

# STAT 2270 Homework 1 Fall 2020

Manuel Alejandro Garcia Acosta

8/27/2020

## Exercise 4

Here we'll make use of the Hitters dataset found in the ISLR package in R. Note that ISLR here stands for the textbook "Introduction to Statistical Learning with applications in R" cited in the syllabus for this class. You may find the lab contained in Chapter 6 of ISLR very useful for the first few parts of this exercise.

### Part (a)

Load in the Hitters dataset found in the ISLR package in R. Following the approach in ISLR, we'll use Salary as our response and treat the rest of the variables as predictors. Remove all rows of the data that contain missing values. We'll treat the remaining data as our working dataset for the remainder of this problem.

First I loaded the dataset and omitted the rows with missing values.

**Note:** I'm turning in the analysis performed after normalizing/scaling the numeric variables. This scaling included the 'Salary' variable. As we discussed, results were similar to those obtained by performing the analysis with the original data (without scaling the numeric variables).

```
# Loading the dataset
if(exists('Hitters')){
  rm(Hitters)
}else{
  fix(Hitters)
}

## Warning in rm(Hitters): object 'Hitters' not found

# Omitting the rows with missing values
Hitters <- na.omit(Hitters)
# Scaling all numeric variables including 'Salary'
ind <- sapply(Hitters, is.numeric)
Hitters[ind] <- lapply(Hitters[ind], scale)
rm(ind)
```

### Part (b)

Construct a linear model and record the resulting MSE of the predictions on the (training) data. Don't worry about fixing up the linear model (e.g. removing terms that are not significant) – simply regress Salary on all remaining predictors and record the error.

#### NOTES:

- For computing the *MSE* I'm taking the mean of the residual sum of squares. I did this throughout this HW.
- As we discussed, the *MSE* I'll be reporting was the *training error*.

```
# Fitting the linear model
fit <- lm(Salary ~ ., data = Hitters)
# I computed the MSE by dividing the residuals squared sum by n
# ISLR book just takes the mean of the residual sum of squares for the MSE
mse <- sum((Hitters$Salary - fit$fitted.values)^2)/(length(Hitters$Salary))
```

The *MSE* for this model is 0.4521583.

## Part (c)

Now construct the best possible linear model you can find. You can do this however you like: removing terms that are not significant, using combinations of forward/backward selection, including interaction terms, including higher order polynomial terms etc., but don't use any of the regularization methods we discussed in class. Once you've found what you think is the best model, record the *MSE*.

## Multicollinearity

I checked the data for collinearity and there seems to be some sort of multicollinearity in the dataset. Some of the covariates like 'CAtBat', 'CHits', 'CRuns' and 'CRBI' have *VIF* over 100. This was really high.

However, I decided not to discard them right away and started doing model selection with the saturated model (all features).

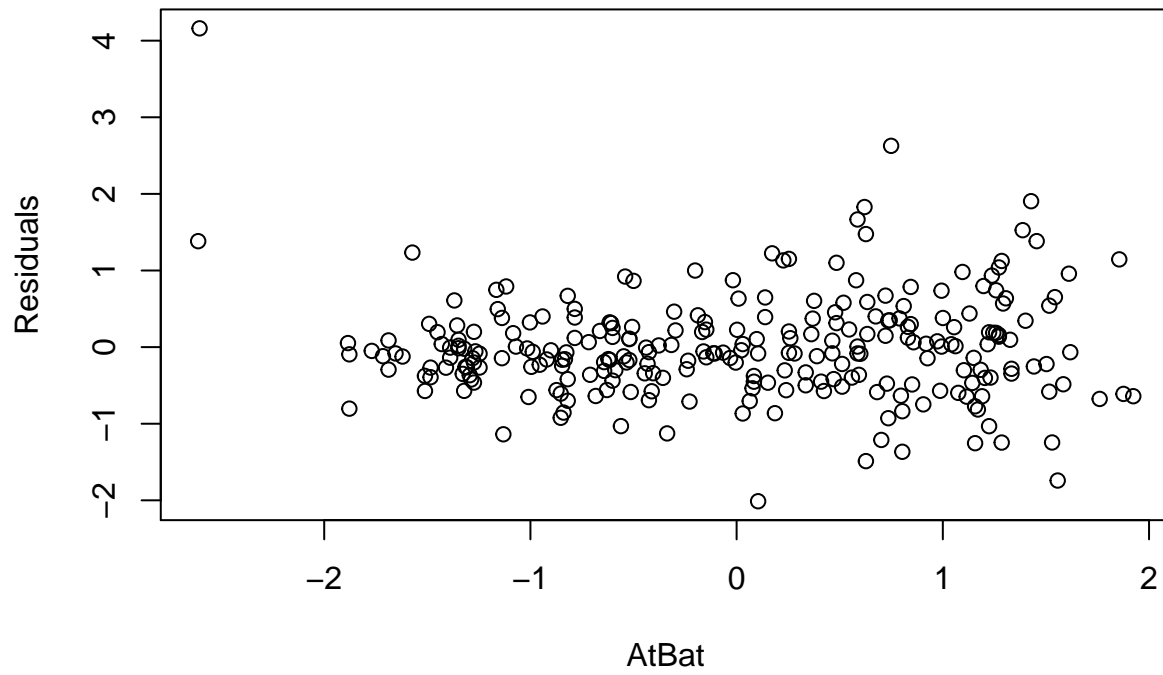
```
# Computing the VIF for all the covariates
vif(lm(Salary ~ ., data = Hitters))
```

```
##      AtBat      Hits      HmRun      Runs      RBI      Walks
## 22.944366 30.281255  7.758668 15.246418 11.921715  4.148712
##      Years      CAtBat      CHits      CHmRun      CRuns      CRBI
##  9.313280 251.561160 502.954289 46.488462 162.520810 131.965858
##      CWalks      League      Division      PutOuts      Assists      Errors
## 19.744105  4.134115  1.075398  1.236317  2.709341  2.214543
##      NewLeague
##  4.099063
```

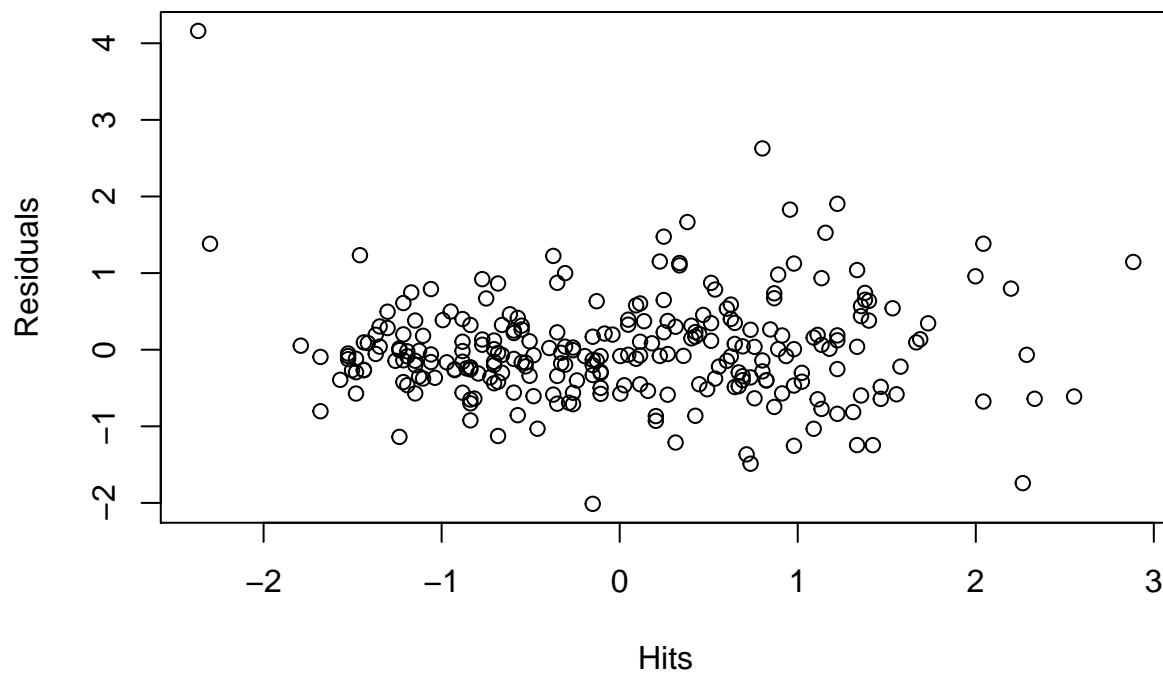
## Adding covariates

After plotting the residuals of the model from part (b) vs the covariates, I tried adding to the model the squares of 'AtBat', 'Hits', 'Runs' and 'Walks'. While testing for the effects of the squares separately (a model with all original covariates plus one of these squares) all the p-values were below  $\alpha = 0.05$ .

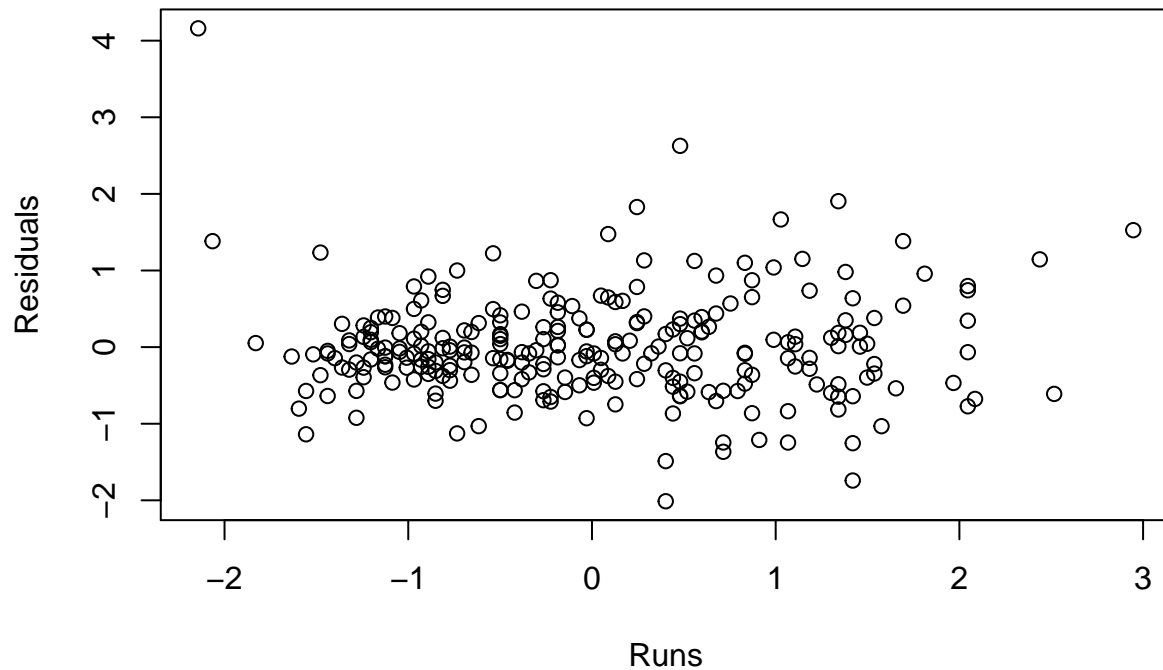
**Residual plot against AtBat**



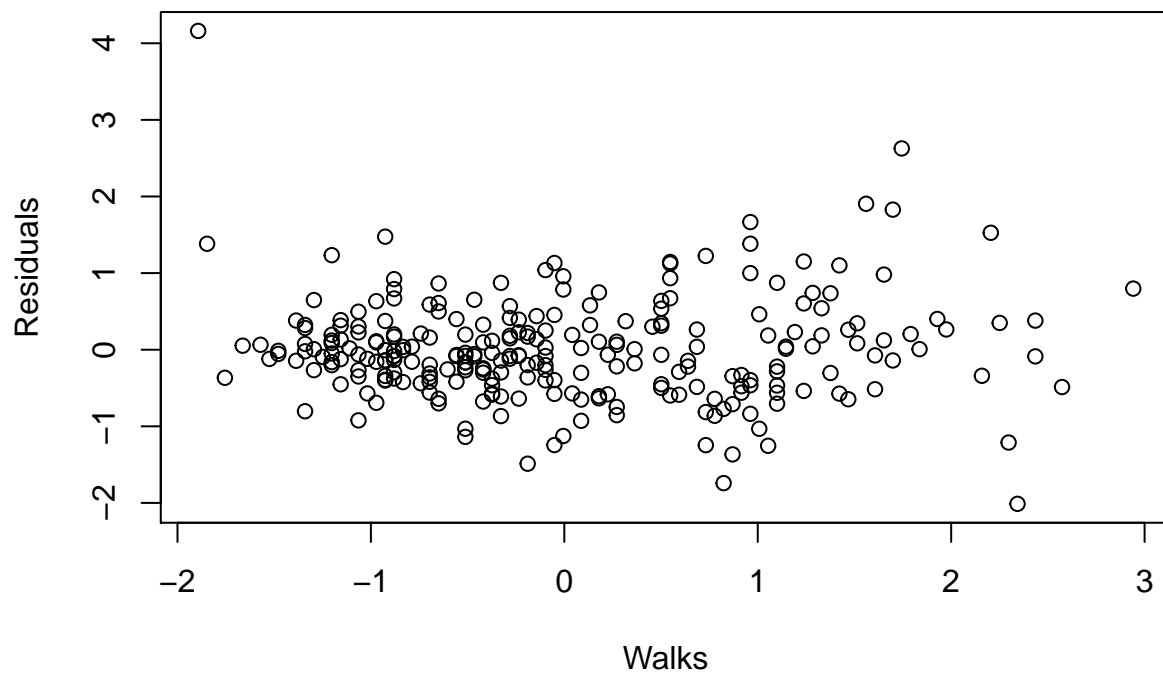
**Residual plot against Hits**



**Residual plot against Runs**



**Residual plot against Walks**



After this I looked for interactions between variables, I did this in a single model.

```
# Fitting a model with all covariates and all possible interaction effects
fit2 <- lm(Salary ~ (.)^2, data = Hitters)
# summary(fit2)
```

While testing for the interaction terms, the interactions which seemed significant at  $\alpha = 0.05$  were  $AtBat * CHmRun$ ,  $AtBat * CRuns$ ,  $Hits * CHmRun$ ,  $Walks * CHmRun$  and  $CHmRun * CRBI$  and  $PutOuts * Assists$ .

## Model selection

I proceeded to perform stepwise selection with ( $\alpha_1 = 0.05, \alpha_2 = 0.1$ ). I considered all original covariates plus the squared and interaction terms that appeared to be significant after my initial exploration.

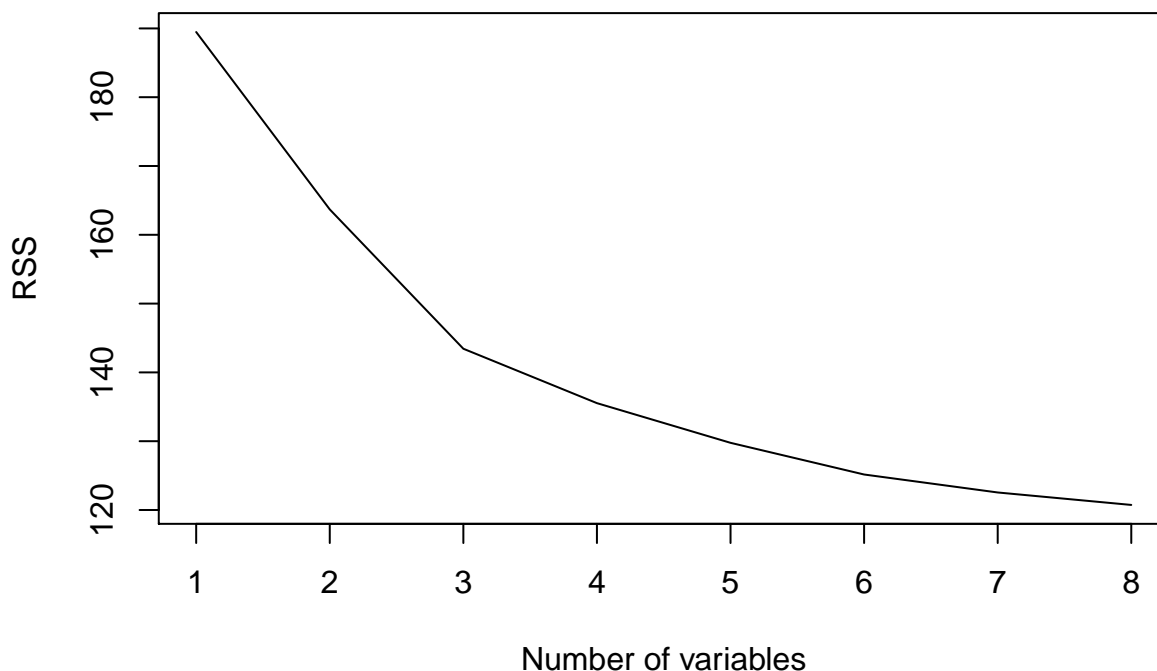
**NOTE:** I commented out the line for stepwise selection since the output was way too long to include it on the pdf file version of the HW. You'll find the code on the Rmd file 'HW1(scaled).Rmd'

```
# Fitting the expanded model
fit3 <- lm(Salary ~ . + I(AtBat^2) + I(Hits^2) + I(Runs^2) + I(Walks^2) +
           AtBat*CHmRun + AtBat*CRuns + Hits*CHmRun + Walks*CHmRun +
           CHmRun*CRBI + PutOuts*Assists, data = Hitters)
# Running stepwise selection for the proposed model
alpha.1 <- 0.05
alpha.2 <- 0.1
# step.wise <- ols_step_both_p(fit3, pent = alpha.1, prem = alpha.2)
```

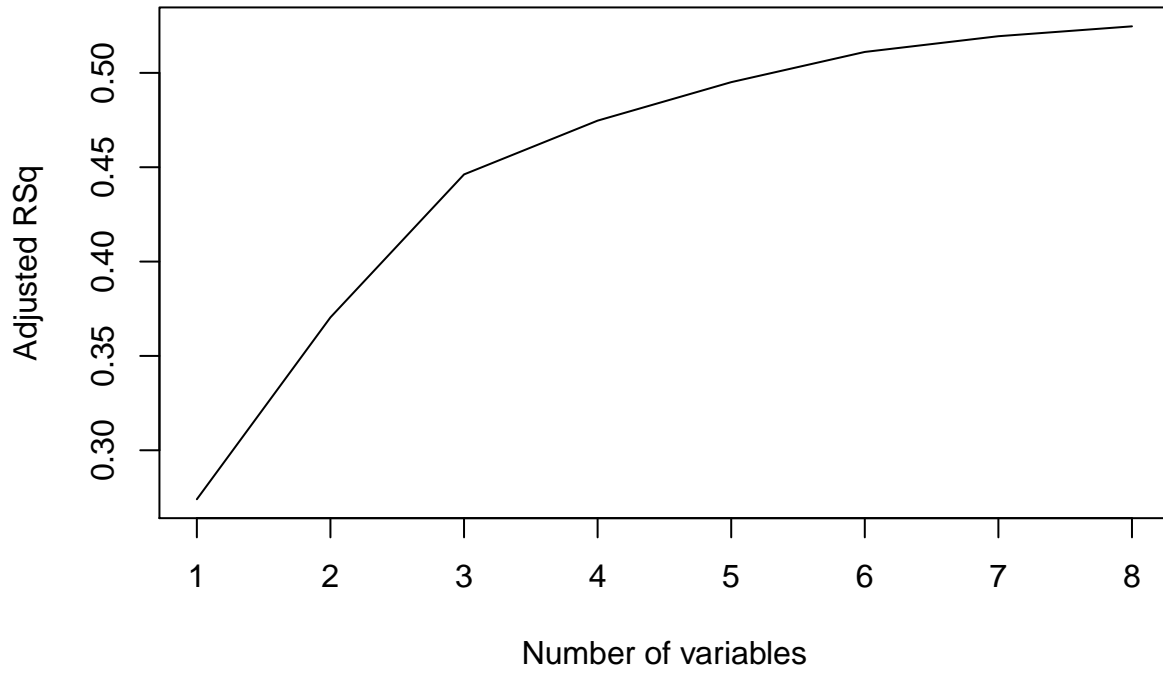
After running stepwise selection I got a model that includes 'Walks', 'CAtBat', 'CHmRun', 'Division', 'AtBat^2', 'Hits^2', 'Walks^2', 'PutOuts\*Assists'. Next thing I wanted to do was look for the best submodel. I used RSS, Adjusted  $R^2$ ,  $C_p$  and  $BIC$  to check which submodel was the 'best'. Next I plot each of these as a function of the number of features each 'best' subset has.

```
regfit <- regsubsets(Salary ~ Walks + CAtBat + CHmRun + Division + I(AtBat^2) +
                    I(Hits^2) + I(Walks^2) + PutOuts*Assists, data = Hitters,
                    nvmax = 8)
reg.summary <- summary(regfit)
```

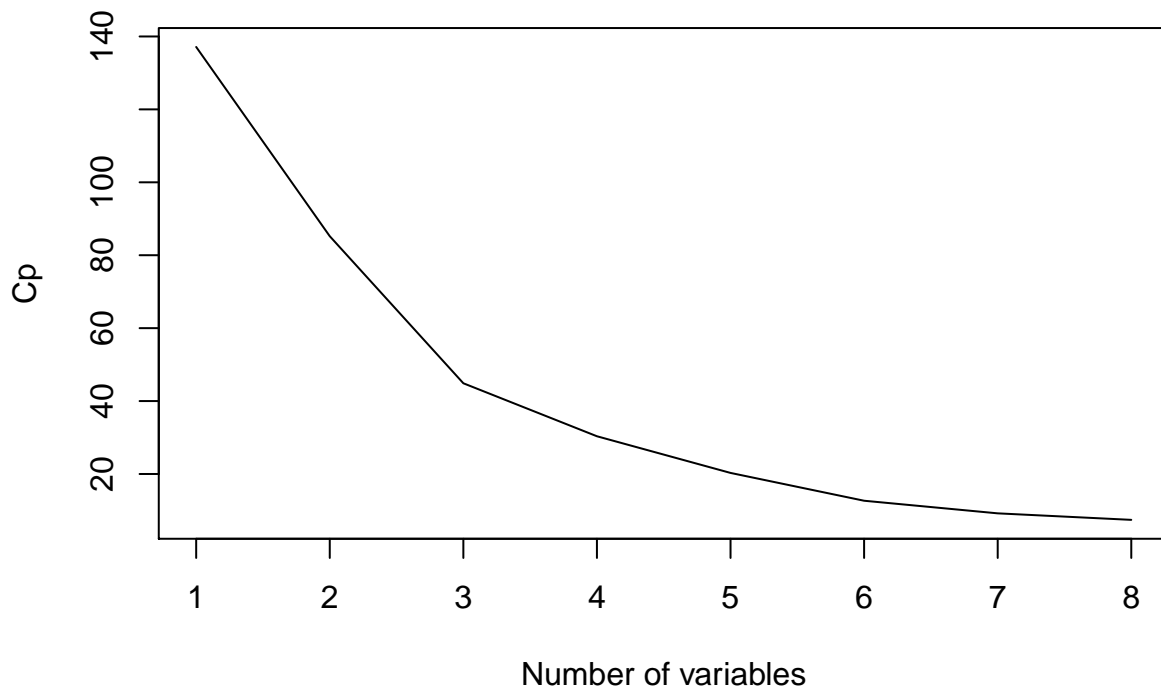
## Residual Sum of Squares VS Num. of Variables



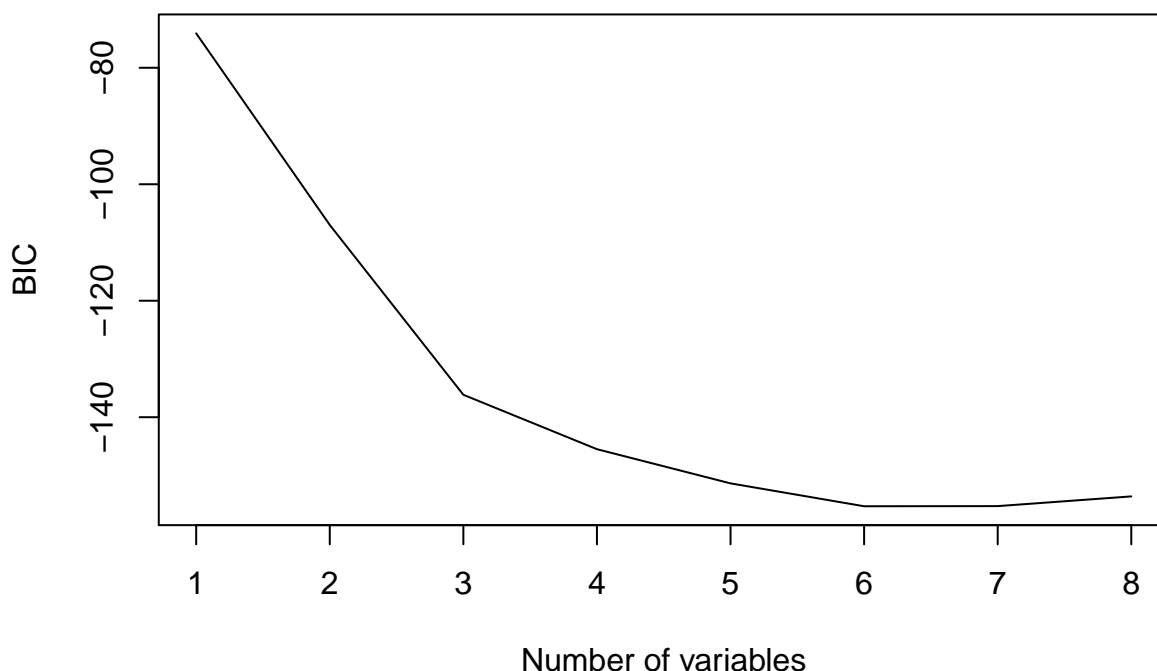
**Adjusted RSq VS Num. of Variables**



**Cp VS Num. of Variables**



## BIC VS Num. of Variables



Results suggest that the best model for all criterias except *BIC* is the one with all 8 covariates ('Walks', 'CAtBat', 'CHmRun', 'Division', 'AtBat^2', 'Hits^2', 'Walks^2', 'PutOuts\*Assists'). This is exactly the same model I got after running stepwise selection. Using *BIC* the best model used 6 covariates, dropping 'AtBat^2' and 'Walks^2' from the full model. I'll pick the one with all 8 features to compute the *MSE*.

```
# Computing the MSE of the model
fit4 <- lm(Salary ~ Walks+CAtBat+CHmRun+Division+I(AtBat^2)+
           I(Hits^2)+I(Walks^2)+PutOuts*Assists, data = Hitters)
mse4 <- mean(fit4$residuals^2)
```

Finally, the *MSE* for the resulting model was 0.4583174, which this time around wasn't an improvement over the model from part (b). So sad : (.

## Part (d)

Find the best model using ridge regression. Consider a range of tuning parameters and select the best model by cross validation. Record the value of the optimal tuning parameter as well as the values of the resulting coefficient estimates from that model.

**NOTE:** For solving this homework I selected the tuning parameters<sup>1</sup> while using 10-fold cross-validation.

For  $\lambda$  I created a vector with one thousand values ranging from  $1 * 10^{10}$  to 0.01.

```
# Creating the design matrix and the response vector
# NOTE: I needed to erase the intercept because glmnet creates the intercept
# automatically
x <- model.matrix(Salary~., data = Hitters, y=Salary)[,-1]
y <- Hitters$Salary
# Creating a set of possible values for lambda
```

<sup>1</sup> $\lambda$  for ridge regression, lasso and elastic net,  $\alpha$  for elastic net

```

lambdas <- 10^seq(10,-2,length.out=1000)
# Fitting ridge regression for the vector of lambdas
fit.ridge <- glmnet(x,y,alpha = 0, lambda = lambdas)

```

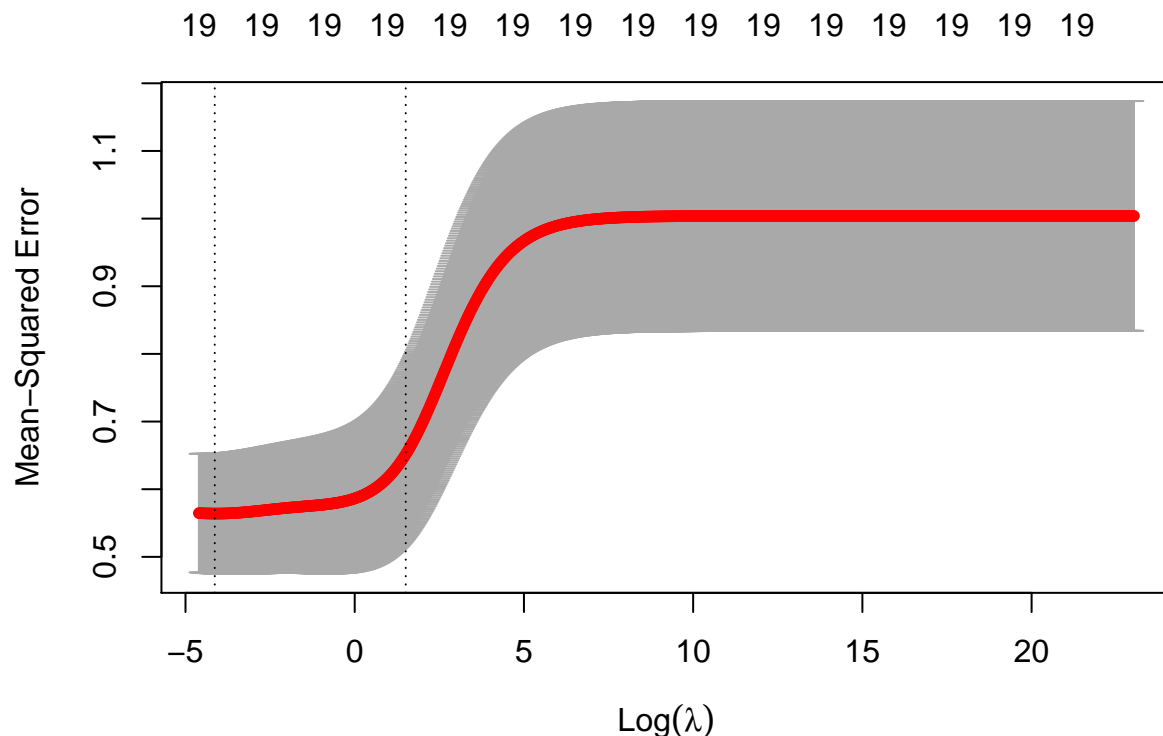
**NOTE:** For reproducibility purposes I set a seed whenever I used  $k$ -fold cross-validation. This holds for all remaining parts of this exercise.

```

# Set a seed so results can be reproduced
set.seed(1989)
# Here I run 10-fold cross-validation
cv.fit.ridge <- cv.glmnet(x,y, alpha = 0, nfolds = 10, lambda = lambdas,
                          type.measure = 'mse')

```

I include a plot of how the  $MSE$  varies alongside  $\log(\lambda)$ .



Next, I obtained the value for  $\lambda$  associated to the smallest cross-validation error as well as the coefficient estimates for that model. Notice that I got my results after running  $k$ -fold cross-validation only once.

```

# Obtain the lambda which gives smallest error
ridge.lambda <- cv.fit.ridge$lambda.min

```

The ‘best’ value for lambda was  $\lambda = 0.0294082$ . For this value, the ridge regression estimates are

```

# Run ridge regression with the 'best' lambda
best.ridge.fit <- glmnet(x,y, alpha = 0, lambda = ridge.lambda)
# Obtain the coefficients
coef.d <- as.matrix(coef(best.ridge.fit))
coef.d

```

```

##                s0
## (Intercept)  0.1044910830
## AtBat       -0.4489162926
## Hits        0.4838906434

```



```
## HmRun      -0.0003714846
## Runs       0.0093836397
## RBI        0.0145780242
## Walks      0.2279359737
## Years      -0.1143765187
## CAtBat     -0.1729717323
## CHits      0.2495951091
## CHmRun     0.1232289847
## CRuns      0.4089154886
## CRBI       0.2573957294
## CWalks     -0.3032058674
## LeagueN    0.1330520778
## DivisionW  -0.2744835044
## PutOuts    0.1715134339
## Assists    0.0830109530
## Errors     -0.0560635510
## NewLeagueN -0.0590067448
```

Finally, I computed the MSE for ridge regression using the best lambda. Here  $MSE = 0.4711957$

```
# Computing the MSE of the model
ridge.pred <- as.numeric(predict(best.ridge.fit, s=ridge.lambda, newx=x))
mse.ridge <- mean((ridge.pred - y)^2)
```

## Part (e)

Repeat part (d) with lasso.

For solving this part I selected the tuning parameter while using 10-fold cross-validation.

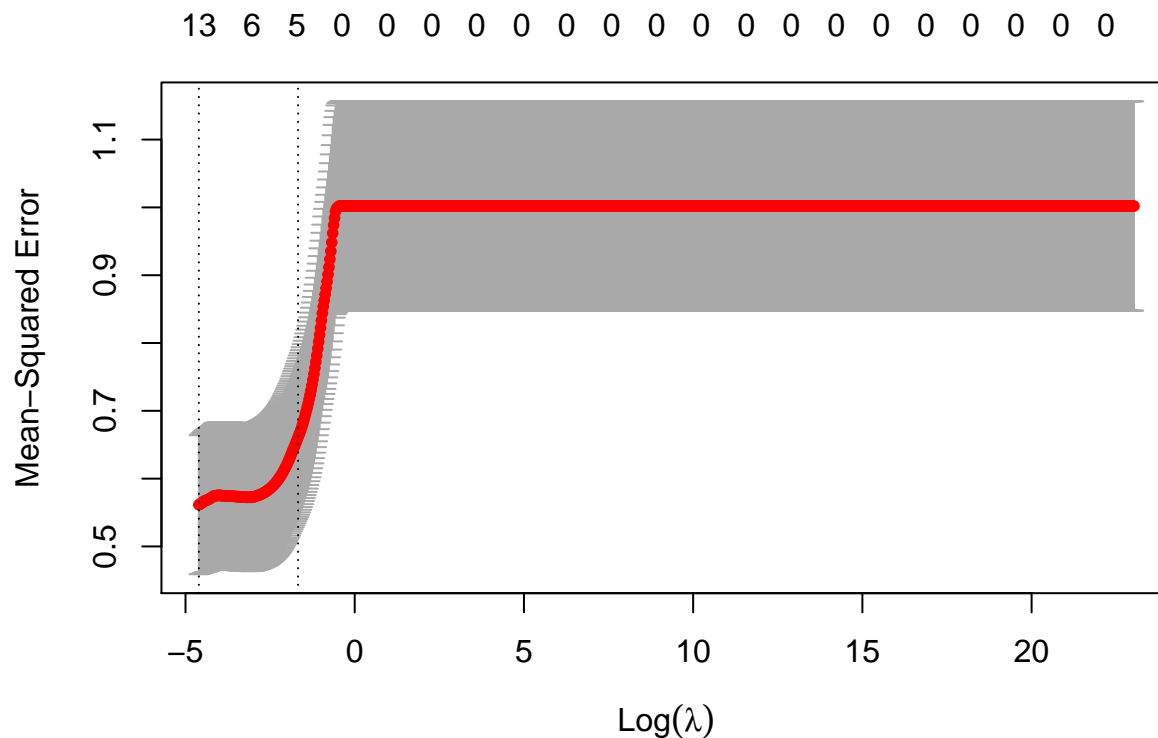
**NOTE:** I'm using the design matrix 'x', response vector 'y' and the vector of lambdas 'lambdas' created in part (d) for parts (e)-(f) also.

```
# Fitting lasso for vector of lambdas
fit.lasso <- glmnet(x,y,alpha = 1, lambda = lambdas)
```

I set the seed and run cross-validation next

```
# Set a seed so results can be reproduced
set.seed(1991)
# Here I run 10-fold cross-validation
cv.fit.lasso <- cv.glmnet(x,y, alpha = 1, nfolds = 10, lambda = lambdas,
                        type.measure = 'mse')
```

I include a plot of how the  $MSE$  varies alongside  $\log(\lambda)$ .



Next, I obtained the value for  $\lambda$  associated to the smallest cross-validation error as well as the coefficient estimates for that model.

```
# Obtain the lambda which gives smallest error
lasso.lambda <- cv.fit.lasso$lambda.min
```

The 'best' value for lambda was  $\lambda = 0.01$ . For this value, the lasso estimates are

```
# Run lasso with the 'best' lambda
best.lasso.fit <- glmnet(x,y, alpha = 1, lambda = lasso.lambda)
# Obtain the coefficients
coef.e <- as.matrix(coef(best.lasso.fit))
coef.e
```

```
##                               s0
## (Intercept)  0.10324285
## AtBat       -0.34463433
## Hits        0.45813891
## HmRun       0.00000000
## Runs        0.00000000
## RBI         0.00000000
## Walks       0.18480554
## Years      -0.06879923
## CAtBat      0.00000000
## CHits       0.00000000
## CHmRun      0.06466104
## CRuns       0.37321427
## CRBI        0.27715798
## CWalks     -0.19318224
## LeagueN     0.06512583
## DivisionW   -0.26289904
## PutOuts     0.16107387
```

```
## Assists      0.02538619
## Errors      -0.01769622
## NewLeagueN   0.00000000
```

Finally, I computed the MSE for lasso using the best lambda. Here  $MSE = 0.4750985$ .

```
# Computing the MSE of the model
lasso.pred <- as.numeric(predict(best.lasso.fit, s=lasso.lambda, newx=x))
mse.lasso <- mean((lasso.pred - y)^2)
```

## Part (f)

Repeat part (d) with elastic net.

**NOTE:** In order to solve this part of the problem I programmed a bunch of functions that allowed me to run elastic net through different values for  $\alpha$ . I'm omitting the code on the pdf but they can be found in the Rmd file 'HW1(scaled).Rmd' of my HW.

Now, for elastic net I did basically three steps

1. Define a set of alphas and run  $k$ -fold cross-validation for each lambda
2. Choose best lambda for each alpha
3. Compare MSE for each of the lambdas obtained. Since each  $\lambda$  corresponds to a single  $\alpha$ , this will give us the best pair of  $(\alpha, \lambda)$ .

The set of  $\alpha$ 's I used were  $\{0, 0.01, 0.02, \dots, 1\}$ . Once again, I set a seed for reproducibility.

```
# Set a seed so results can be reproduced
set.seed(1995)
# Create vector of alphas
alphas <- seq(from = 0, to = 1, by = 0.01)
# Run k-fold cross-validation for the set of alphas defined above
out <- cv.alpha.vect(alphas = alphas, feat = x, res = y, l = lambdas)
# Get a vector of the MSE associated to the 'best' lambda for each alpha
elastic.mse <- cv.mse.vect(out, featu = x, resp = y)
# Select the lowest MSE over elastic.mse
index <- which.min(elastic.mse)
# Gets the minimum MSE over all pairs (alpha, lambda)
mse.elastic <- elastic.mse[index]
```

The best pair  $(\alpha, \lambda)$  was  $(0, 0.01)$  with  $MSE_{\alpha, \lambda} = 0.459424$ . This intuitively tells us that lasso performs a little bit better than ridge regression while using the original covariates since the 'best' alpha was close to 1. The elastic net estimates are

```
# Get best pair (alpha, lambda)
elastic.alpha <- alphas[index]
elastic.lambda <- get.lambda(out[[index]])
# Get coefficient estimates
best.elastic.fit <- glmnet(x, y, alpha = elastic.alpha, lambda = elastic.lambda)
coef.f <- as.matrix(coef(best.elastic.fit))
coef.f
```

```
##              s0
## (Intercept)  0.103585498
## AtBat       -0.501385053
## Hits        0.539513150
## HmRun       0.010386175
```

```
## Runs      -0.010381587
## RBI       0.005137531
## Walks     0.243671407
## Years     -0.110240346
## CAtBat    -0.249177748
## CHits     0.260499762
## CHmRun    0.117605909
## CRuns     0.484681834
## CRBI      0.279534371
## CWalks    -0.336891060
## LeagueN   0.134818178
## DivisionW -0.272915060
## PutOuts   0.172680010
## Assists   0.089869591
## Errors    -0.055339259
## NewLeagueN -0.060572313
```

## Part (g)

Compare the models in parts (b)-(f). Does any one model (or a few models) stand out as being substantially better than the others?

By comparing  $MSE$ 's of the different models. I can state that between the models which only use the original features,  $OLS$  (0.4521583) and elastic net (0.459424) performed better than ridge regression (0.4711957) and lasso (0.4750985). The model proposed in part (c) outperformed all regularization methods (0.4583174) but not  $OLS$  using MSE as **selection criteria**.

## Part (h)

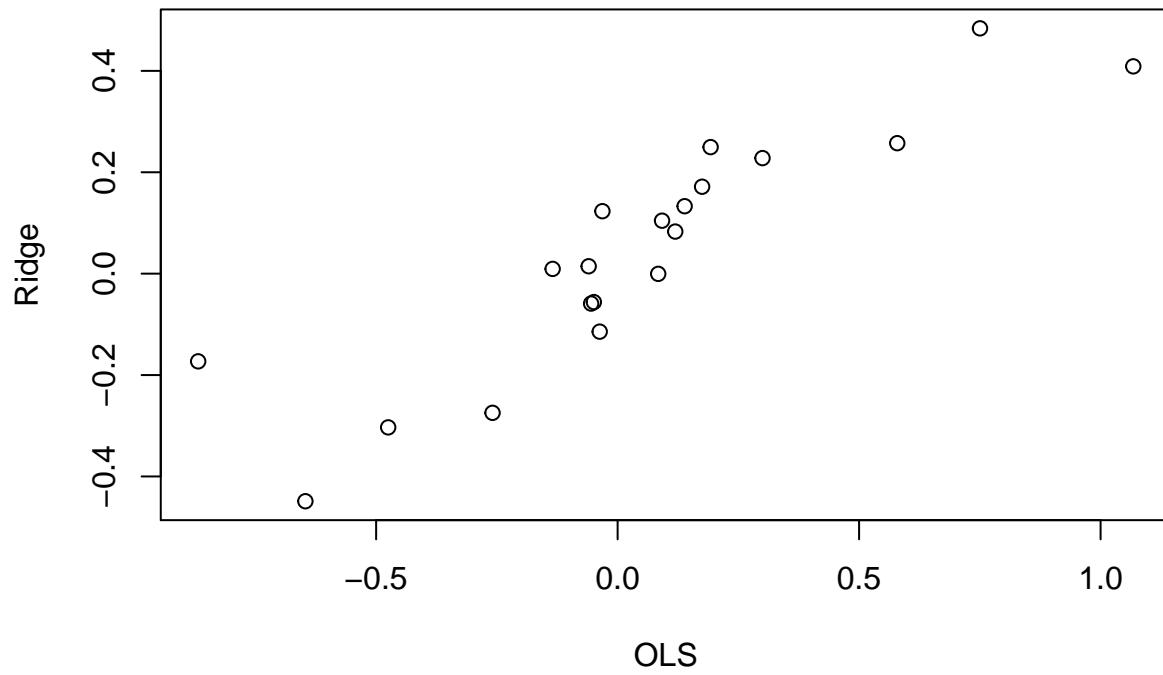
Make a scatterplot of coefficient estimates from part (b) vs those found in part (d). Repeat this to compare the results in (b) with those from (e) and also from (f). Are those coefficients with the smallest estimates those most penalized in parts (d)-(f)?

Here I will compute the coefficients for OLS and regularization methods after scaling the numeric variables.

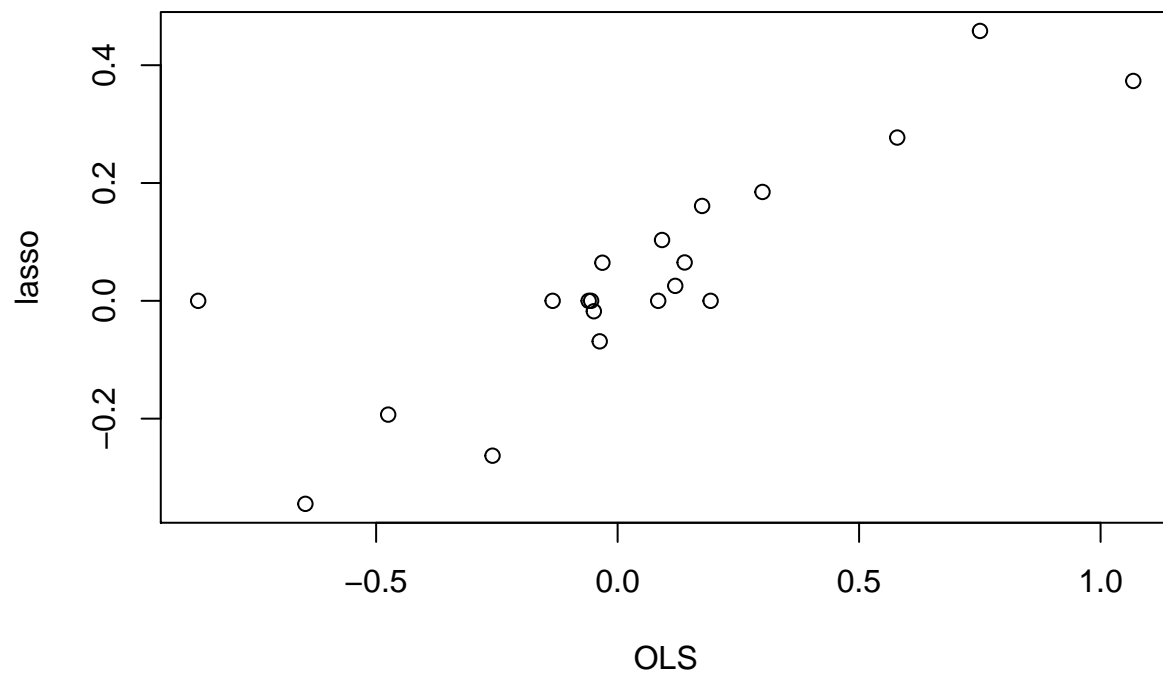
While plotting the coefficients estimates from  $OLS$  against those obtained by using ridge regression, lasso and elastic net we can tell that  $OLS$  estimates which were close to 0 (either negative or positive) were the most penalized regardless of the regularization method used.

In all three plots, visible clusters around (0,0) appear.

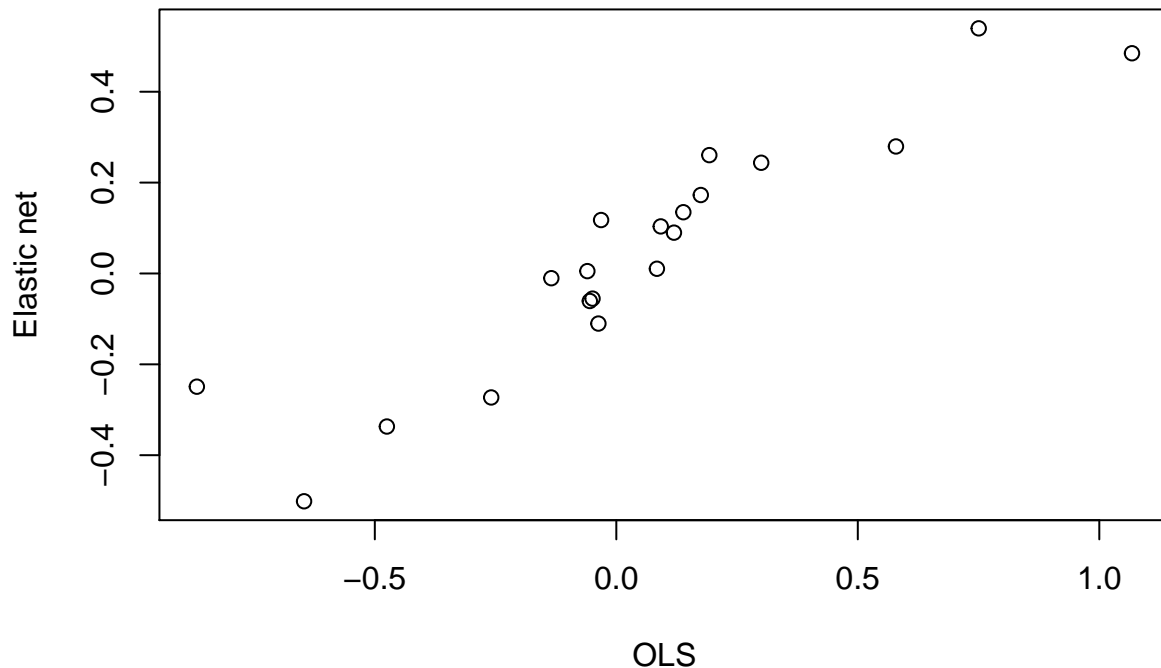
**Scatterplot of OLS VS Ridge reg. estimates**



**Scatterplot of OLS VS lasso estimates**



## Scatterplot of OLS VS Elastic net estimates



### Part (i)

Repeat parts (d)-(f) with a dataset that includes interactions and higher order polynomial terms. Do these models appear to be better than the others you've fit thus far?

For this exercise I'll be adding the extra variables that I proposed on part (c).

**NOTE:** Here some comments about what I did for this part.

- I created a new design matrix 'x.ext'. It will be used for all regularization methods.
- Since I'm still trying to predict 'Salary' I'll keep up using the response vector 'y' created in part (d).
- I'll use the same set of vectors for the hyper-parameters that I used to tune  $\lambda$  ('lambdas') and  $\alpha$  ('alphas') in parts (d)-(f).

### Ridge regression

I did pretty much the same as in part (d) after adding the higher order polynomial and interaction terms.

```
# Create the new design matrix
# NOTE: Needed to erase the intercept because glmnet creates the intercept
# automatically
x.ext <- model.matrix(Salary ~ . + I(AtBat^2) + I(Hits^2) + I(Runs^2) + I(Walks^2) +
  AtBat*CHmRun + AtBat*CRuns + Hits*CHmRun + Walks*CHmRun +
  CHmRun*CRBI + PutOuts*Assists, data = Hitters, y=Salary)[,-1]
```

After setting the seed I used  $k$ -fold cross-validation to select  $\lambda$ .

```
# Set a seed so results can be reproduced
set.seed(1990)
```

```

# Here I run 10-fold cross-validation
cv.fit.ext.ridge <- cv.glmnet(x.ext,y, alpha = 0, nfolds = 10, lambda = lambdas,
                             type.measure = 'mse')

# Obtain the lambda which gives smallest error
ext.ridge.lambda <- cv.fit.ext.ridge$lambda.min

```

The 'best' value for lambda was  $\lambda = 0.02110203$ . For this value, the ridge regression estimates are

```

# Run ridge regression with the 'best' lambda
best.ext.ridge.fit <- glmnet(x.ext,y, alpha = 0, lambda = ext.ridge.lambda)
# Obtain the coefficients
coef.ext.d <- as.matrix(coef(best.ext.ridge.fit))
coef.ext.d

```

```

##                               s0
## (Intercept)      -0.070499610
## AtBat            -0.137799284
## Hits             0.342048666
## HmRun            -0.107711776
## Runs            -0.005014960
## RBI             -0.053801033
## Walks           0.126911104
## Years           -0.113469482
## CAtBat          -0.136970481
## CHits           0.067587271
## CHmRun          0.566556520
## CRuns           0.300544557
## CRBI            0.304031212
## CWalks          -0.131248686
## LeagueN         0.151420882
## DivisionW       -0.207720387
## PutOuts         0.072332920
## Assists         -0.003224363
## Errors          -0.043576005
## NewLeagueN      -0.022816853
## I(AtBat^2)       0.044984872
## I(Hits^2)        0.102863586
## I(Runs^2)        -0.034205632
## I(Walks^2)       0.128600354
## AtBat:CHmRun     -0.185685390
## AtBat:CRuns      0.061420493
## Hits:CHmRun      0.343845804
## Walks:CHmRun     0.014520894
## CHmRun:CRBI      -0.203398031
## PutOuts:Assists -0.394998988

```

Finally, the *MSE* for this model was 0.3148338.

```

# Computing the MSE of the model
ext.ridge.pred <- as.numeric(predict(best.ext.ridge.fit, s=ext.ridge.lambda,
                                     newx=x.ext))
mse.ext.ridge <- mean((ext.ridge.pred - y)^2)

```

## Lasso

I did pretty much the same as in part (e) after adding the higher order polynomial and interaction terms.

**NOTE:** As mentioned, I'll keep using the same design matrix as above.

I set the seed and used  $k$ -fold cross-validation to select  $\lambda$ .

```
# Set a seed so results can be reproduced
set.seed(2017)
# Here I run 10-fold cross-validation
cv.fit.ext.lasso <- cv.glmnet(x.ext,y, alpha = 1, nfolds = 10, lambda = lambdas,
                             type.measure = 'mse')

# Obtain the lambda which gives smallest error
ext.lasso.lambda <- cv.fit.ext.lasso$lambda.min
```

The 'best' value for lambda was  $\lambda = 0.01$ . For this value, the lasso estimates are

```
# Run lasso with the 'best' lambda
best.ext.lasso.fit <- glmnet(x.ext,y, alpha = 1, lambda = ext.lasso.lambda)
# Obtain the coefficients
coef.ext.e <- as.matrix(coef(best.ext.lasso.fit))
coef.ext.e
```

```
##                               s0
## (Intercept)      -0.10055869
## AtBat            0.00000000
## Hits            0.20844358
## HmRun           -0.08567701
## Runs            0.00000000
## RBI             0.00000000
## Walks           0.08171496
## Years           0.00000000
## CAtBat          0.00000000
## CHits           0.00000000
## CHmRun          0.45219060
## CRuns           0.12785674
## CRBI            0.19233666
## CWalks          0.00000000
## LeagueN        0.10652381
## DivisionW      -0.19064574
## PutOuts         0.06728482
## Assists         0.00000000
## Errors         -0.01745085
## NewLeagueN     0.00000000
## I(AtBat^2)      0.00000000
## I(Hits^2)       0.12092918
## I(Runs^2)       0.00000000
## I(Walks^2)      0.11343796
## AtBat:CHmRun    0.00000000
## AtBat:CRuns     0.02108065
## Hits:CHmRun     0.22668101
## Walks:CHmRun    0.00000000
## CHmRun:CRBI    -0.16015669
## PutOuts:Assists -0.33067231
```

Finally, the  $MSE$  for this model was 0.3270579.



```

# Computing the MSE of the model
ext.lasso.pred <- as.numeric(predict(best.ext.lasso.fit, s=ext.lasso.lambda,
                                   newx=x.ext))
mse.ext.lasso <- mean((ext.lasso.pred - y)^2)

```

## Elastic net

**NOTE:** I'll be using the functions I defined for part (f). Also, followed the same steps as in part (f).

```

# Set a seed so results can be reproduced
set.seed(2018)
# Run k-fold cross-validation for the set of alphas
out.ext <- cv.alpha.vect(alphas = alphas, feat = x.ext, res = y, l = lambdas)
# Get a vector of the MSE associated to the 'best' lambda for each alpha
ext.elastic.mse <- cv.mse.vect(out.ext, featu = x.ext, resp = y)
# Select the lowest MSE over elastic.mse
index.ext <- which.min(ext.elastic.mse)
# Gets the minimum MSE over all pairs (alpha, lambda)
mse.ext.elastic <- ext.elastic.mse[index.ext]

```

The best pair  $(\alpha, \lambda)$  was  $(0.02, 0.01056876)$  with  $MSE_{\alpha, \lambda} = 0.3091055$ . This time elastic net came closer to ridge regression. The elastic net estimates are

```

# Get best pair (alpha, lambda)
elastic.alpha <- alphas[index.ext]
elastic.lambda <- get.lambda(out[[index.ext]])
# Get coefficient estimates
best.elastic.fit <- glmnet(x,y, alpha = elastic.alpha, lambda = elastic.lambda)
coef.f <- as.matrix(coef(best.elastic.fit))
coef.f

```

```

##              s0
## (Intercept) 0.078165873
## AtBat      0.000000000
## Hits       0.113280175
## HmRun      0.000000000
## Runs       0.060200212
## RBI        0.043927601
## Walks      0.094121669
## Years      0.000000000
## CAtBat     0.050945075
## CHits      0.100880948
## CHmRun     0.085086665
## CRuns      0.102835182
## CRBI       0.107026343
## CWalks     0.000000000
## LeagueN    0.062306592
## DivisionW  -0.214671980
## PutOuts    0.125899524
## Assists    0.007182533
## Errors     -0.020555784
## NewLeagueN 0.003954126

```

Using the MSE as **performance criteria**, we can say that these models appear to be ‘better’. All new models performed better than any of the models in parts (b)-(f). This means also that the models improved

the first iteration of each regularization method; elastic net (0.3091055) performed better than ridge regression (0.3148338) and lasso (0.3270579). The ‘best’ model overall judging by the  $MSE$  was elastic net with the added features.