Ridge and Lasso Regression - R Code

Example Dataset `Hitters` in ISLR2 package. Major League Baseball Data from the 1986 and 1987 seasons.

References James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) An Introduction to Statistical Learning with applications in R, https://www.statlearning.com, Springer-Verlag, New York

Install package `glmnet`

We will use the package `glmnet` to fit the ridge and lasso regression. Make sure it is installed and loaded.

Load the data (already looked at the data in previous labs)

```
Hitters    <- na.omit(Hitters) # Eliminate all rows with NAs
Hitters_tr <- Hitters %>% mutate(Years_tr = log(Years),
                                 CAtBat_tr = log(CAtBat),
                                 CHits_tr = log(CHits),
                                 CHmRun_tr = log(CHmRun+1),
                                 CRuns_tr  = log(CRuns),
                                 CRBI_tr   = log(CRBI),
                                 CWalks_tr = log(CWalks),
                                 Salary_tr = log(Salary))
Hitters_tr <- Hitters_tr %>% select(!c(Years,CAtBat,CHmRun,CRuns,CRBI,CWalks,Salary, CHits,
                                       League, Division,NewLeague))
dim(Hitters_tr)
```

```
## [1] 263  17
```

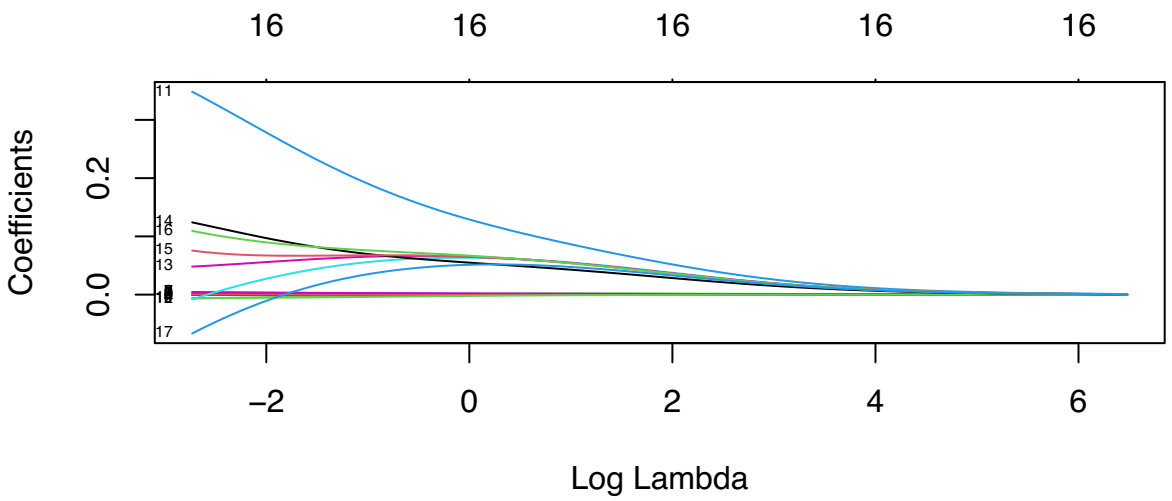
The `glmnet` package does not use the "model formula language" as we are used to with the `lm` function. We need to manually setup an X matrix and a vector for the response variable Y

```
x = model.matrix(Salary_tr~., data = Hitters_tr)
y = Hitters_tr$Salary_tr
```

Ridge regression

First we will fit a ridge-regression model. This is achieved with the `glmnet` function with the argument `alpha = 0`. (`alpha = 1` then a lasso model is fit.)

```
ridge.fit <- glmnet(x, y, alpha = 0)
plot(ridge.fit, xvar = "lambda", label = TRUE)
```

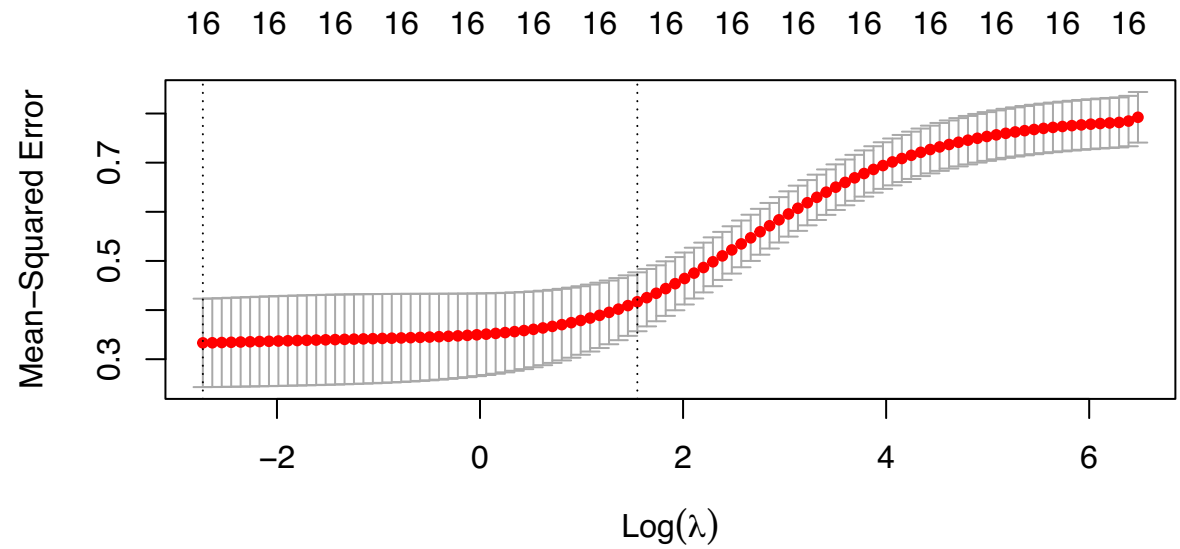


Do the CV step to pick the λ parameter. The library `glmnet` has a function `cv.glmnet` that does that for us.

```
cv.ridge  <- cv.glmnet(x,y, alpha = 0)
cv.ridge
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure      SE Nonzero
## min  0.065   100  0.3332 0.08999      16
## 1se  4.710    54  0.4169 0.05999      16
```

```
plot(cv.ridge)
```



Fit the model with the selected lambda. Using the function `coef`, find the coefficients for the lasso model fit.

```
ridge.fit.minlambda <- glmnet(x, y, alpha = 0, lambda = 0.065)
coef(ridge.fit.minlambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
```

```
## (Intercept)  3.424866e+00
## (Intercept)  .
## AtBat       -5.413692e-04
## Hits         3.464200e-03
## HmRun       -6.814971e-03
## Runs         3.883505e-03
## RBI          3.341926e-05
## Walks        4.570778e-03
## PutOuts      3.391169e-04
## Assists      3.192884e-04
## Errors      -5.715472e-03
## Years_tr     3.490788e-01
## CAtBat_tr   -5.588398e-03
## CHits_tr     4.682632e-02
## CHmRun_tr    1.228972e-01
## CRuns_tr     7.317496e-02
## CRBI_tr      1.108628e-01
## CWalks_tr   -6.643194e-02
```

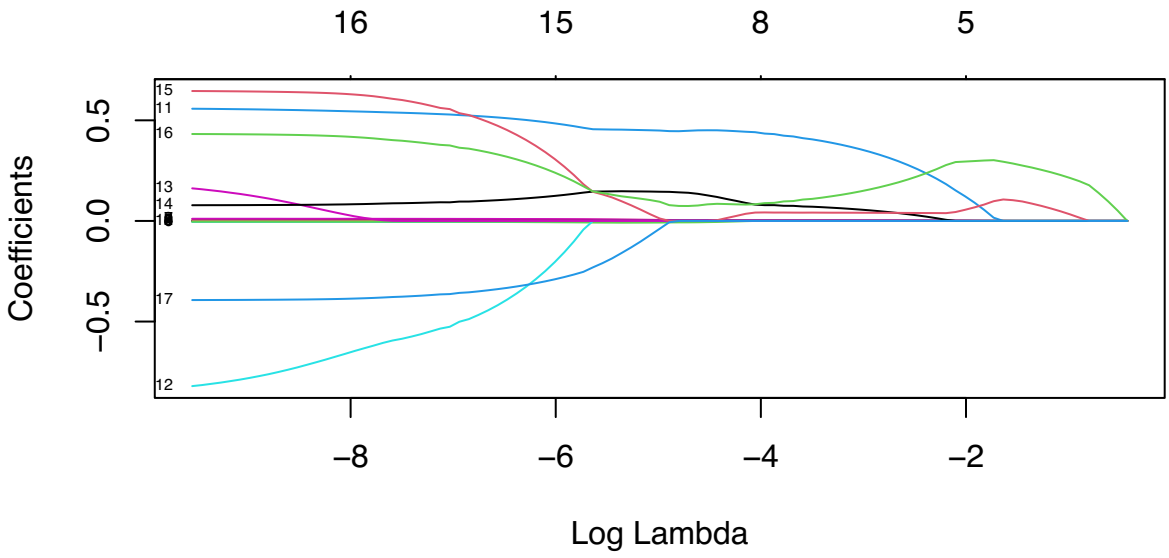
Lasso regression

Now, fit a lasso-regression model.

```
# Achieved with the `glmnet` function with the argument `alpha = 1`
lasso.fit <- glmnet(x, y, alpha = 1)
```

Now plot the coefficients against the log of lambda. What is the maximum number of variables considered in the model as lambda changes? The minimum? How does that compare with the ridge regression?

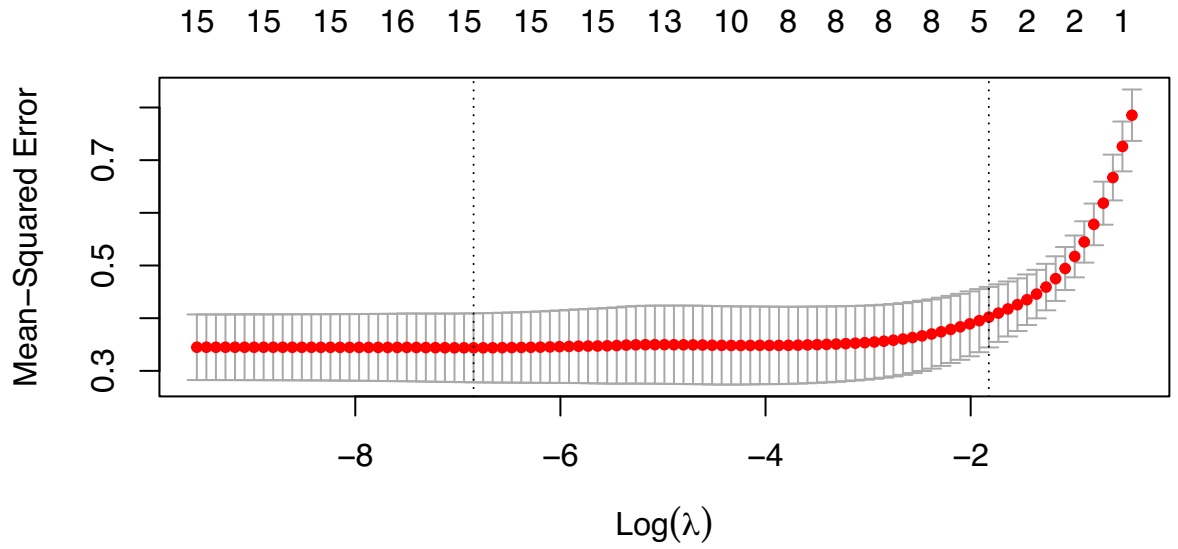
```
# Plot the results
plot(lasso.fit, xvar = "lambda", label = TRUE)
```



The maximum number of variables is 16 and the minimum is zero. Under the ridge regression, there were always 16 variables included in the model, independently of the value of lambda.

Perform cross validation to pick the λ parameter and plot the results (estimated test mse vs log of lambda). Which value of lambda would you pick?

```
cv.lasso  <- cv.glmnet(x, y, alpha = 1)
plot(cv.lasso)
```




```
cv.lasso
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00106    70  0.3437 0.06542      15
## 1se 0.16156    16  0.4020 0.05747       5
```

Using the function `coef`, find the coefficients for the lasso model fit.

```
coef(cv.lasso)  # by default, the coefficient from the second vertical line
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 3.6654673954
## (Intercept) .
## AtBat       .
## Hits        0.0010710846
## HmRun       .
## Runs        .
## RBI         .
## Walks       0.0006611395
## PutOuts     .
## Assists     .
## Errors      .
## Years_tr    0.0550544848
## CAtBat_tr   .
## CHits_tr    .
## CHmRun_tr   .
## CRuns_tr    0.0813470627
## CRBI_tr     0.3003137478
## CWalks_tr   .
```

```r
# coef(lasso.fit) # coefficients for all the lasso fit (all the values of lambda)
lasso.fit.best <- glmnet(x, y, alpha = 1, lambda = 0.00106)
coef(lasso.fit.best) #coefficients for the best choice of lambda
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  4.9319150610
## (Intercept)  .
## AtBat       -0.0017433794
## Hits         0.0086818169
## HmRun       -0.0044263172
## Runs         0.0003716128
## RBI         -0.0033077471
## Walks        0.0098572506
## PutOuts      0.0003583778
## Assists      0.0003895073
## Errors      -0.0032370495
## Years_tr     0.5302635453
## CAtBat_tr   -0.5458471452
## CHits_tr     .
## CHmRun_tr    0.0906836675
## CRuns_tr     0.5603362779
## CRBI_tr      0.3735275846
## CWalks_tr   -0.3462449026
```

Lasso regression - lambda selected through the validation set approach

Suppose that we now want to use the validation set method to select the lambda for the lasso regression.

Set a seed and split the data set into a train and test data.

```r
# Set seed for reproducibility
set.seed(7304)

#  Divide into training and test sets
train_inds <- caret::createDataPartition(
  y = Hitters_tr$Salary_tr, # response variable as a vector
  p = 0.75       # approx. proportion of data used for training
)

# Create the training and test data sets
Hitters_train <- Hitters_tr %>% slice(train_inds[[1]])
Hitters_test  <- Hitters_tr %>% slice(-train_inds[[1]])
```

Create the x matrix and y vector associated with the train and test data.

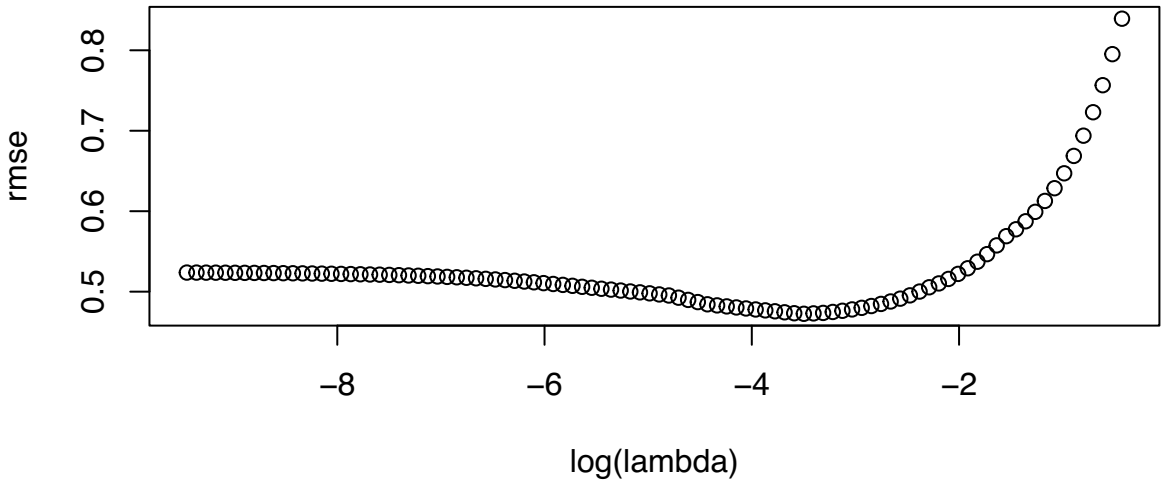
```r
x_train       <- model.matrix(Salary_tr~., data = Hitters_train)
x_test        <- model.matrix(Salary_tr~., data = Hitters_test)
y_train       <- Hitters_train$Salary_tr
y_test        <- Hitters_test$Salary_tr
```

Fit the lasso model on the train data and estimate the rmse

```
lasso.tr      <- glmnet(x_train, y_train)
pred_test     <- predict(lasso.tr, x_test)
rmse          <- sqrt(apply((y_test - pred_test)^2,2, mean))
```

```
plot(log(lasso.tr$lambda), rmse, type = "b", xlab = "log(lambda)")
```

Plot the rmse agains the log lambda and find the best choice of lambda




```
lambda.best <- lasso.tr$lambda[order(rmse)][1]
lambda.best
```

```
## [1] 0.03027065
```

```
lambda.best    <- lasso.tr$lambda[order(rmse)][1]
lasso.fit.best <- glmnet(x, y, alpha = 1, lambda = lambda.best)
coef(lasso.fit.best)
```

Use your choice of lambda to fit the model with all the data

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) 3.5397920389
## (Intercept) .
## AtBat       .
## Hits        0.0034959706
## HmRun       .
## Runs        0.0006519025
## RBI         .
## Walks       0.0034887636
## PutOuts     0.0002393336
## Assists     .
## Errors      .
## Years_tr    0.4035794271
## CAtBat_tr   .
## CHits_tr    .
## CHmRun_tr   0.0745578300
## CRuns_tr    0.0535250136
```

```
## CRBI_tr     0.0955108528
## CWalks_tr   .
```