# Regularization

Martyna

9/28/2020

**Load libraries, get data & set seed for reproducibility**

```
set.seed(222)    # seef for reproducibility
library(glmnet)  # for ridge regression
```

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

```
## Loaded glmnet 4.0-2
```

```
library(dplyr)   # for data cleaning
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(psych)   # for function tr() to compute trace of a matrix
data("mtcars")
```

## RIDGE REGRESSION

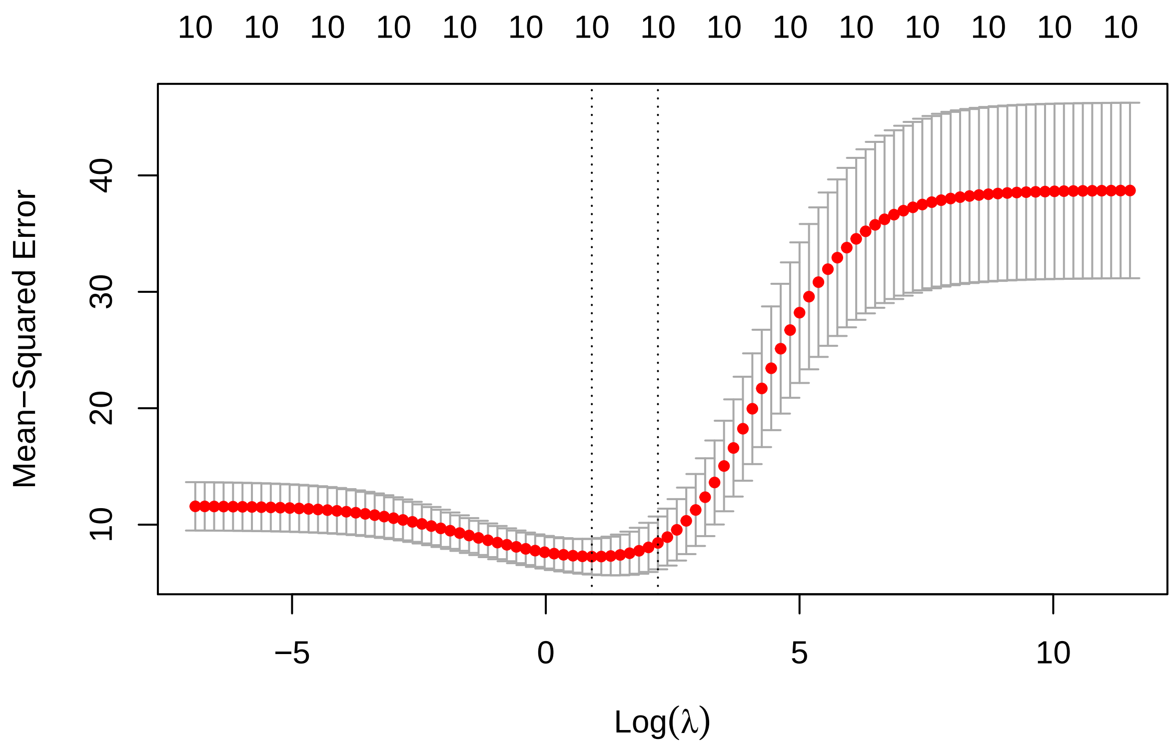
```
#1. RIDGE REGRESSION
# Center y, X will be standardized in the modelling function
#(ridge regression assumes the predictors are standardized and the response is centered)
y <- mtcars %>% select(mpg) %>% scale(center = TRUE, scale = FALSE) %>% as.matrix()
X <- mtcars %>% select(-mpg) %>% as.matrix()
```

Explanation: We scale the X matrix to ensure that the penalty term penalizes each coefficient equally. Instead of solving this heteroskedasticity problem by equalizing the variances of all predictors via scaling, we could just as well use them as weights in the estimation process! This is the idea behind the Differentially-weighted or Heteroskedastic Ridge Regression.

**Perform 10-fold cross-validation to select lambda:**

```
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)
#10-fold cross-validation to sel ect lambda
```

```r
# Setting alpha = 0 implements ridge regression
ridge_cv <- cv.glmnet(X, y, alpha = 0, lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)
# Plot cross-validation results
plot(ridge_cv)
```




```r
# Best cross-validated lambda
lambda_cv <- ridge_cv$lambda.min
# Fit final model, get its sum of squared residuals and multiple R-squared
model_cv <- glmnet(X, y, alpha = 0, lambda = lambda_cv, standardize = TRUE)
y_hat_cv <- predict(model_cv, X)
ssr_cv <- t(y - y_hat_cv) %*% (y - y_hat_cv)
rsq_ridge_cv <- cor(y, y_hat_cv)^2
rsq_ridge_cv
```

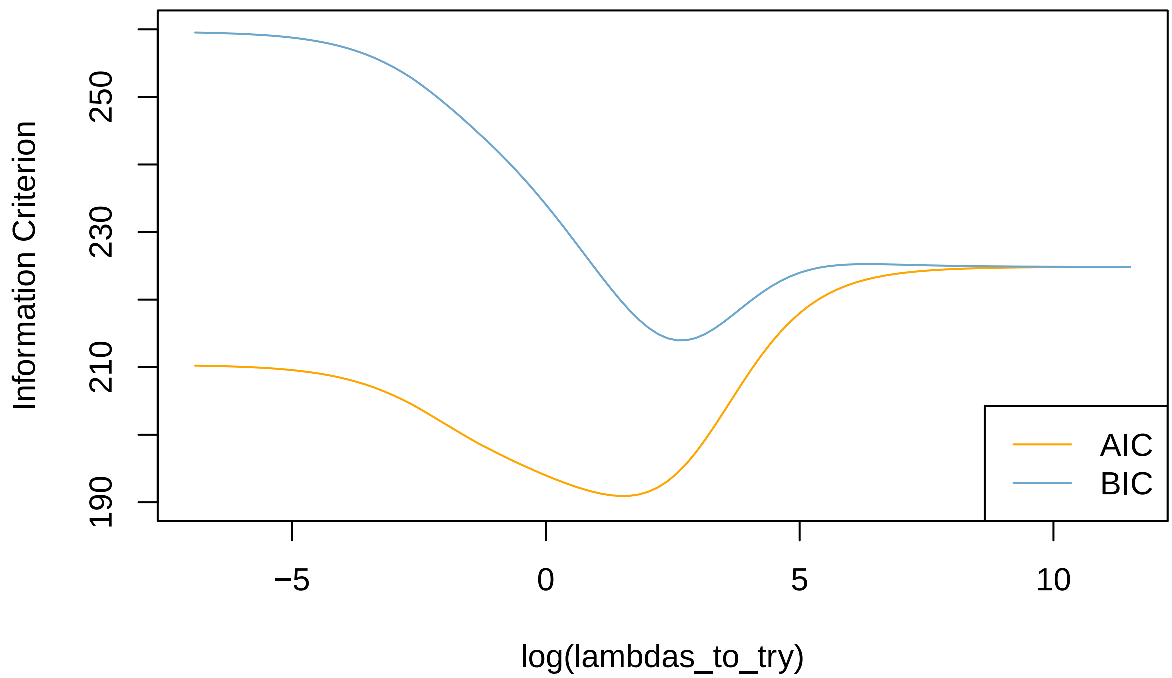
```
##            s0
## mpg 0.8536968
```

**Use information criteria to select lambda:**

```r
X_scaled <- scale(X)
aic <- c()
bic <- c()
for (lambda in seq(lambdas_to_try)) {
  # Run model
  model <- glmnet(X, y, alpha = 0, lambda = lambdas_to_try[lambda], standardize = TRUE)
  # Extract coefficients and residuals (remove first row for the intercept)
  betas <- as.vector((as.matrix(coef(model))[-1, ]))
  resid <- y - (X_scaled %*% betas)
```

```
  # Compute hat-matrix and degrees of freedom
  ld <- lambdas_to_try[lambda] * diag(ncol(X_scaled))
  H <- X_scaled %*% solve(t(X_scaled) %*% X_scaled + ld) %*% t(X_scaled)
  df <- tr(H)
  # Compute information criteria
  aic[lambda] <- nrow(X_scaled) * log(t(resid) %*% resid) + 2 * df
  bic[lambda] <- nrow(X_scaled) * log(t(resid) %*% resid) + 2 * df * log(nrow(X_scaled))
}

# Plot information criteria against tried values of lambdas
plot(log(lambdas_to_try), aic, col = "orange", type = "l",
     ylim = c(190, 260), ylab = "Information Criterion")
lines(log(lambdas_to_try), bic, col = "skyblue3")
legend("bottomright", lwd = 1, col = c("orange", "skyblue3"), legend = c("AIC", "BIC"))
```



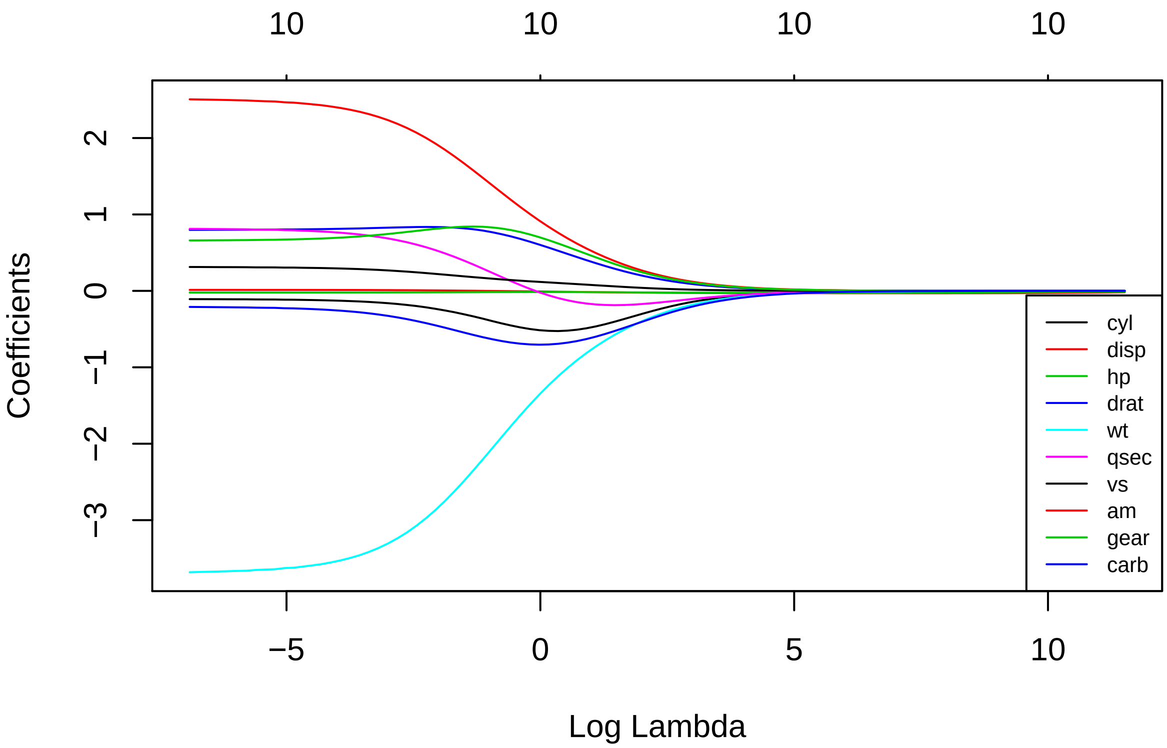

```
# Optimal lambdas according to both criteria
lambda_aic <- lambdas_to_try[which.min(aic)]
lambda_bic <- lambdas_to_try[which.min(bic)]

# Fit final models, get their sum of squared residuals and multiple R-squared
model_aic <- glmnet(X, y, alpha = 0, lambda = lambda_aic, standardize = TRUE)
y_hat_aic <- predict(model_aic, X)
ssr_aic <- t(y - y_hat_aic) %*% (y - y_hat_aic)
rsq_ridge_aic <- cor(y, y_hat_aic)^2

model_bic <- glmnet(X, y, alpha = 0, lambda = lambda_bic, standardize = TRUE)
y_hat_bic <- predict(model_bic, X)
ssr_bic <- t(y - y_hat_bic) %*% (y - y_hat_bic)
rsq_ridge_bic <- cor(y, y_hat_bic)^2
```

**See how increasing lambda shrinks the coefficients :**

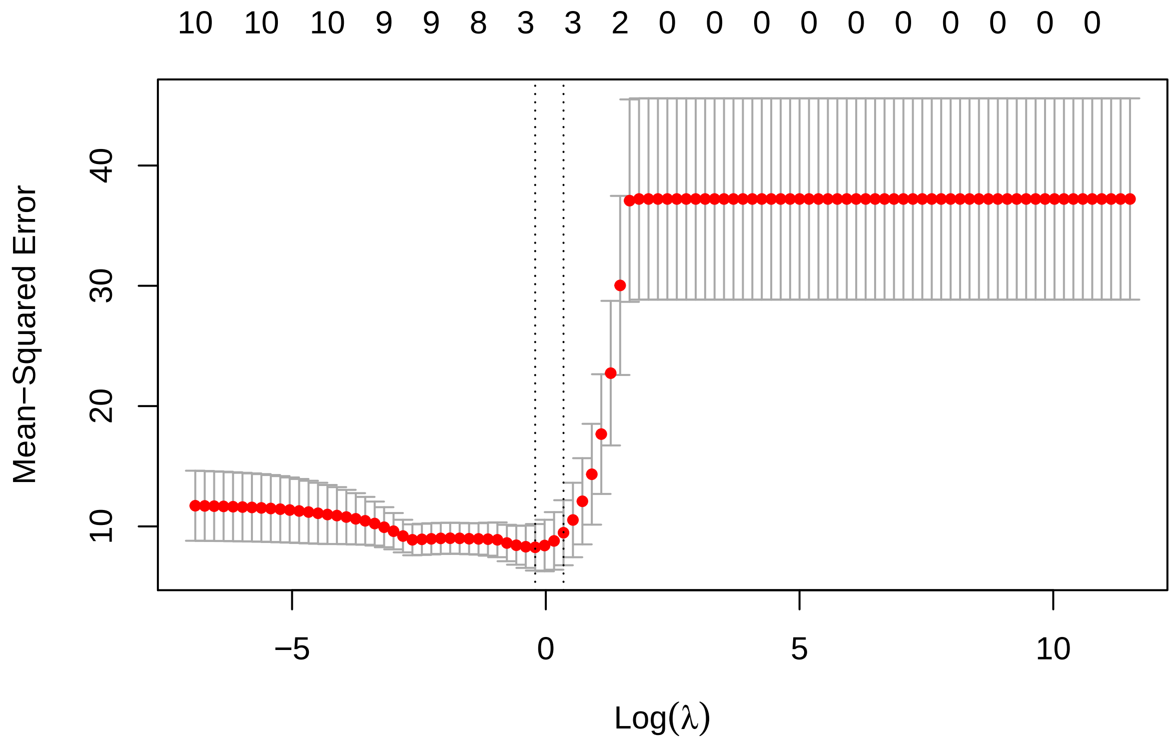
```r
# Each line shows coefficients for one variables, for different lambdas.
# The higher the lambda, the more the coefficients are shrinked towards zero.
res <- glmnet(X, y, alpha = 0, lambda = lambdas_to_try, standardize = FALSE)
plot(res, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(X), cex = .7)
```




## LASSO REGRESSION

Lasso = Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (the so-called L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty). As a result, for high values of lambda, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.

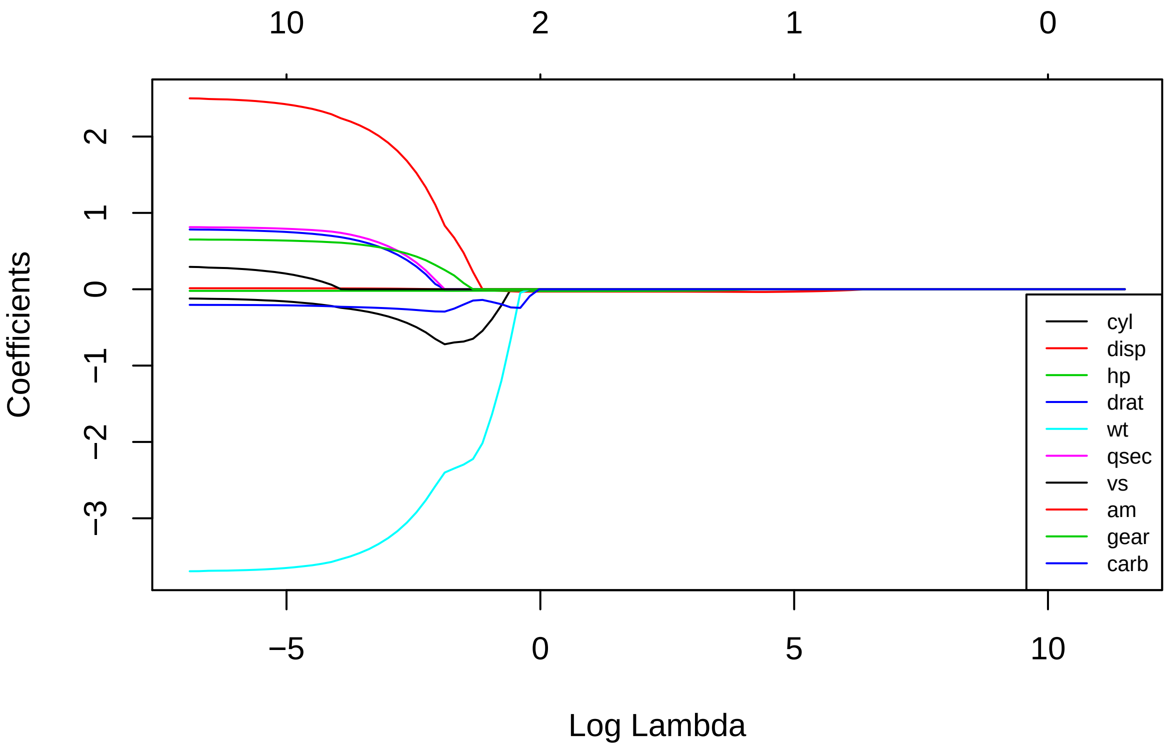
```r
# Perform 10-fold cross-validation to select lambda
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)
# Setting alpha = 1 implements lasso regression
lasso_cv <- cv.glmnet(X, y, alpha = 1, lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)
# Plot cross-validation results
plot(lasso_cv)
```




```
# Best cross-validated lambda
lambda_cv <- lasso_cv$lambda.min
# Fit final model, get its sum of squared residuals and multiple R-squared
model_cv <- glmnet(X, y, alpha = 1, lambda = lambda_cv, standardize = TRUE)
y_hat_cv <- predict(model_cv, X)
ssr_cv <- t(y - y_hat_cv) %*% (y - y_hat_cv)
rsq_lasso_cv <- cor(y, y_hat_cv)^2
```

**See how increasing lambda shrinks the coefficients:**

```
# Each line shows coefficients for one variables, for different lambdas.
# The higher the lambda, the more the coefficients are shrinked towards zero.
res <- glmnet(X, y, alpha = 1, lambda = lambdas_to_try, standardize = FALSE)
plot(res, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(X), cex = .7)
```




Let us compare the multiple R-squared of various models we have estimated:

## RIDGE VS. LASSO

```
rsq <- cbind("R-squared" = c(rsq_ridge_cv, rsq_ridge_aic, rsq_ridge_bic, rsq_lasso_cv))
rownames(rsq) <- c("ridge cross-validated", "ridge AIC", "ridge BIC", "lasso cross_validated")
print(rsq)
```

```
##                       R-squared
## ridge cross-validated 0.8536968
## ridge AIC             0.8496310
## ridge BIC             0.8412011
## lasso cross_validated 0.8424217
```

## ELASTIC NET

Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds:

Check multiple R-squared:

```
y_hat_enet <- predict(elastic_net_model, X)
y_hat_enet
```

```
##          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##         2.05551135          1.74280776          6.28216492          0.52249594
##  Hornet Sportabout             Valiant          Duster 360           Merc 240D
##        -3.27605721          0.15274583         -5.58554614          2.91074947
```

```
##            Merc 230            Merc 280           Merc 280C          Merc 450SE
##          3.20191054         -0.23145256         -0.15061543         -4.52522390
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##         -3.98074275         -4.00295948         -8.12176404         -8.41163995
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##         -8.22108855          7.26534418          8.46202739          8.05969802
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##          3.86926293         -3.30051536         -2.67542264         -5.50883573
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##         -4.13174939          7.59060420          6.05949380          6.85211961
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##         -1.83268991         -0.04290647         -6.03188981          5.00416337
```

```r
rsq_enet <- cor(y, y_hat_enet)^2
rsq_enet
```

```
##          [,1]
## mpg 0.8570287
```