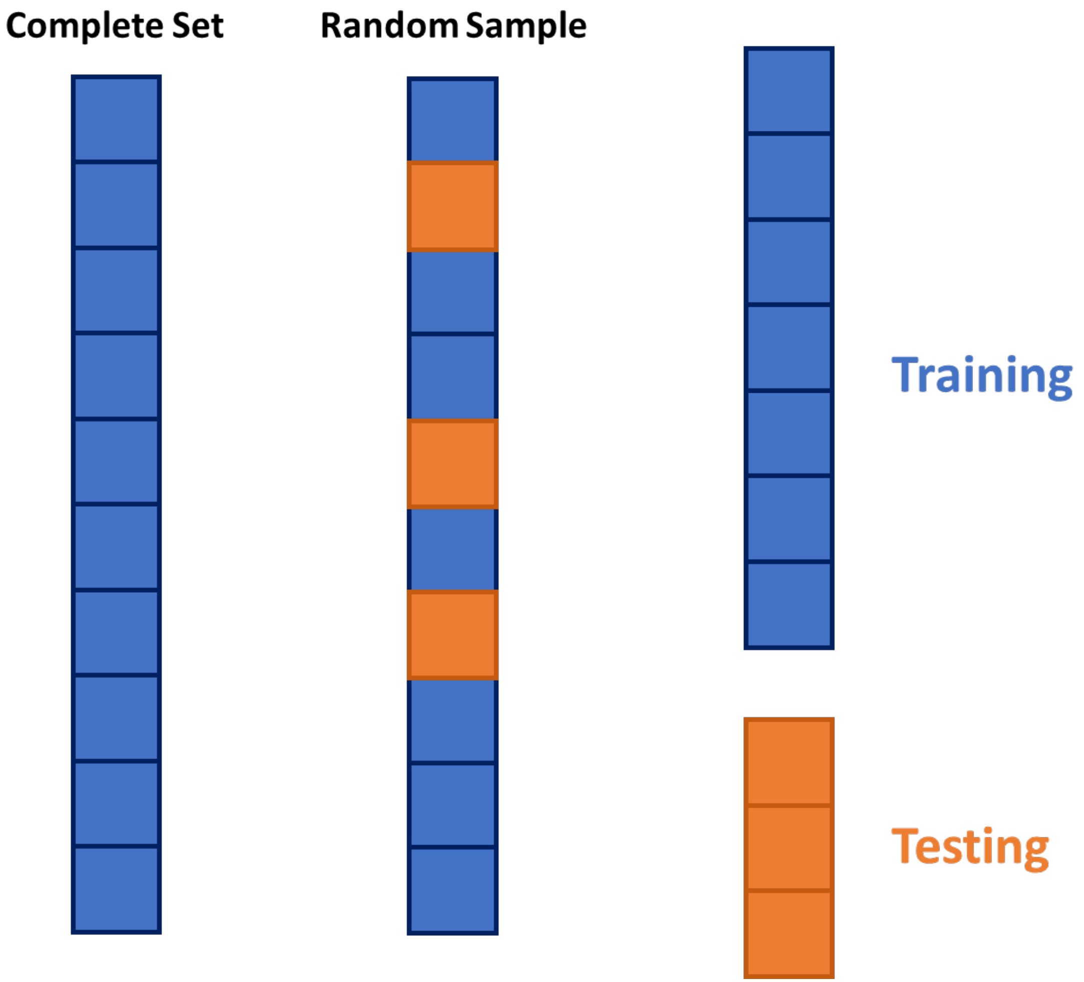
RESAMPLING, MODEL SELECTION, & REGULARIZATION

Dr. Aric LaBarr

Institute for Advanced Analytics

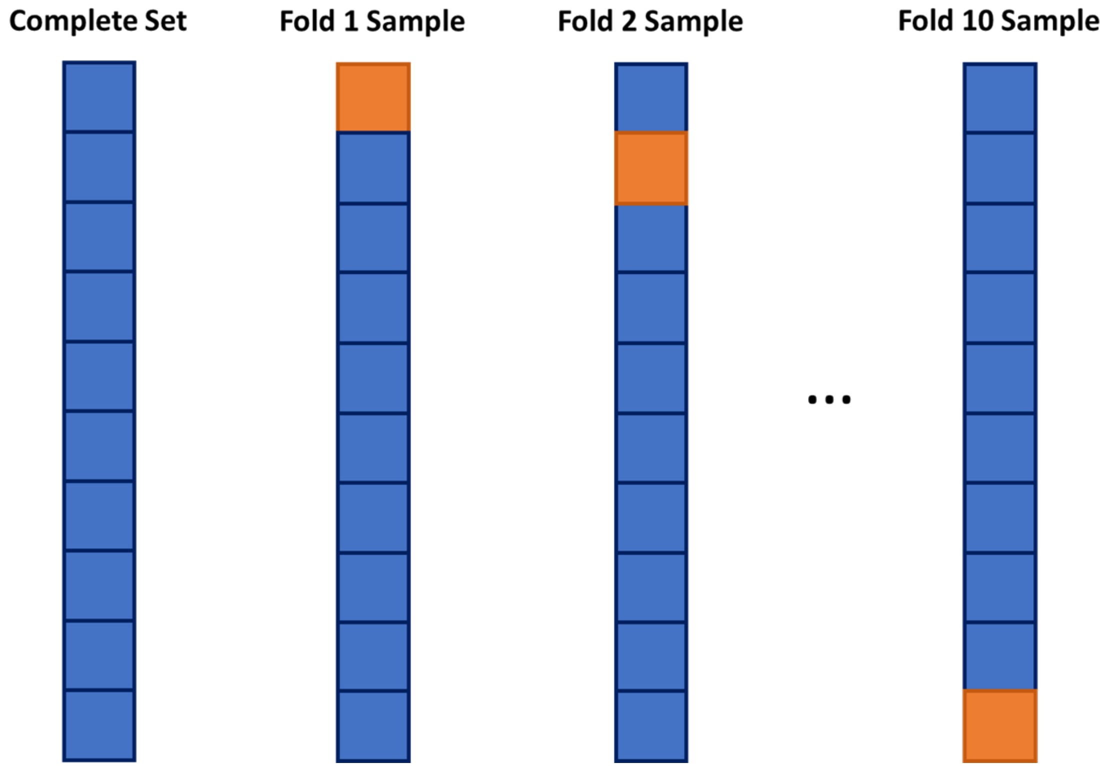
RESAMPLING REVISITED

Training, Validation, Testing



- Split your data into two or three sections of data
  - Training
  - Validation
  - Testing
- Common percentages:
  - 60-20-20
  - 70-20-10
  - 40-40-20
  - Etc.

Cross-Validation



- Divide your data into k-equally sized groups (folds, samples, etc.)
- Model evaluation
  - Average goodness-of-fit across all folds.
- Parameter/Model tuning

Ames Real Estate Data

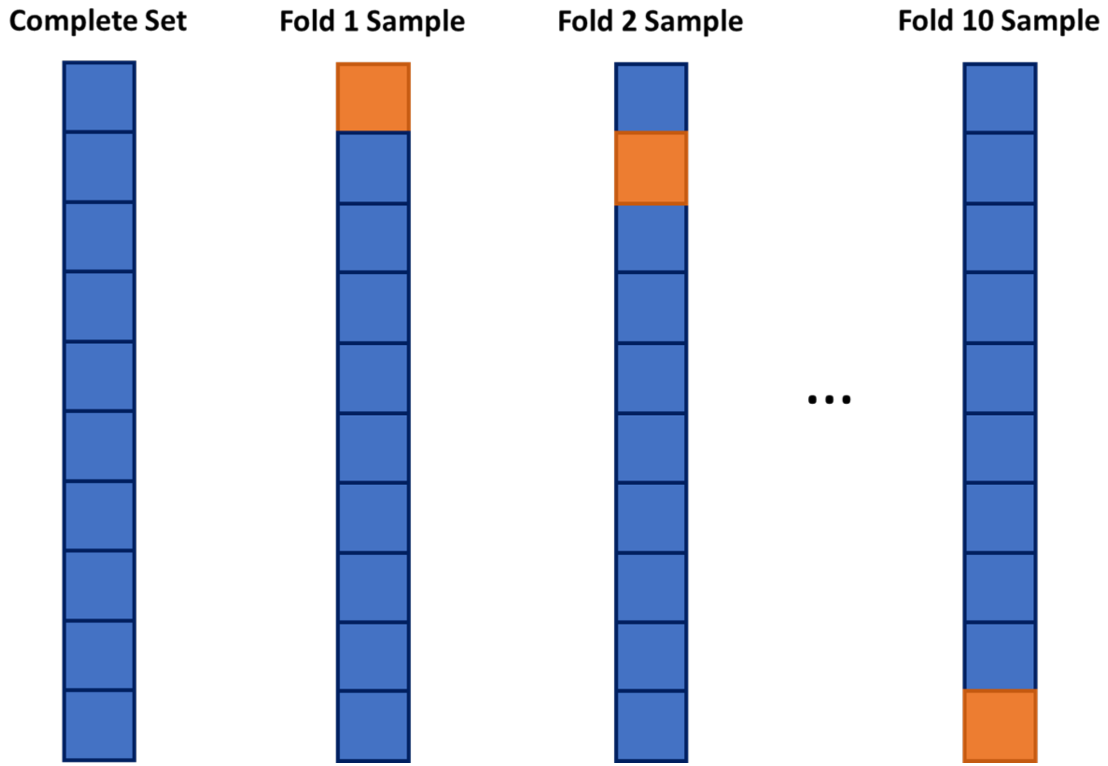
- 2930 homes in Ames, Iowa in the early 2000's.
- Physical attributes of homes along with sales price of home.

Training and Testing Split (No Validation Yet…)

```
ames <- make_ordinal_ames()

ames <- ames %>% mutate(id = row_number())

set.seed(4321)

training <- ames %>% sample_frac(0.7)
testing <- anti_join(ames, training, by = 'id')

training <- training %>%
  select(Sale_Price, Bedroom_AbvGr, Year_Built, Mo_Sold, Lot_Area,
         Street, Central_Air, First_Flr_SF, Second_Flr_SF, Full_Bath,
         Half_Bath, Fireplaces, Garage_Area, Gr_Liv_Area, TotRms_AbvGrd)
```

MODEL SELECTION

Cross-Validation



- Divide your data into k-equally sized groups (folds, samples, etc.)
- Model evaluation
  - Average goodness-of-fit across all folds.
- Parameter/Model tuning

Variable Selection in Linear Models

- Linear models contains many different models (linear, logistic, etc.).
- **ALWAYS** start by narrowing a list of reasonable predictor variables through exploratory analysis.

- Explanation/Inference:
  - Forward, Backward, Stepwise

- Prediction:
  - LASSO, Ridge, Elastic Net
  - Potentially provides better predictive models, but at the cost of lack of interpretability

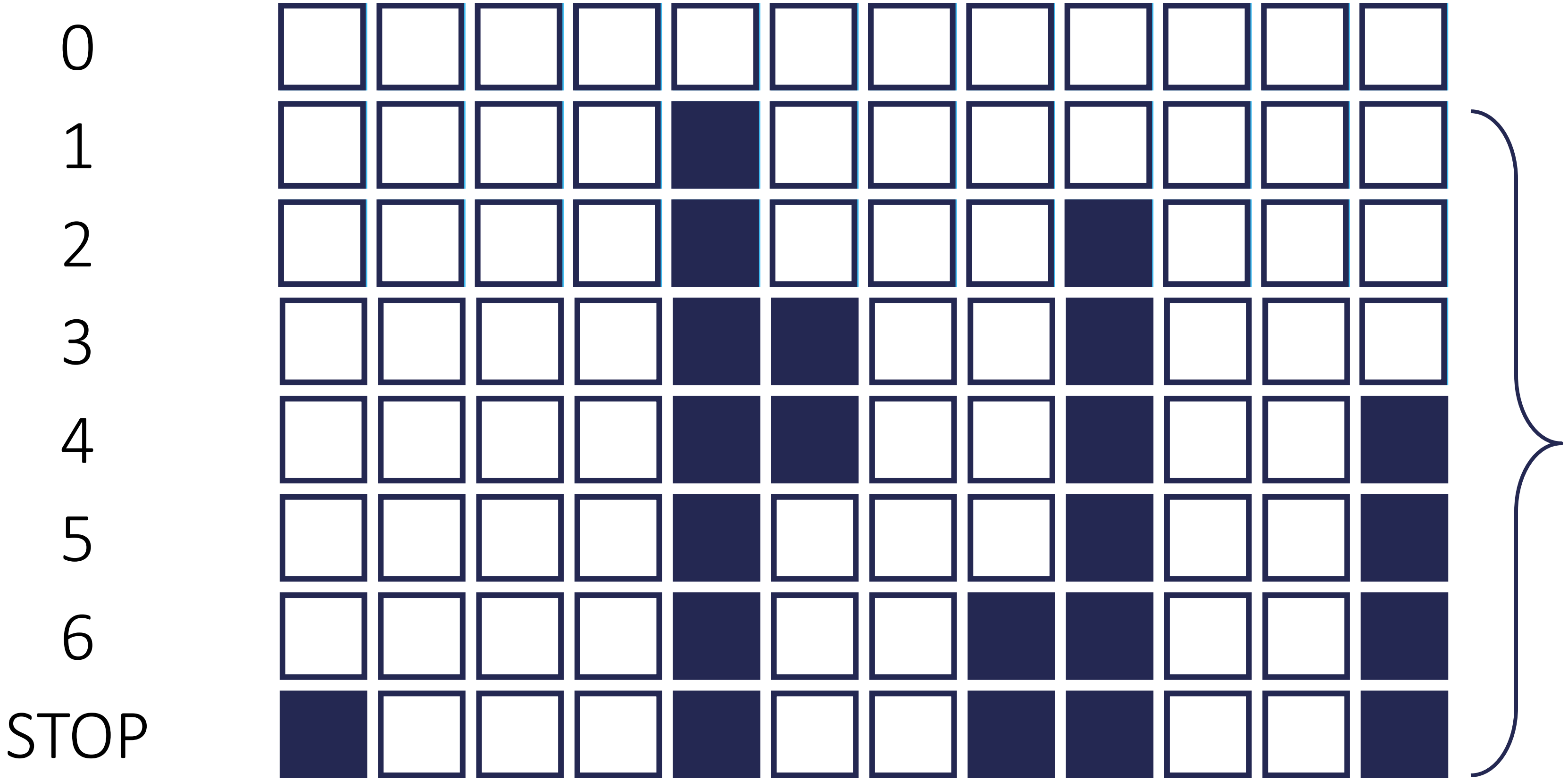
Variable Selection in Linear Models

- Linear models contains many different models (linear, logistic, etc.).
- **ALWAYS** start by narrowing a list of reasonable predictor variables through exploratory analysis.

- Explanation/Inference:
  - Forward, Backward, Stepwise

- Prediction:
  - LASSO, Ridge, Elastic Net
  - Potentially provides better predictive models, but at the cost of lack of interpretability

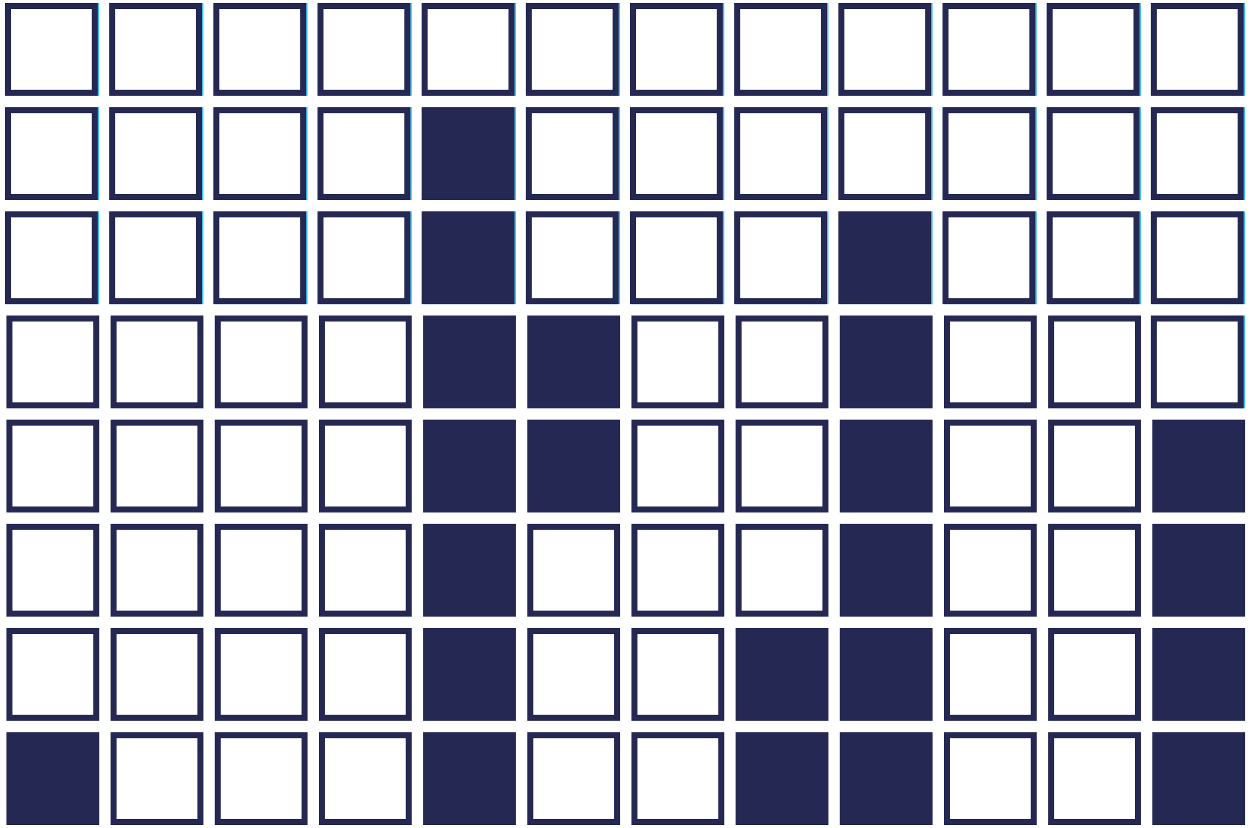
Stepwise Selection through Validation Set



Look at **validation** instead of training for each step. Which is better in **validation set**?

Stepwise Selection through Cross-Validation

0
1
2
3
4
5
6
STOP



Look at **validation** instead of training for each step. Which is better in avg. MSE in **cross-validation**?

Stepwise Selection through Cross-Validation

```
set.seed(9876)

step.model <- train(Sale_Price ~ ., data = training,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:14),
                    trControl = trainControl(method = 'cv', number = 10))
```

From caret package (similar to scikit learn in Python)

Stepwise Selection through Cross-Validation

```
set.seed(9876)

step.model <- train(Sale_Price ~ ., data = training,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:14),
                    trControl = trainControl(method = 'cv', number = 10))
```

Cross validation – 10 fold

2 Views of Parameter Tuning

Classical View

- Use validation to evaluate which model is “best” at each step of the procedure.
- Final model contains variables remaining at end of procedure.
- Example: Age, Income, Credit Score

”Modern” View

- Use validation to evaluate which model is “best” at each step of the procedure.
- Final model contains **same number of variables** as model at end of procedure.
- Example: 3 variable model

2 Views of Parameter Tuning

Classical View

- Combine training and validation.
- Update parameter estimates on the chosen variables (ex: Age, Income, Credit Score).

”Modern” View

- Combine training and validation.
- Do not restrict yourself to any variable, just the number of variables (ex: find best 3 variable model).

Stepwise Selection through Cross-Validation

```
step.model$results
```

```
##    nvmax     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      1 61611.97 0.3942384 45180.98 5472.666 0.05620948 2526.312
## 2      2 49940.14 0.6074530 34262.71 7925.632 0.09929920 3450.872
## 3      3 42271.12 0.7153532 28108.76 7112.714 0.07760796 2655.431
## 4      4 41519.38 0.7274272 27291.26 7807.909 0.08250703 2626.796
## 5      5 39709.65 0.7505967 26396.47 7761.820 0.08026905 2621.732
## 6      6 39293.66 0.7556266 26266.02 7486.423 0.07609118 2547.016
## 7      7 39403.58 0.7542579 26256.86 7471.045 0.07604703 2553.544
## 8      8 39436.99 0.7538030 26265.14 7447.858 0.07528998 2562.365
## 9      9 39547.27 0.7525961 26324.07 7624.466 0.07703777 2651.493
## 10    10 39466.09 0.7536853 26281.15 7644.350 0.07719334 2660.940
## 11    11 39395.64 0.7546030 26253.80 7674.807 0.07729593 2706.320
## 12    12 39344.06 0.7552931 26195.33 7685.764 0.07737416 2753.871
## 13    13 39340.86 0.7553046 26182.74 7675.699 0.07725826 2737.195
## 14    14 39347.25 0.7553214 26190.40 7671.560 0.07724606 2724.610
```

```
step.model$bestTune
```

```
##   nvmax
## 6     6
```

Stepwise Selection through Cross-Validation

```
summary(step.model$finalModel)
```

```
## 1 subsets of each size up to 6
## Selection Algorithm: backward
##          Bedroom_AbvGr Year_Built Mo_Sold Lot_Area StreetPave Central_AirY
## 1  ( 1 ) " "           " "        " "     " "      " "        " "
## 2  ( 1 ) " "           " "        " "     " "      " "        " "
## 3  ( 1 ) " "           "*"        " "     " "      " "        " "
## 4  ( 1 ) " "           "*"        " "     " "      " "        " "
## 5  ( 1 ) "*"           "*"        " "     " "      " "        " "
## 6  ( 1 ) "*"           "*"        " "     " "      " "        " "
##          First_Flr_SF Second_Flr_SF Full_Bath Half_Bath Fireplaces Garage_Area
## 1  ( 1 ) "*"          " "           " "       " "       " "        " "
## 2  ( 1 ) "*"          "*"           " "       " "       " "        " "
## 3  ( 1 ) "*"          "*"           " "       " "       " "        " "
## 4  ( 1 ) "*"          "*"           " "       " "       " "        "*"
## 5  ( 1 ) "*"          "*"           " "       " "       " "        "*"
## 6  ( 1 ) "*"          "*"           " "       " "       "*"        "*"
##          Gr_Liv_Area TotRms_AbvGrd
## 1  ( 1 ) " "         " "
## 2  ( 1 ) " "         " "
## 3  ( 1 ) " "         " "
## 4  ( 1 ) " "         " "
## 5  ( 1 ) " "         " "
## 6  ( 1 ) " "         " "
```

"Classical" View of Parameter Tuning

```
final.model1 <- glm(Sale_Price ~ First_Flr_SF + Second_Flr_SF + Year_Built + Garage_Area +
                                  Bedroom_AbvGr + Fireplaces,
                    data = training)

summary(final.model1)


## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.407e+06  6.439e+04 -21.852  < 2e-16 ***
## First_Flr_SF   1.128e+02  3.236e+00  34.871  < 2e-16 ***
## Second_Flr_SF  8.252e+01  2.812e+00  29.342  < 2e-16 ***
## Year_Built     7.256e+02  3.306e+01  21.945  < 2e-16 ***
## Garage_Area    6.012e+01  5.366e+00  11.203  < 2e-16 ***
## Bedroom_AbvGr -1.265e+04  1.317e+03  -9.607  < 2e-16 ***
## Fireplaces     1.113e+04  1.555e+03   7.157 1.14e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## AIC: 49246
```

“Modern” View of Parameter Tuning

```
empty.model <- glm(Sale_Price ~ 1, data = training)
full.model <- glm(Sale_Price ~ ., data = training)

final.model2 <- step(empty.model, scope = list(lower = formula(empty.model),
                                               upper = formula(full.model)),
                     direction = "both", steps = 6)


summary(final.model2)
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.441e+06  6.451e+04 -22.343  < 2e-16 ***
## Gr_Liv_Area     8.116e+01  2.790e+00  29.086  < 2e-16 ***
## Year_Built      7.433e+02  3.313e+01  22.438  < 2e-16 ***
## First_Flr_SF    3.053e+01  2.944e+00  10.370  < 2e-16 ***
## Garage_Area     6.110e+01  5.373e+00  11.372  < 2e-16 ***
## Bedroom_AbvGr  -1.258e+04  1.322e+03  -9.518  < 2e-16 ***
## Fireplaces      1.138e+04  1.558e+03   7.305 3.95e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ‘ 1
##
## AIC: 49257
```

"Modern" View of Parameter Tuning

```
empty.model <- glm(Sale_Price ~ 1, data = training)
full.model <- glm(Sale_Price ~ ., data = training)

final.model2 <- step(empty.model, scope = list(lower = formula(empty.model),
                                               upper = formula(full.model)),
                     direction = "both", steps = 6)


summary(final.model2)
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.441e+06  6.451e+04 -22.343  < 2e-16 ***
## Gr_Liv_Area     8.116e+01  2.790e+00  29.086  < 2e-16 ***
## Year_Built      7.433e+02  3.313e+01  22.438  < 2e-16 ***
## First_Flr_SF    3.053e+01  2.944e+00  10.370  < 2e-16 ***
## Garage_Area     6.110e+01  5.373e+00  11.372  < 2e-16 ***
## Bedroom_AbvGr -1.258e+04  1.322e+03  -9.518  < 2e-16 ***
## Fireplaces      1.138e+04  1.558e+03   7.305 3.95e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## AIC: 49257
```

Different 6 variables!

?

REGULARIZATION

Variable Selection in Linear Models

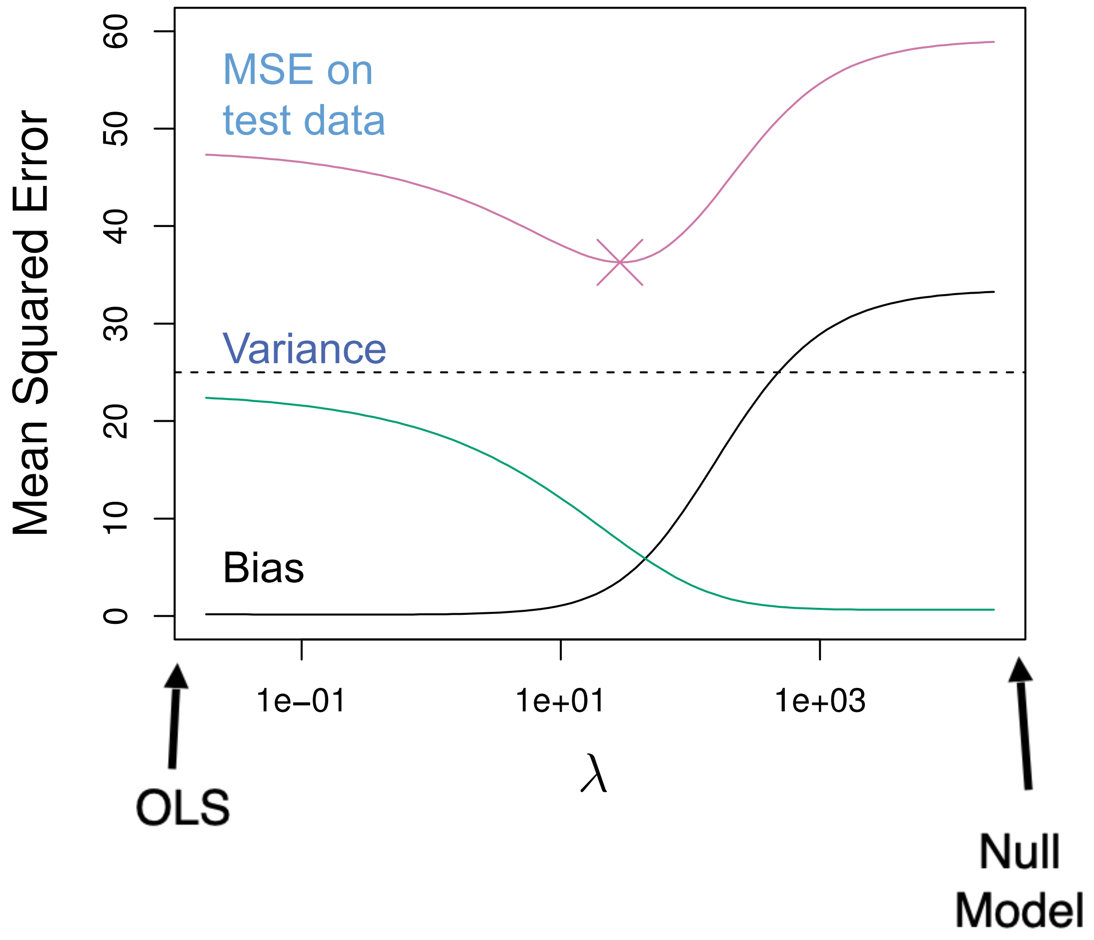
- Linear models contains many different models (linear, logistic, etc.).
- **ALWAYS** start by narrowing a list of reasonable predictor variables through exploratory analysis.

- Explanation/Inference:
  - Forward, Backward, Stepwise

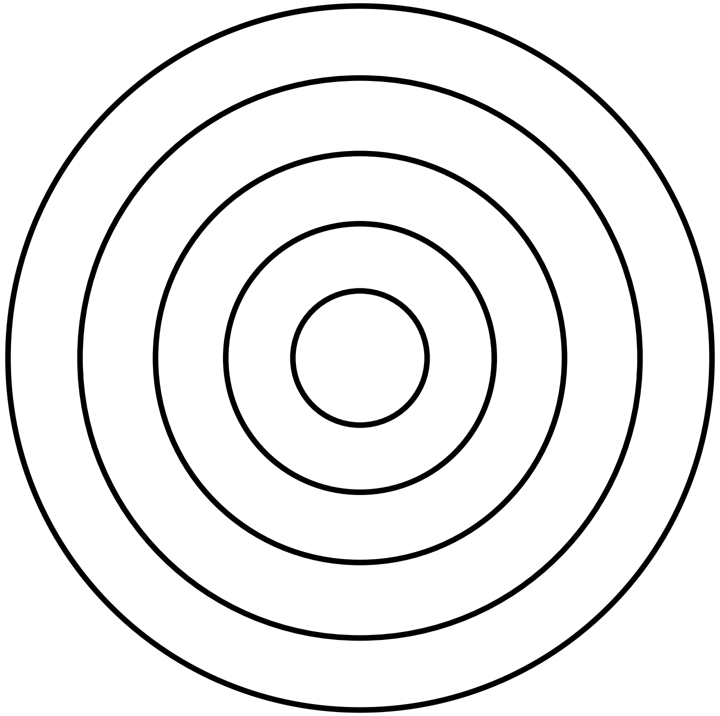
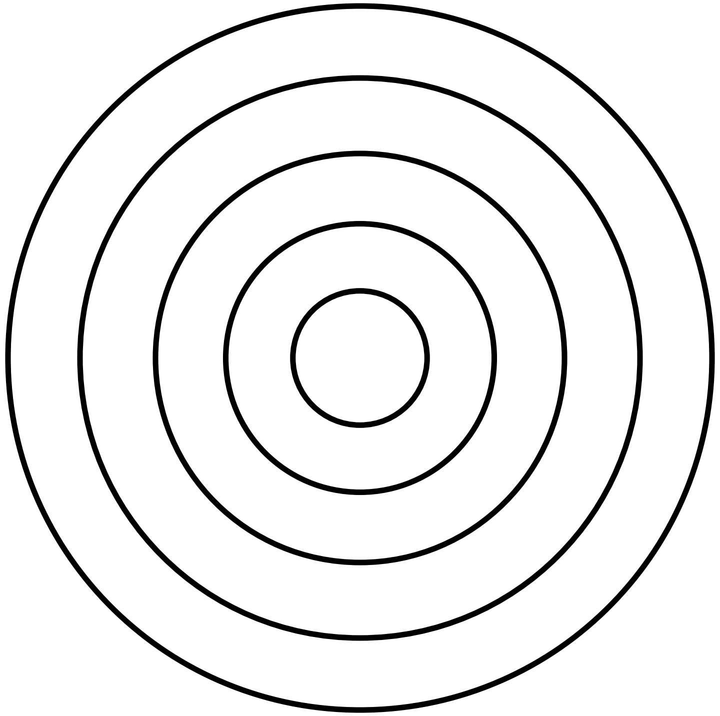
- Prediction:
  - LASSO, Ridge, Elastic Net
  - Potentially provides better predictive models, but at the cost of lack of interpretability

Regularization

- **Regularization** (or penalization / shrinkage) is a common tool to control the complexity/flexibility of a model.
- Adds penalty term to penalize model complexity.
- Model becomes biased, but potentially improve variance of the model.

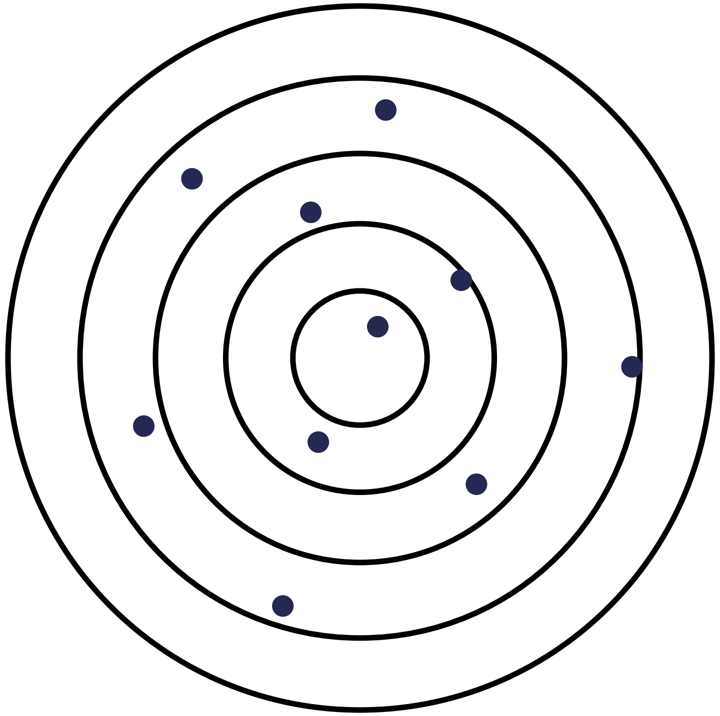


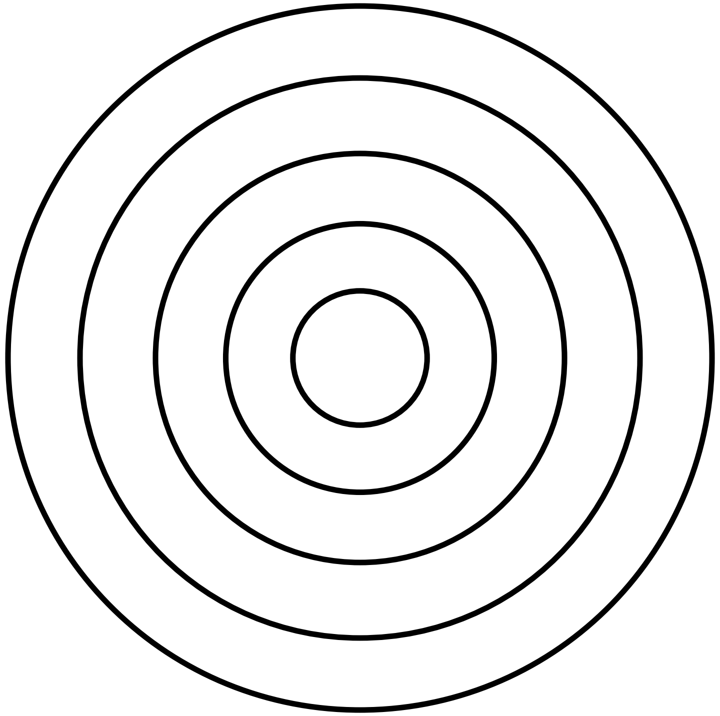
Biased Regression Techniques

Biased Regression Techniques

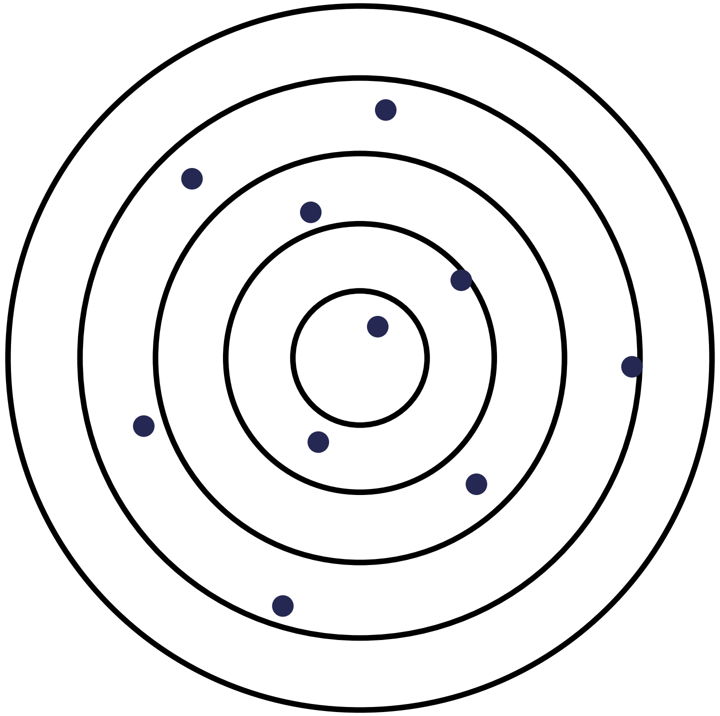
Unbiased but not precise



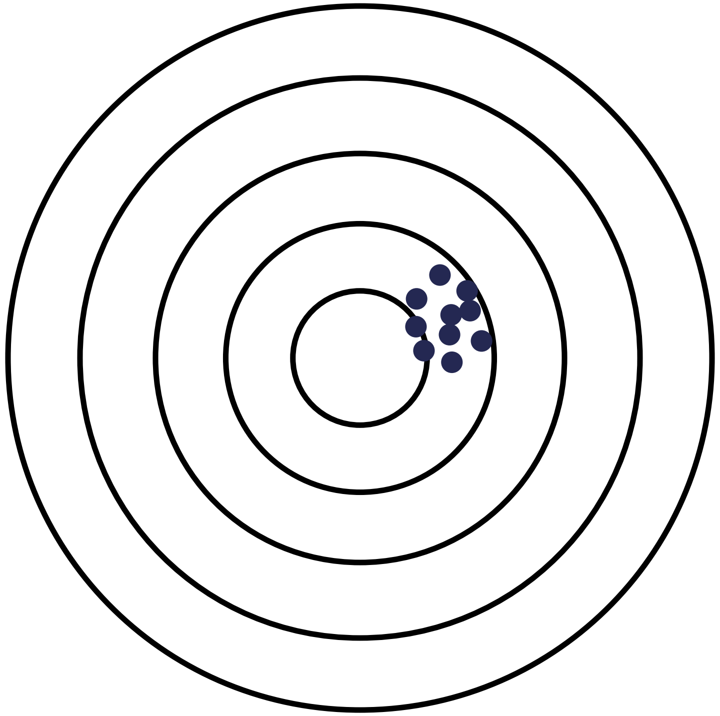


Biased Regression Techniques

Unbiased but not precise

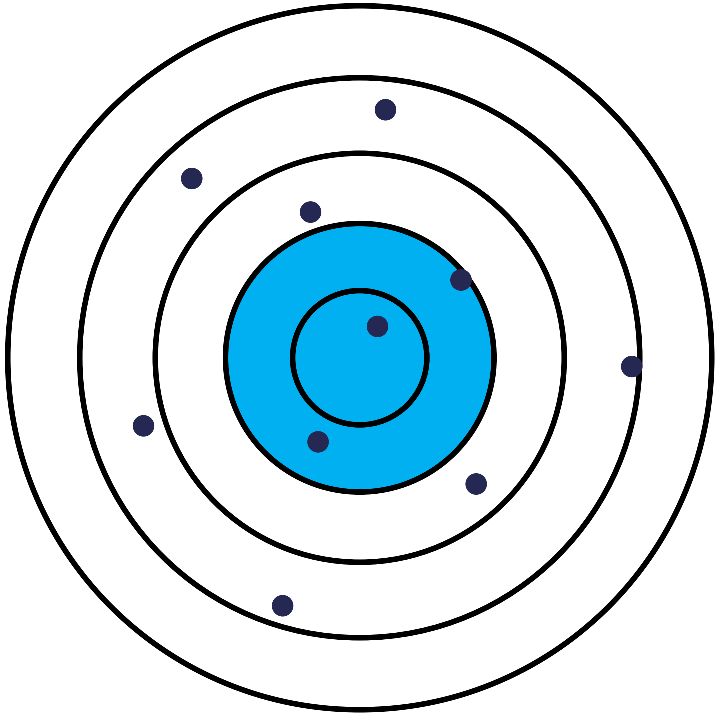


Biased but precise

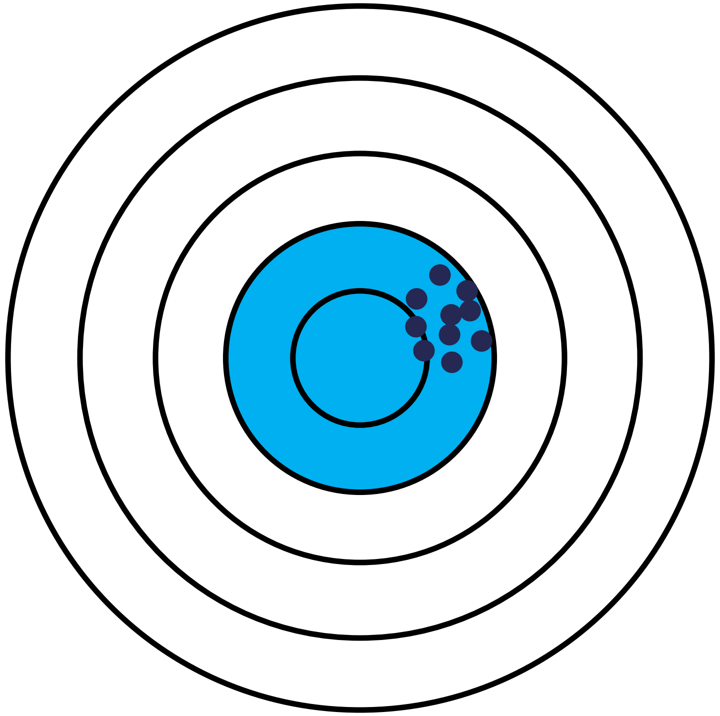


Biased Regression Techniques

Unbiased but not precise



Biased but precise



Regularized Regression

- **Regularized regression** (or penalized / shrinkage regression) puts constraints on the estimated coefficients in our model and *shrink* these estimates to 0.

- Coefficients become biased, but potentially improve variance of the model.

- 3 Common Approaches – Ridge, LASSO, Elastic Net

Penalties in Models

- OLS regression minimizes the sum of squared errors:

$$\min\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right) = \min(SSE)$$

- Regularized regression introduces a penalty term to the minimization:

$$\min\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + Penalty\right) = \min(SSE + Penalty)$$

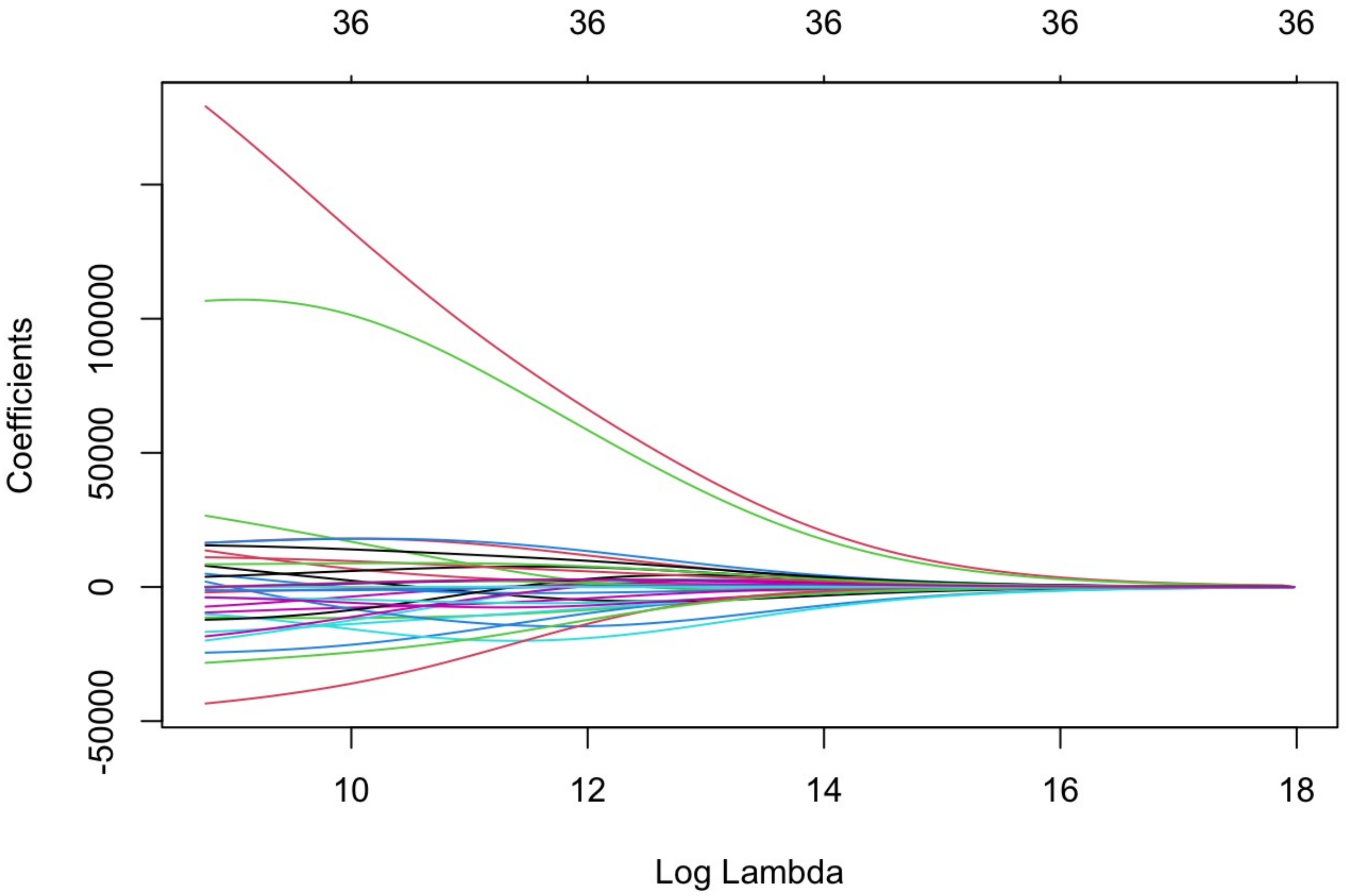
Ridge Regression

- Ridge regression introduces an "L_2" penalty term to the minimization:

$$\min\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\hat{\beta}_j^2\right) = \min\left(SSE + \lambda\sum_{j=1}^{p}\hat{\beta}_j^2\right)$$

- Penalty is controlled by **tuning parameter**, λ.
  - If $\lambda = 0$, then OLS.
  - As $\lambda \to \infty$, coefficients shrink to 0.

Ridge Regression



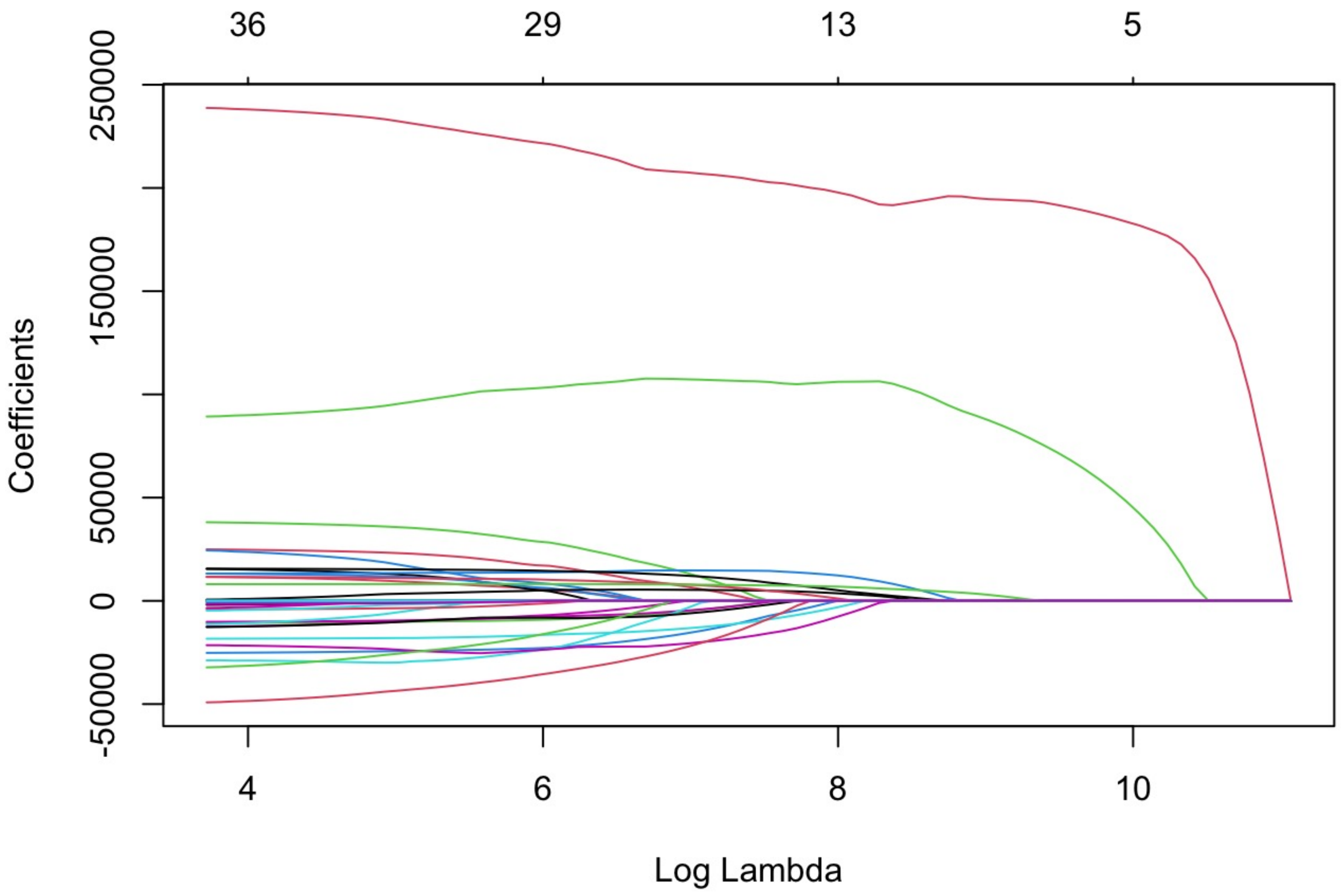
LASSO Regression

- Least absolute shrinkage and selection operator (LASSO) regression introduces an "L_1" penalty term to the minimization:

$$\min\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}|\hat{\beta}_j|\right) = \min\left(SSE + \lambda\sum_{j=1}^{p}|\hat{\beta}_j|\right)$$

- Penalty is controlled by **tuning parameter**, λ.
  - If $\lambda = 0$, then OLS.
  - As $\lambda \to \infty$, coefficients shrink to 0.

LASSO Regression



Differences in Effects

- Penalty is controlled by **tuning parameter**, λ.
  - If $\lambda = 0$, then OLS.
  - As $\lambda \to \infty$, coefficients $\boxed{\text{shrink to 0.}}$

Differences in effects are due to differences in penalty.

When solving the system of equations for the different penalties we get the following:

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y \qquad \hat{\beta}_R = (X^T X + \lambda I)^{-1} X^T Y \qquad \hat{\beta}_L = (X^T X)^{-1} (X^T Y - \lambda I)$$

Differences in Effects

- Penalty is controlled by **tuning parameter**, λ.
  - If $\lambda = 0$, then OLS.
  - As $\lambda \to \infty$, coefficients shrink to 0.

Differences in effects are due to differences in penalty.

When solving the system of equations for the different penalties we get the following:

$$\hat{\beta}_{OLS} = (X^TX)^{-1}X^TY \qquad \hat{\beta}_R = (X^TX + \lambda I)^{-1}X^TY \qquad \hat{\beta}_L = (X^TX)^{-1}(X^TY - \lambda I)$$

As $\lambda \to \infty$, $\hat{\beta}_R$ gets infinitely close to 0

Differences in Effects

- Penalty is controlled by **tuning parameter**, λ.
  - If $\lambda = 0$, then OLS.
  - As $\lambda \to \infty$, coefficients shrink to 0.

Differences in effects are due to differences in penalty.

When solving the system of equations for the different penalties we get the following:

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y \qquad \hat{\beta}_R = (X^T X + \lambda I)^{-1} X^T Y \qquad \hat{\beta}_L = (X^T X)^{-1}(X^T Y - \lambda I)$$

If $\lambda = X^T Y$, $\hat{\beta}_L$ can actually equal 0

Elastic Net

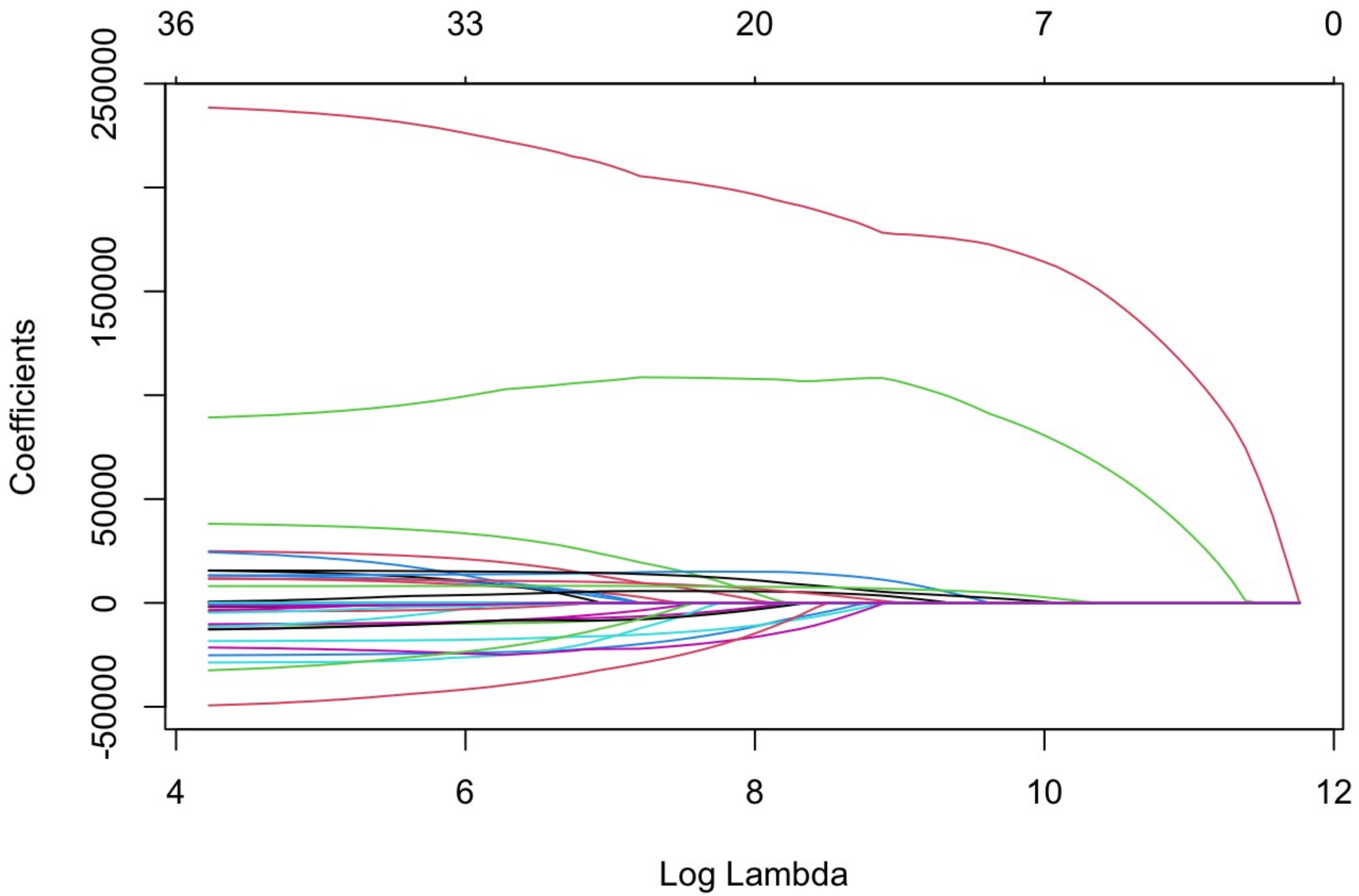
- The `glmnet` function in R takes slightly different approach:

$$\min\left(\sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\lambda\left[\alpha\sum_{j=1}^{p}|\hat{\beta}_j|+(1-\alpha)\sum_{j=1}^{p}\hat{\beta}_j^2\right]\right)$$

Why R has the "`alpha = `" option.

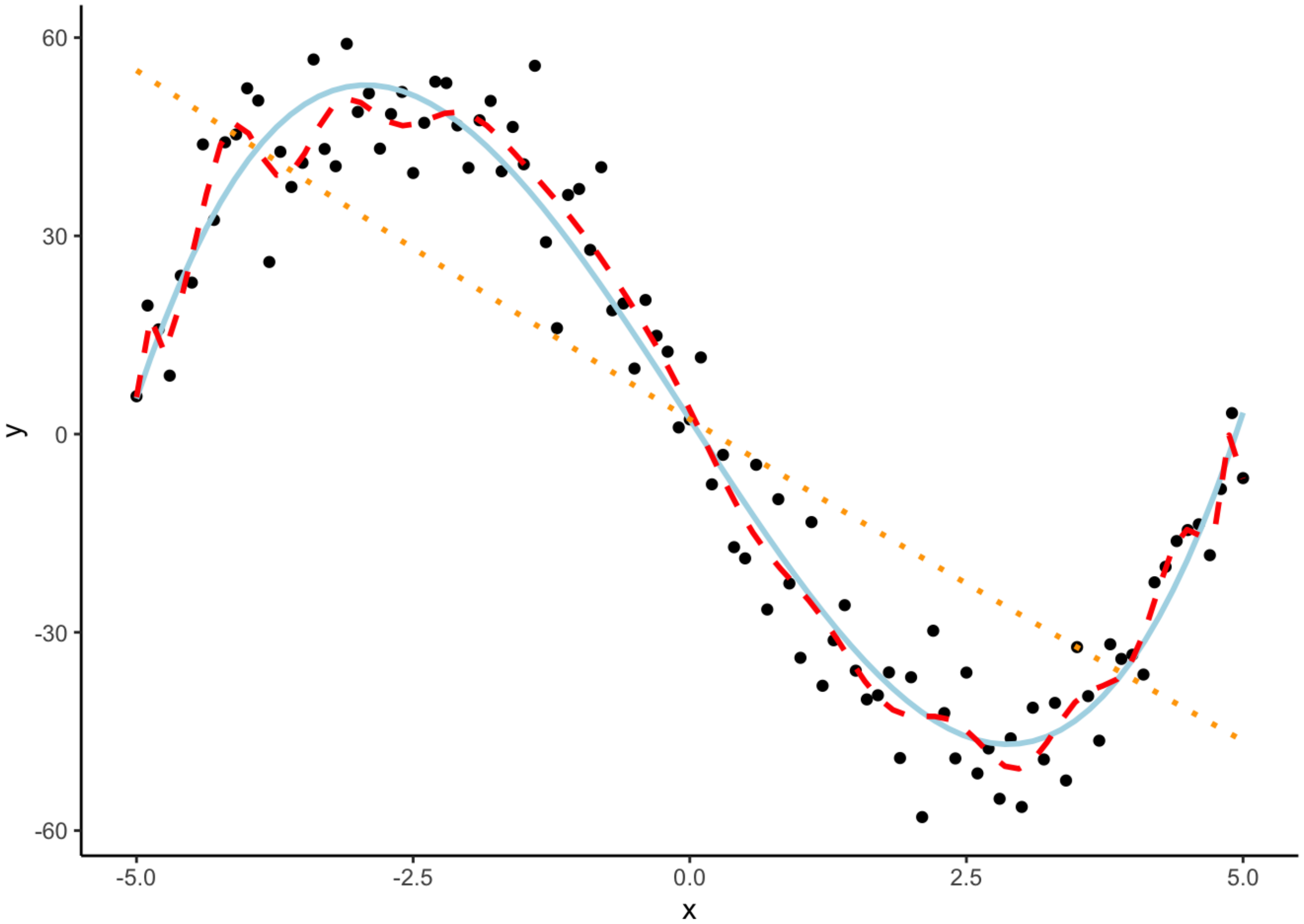
- Any value of `alpha` between 0 and 1 gives a combination of both penalties (elastic net).

Elastic Net Regression



Fear of Overfitting

- Need to select λ for any of the regularized regression approaches.
- Don't want to minimize variance to the point of overfitting our model to the training data.

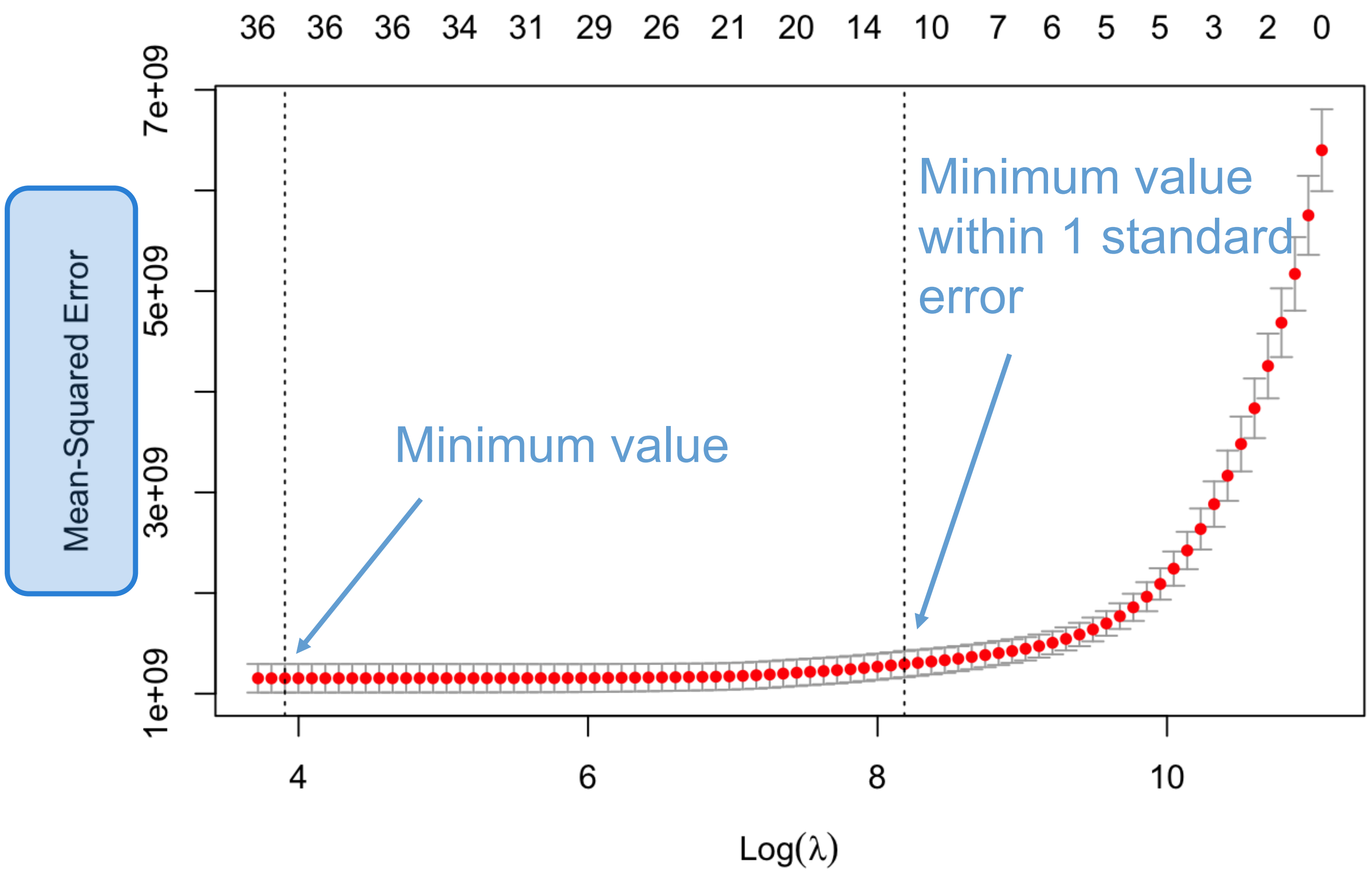


Cross-Validation

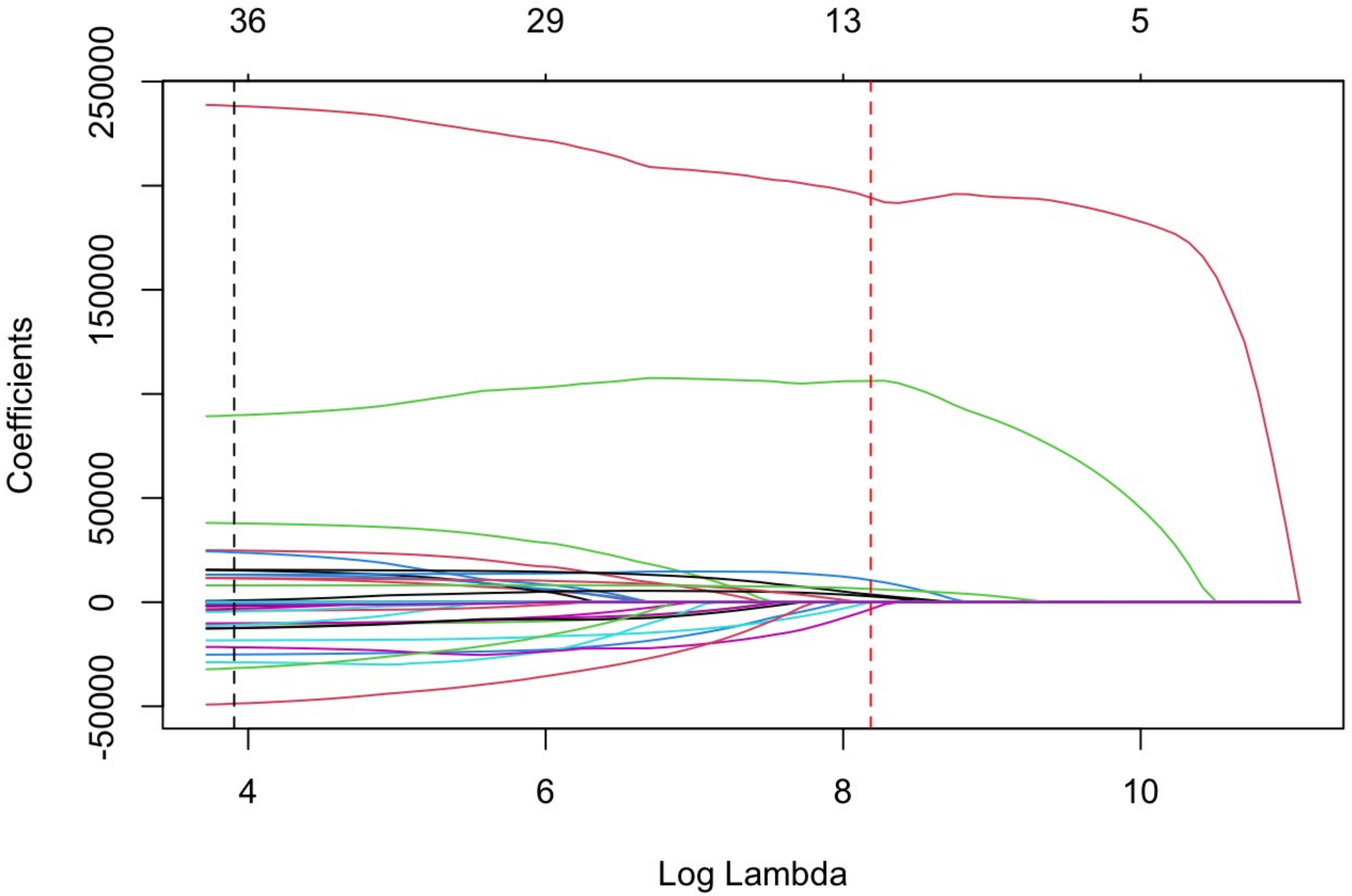
- **Cross-validation** (CV) is common approach to prevent overfitting when tuning a parameter.
- Concept:
  - Split training data into multiple pieces
  - Build model on majority of pieces
  - Evaluate on remaining piece
  - Repeat process with switching out pieces for building and evaluation

LASSO Regression

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



LASSO Regression



Elastic Net Optimization

```
set.seed(5)

en.model <- train(Sale_Price ~ ., data = training,
                  method = "glmnet",
                  tuneGrid = expand.grid(.alpha = seq(0,1, by = 0.05),
                                         .lambda = seq(100,60000, by = 1000)),
                  trControl = trainControl(method = 'cv', number = 10))
```

Defining the parameters we want to tune

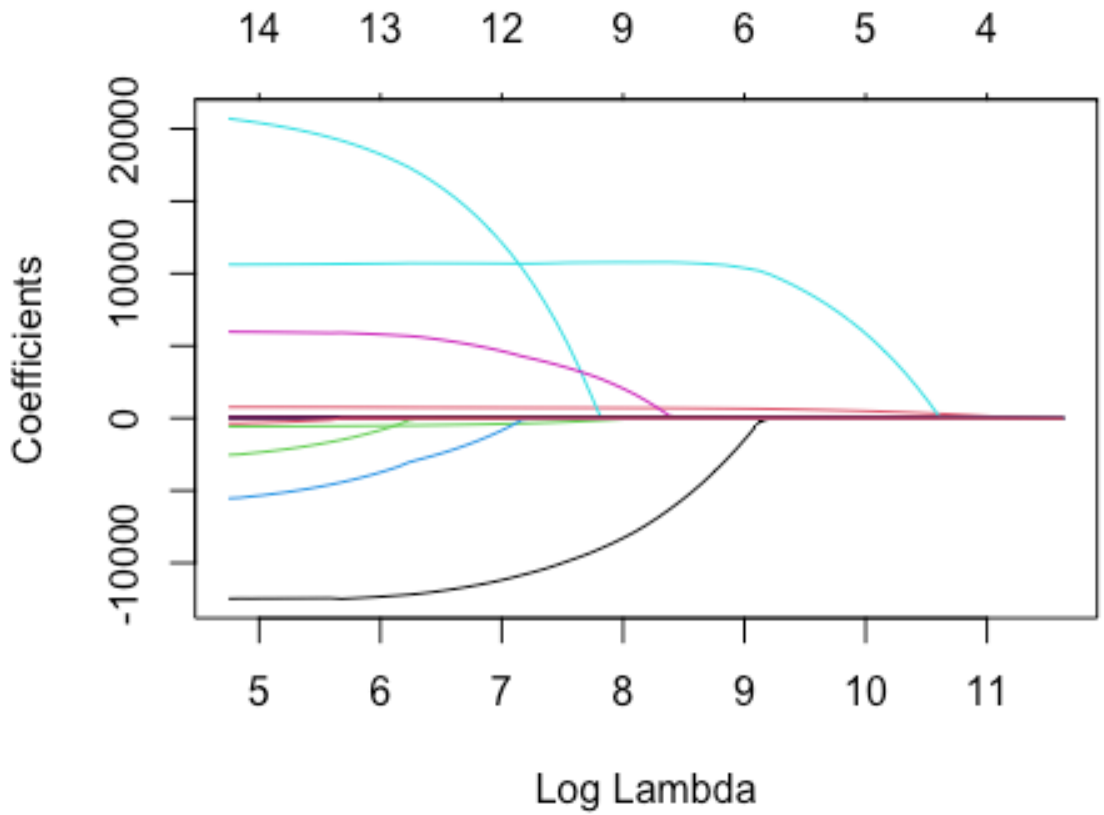
Elastic Net Optimization

```
en.model
```

```
## glmnet
##
## 2051 samples
##   14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1847, 1846, 1846, 1846, 1846, 1845, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  RMSE      Rsquared   MAE
##   0.00     100   39425.23  0.7549646  26190.91
##   0.00    1100   39425.23  0.7549646  26190.91
##   0.00    2100   39425.23  0.7549646  26190.91
…
##   1.00   56100   78334.99  0.5086181  57328.76
##   1.00   57100   78607.53  0.4385170  57550.28
##   1.00   58100   78616.60        NaN  57557.27
##   1.00   59100   78616.60        NaN  57557.27
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.5 and lambda = 100.
```

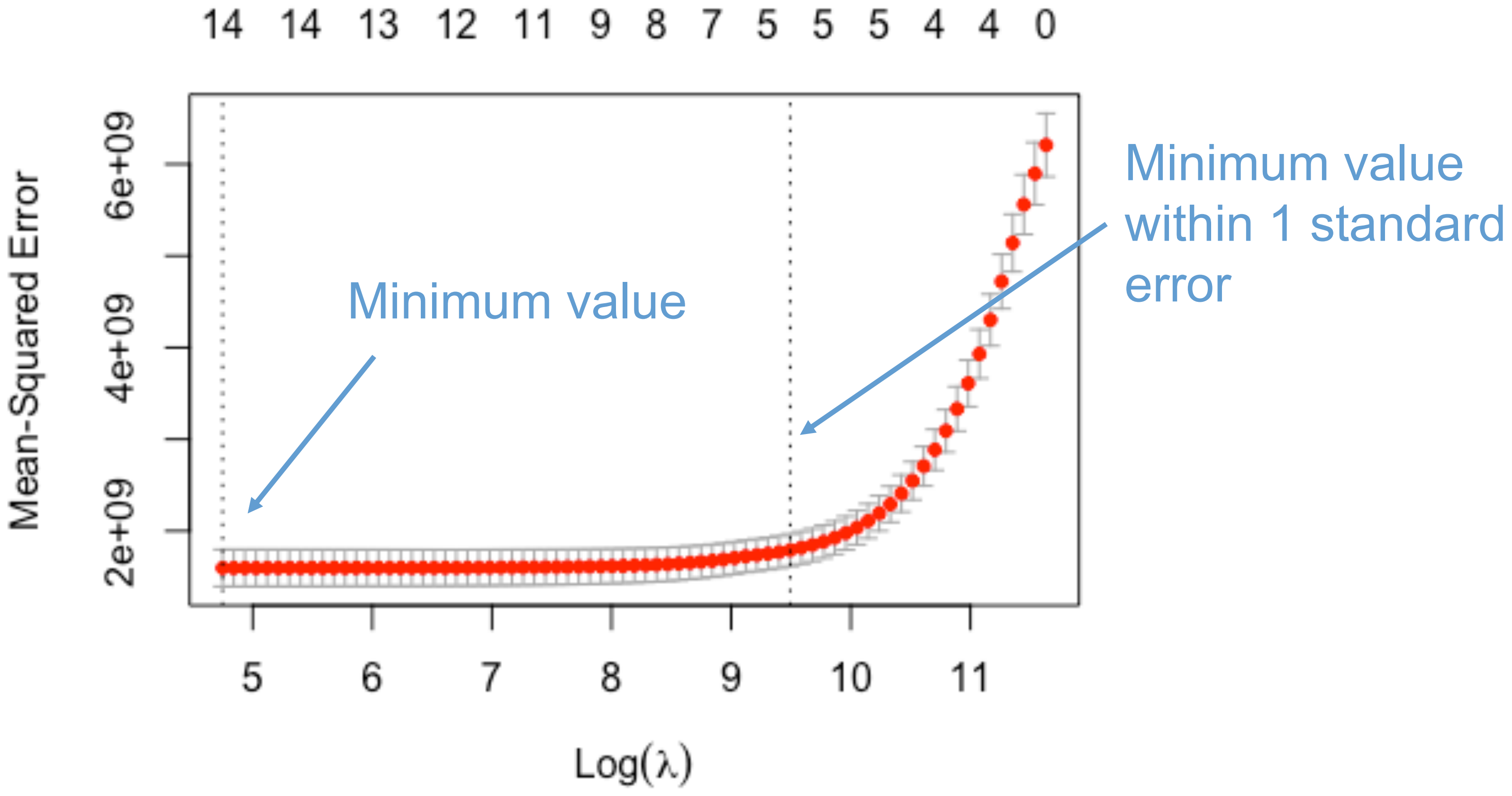
Elastic Net Optimization

```
train_x <- model.matrix(Sale_Price ~ ., data = training)[, -1]
train_y <- training$Sale_Price

ames_en <- glmnet(x = train_x,  y = train_y,
                  alpha = 0.5)

plot(ames_en, xvar = "lambda")
```



Elastic Net Optimization

```
set.seed(5)
ames_en_cv <- cv.glmnet(x = train_x,  y = train_y,  alpha = 0.5)

plot(ames_en_cv)
```



Elastic Net Optimization

```
ames_en_cv$lambda.min
## [1] 115.4119
```

Similar to our value of 100

```
ames_en_cv$lambda.1se
## [1] 13269.57
```