Model Building & Scoring for Prediction

Institute for Advanced Analytics
MSA Class of 2022

Model Building

- Linear regression is a great initial approach to model building, but it isn’t the only form of regression.
- Linear regression is the **best linear unbiased estimator (BLUE)**.

Best Linear Unbiased Estimator

- What does it mean to be **unbiased**?

$$\hat{\beta}_j \sim N\left(\beta_j, s_{\hat{\beta}_j}\right)$$

Best Linear Unbiased Estimator

- What does it mean to be **unbiased**?

$$\hat{\beta}_j \sim N\left(\beta_j, s_{\hat{\beta}_j}\right)$$

On average, coefficients from all samples are centered around the true coefficient.

Best Linear Unbiased Estimator

- What does it mean to be **unbiased**?

$$\hat{\beta}_j \sim N\left(\beta_j, s_{\hat{\beta}_j}\right)$$

- What does it mean to be **best**?
  - *IF* assumptions hold, $s_{\hat{\beta}_j}$ is the minimum variance of all the unbiased estimators.

Best Linear Unbiased Estimator

- What does it mean to be **unbiased**?

$$\hat{\beta}_j \sim N\left(\beta_j, s_{\widehat{\beta}_j}\right)$$

What if assumptions don’t hold?

- What does it mean to be **best**?
  - ***IF* assumptions hold**, $s_{\widehat{\beta}_j}$ is the minimum variance of all the unbiased estimators.

Best Linear Unbiased Estimator

- What does it mean to be **unbiased**?

$$\hat{\beta}_j \sim N\left(\beta_j, s_{\hat{\beta}_j}\right)$$

What if assumptions don't hold?

- What does it mean to be **best**?
  - ***IF* assumptions hold**, $s_{\hat{\beta}_j}$ is the **minimum variance of all the unbiased estimators**.

What if biased estimators had smaller variance?

Regularized Regression

Potential Problems

- As the number of variables increases, more problems tend to arise.
  - Assumptions start to fail.
  - Multicollinearity concerns.

- Multicollinearity problems → coefficients vary widely.
  - Variations lead to **overfitting** (only predicting the training data well, but not generalizing to the test dataset).
  - Higher variance than desired.

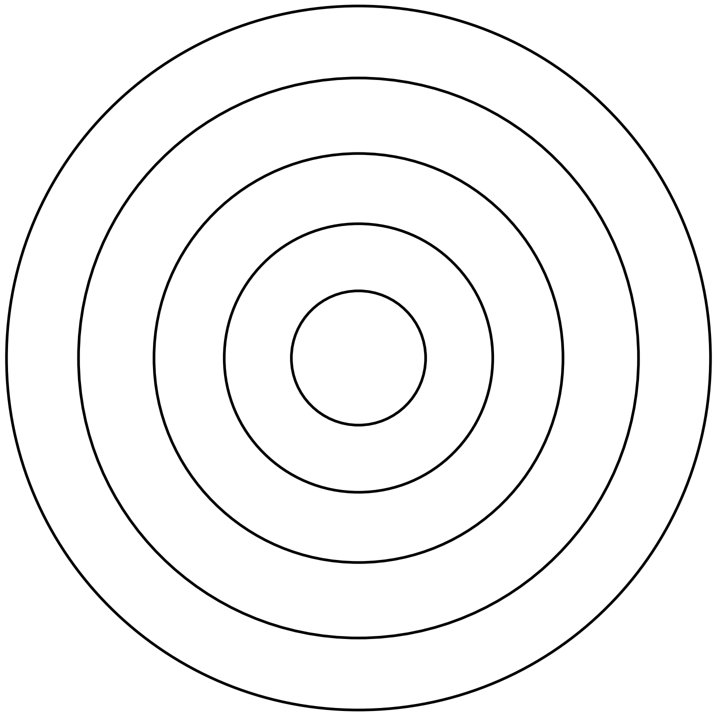
- More variables than observations (genetic modeling).

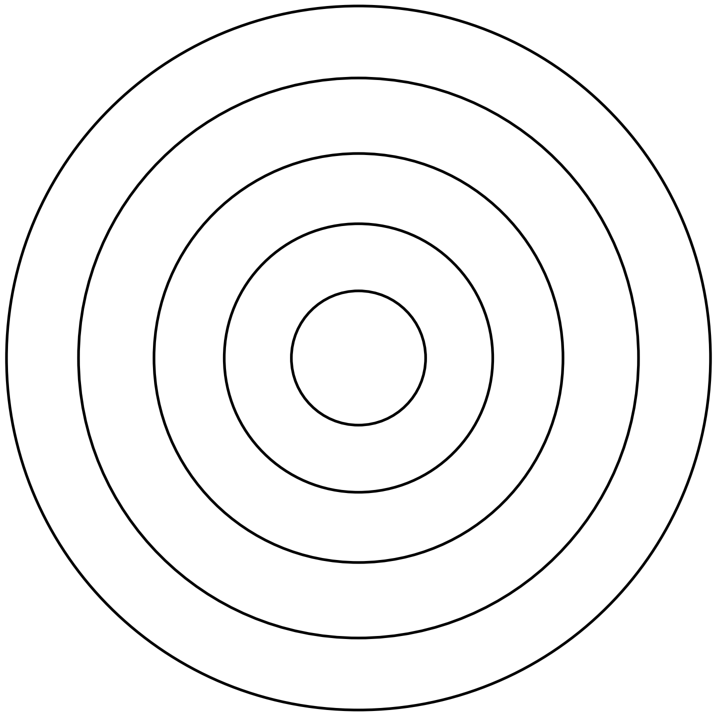
Regularized Regression

- **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.

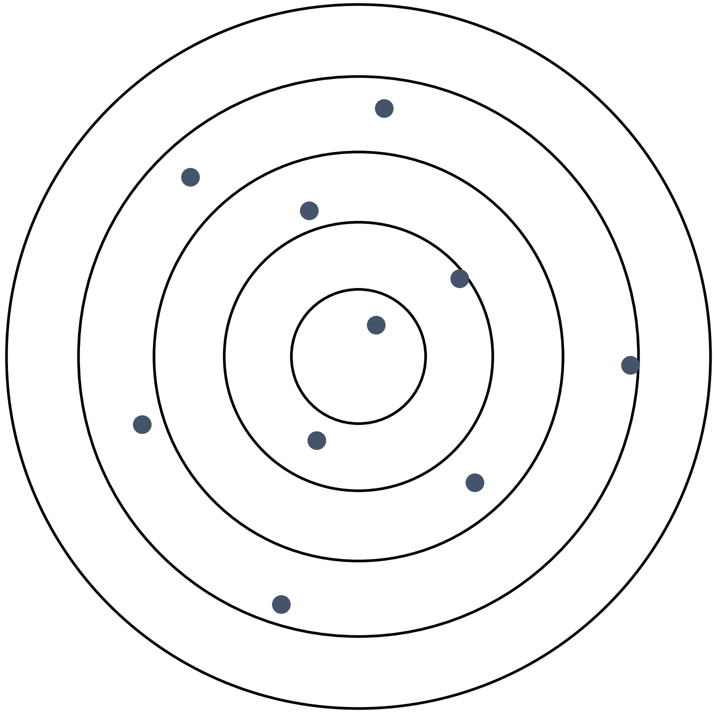
Biased Regression Techniques

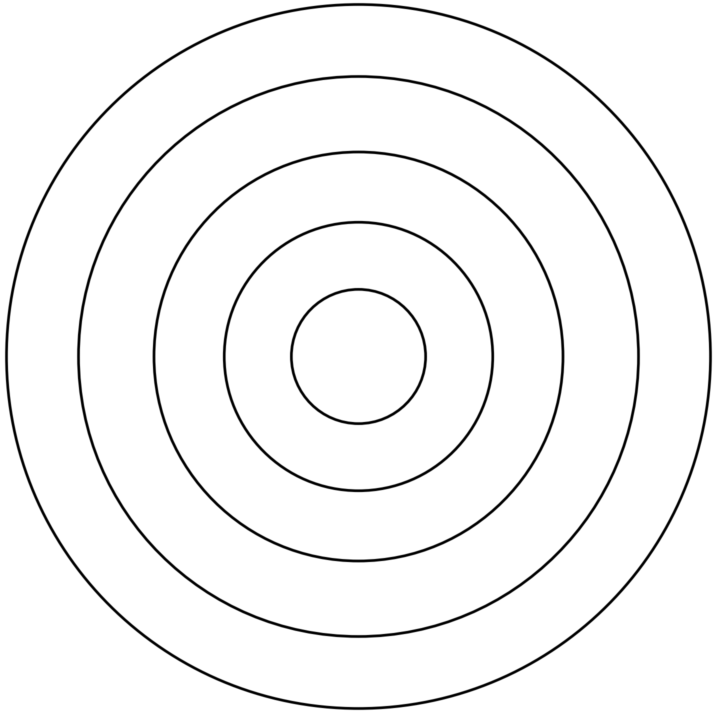




Biased Regression Techniques

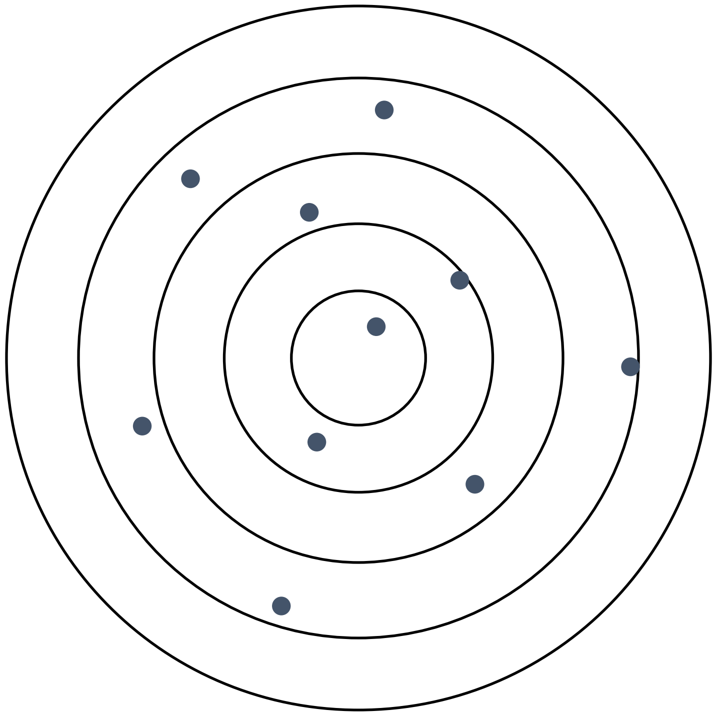
Unbiased but not precise



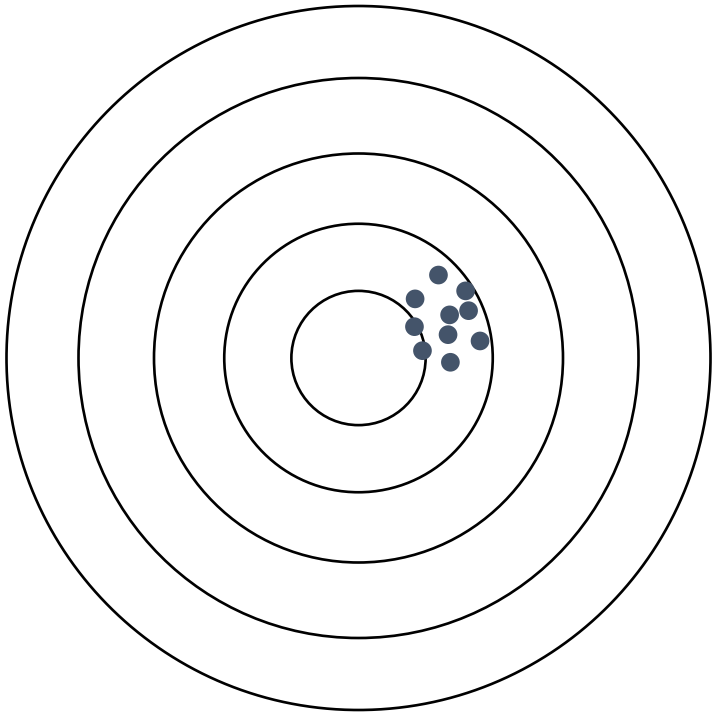


Biased Regression Techniques

Unbiased but not precise

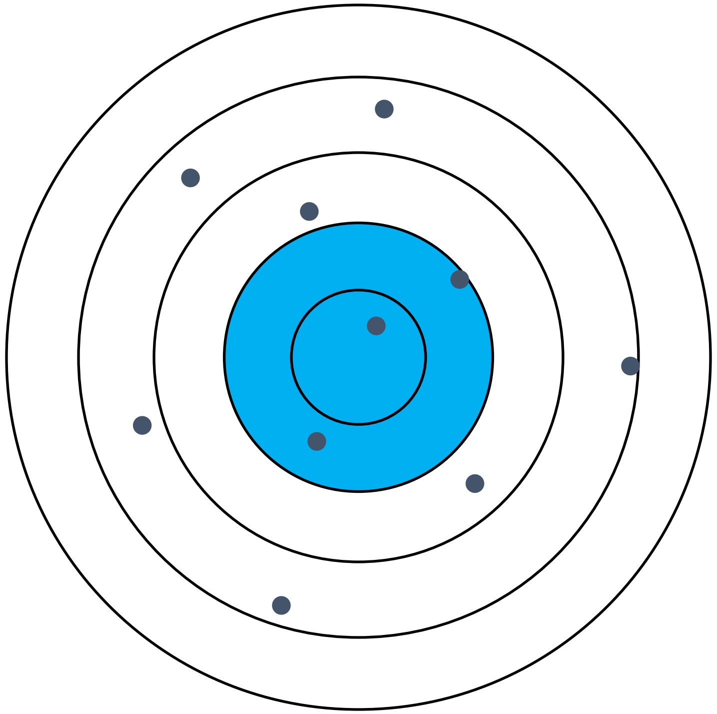


Biased but precise

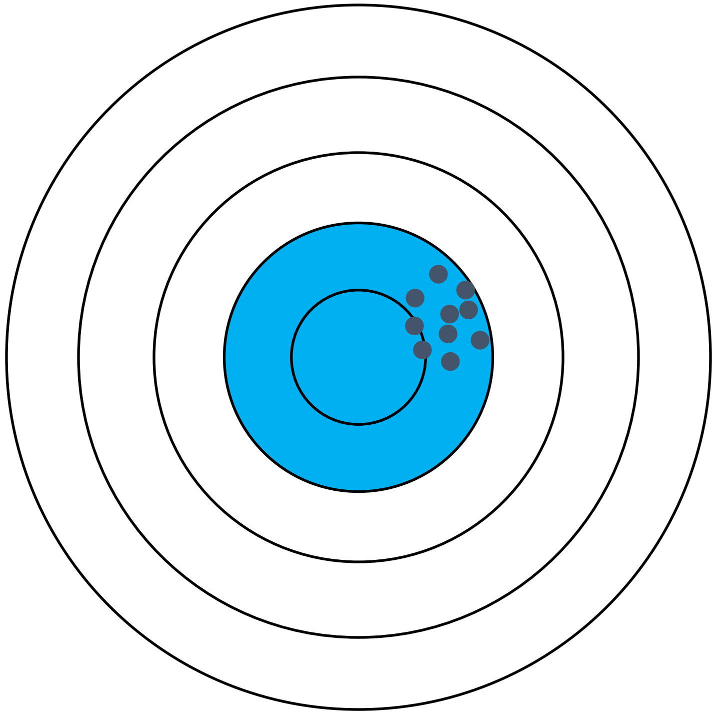


Biased Regression Techniques

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:

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

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:

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

?

Regularized Regression

RIDGE REGRESSION

Penalties in Models

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

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

Penalties in Models

- 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

```
train_reg <- train %>%
               dplyr::select(Sale_Price, Lot_Area, Street,
                             Bldg_Type, House_Style, Overall_Qual,
                             Roof_Style, Central_Air, First_Flr_SF,
                             Second_Flr_SF, Full_Bath, Half_Bath,
                             Fireplaces, Garage_Area, Gr_Liv_Area,
                             TotRms_AbvGrd) %>%
               replace(is.na(.), 0)
train_x <- model.matrix(Sale_Price ~ ., data = train_reg)[, -1]
train_y <- train_reg$Sale_Price
```

Ridge Regression

```
train_reg <- train %>%
             dplyr::select(Sale_Price, Lot_Area, Street,
                           Bldg_Type, House_Style, Overall_Qual,
                           Roof_Style, Central_Air, First_Flr_SF,
                           Second_Flr_SF, Full_Bath, Half_Bath,
                           Fireplaces, Garage_Area, Gr_Liv_Area,
                           TotRms_AbvGrd) %>%
             replace(is.na(.), 0)
train_x <- model.matrix(Sale_Price ~ ., data = train_reg)[, -1]
train_y <- train_reg$Sale_Price
```

Ridge Regression

```
test_reg <- test %>%
              dplyr::select(Sale_Price, Lot_Area, Street,
                            Bldg_Type, House_Style, Overall_Qual,
                            Roof_Style, Central_Air, First_Flr_SF,
                            Second_Flr_SF, Full_Bath, Half_Bath,
                            Fireplaces, Garage_Area, Gr_Liv_Area,
                            TotRms_AbvGrd) %>%
              replace(is.na(.), 0)
test_x <- model.matrix(Sale_Price ~ ., data = test_reg)[, -1]
test_y <- test_reg$Sale_Price
```

Ridge Regression

```r
library(glmnet)
ames_ridge <- glmnet(x = train_x, y = train_y, alpha = 0)
plot(ames_ridge, xvar = "lambda")
```

Ridge Regression

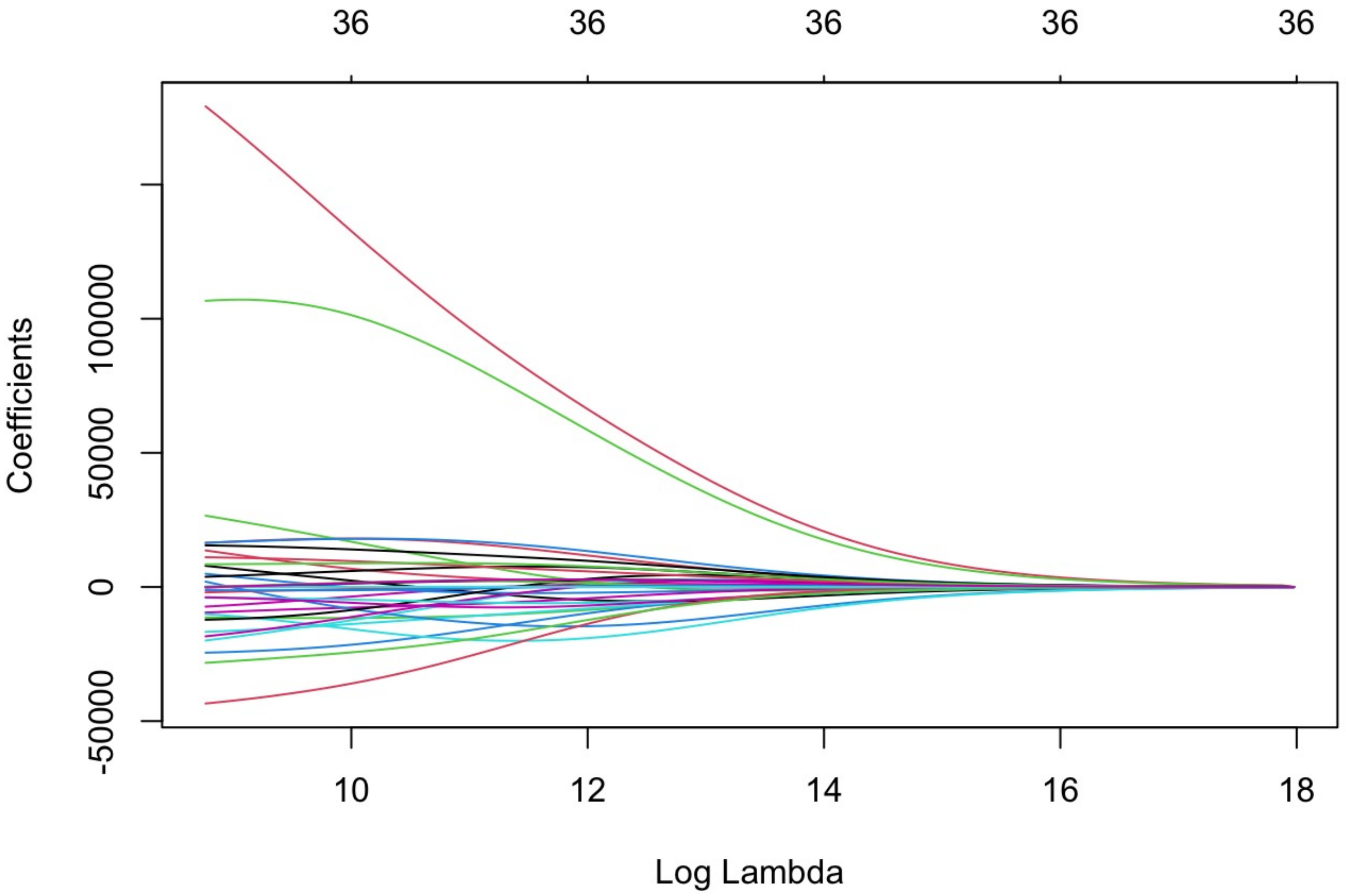
```r
library(glmnet)

ames_ridge <- glmnet(x = train_x, y = train_y, alpha = 0)

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

Option to use ridge penalty

Ridge Regression



Regularized Regression

LASSO REGRESSION

Penalties in Models

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

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

Penalties in Models

- 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.

Differences in Effects

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

Ridge regression approaches 0 asymptotically.

LASSO can have coefficients equal to 0
(variable removed from model).

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:

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

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

LASSO Regression

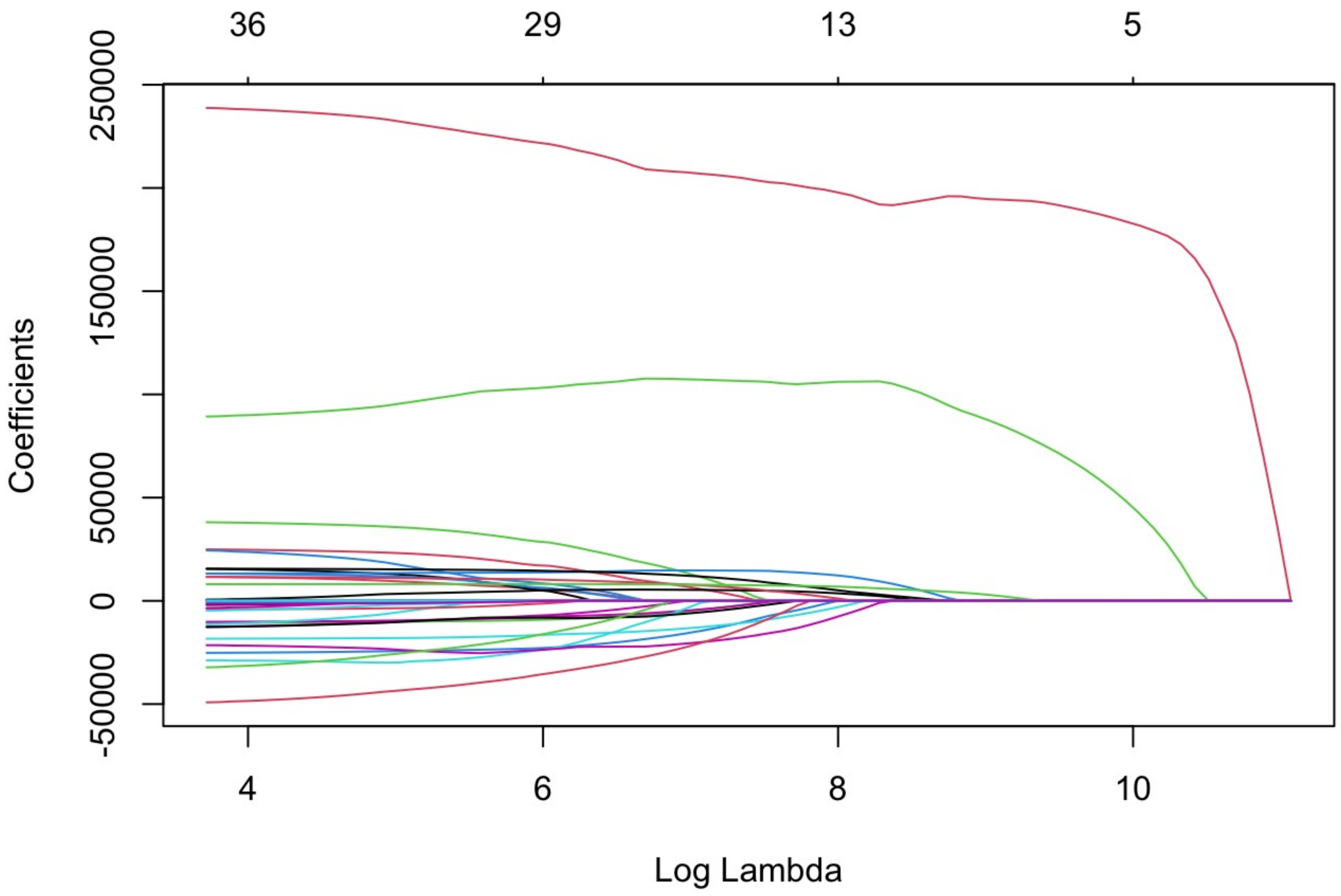
```r
library(glmnet)
ames_lasso <- glmnet(x = train_x, y = train_y, alpha = 1)
plot(ames_lasso, xvar = "lambda")
```

LASSO Regression

```
library(glmnet)
ames_lasso <- glmnet(x = train_x, y = train_y, alpha = 1)
plot(ames_lasso, xvar = "lambda")
```

Option to use LASSO penalty

LASSO Regression



Regularized Regression

ELASTIC NET REGRESSION

Penalties in Models

- Both ridge and LASSO have advantages and disadvantages.
  - LASSO does variable selection.
  - Ridge keeps all variables (LASSO drops arbitrarily)

- Elastic net regression combines both penalty terms in the minimization:

$$\min\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{p} |\hat{\beta}_j| + \lambda_2 \sum_{j=1}^{p} \hat{\beta}_j^2\right)$$

Penalties in Models

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

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

Penalties in Models

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

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

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

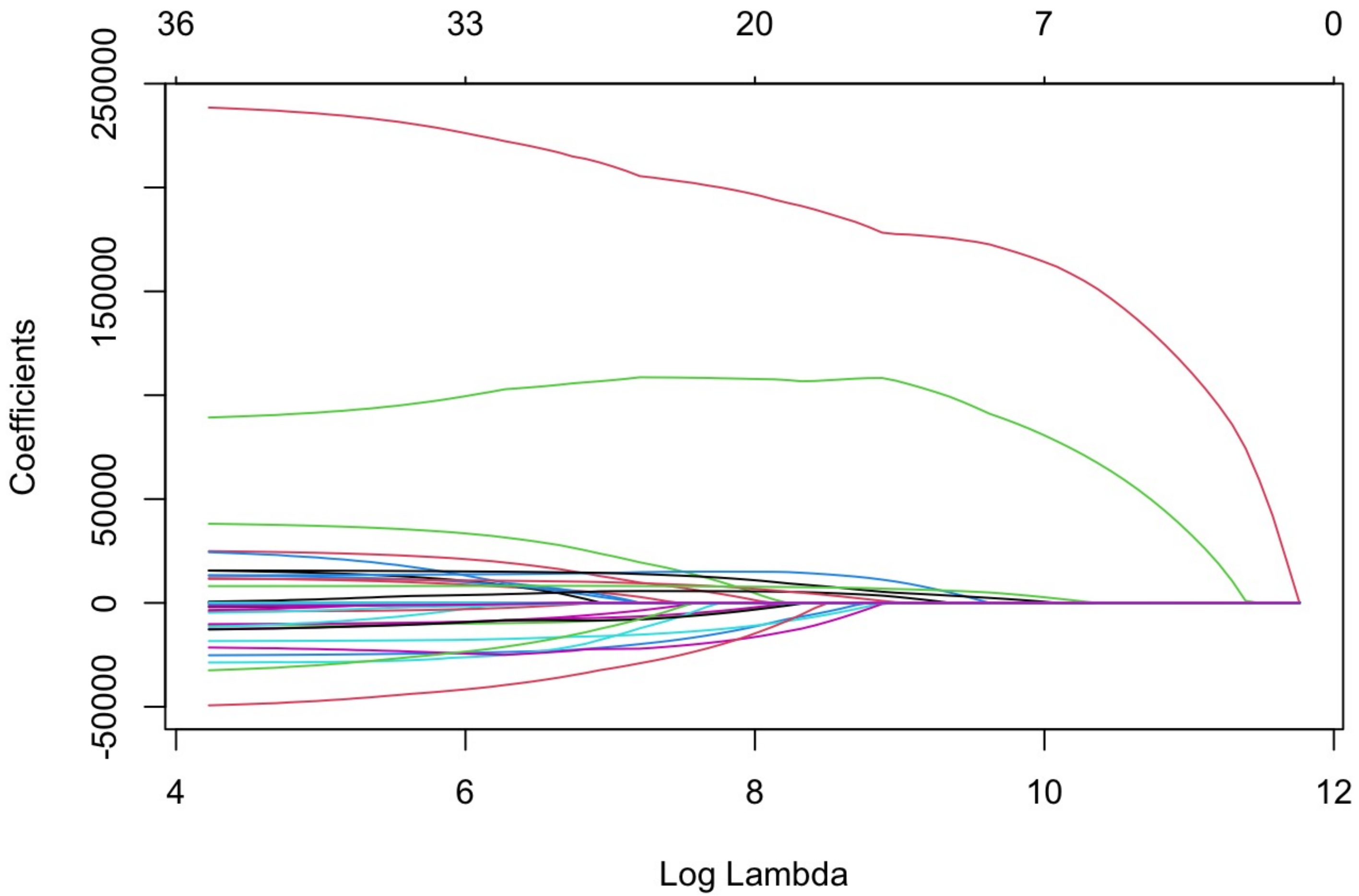
```
library(glmnet)
ames_en <- glmnet(x = train_x, y = train_y, alpha = 0.5)
plot(ames_en, xvar = "lambda")
```

Elastic Net Regression

```
library(glmnet)
ames_en <- glmnet(x = train_x, y = train_y, alpha = 0.5)
plot(ames_en, xvar = "lambda")
```

Use elastic net since value between 0 and 1.

Elastic Net Regression

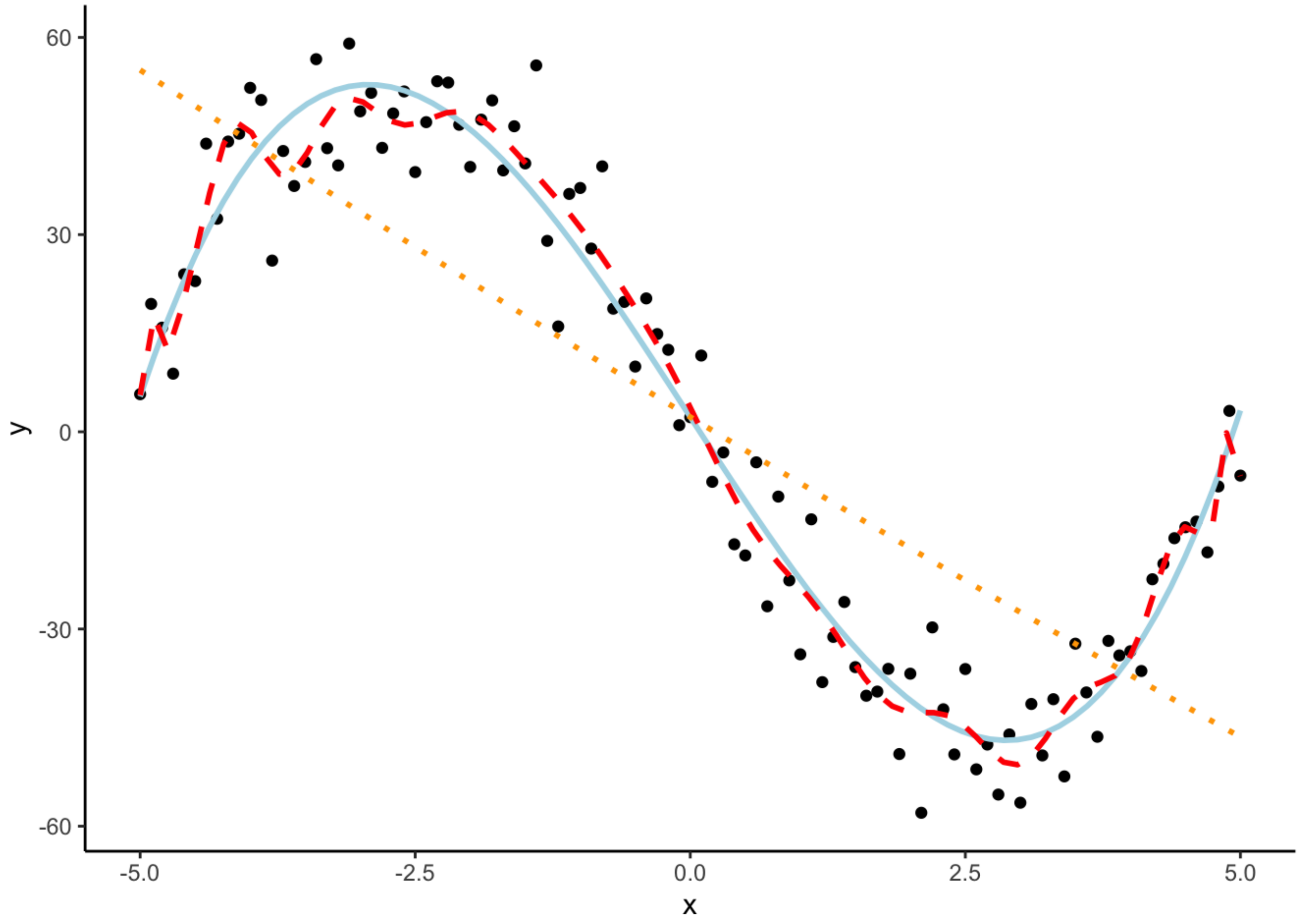


?

Optimizing Penalties

Fear of Overfitting

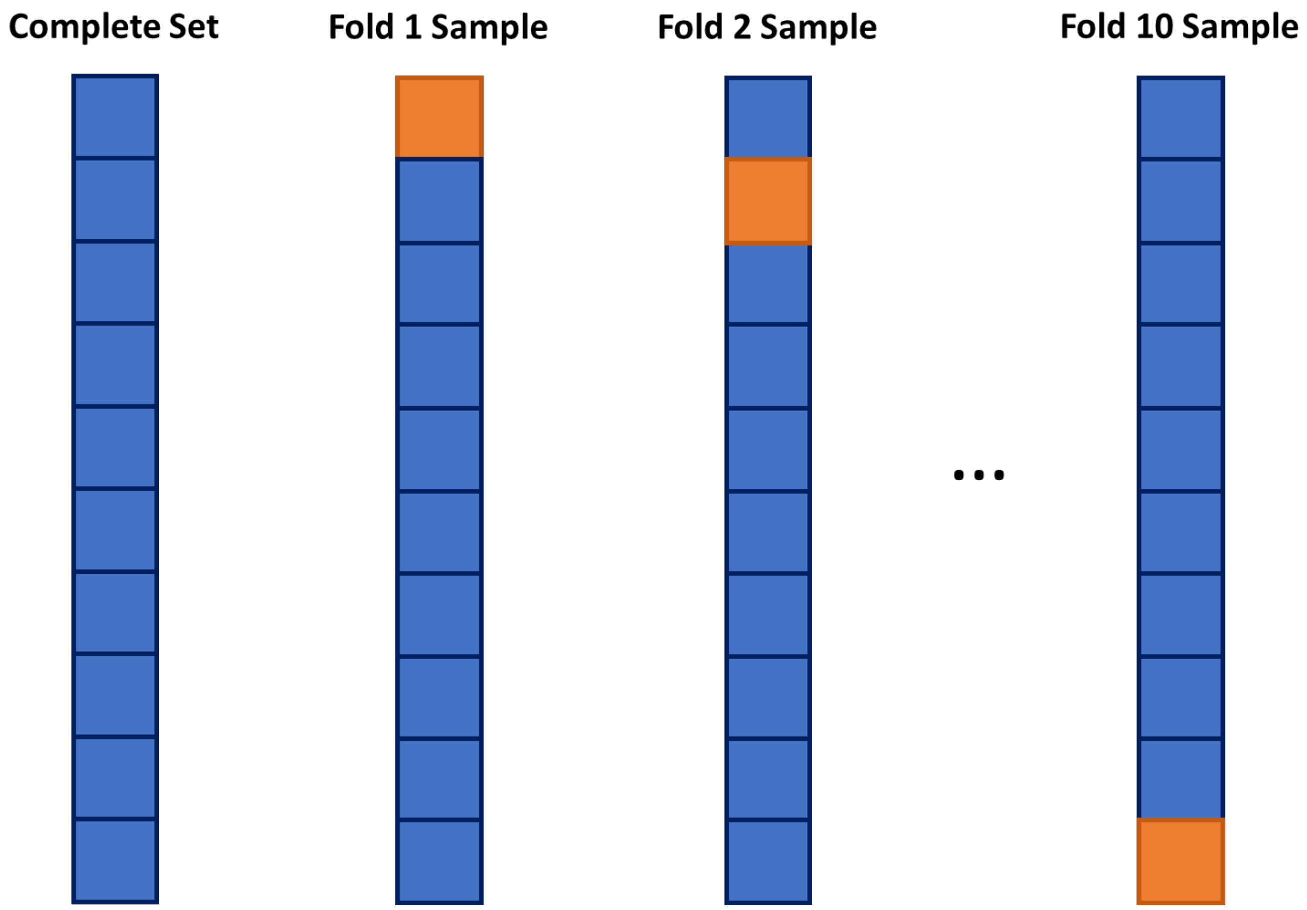
- Need to select λ for any of the regularized regression approaches.
- Don't want to minimize variance to the point of overfitting our model to the training data.



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

k-fold Cross-Validation

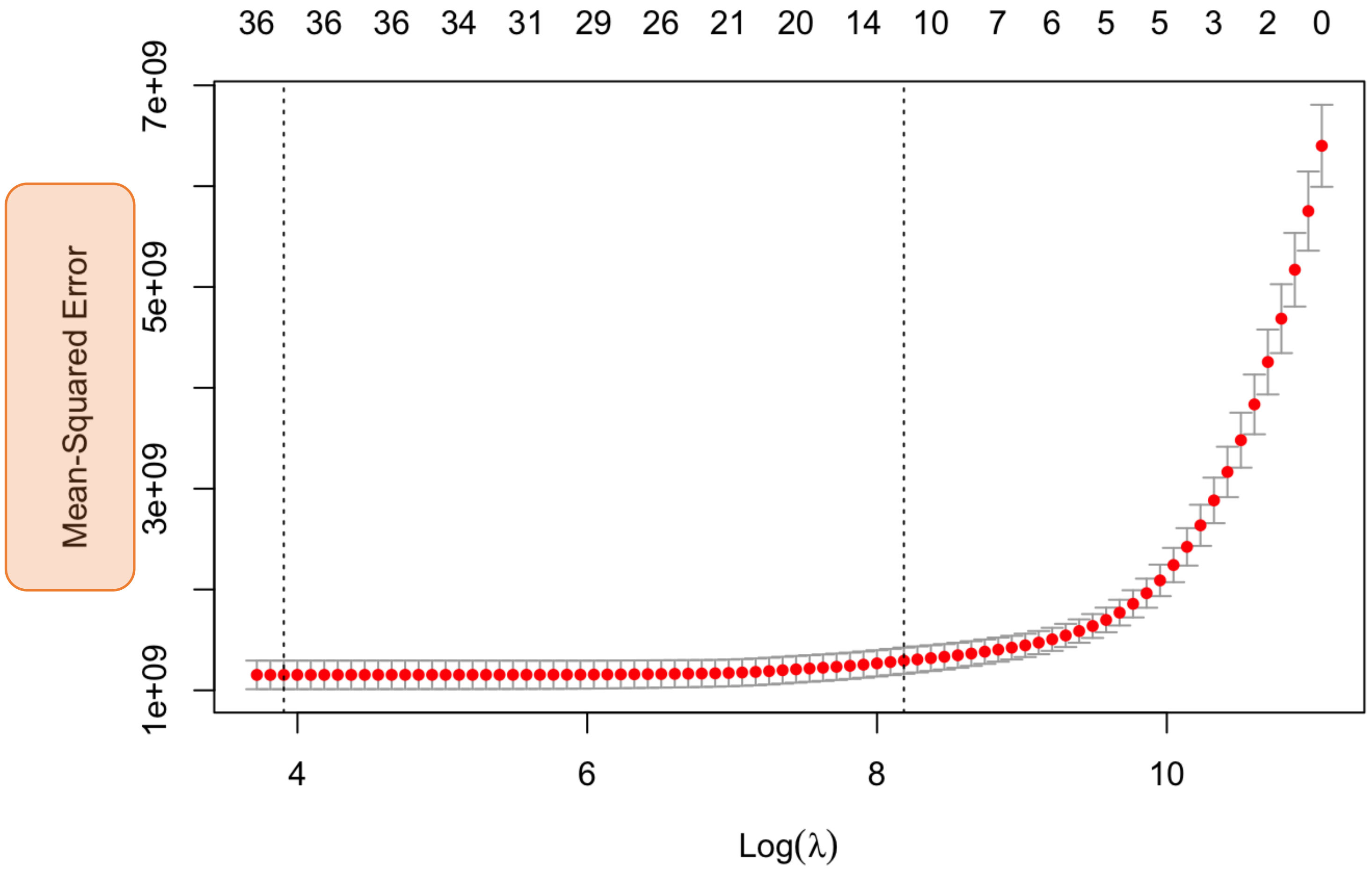


LASSO Regression

```
ames_lasso_cv <- cv.glmnet(x = train_x, y = train_y, alpha = 1)
plot(ames_lasso_cv)
```

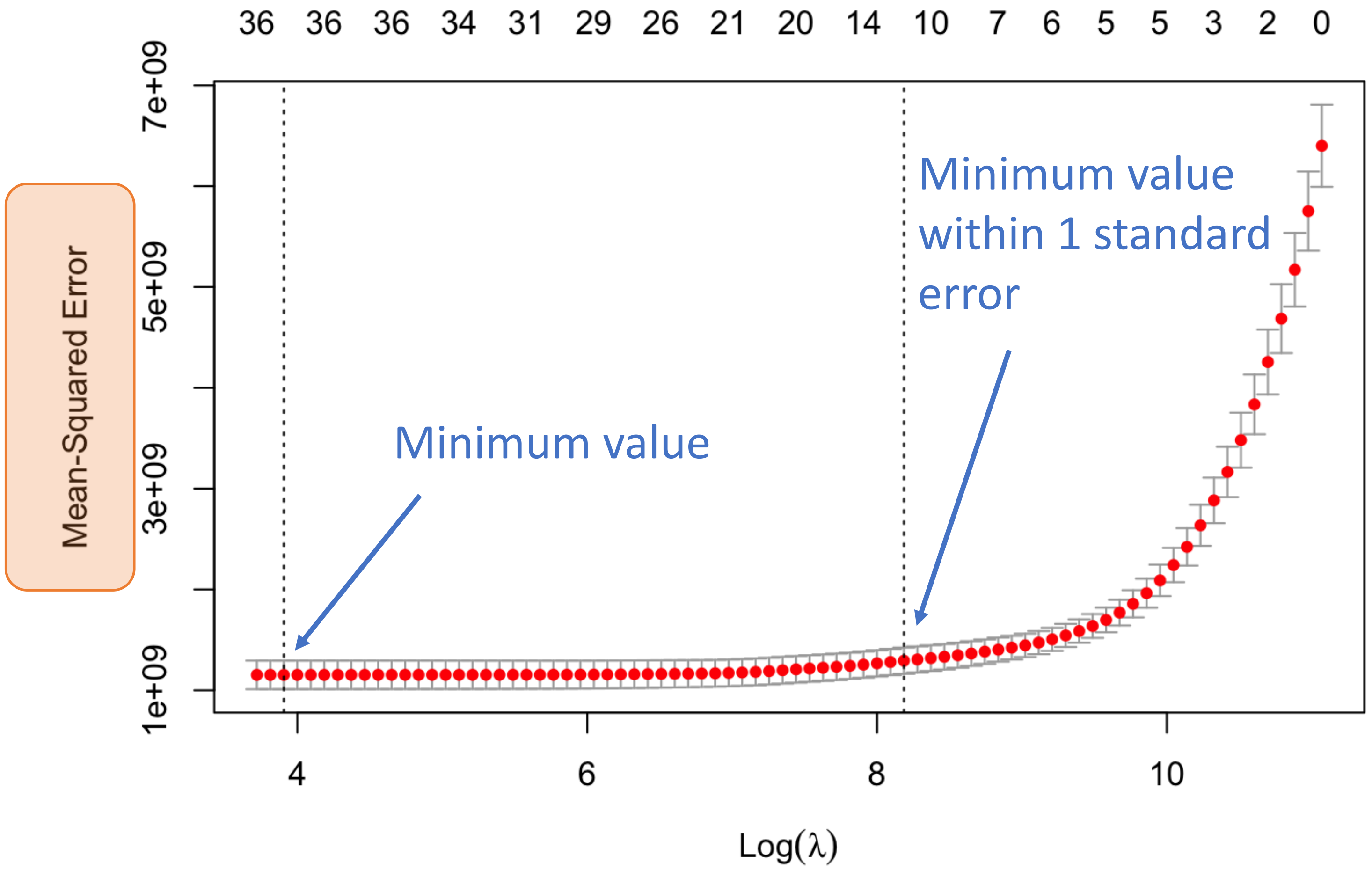
LASSO Regression

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



LASSO Regression

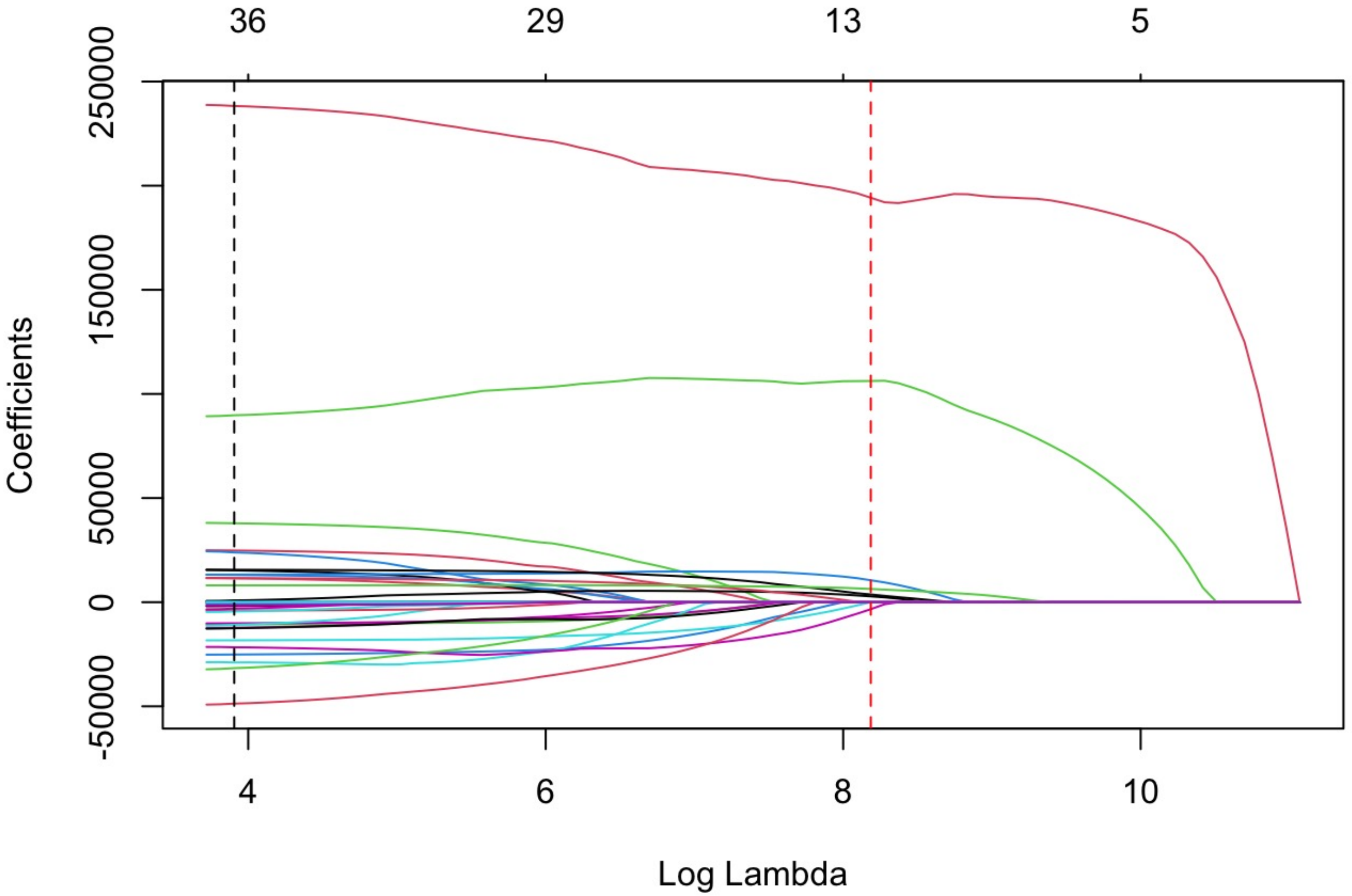
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



LASSO Regression

```r
plot(ames_lasso, xvar = "lambda")
abline(v = log(ames_lasso_cv$lambda.1se), col = "red", lty = "dashed")
abline(v = log(ames_lasso_cv$lambda.min), col = "black", lty = "dashed")
```

LASSO Regression



Important Variables

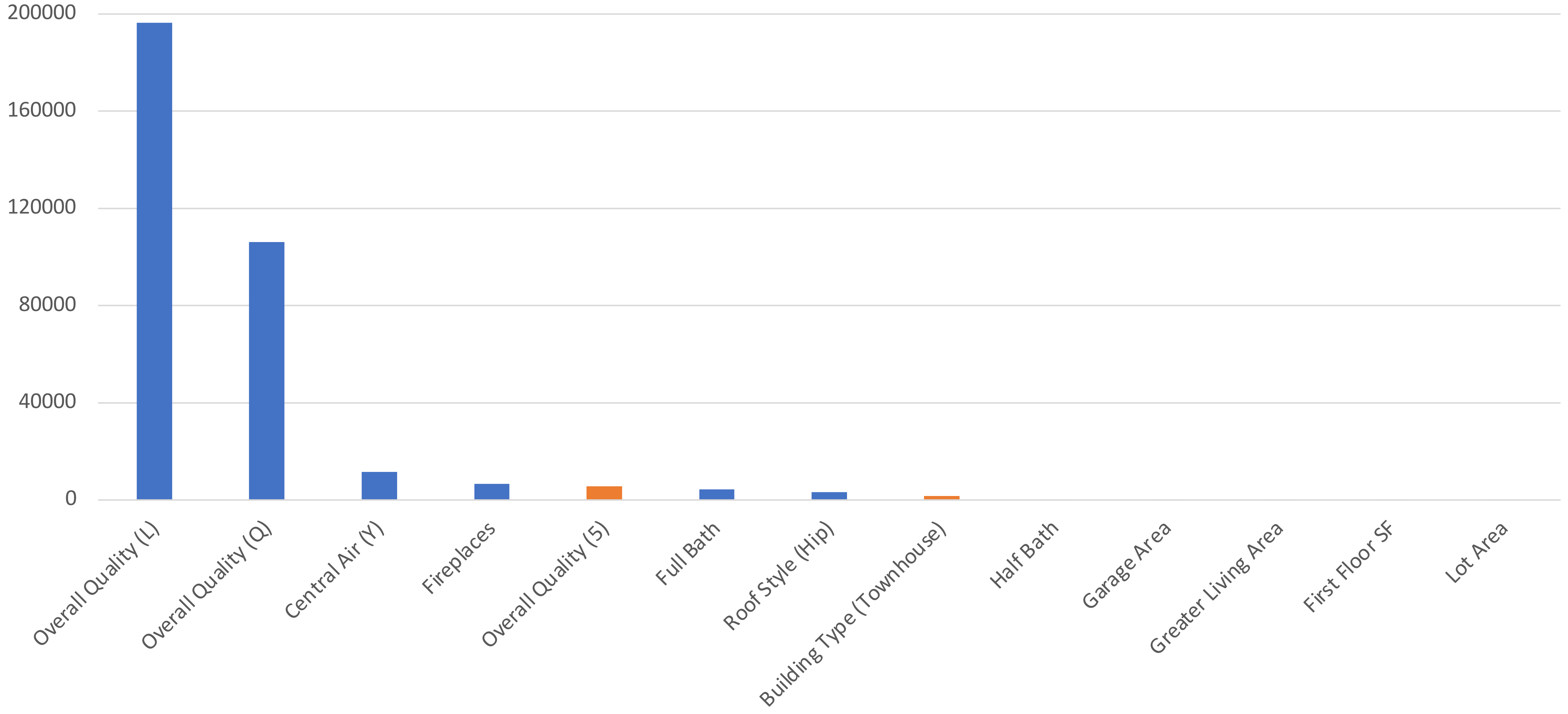
```
coef(ames_lasso, s = c(ames_lasso_cv$lambda.min, ames_lasso_cv$lambda.1se))
```

Important Variables

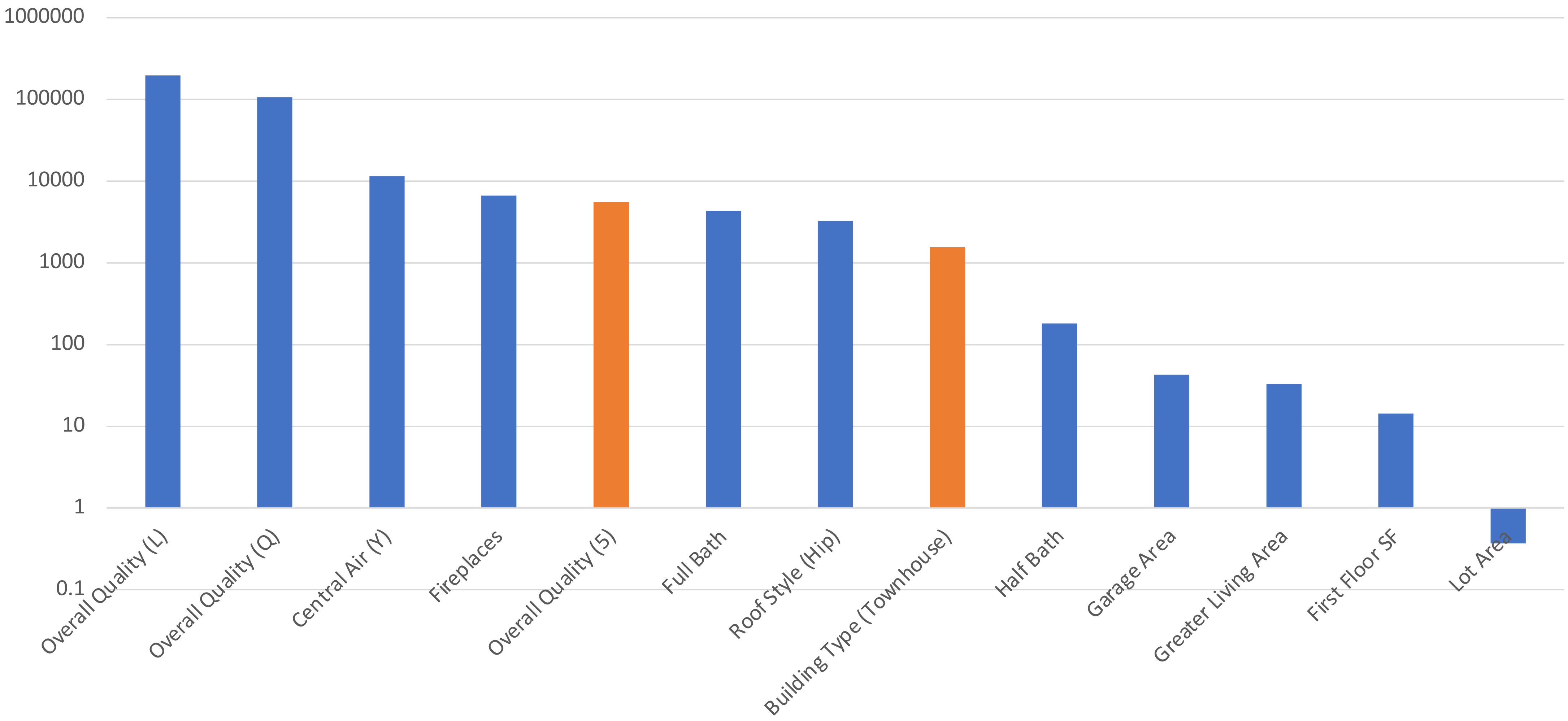
```
                                     s1            s2
(Intercept)                4.809883e+04  8.326132e+04
Lot_Area                   5.455632e-01  3.727557e-01
StreetPave                 7.742774e+03  .
Bldg_TypeTwoFmCon         -9.791571e+03  .
Bldg_TypeDuplex           -2.380411e+04  .
Bldg_TypeTwnhs            -1.755640e+04 -1.552627e+03
Bldg_TypeTwnhsE           -8.901776e+03  .
House_StyleOne_and_Half_Unf  1.006755e+04  .
House_StyleOne_Story       2.100854e+04  .
House_StyleSFoyer          3.314566e+04  .
House_StyleSLvl            9.806126e+03  .
House_StyleTwo_and_Half_Fin -2.786798e+04  .
House_StyleTwo_and_Half_Unf -8.735039e+03  .
House_StyleTwo_Story       .              .
```

⋮

Important Variables



Important Variables



?

Model Comparisons

Comparing Models

- The model results in a formula or rules.
- The data require modifications:
  - Derived inputs
  - Transformations
  - Missing value imputation

- To score/compare, you **do not rerun the algorithm**!
- Apply score code (equations) obtained from the final model to the test data for comparing.

Comparing Models

- Test dataset is for comparing final models and reporting final metrics.
- **DO NOT GO BACK AFTER TO REBUILD MODEL!**
- **DO NOT JUST BUILD 1000’s OF MODELS TO COMPARE IN THE TEST SET!**
- We do not want to fit to the test dataset as it is our honest assessment of how good our models can do.

Model Metrics

- Root MSE (RMSE):

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

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Mean Absolute Percentage Error (MAPE):

$$MAPE = 100\times\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Model Metrics

- Root MSE (RMSE):

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

Problems:

- Not easily interpretable

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Problems:

- Not scale invariant

- Mean Absolute Percentage Error (MAPE):

Problems:

- Not symmetric

$$MAPE = 100\times\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Predictions

```r
test$pred_lm <- predict(ames_lm, newdata = test)
head(test$pred_lm)
```

```
##        1        2        3        4        5        6
## 142107.3 142107.3 228909.6 142107.3 142107.3 142107.3
```

```r
test_reg$pred_lasso <- predict(ames_lasso, s = ames_lasso_cv$lambda.1se, newx = test_x)
head(test_reg$pred_lasso)
```

```
##        1        2        3        4        5        6
## 156677.8 172432.5 239922.1 105713.6 200908.8 124913.5
```

Predictions – MAPE

```
test %>%
  mutate(lm_APE = 100*abs((Sale_Price - pred_lm)/Sale_Price)) %>%
  dplyr::summarise(MAPE_lm = mean(lm_APE))
```

```
##   MAPE_lm
##     <dbl>
## 1    23.2
```

```
test_reg %>%
  mutate(lasso_APE = 100*abs((Sale_Price - pred_lasso)/Sale_Price)) %>%
  dplyr::summarise(MAPE_lasso = mean(lasso_APE))
```

```
##   MAPE_lasso
##        <dbl>
## 1       13.4
```

?