
    type.measure = "class"
  )

  pred <- predict(cv, train.x, type = "class", s = cv$lambda.1se)
  confusion <- table(pred, train.y)
  tuning$accuracy[i] <- sum(diag(confusion)) / sum(confusion)
}
```

We can apply an elastic net model by choosing `0 < alpha < 1`

1. Let's create a tuning grid
2. Now we'll loop through each alpha value, apply a glmnet model, compute the accuracy and add that value into our tuning grid.

ELASTIC NET

```r
# Let's assess how an elastic net compares
ggplot(tuning, aes(alpha, accuracy)) +
  geom_line() +
  ylim(c(0, 1))
```



We can apply an elastic net model by choosing `0 < alpha < 1`

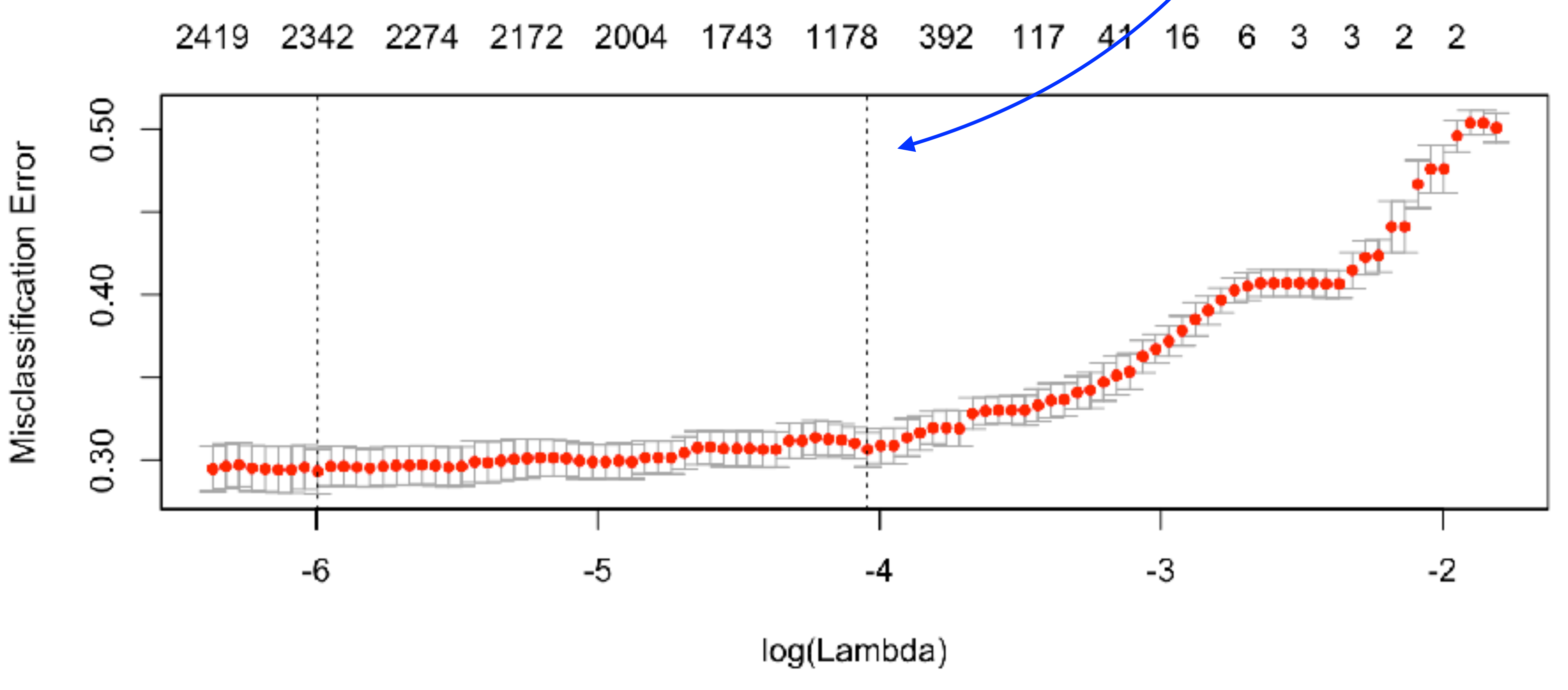
1. Let's create a tuning grid
2. Now we'll loop through each alpha value, apply a glmnet model, compute the accuracy and add that value into our tuning grid.
3. Now we can plot our results across all alpha values.

ELASTIC NET

```
# let's apply an elastic net with alpha = .65
cv <- cv.glmnet(
  x = train.x,
  y = as.factor(train.y),
  alpha = .65,
  family = "binomial",
  nfolds = 10,
  type.measure = "class"
)
```

Let's apply an elastic net with `alpha = .65`

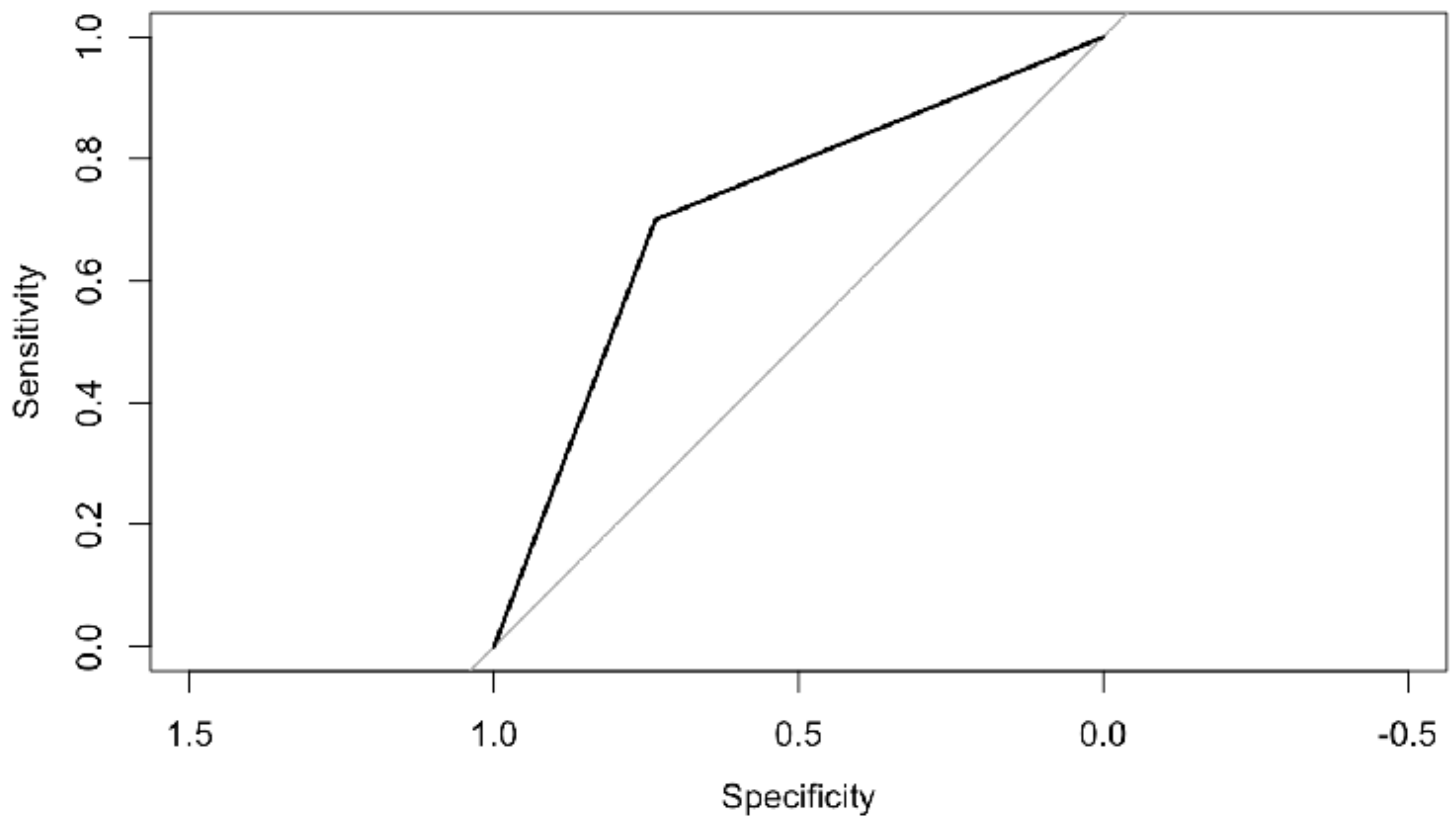
We can reduce our feature set down to about 1000 if we wanted/needed to.



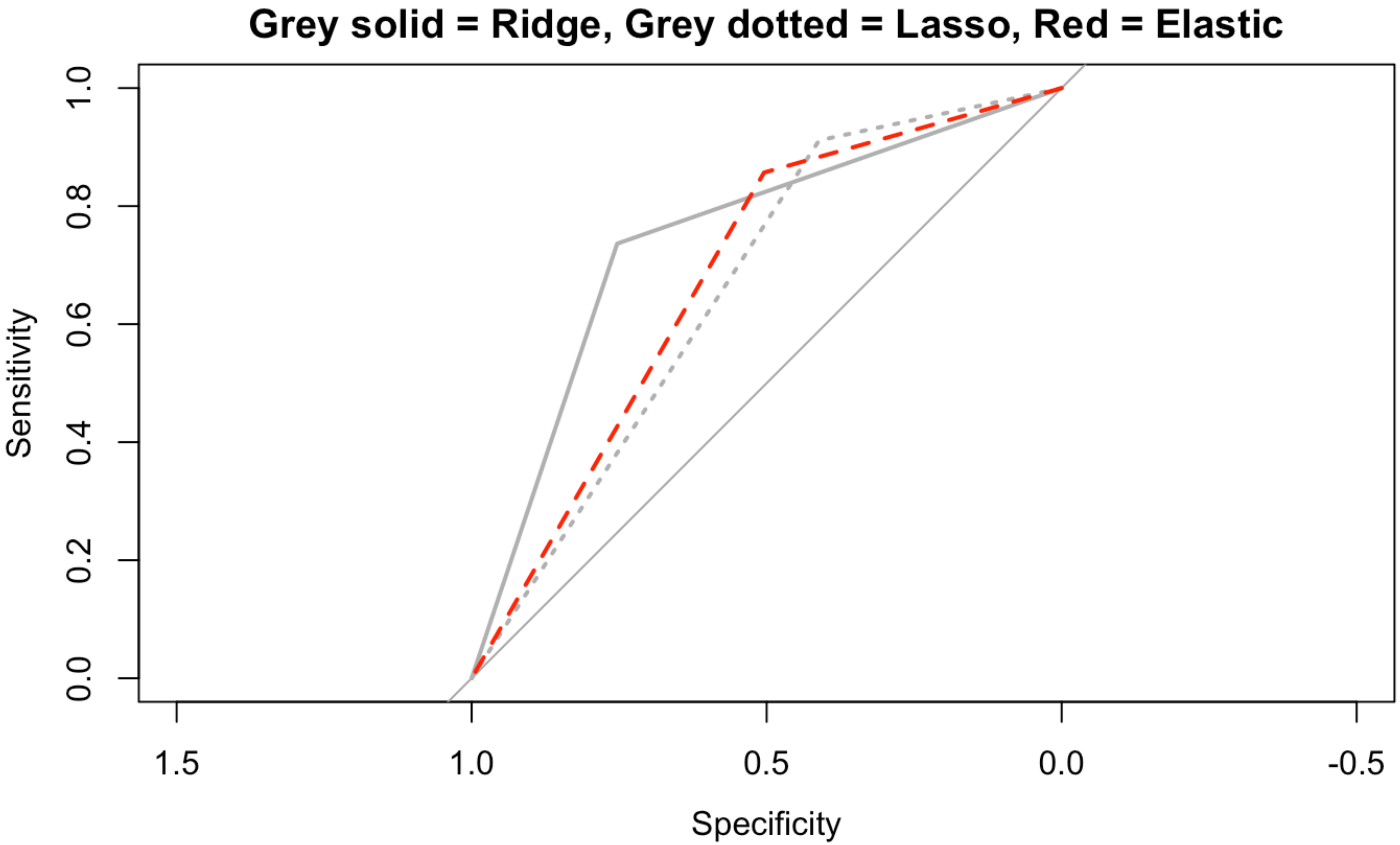
ELASTIC NET

```
# Now let's predict…but with our test set
pred.test <- predict(
  cv,
  test.x,
  type = "class",
  s = cv$lambda.min
)

auc <- roc(test.y, as.numeric(pred.test))
auc
Data: as.numeric(pred.test) in 450 controls (test.y 0) <
440 cases (test.y 1).
Area under the curve: 0.7178

plot(auc)
```

Now let's assess how well our model generalizes to an unseen data set (`test.x`)



ELASTIC NET


Grey solid = Ridge, Grey dotted = Lasso, Red = Elastic
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.5
1.0
0.5
0.0
-0.5
Specificity

UNDERSTANDING THE RESULTS

So we found a decent model, now how do we understand the results?

UNDERSTANDING THE RESULTS

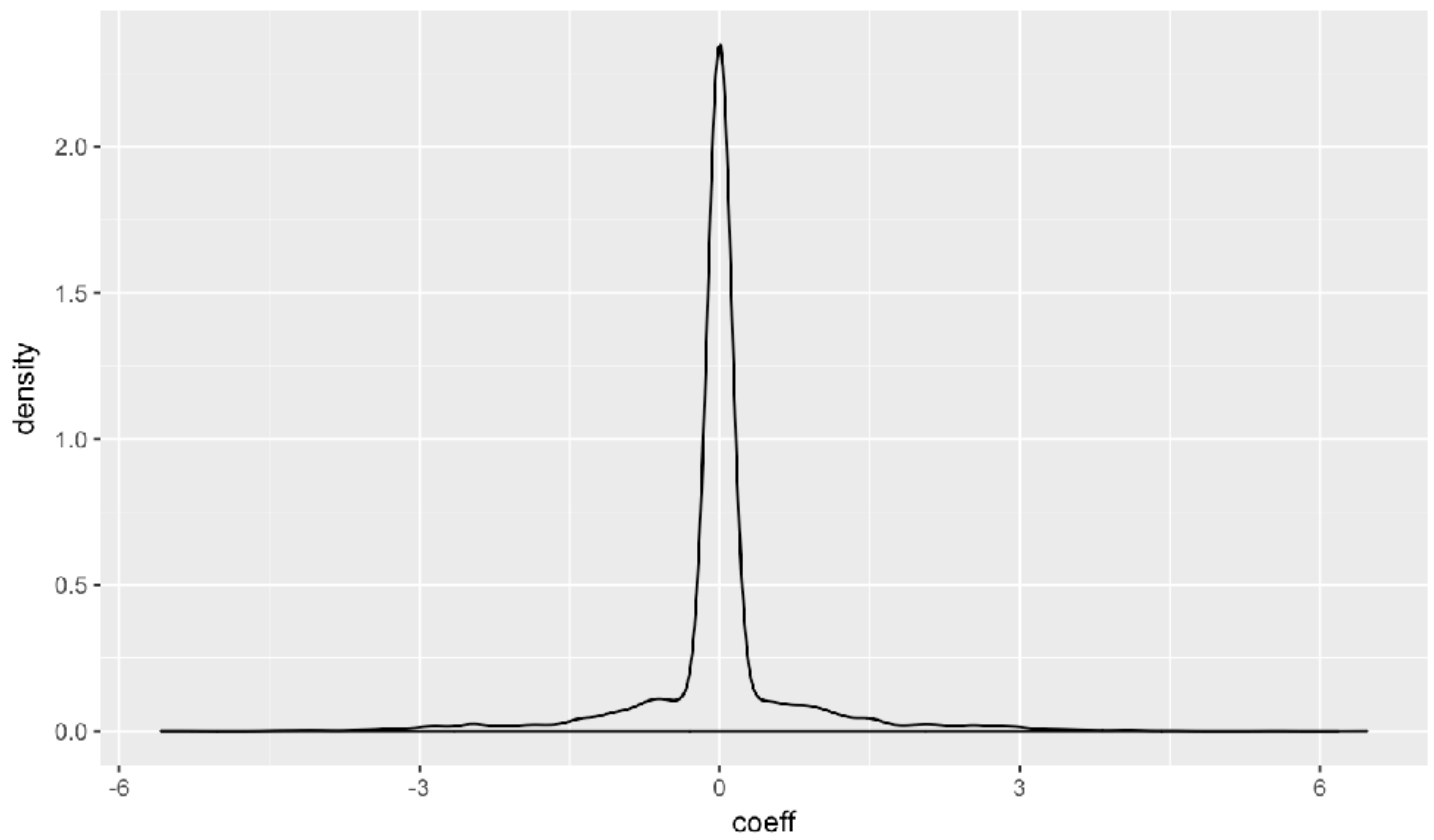
```
# finding the most impactful words
glmnet.coef <- as.matrix(coef(cv, s = "lambda.min"))
glmnet.coef <- tibble(
  words = row.names(glmnet.coef),
  coeff = glmnet.coef[, 1]
) %>%
  arrange(desc(coeff)) %>%
  mutate(words = fct_inorder(words))

glmnet.coef
# A tibble: 7,150 x 2
   words     coeff
   <fct>     <dbl>
 1 overthrow  6.47
 2 pummel     5.55
 3 u.k        4.54
 4 doug       4.54
```

We can extract each feature (word) and the coefficient

UNDERSTANDING THE RESULTS

```
# finding the most impactful words
summary(glmnet.coef$coeff)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-5.581671  0.000000  0.000000  0.006956  0.000000  6.472727

ggplot(glmnet.coef, aes(coeff)) +
  geom_density()
```

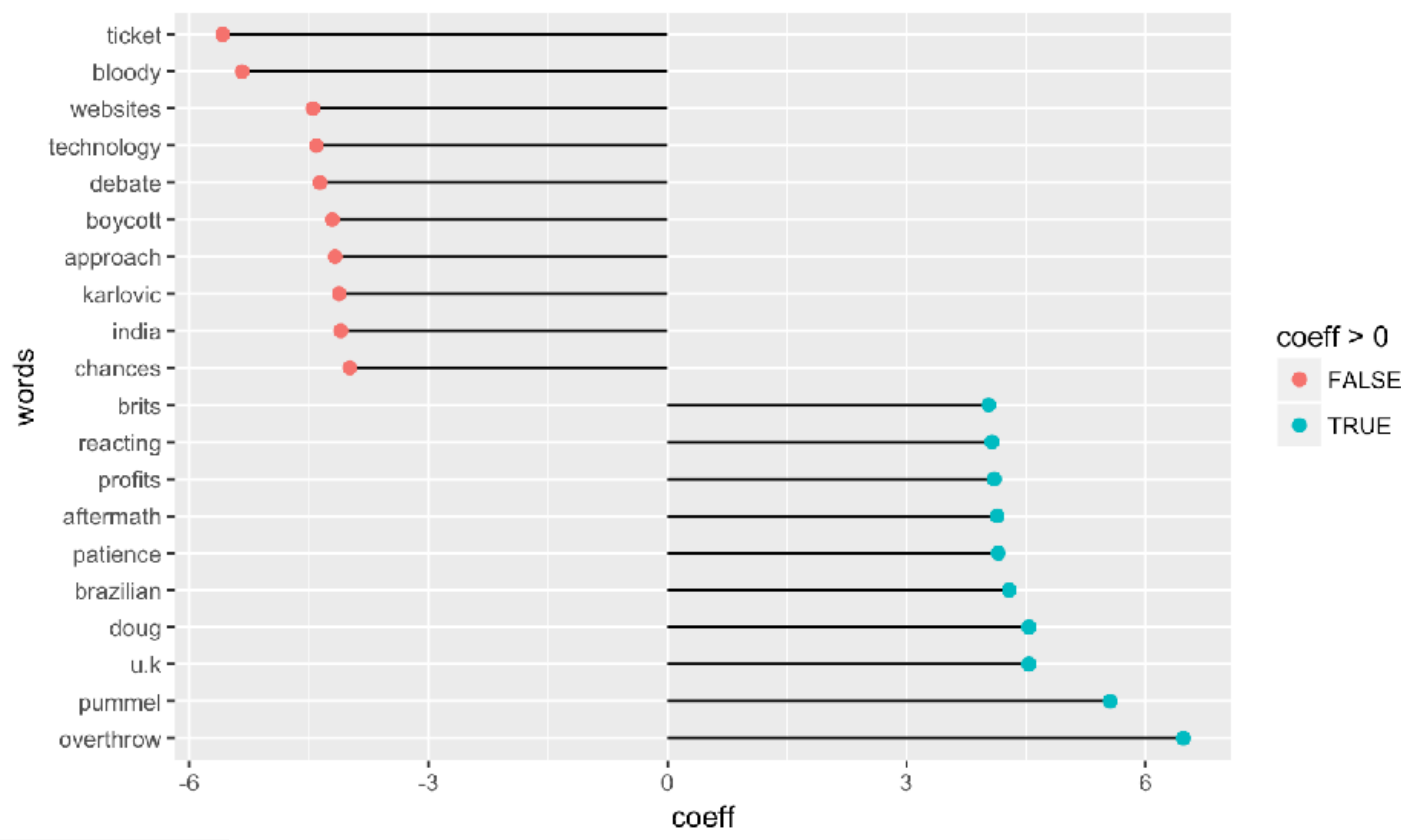
We see that many of the features have coefficients that have been shrunk to zero



UNDERSTANDING THE RESULTS

```
# top 10 positive and negative words
rbind(
  top_n(glmnet.coef, 10),
  top_n(glmnet.coef, -10)
) %>%
  ggplot(aes(x = coeff, y = words)) +
  geom_segment(aes(yend = words, xend = 0)) +
  geom_point(aes(color = coeff > 0), size = 2)
```

We can look at the top 20 influential words



UNDERSTANDING THE RESULTS

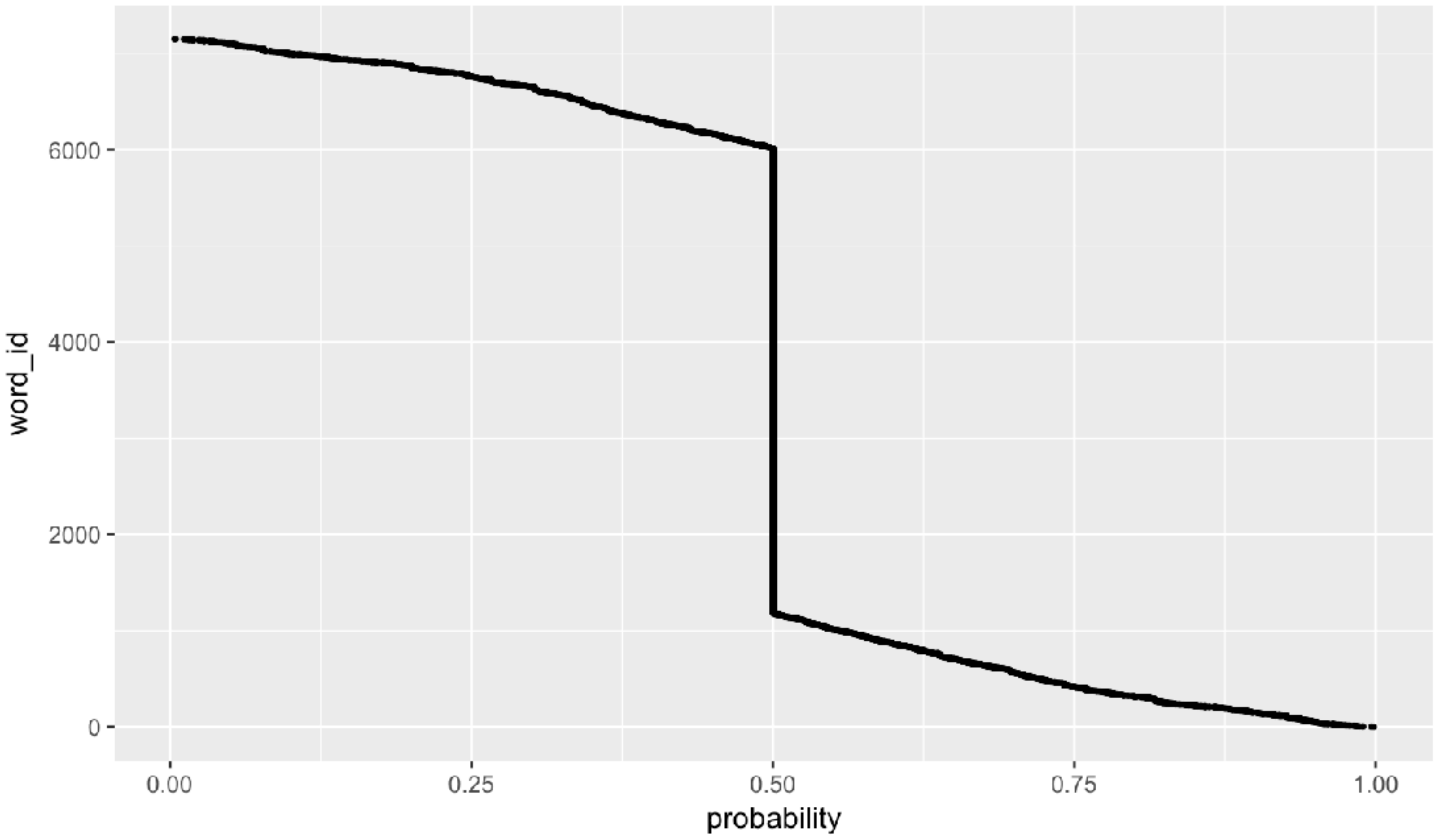
```
# convert to probabilities
glmnet_coef <- glmnet.coef %>%
  mutate(
    probability = arm::invlogit(coeff),
    word_id     = row_number(words)
    )

glmnet_coef
# A tibble: 7,150 x 4
   words     coeff probability word_id
   <fct>     <dbl>       <dbl>   <int>
 1 overthrow  6.47       0.998       1
 2 pummel     5.55       0.996       2
 3 u.k        4.54       0.989       3
 4 doug       4.54       0.989       4
 5 brazilian  4.29       0.986       5
 6 patience   4.15       0.984       6
```

Let’s compute the probabilities

UNDERSTANDING THE RESULTS

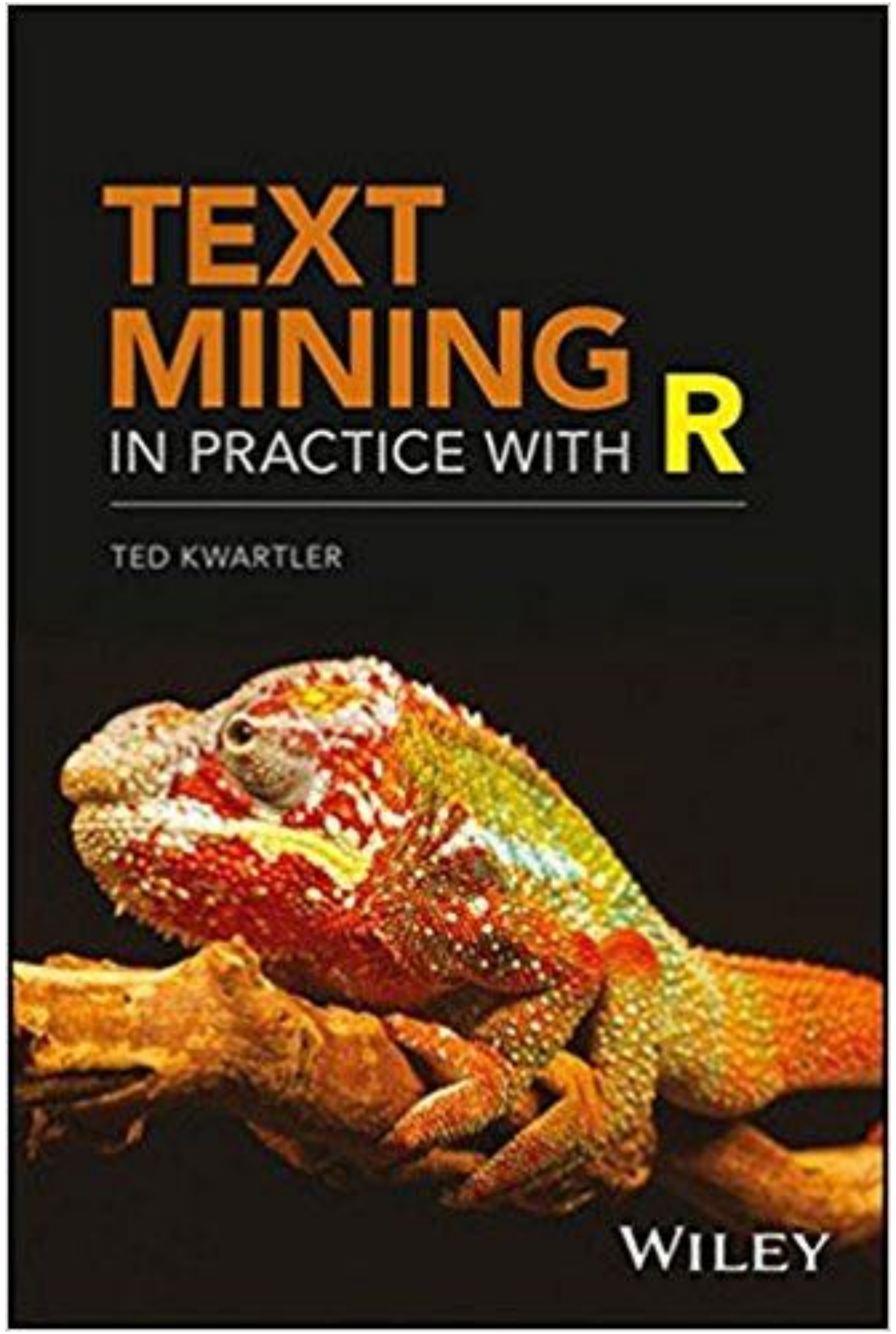
```r
# plot word probability distributions
ggplot(glmnet_coef, aes(probability, word_id)) +
  geom_point(size = .5)
```

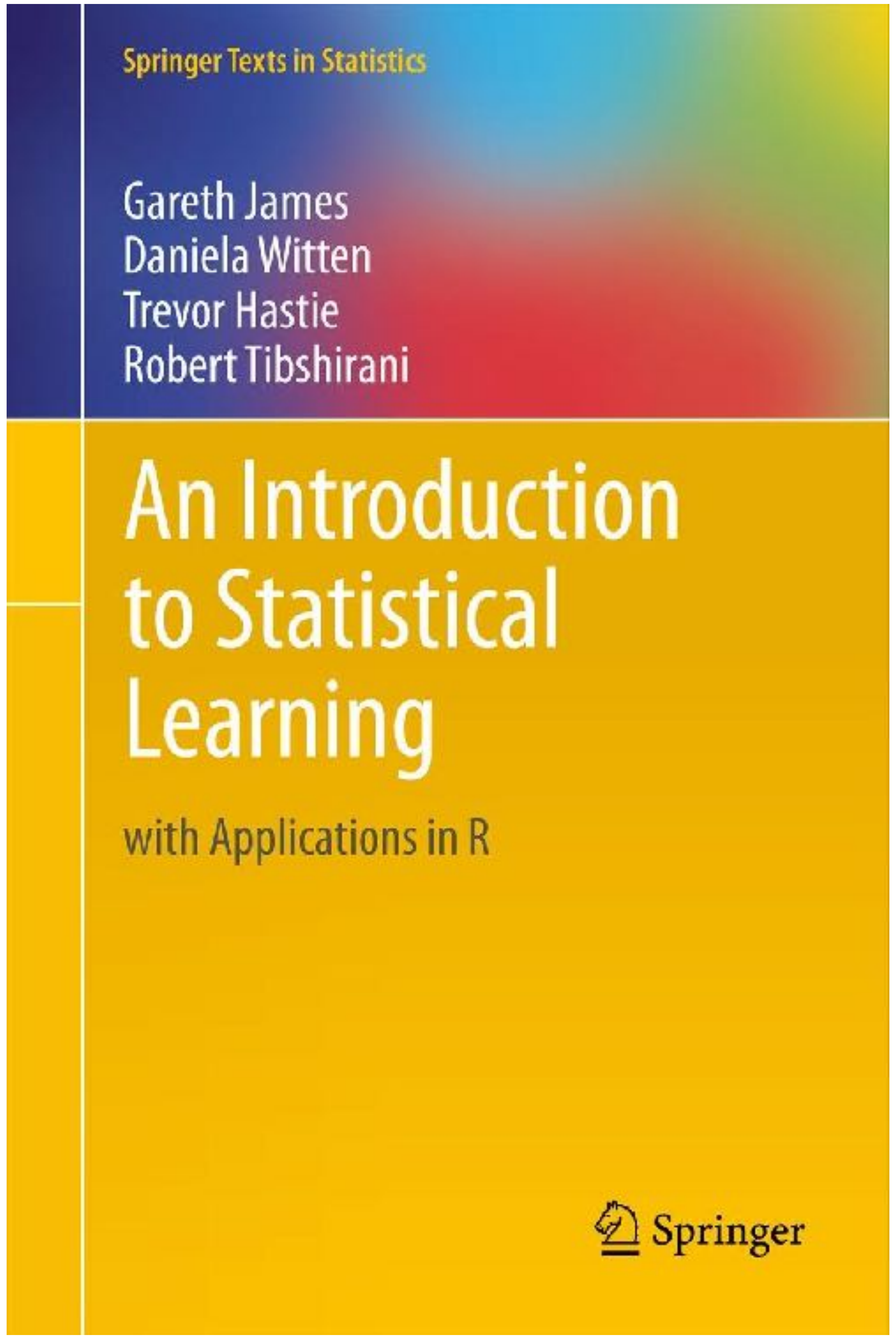
Let's visualize the probabilities

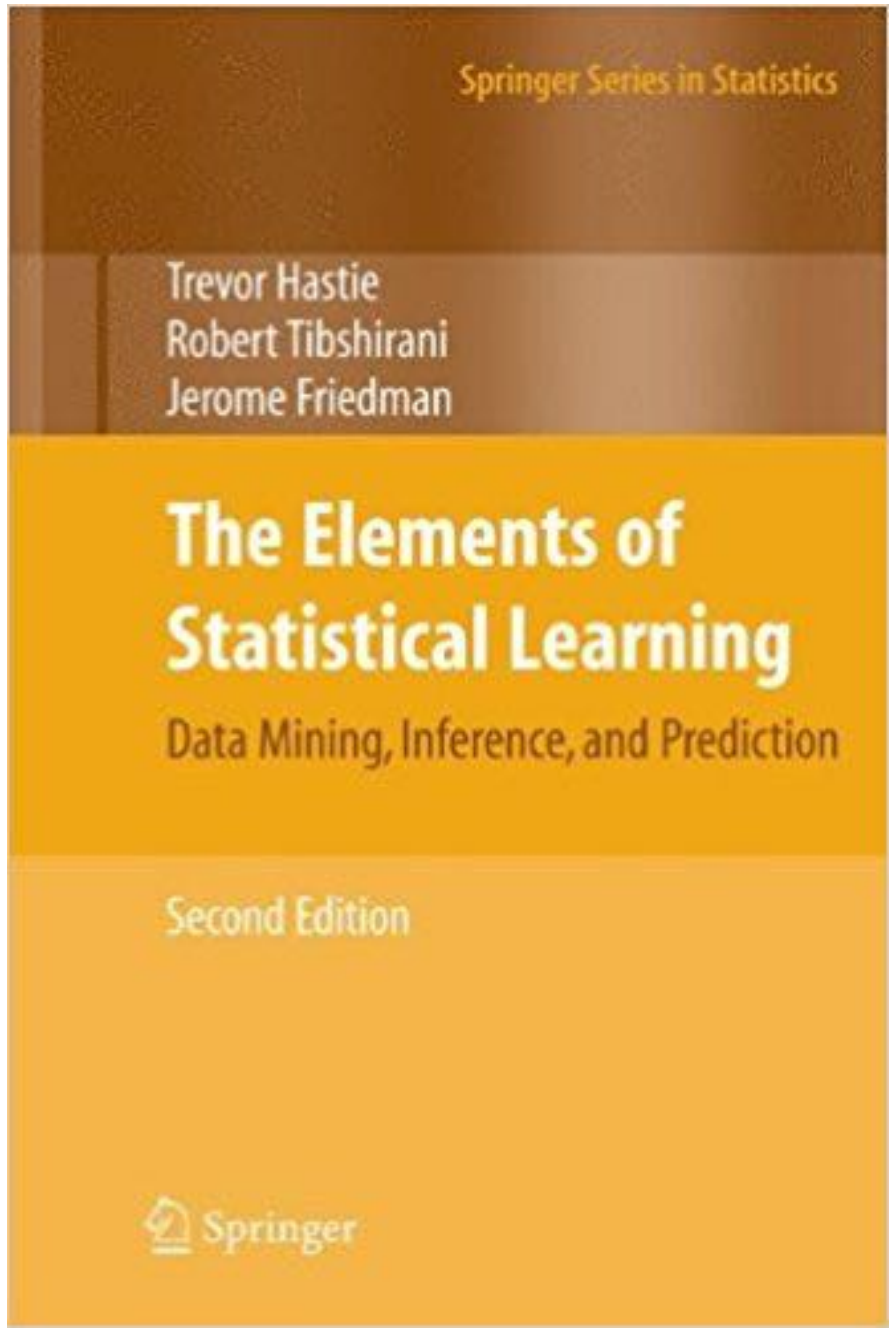


SO LITTLE TIME!

LEARN MORE


TEXT MINING IN PRACTICE WITH R
TED KWARTLER
WILEY


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer

WHAT TO REMEMBER

FUNCTIONS TO REMEMBER

Operator/Function	Description
`Matrix::Matrix`	Create a sparse matrix
`glmnet::cv.glment`	Compute a cross validated regularized regression model (Ridge: `alpha = 0`; Lasso: `alpha = 1`; Elastic net: `0 < alpha < 1`)
`pROC::roc`	Create a ROC curve and compute area under the curve
`predict`	Predict response values for a new feature set