# 2. Regularization and Variable Selection

Hair Parra

2024-01-13

## Variable Selection in Linear Regression

### Example: Synthetic data variable selection

Consider the process with three covariates $X_1, X_2, X_3$ and numeric target $Y$ given by:

$$
\begin{aligned}
X_1 &\sim \mathcal{N}(0,1) \\
X_2 &= X_1 + \epsilon_X \quad , \quad \epsilon_X \sim \mathcal{N}(0, 0.5^2) \\
X_3 &\sim \mathcal{N}(0,1) \\
Y &= X_1 + X_2 + X_3 + \epsilon_Y \quad , \quad \epsilon_Y \sim \mathcal{N}(0, 3^2)
\end{aligned}
$$

The goal is to find the best subset of covariates that explain the target $Y$.

```
# Generate training and test data

# Set the random seed for reproducibility
set.seed(364575)

# Produce train data with the described process
n = 100
x1 = rnorm(n)
x2 = x1 + 0.5 * rnorm(n)
x3 = rnorm(n)
y = x1 + x2 + x3 + 3 * rnorm(n)
matx = cbind(x1, x2, x3)
dat = data.frame(y, x1, x2, x3)
names(dat) = c("y", "x1", "x2", "x3")

# Test data
ntest = 10000
x1new = rnorm(ntest)
x2new = x1new + 0.5 * rnorm(ntest)
x3new = rnorm(ntest)
ynew = x1new + x2new + x3new + 3 * rnorm(ntest)
matxnew = cbind(x1new, x2new, x3new)
datnew = data.frame(x1new, x2new, x3new)
names(datnew) = c("x1", "x2", "x3")

# display both datasets
head(dat)
```

```
##           y          x1         x2          x3
## 1 -2.831217 -1.82557131 -1.4850248 -0.09075717
## 2  1.251882  1.00989334  1.6541838 -0.89472200
## 3 -1.144341 -0.03561269  0.9470485  1.32178335
## 4 -4.390979  0.06701368 -0.3592665 -1.38134179
```

```
## 5 -3.329712  1.18924992  0.3959552  0.25001775
## 6 -3.744215 -1.12926049 -1.0053691 -1.01140464
```

```
head(datnew) # test set does not contain Y
```

```
##           x1          x2         x3
## 1  0.4555318  0.03448935 -0.3022184
## 2  1.1450641  1.12901501  0.5364467
## 3 -0.4777378 -1.44942379 -0.2828134
## 4  0.2076905 -0.25621060  0.5748806
## 5  0.5715914  0.12289959 -0.4796574
## 6 -1.1856869 -1.22423062  0.4910526
```

```
# inspect the correlation between the data
cor(cbind(y,x1,x2,x3))
```

```
##            y        x1        x2        x3
## y  1.0000000 0.6080312 0.5857846 0.3119057
## x1 0.6080312 1.0000000 0.8818763 0.1216682
## x2 0.5857846 0.8818763 1.0000000 0.0914663
## x3 0.3119057 0.1216682 0.0914663 1.0000000
```

We note that $X_1$ and $X_2$ are highly correlated.

Next, we fit the model:

```
# ordinary OLS
lm.fit=lm(y~x1+x2+x3)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1025 -1.8028  0.2313  2.1827  8.6000
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1114     0.3318   0.336  0.73784
## x1            1.4948     0.6640   2.251  0.02665 *
## x2            0.8642     0.5824   1.484  0.14115
## x3            1.0087     0.3165   3.187  0.00194 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.166 on 96 degrees of freedom
## Multiple R-squared:   0.44,  Adjusted R-squared:  0.4225
## F-statistic: 25.14 on 3 and 96 DF,  p-value: 4.331e-12
```

We see that $\hat{\beta_1}$ and $\hat{\beta_2}$ are not well estimated! (True value is 1). Additionally, the errors are twice as large as compared to $\hat{\beta_3}$.

## Stepwise Regression

AIC and BIC in latex:

$$\text{AIC} = -2\log(\hat{\theta}_{MLE}) + 2p$$
$$\text{BIC} = -2\log(\hat{\theta}_{MLE}) + \log(n)p$$

```r
# load librarires
library(MASS)

# stepwise with AIC as the criterion
step.AIC=stepAIC(lm.fit,direction="both")
```

```
## Start:  AIC=234.4
## y ~ x1 + x2 + x3
##
##        Df Sum of Sq     RSS    AIC
## <none>               962.16 234.40
## - x2    1    22.065  984.22 234.67
## - x1    1    50.798 1012.95 237.55
## - x3    1   101.793 1063.95 242.46
```

```r
# produce predictions on the test set
predaic=predict(lm.fit,newdata=datnew)
```

```r
# stepwise with BIC as the criterion
stepAIC(lm.fit,direction="both",k=log(n))
```

```
## Start:  AIC=244.82
## y ~ x1 + x2 + x3
##
##        Df Sum of Sq     RSS    AIC
## - x2    1    22.065  984.22 242.48
## <none>               962.16 244.82
## - x1    1    50.798 1012.95 245.36
## - x3    1   101.793 1063.95 250.27
##
## Step:  AIC=242.48
## y ~ x1 + x3
##
##        Df Sum of Sq     RSS    AIC
## <none>               984.22 242.48
## + x2    1     22.06  962.16 244.82
## - x3    1     98.73 1082.95 247.44
## - x1    1    566.78 1551.00 283.36

##
## Call:
## lm(formula = y ~ x1 + x3)
##
## Coefficients:
## (Intercept)           x1           x3
##      0.1941       2.3627       0.9928
```

```r
# refit model with covariates selected by BIC
lm.fitbic=lm(y~x1+x3)

# produce preditions on the test set
predbic=predict(lm.fitbic,newdata=datnew)
```

## Applying Ridge Regression

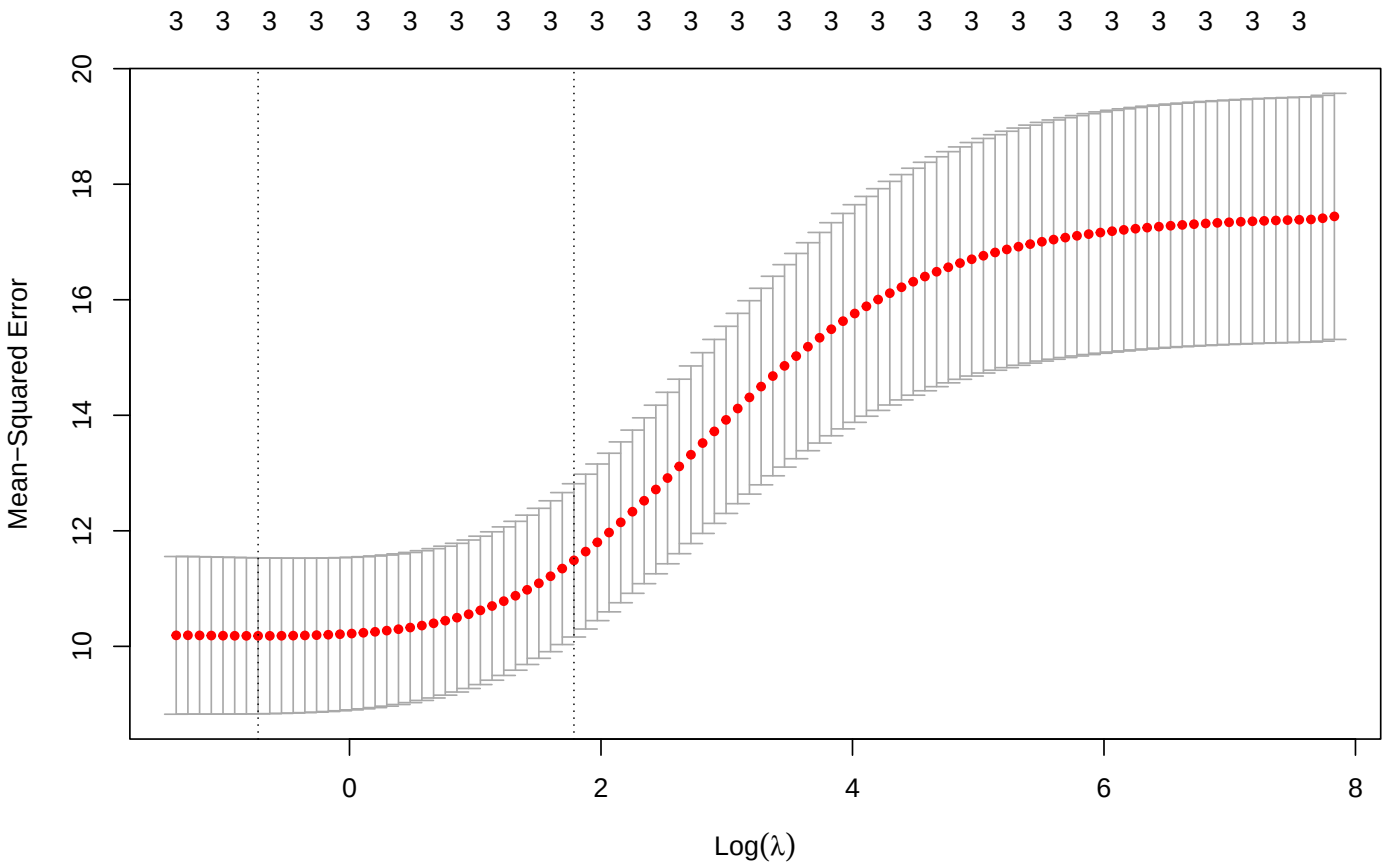
```r
# load libraries
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
# Cross-validation to select the best value of lambda for ridge regression (alpha=0)
glmnet.cv = cv.glmnet(matx, y, alpha = 0)

# Plot the cross-validation results
plot(glmnet.cv)
```




We can then extract the best value of $\lambda$ and get the predictions with the model that uses it.

```r
# Find the best lambda value for ridge regression
bestlridge = glmnet.cv$lambda.min

# Predict using ridge regression with the best lambda
predridge = predict(glmnet.cv, new = matxnew, s = bestlridge)

# Display the coefficients for the selected lambda
predict(glmnet.cv, s = bestlridge, type = "coefficients")
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 0.09284517
```

```
## x1          1.30583677
## x2          0.91164666
## x3          0.91963083
```

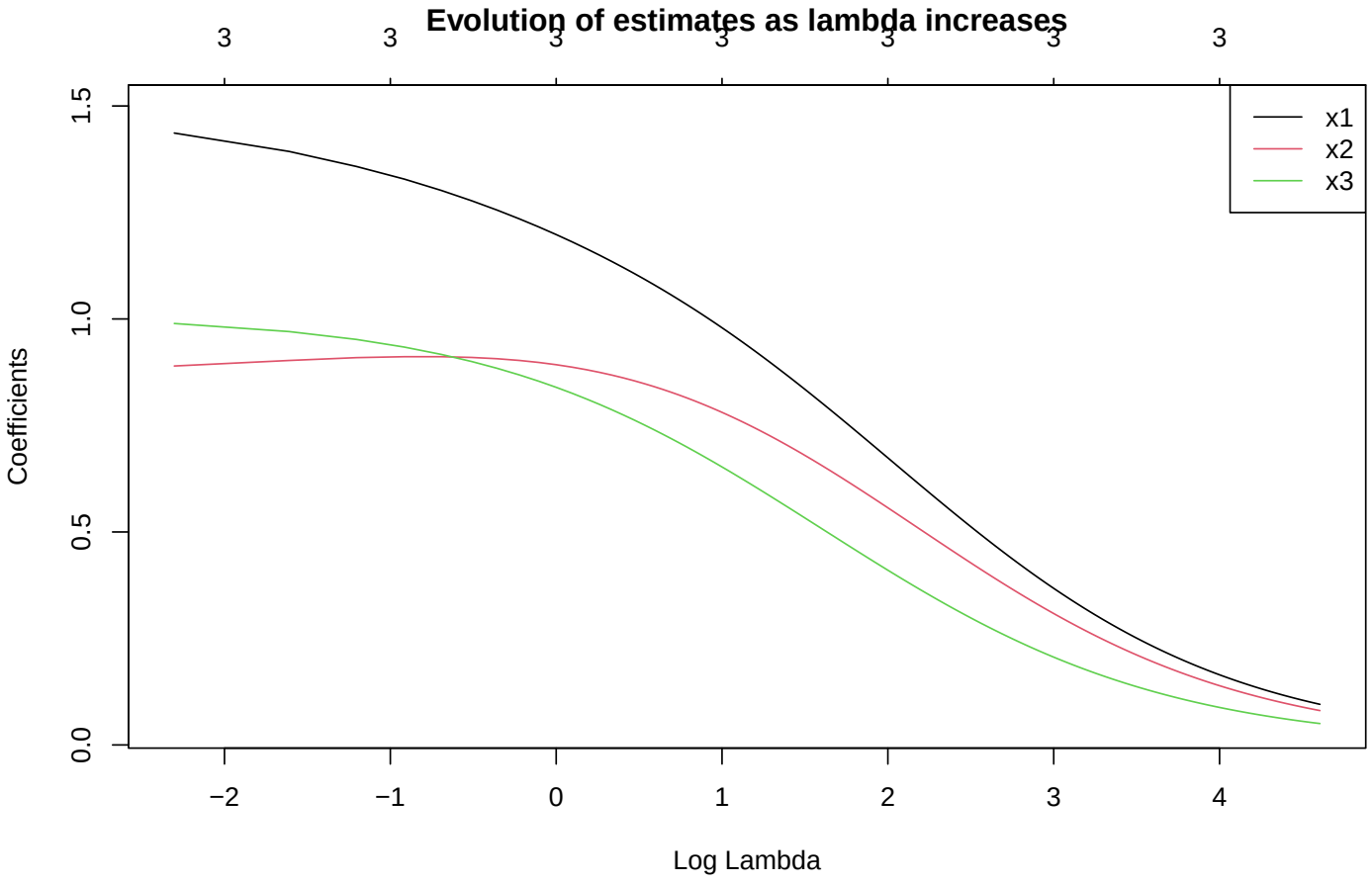
```
print(paste0("best lambda: ", bestlridge))
```

```
## [1] "best lambda: 0.483375692426974"
```

- $\hat{\beta}_1$ and $\hat{\beta}_2$ are closer to their true value!
- In general, $\hat{\beta} \to 0$ as $\lambda \to \infty$.
- But some coefficients might also start to increase when $\lambda$ begins to increase.

```
# Plot the Ridge (L2 regularization) regularization path for a range of lambda values
plot(glmnet(matx, y, alpha = 0, lambda = seq(0, 100, 0.1)),
     main = "Evolution of estimates as lambda increases",
     xvar = "lambda", label = TRUE)

# Add labels to the lines representing coefficients
legend("topright", legend = c("x1", "x2", "x3"), col = 1:3, lty = 1)
```




Finally, we can compute the mean squared error on the test set for the three models: - OLS (using stepwise with AIC) - OLS (using stepwise withBIC) - ridge regression with $\lambda$ selected by cross-validation.

```
# Calculate the mean squared error for the three models: AIC, BIC, ridge
mse_results <- c(
  "OLS with AIC" = mean((predaic-ynew)^2),
  "OLS with BIC" = mean((predbic-ynew)^2),
  "Ridge Regression" = mean((predridge-ynew)^2)
```

```
)

# Display the mean squared error for each model with their names
mse_results
```

```
##     OLS with AIC     OLS with BIC Ridge Regression
##         9.175325         9.451781         9.098615
```

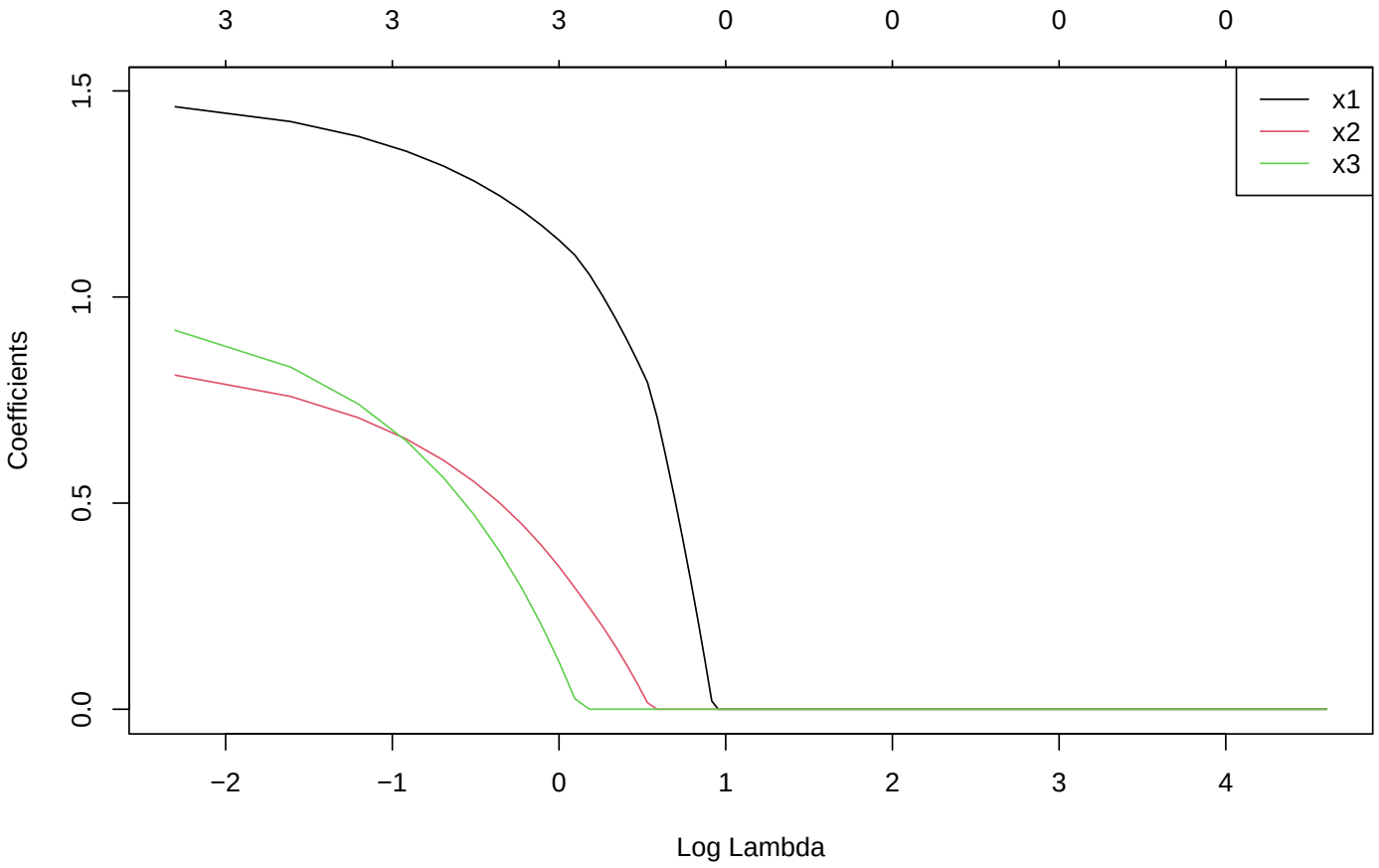
We see that the best performance is achieved by ridge regression followed by OLS (stepwise with AIC).

**Remark:** Since $\epsilon_Y \sim \mathcal{N}(0, 3^2) \implies \mathbb{V}[\epsilon] = 9$ which is the irreducible variance. So we are close to the best possible performance.

## Applying Lasso Regression

```
# Plot the Lasso (L1 regularization) regularization path for a range of lambda values
plot(glmnet(matx, y, alpha = 1, lambda = seq(0, 100, 0.1)), xvar = "lambda", label = TRUE)

# Add labels to the lines representing coefficients
legend("topright", legend = c("x1", "x2", "x3"), col = 1:3, lty = 1)
```




We see that $\hat{\beta}_3$ reaches 0 first, followed by $\hat{\beta}_2$, and when $\lambda$ is about $2.7 = e^1$, all coefficients are 0. Thus the model selects: 1. All variables 2. Only $X_1$ and $X_2$ 3. Only $X_1$

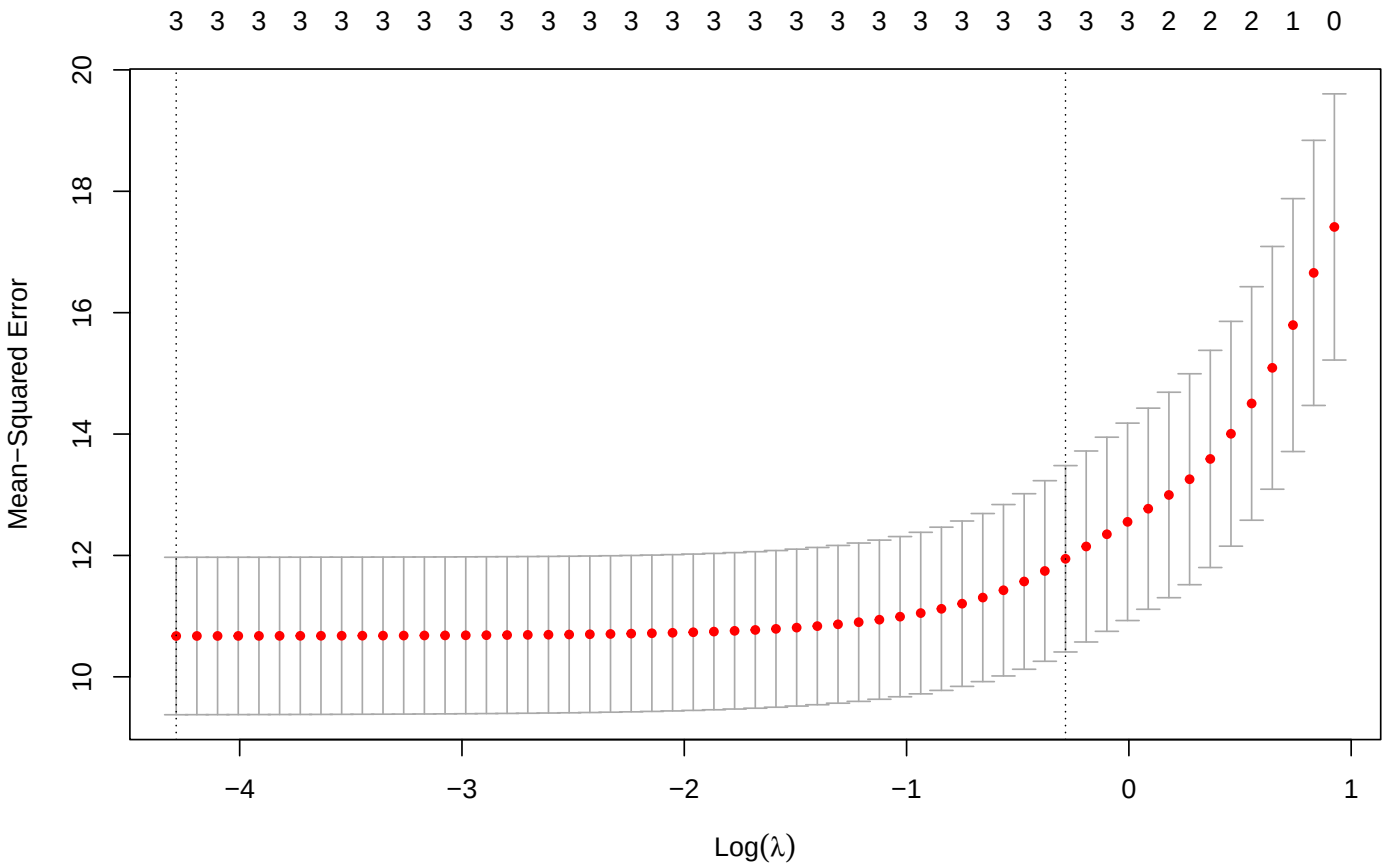
Recall the original generation process:

$$
\begin{aligned}
X_1 &\sim \mathcal{N}(0,1) \\
X_2 &= X_1 + \epsilon_X \quad , \quad \epsilon_X \sim \mathcal{N}(0, 0.5^2) \\
X_3 &\sim \mathcal{N}(0,1) \\
Y &= X_1 + X_2 + X_3 + \epsilon_Y \quad , \quad \epsilon_Y \sim \mathcal{N}(0, 3^2)
\end{aligned}
$$

### Lasso: Optimal $\lambda$ with cross-validation

```
# Perform cross-validation to select the best value of lambda for Lasso regression (alpha=1)
glmnet.cv = cv.glmnet(matx, y, alpha = 1)

# plot the cross-validation results
plot(glmnet.cv)
```




```
# Get the lambda value with the minimum cross-validation error for Lasso
bestllasso = glmnet.cv$lambda.min

# Display the best lambda value for Lasso
print(paste0("best lambda (lasso) = ", bestllasso))
```

```
## [1] "best lambda (lasso) = 0.0137667130996201"
```

```
# Predict using Lasso regression with the best lambda
predlasso = predict(glmnet.cv, new = matxnew, s = bestllasso)

# Display the coefficients for the selected lambda in Lasso regression
predict(glmnet.cv, s = bestllasso, type = "coefficients")
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 0.1117049
## x1          1.4912090
## x2          0.8560317
## x3          0.9963326
```

Let's inspect the MSE:

```
# Calculate the mean squared error for different models and store them in a named vector
mse_results <- c(
  "OLS with AIC" = mean((predaic - ynew)^2),
  "OLS with BIC" = mean((predbic - ynew)^2),
  "Ridge Regression" = mean((predridge - ynew)^2),
  "Lasso Regression" = mean((predlasso - ynew)^2)
)

# Display the names of models and their respective mean squared errors
mse_results
```

```
##     OLS with AIC     OLS with BIC Ridge Regression Lasso Regression
##         9.175325         9.451781         9.098615         9.167900
```

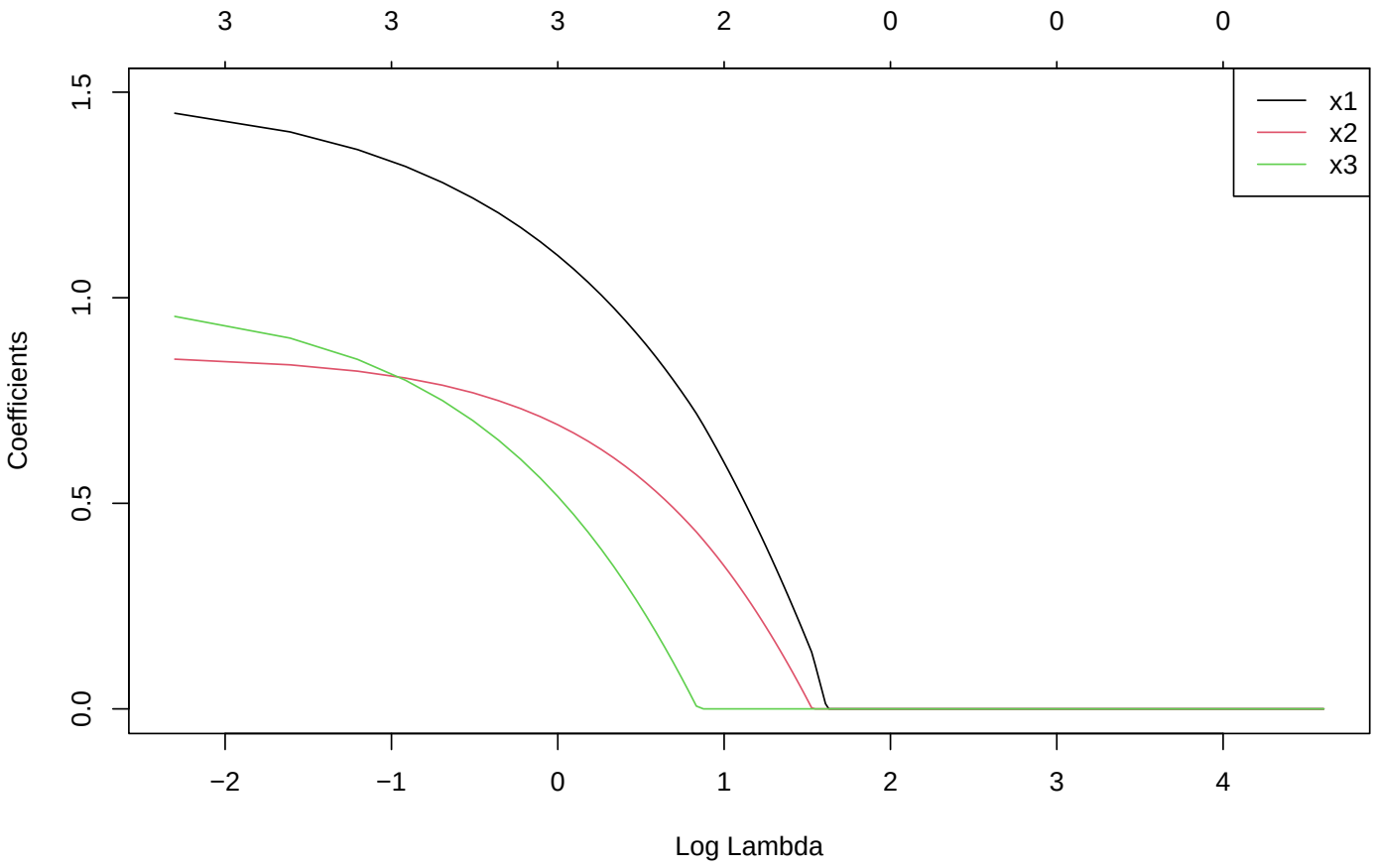
- Here, Ridge » Lasso > OLS (AIC) » OLS (BIC).
- **Ridge** tens to shrink the coefficients towards 0 to produce an "average" effect.
- **Lasso** will tend to favour one of the correlated covariates.
- Similar predictive performance.

## Elastic Net Regression

We can observe the effect on the parameters with $\alpha = 0.5$

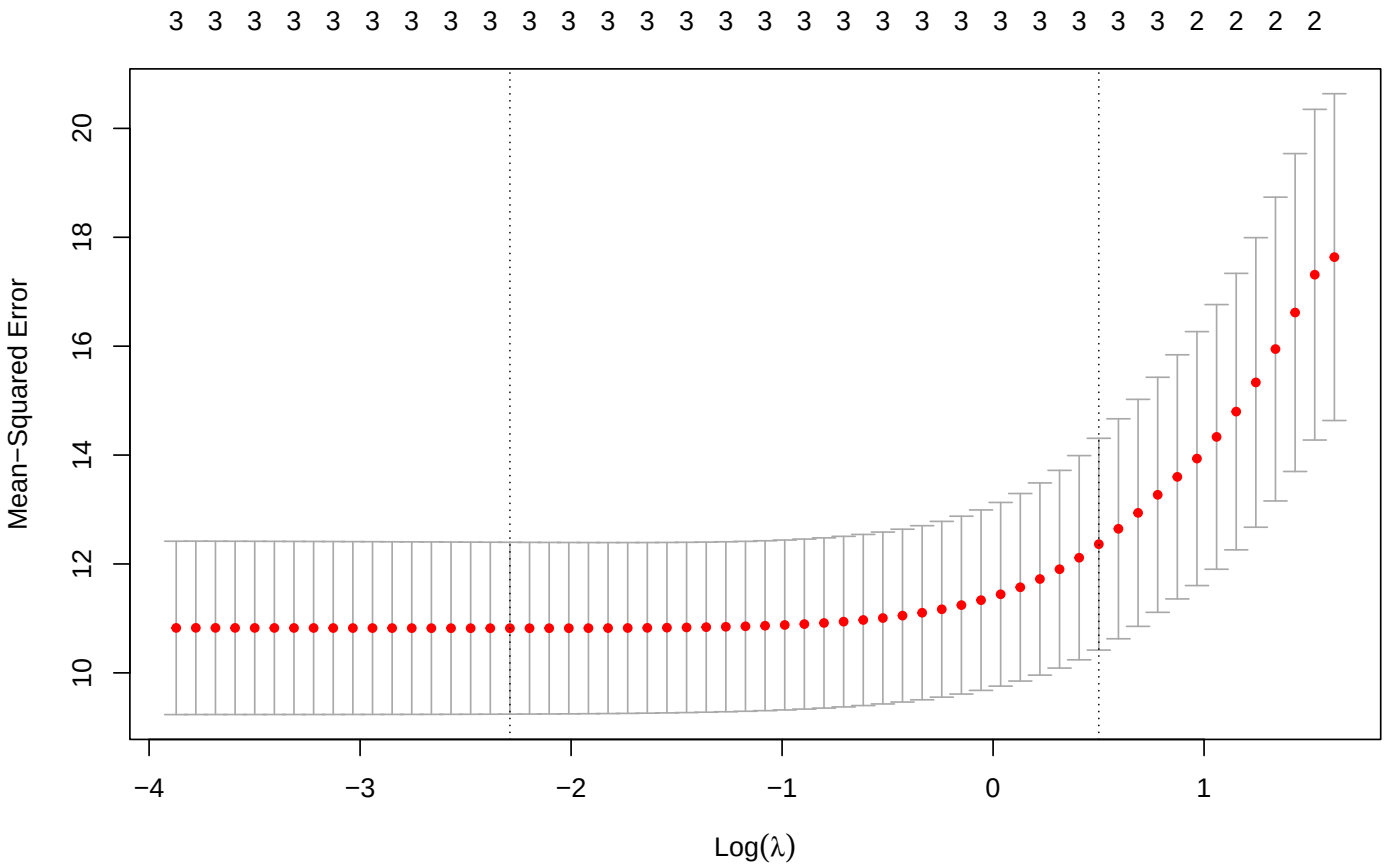
```
# elastic net regression with the glmnet package
plot(glmnet(matx, y, alpha=.5,lambda=seq(0,100,.1)),xvar = "lambda", label = TRUE)

# Add labels to the lines representing coefficients
legend("topright", legend = c("x1", "x2", "x3"), col = 1:3, lty = 1)
```




Similarly, the cross-validation for $\lambda$:

```
# Perform cross-validation to select the best value of lambda for elastic net regression (alpha=0.5)
glmnet.cv = cv.glmnet(matx, y, alpha=.5)
plot(glmnet.cv)
```




```
# Elastic Net regression with the glmnet package

# Get the lambda value with the minimum cross-validation error for Elastic Net
bestllel = glmnet.cv$lambda.min

# Display the best lambda value for Elastic Net
bestllel
```

```
## [1] 0.1012784
```

```
# Predict using Elastic Net regression with the best lambda
predelasticnet = predict(glmnet.cv, new = matxnew, s = bestllel)

# Display the coefficients for the selected lambda in Elastic Net regression
predict(glmnet.cv, s = bestllel, type = "coefficients")
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 0.1090233
## x1          1.4472756
## x2          0.8511481
## x3          0.9540014
```

```
# Calculate the mean squared error for different models and store them in a named vector
mse_results <- c(
  "OLS with AIC" = mean((predaic - ynew)^2),
  "OLS with BIC" = mean((predbic - ynew)^2),
  "Ridge Regression" = mean((predridge - ynew)^2),
```

```
  "Lasso Regression" = mean((predlasso - ynew)^2),
  "Elastic Net (alpha=0.5)" = mean((predelasticnet - ynew)^2)
)

# Display the names of models and their respective mean squared errors
mse_results
```

```
##            OLS with AIC            OLS with BIC        Ridge Regression
##                9.175325                9.451781                9.098615
##        Lasso Regression Elastic Net (alpha=0.5)
##                9.167900                9.139655
```

- In this case, Elastic Net performs better than Lasso slightly, but not better than Ridge.
- Note that `glmnet` does not have a function to automatically select the best value of $\alpha$.

## Ames Data Example

- alternative to the well-known Boston Housing data set.
- It has 2,330 observations and 82 variables and contains information from the Ames Assessor's Office used in computing assessed values for individual residential properties sold in Ames, IA from 2006 to 2010.
- https://ww2.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt