

Project - STAT 151A

Aditya Jhanwar

12/8/2019

```
load(url("http://www.stat.berkeley.edu/users/nolan/data/baseball2012.rda"))
baseball = as_tibble(baseball)
```

Data Exploration and Feature Creation

1)

The first step is to clean the `baseball` data by removing unnecessary explanatory variables and entries missing a salary (observed `Yi`) value.

```
baseball = baseball %>% dplyr::select(-c("ID", "yearID", "teamID", "lgID", "nameFirst",
                                         "nameLast", "G_batting")) # remove unnecessary variables
baseball = baseball %>% drop_na(salary) # remove units with no salary values
```

Next, I followed the author's process in creating new features as described in the textbook.

```
baseball = baseball %>% mutate(AVG = CH/CAB,
                               OBP = 100*(CH + CBB)/(CAB + CBB),
                               CAB.avg = CAB/years,
                               CH.avg = CH/years,
                               CHR.avg = CHR/years,
                               CR.avg = CR/years,
                               CRBI.avg = CRBI/years)
```

Finally, I cleaned the `Position` and `Years` explanatory variables through reimpementing them as dummy variables.

According to Fox's description of his analysis, he mentions **middle infielders** as players who consistently played second base or shortstop so I classified all individuals with either position as such.

```
new.pos = c()
MI = c("12", "1S", "23", "2B", "2S", "3S", "02", "SS")
C = c("C", "C1", "OC")
CF = c("CF")

# Assign new factor assignment for MI (middle infielders), C (catcher),
# CF (center field), and O (other)
for (i in baseball$POS){
  if (i %in% MI) { new.pos = c(new.pos, "MI") }
  else if (i %in% C) { new.pos = c(new.pos, "C") }
  else if (i %in% CF) { new.pos = c(new.pos, "CF") }
  else { new.pos = c(new.pos, "O") }
}

baseball$POS = relevel(factor(new.pos), "O")
```

```

new.years = c()
neg.cont  = 6
neg.sal   = 3

for (i in baseball$years){
  if      (i >= neg.cont) { new.years = c(new.years, ".cont") }
  else if (i >= neg.sal) { new.years = c(new.years, ".sal")   }
  else                                     { new.years = c(new.years, "other")       }
}

baseball$neg = relevel(factor(new.years), "other")

lm.fit = lm(salary ~ ., data = baseball)
new.baseball = as_tibble(model.matrix(lm.fit)[,-1])
new.baseball$salary = baseball$salary

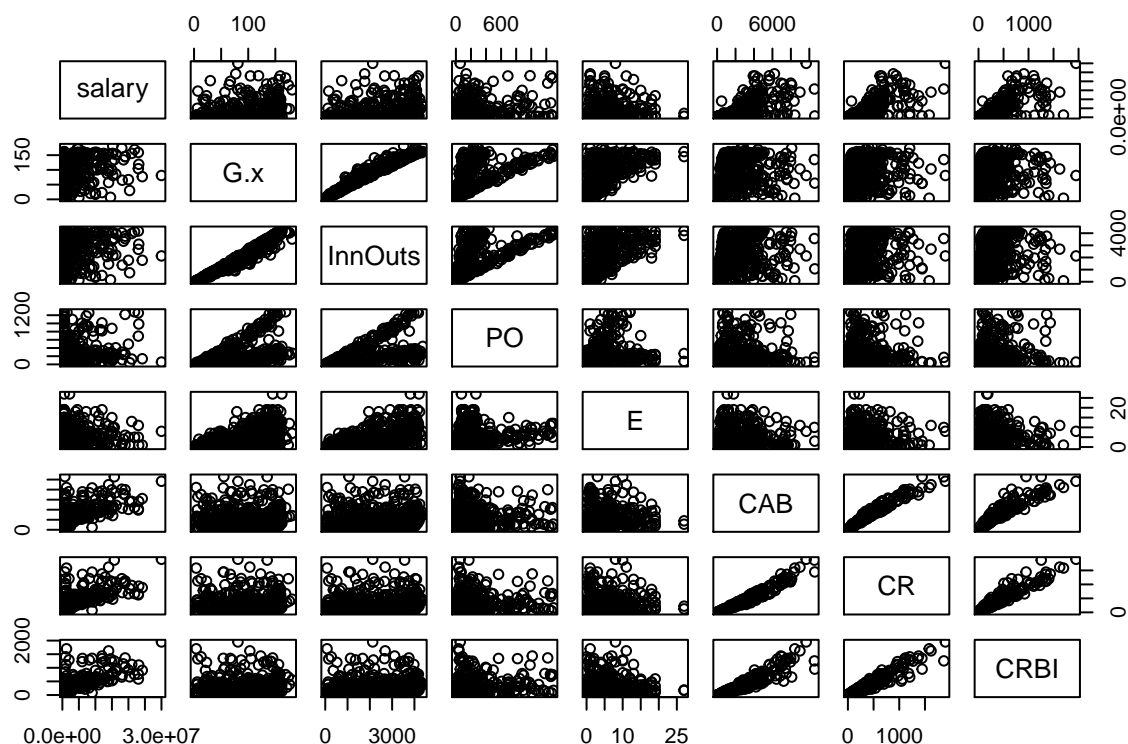
```

Now that we have completed the feature creation process, the next step is to analyze the data itself.

Firstly, I'll look at the structure of the data itself and how the different variables are associated with each other. Since there are a lot of explanatory variables within the data, I will select a few key variables I believe to be the most influential in the model and investigate the structure.

Note: - `G.x` = Position played at specified position - `InnOuts` = Time played in the field expressed as outs - `PO` = Putouts - `E` = Errors - `CAB` = Career at bats - `CR` = Career runs - `CRBI` = Career runs batted in

```
pairs(salary ~ G.x + InnOuts + PO + E + CAB + CR + CRBI, data = new.baseball)
```



From observing the paired structures of data it is evident that some features are uncorrelated whereas others are strongly correlated. However, this is mostly expected as certain features relate to one another. For example, a player's career at bats would be associated with his career runs or career runs batted in since all tie into a player's capability of scoring bases.

This indicates a possible issue in inference of coefficients through linear modeling since the standard error calculation will be grossly inflated.

In addition, I noticed some of the variables have a stronger correlation with the salary than that of other variables. For example, `G.x` and `InnOuts` do not seem to have a strong association with the other variables whereas `CAB`, `CR`, and `CRBI` have comparatively stronger correlations with each other. This indicates some sort of variable selection and model pruning may be of benefit.

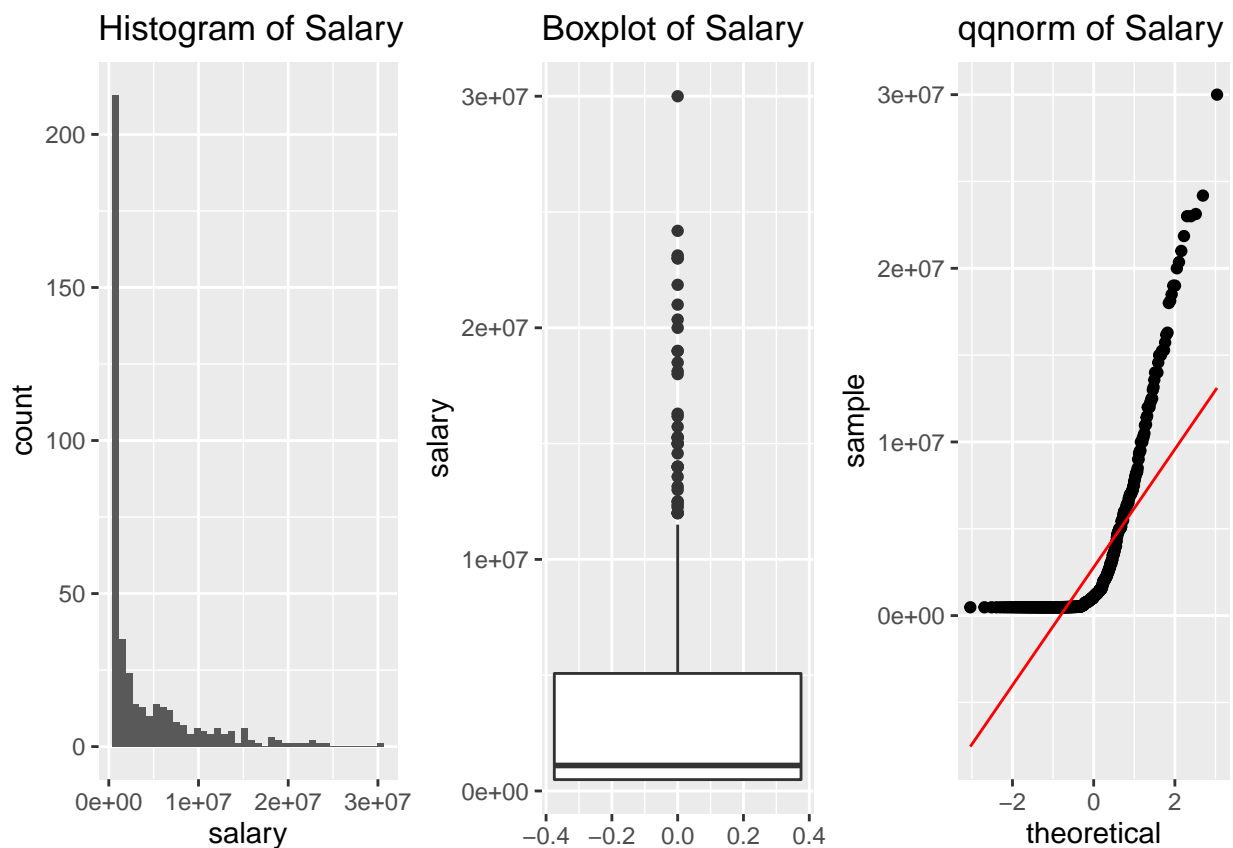
Next, I'd like to look into whether the data is distributed normally as per an assumption of gaussian distributed errors in linear modelling.

```
sal1 = ggplot(data = new.baseball, aes(x = salary)) +
  geom_histogram(bins = 40) +
  ggtitle("Histogram of Salary")

sal2 = ggplot(data = new.baseball, aes(y = salary)) +
  geom_boxplot() +
  ggtitle("Boxplot of Salary")

sal3 = ggplot(data = new.baseball, aes(sample=salary)) +
  geom_qq() +
  geom_qq_line(color = "red") +
  ggtitle("qqnorm of Salary")

grid.arrange(sal1, sal2, sal3, nrow = 1)
```



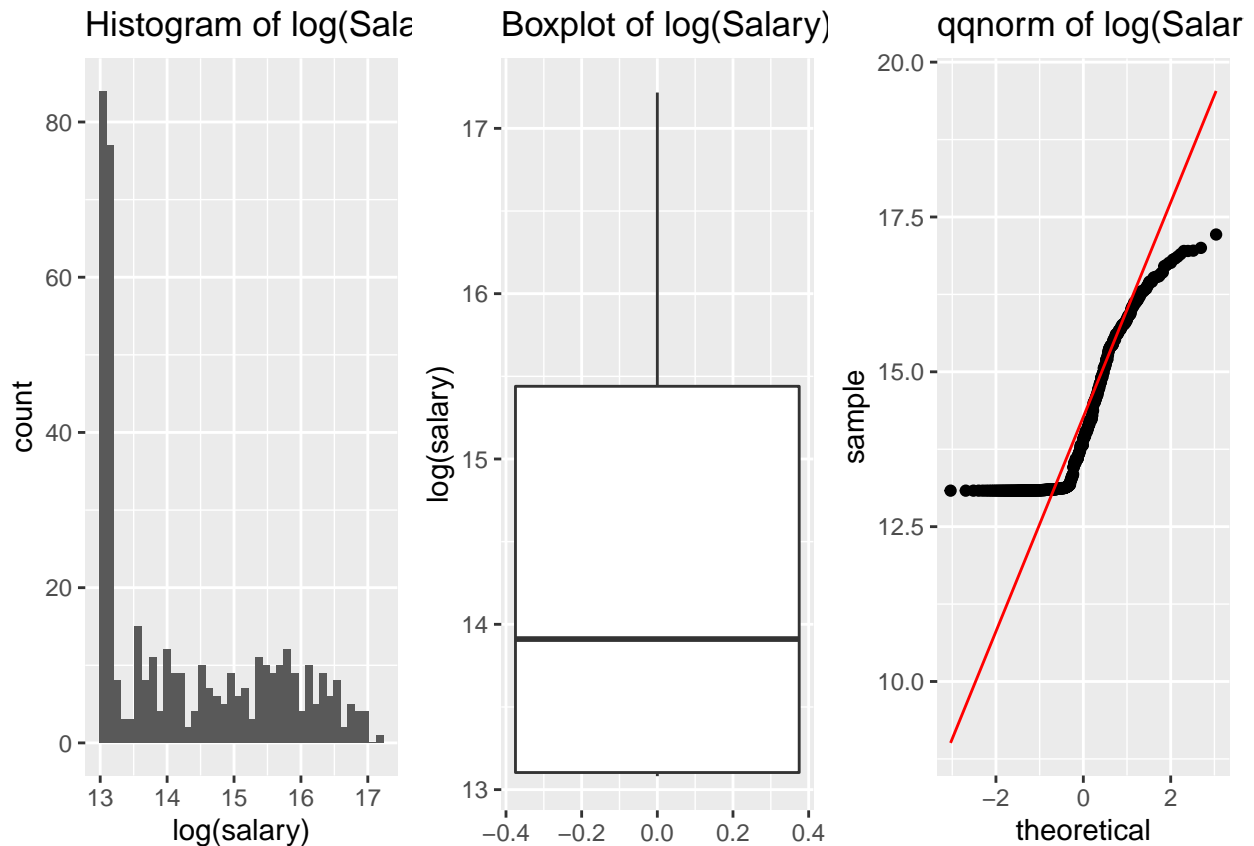
The plots above show that the observed outcome values are not distributed normally. Hence, some sort of transformation of the data is necessary in order to use linear modelling for proper inference.

```
log.sal1 = ggplot(data = new.baseball, aes(x = log(salary))) +
  geom_histogram(bins=40) +
  ggtitle("Histogram of log(Salary)")

log.sal2 = ggplot(data = new.baseball, aes(y = log(salary))) +
  geom_boxplot() +
  ggtitle("Boxplot of log(Salary)")
```

```
log.sal3 = ggplot(data = new.baseball, aes(sample=log(salary))) +
  geom_qq() +
  geom_qq_line(color = "red") +
  ggtitle("qqnorm of log(Salary)")

grid.arrange(log.sal1, log.sal2, log.sal3, nrow = 1)
```



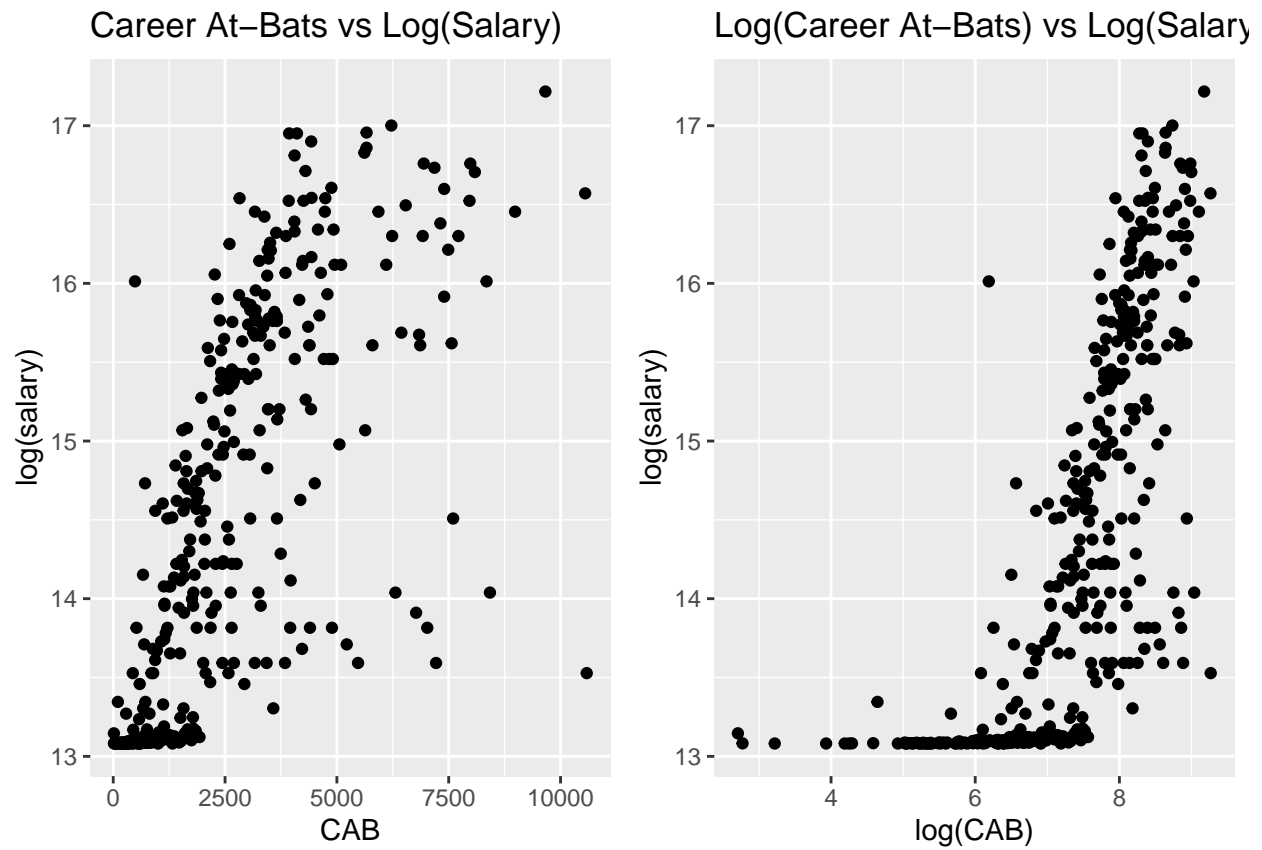
Fox mentions log transforming the **salary** data in his linear modelling analysis for this data and this is in line with the observed plots above. The histogram of the original salaries is right skewed whereas the histogram of the log transformed salaries is somewhat more stabilized. In addition, the boxplots and qqnorm plots shed more insight into how normality is improved through log transforming the data and so it would be better to use the log transformed salary values as the outcome instead.

Fox also suggests log transforming some feature variables (i.e. years in the majors, career at-bats) through preliminary examination, and so I will do the same to carry forward analysis in a similar manner. An argument for why this may be beneficial is that it might garner a stronger linear relationship between the seemingly most influential explanatory variables and the salary and thus improving the model's predictive capability overall which in turn helps in equalizing the variance.

```
cab.plot1 = ggplot(data = new.baseball, aes(x = CAB, y = log(salary))) +
  geom_point() +
  ggtitle(label = "Career At-Bats vs Log(Salary)")

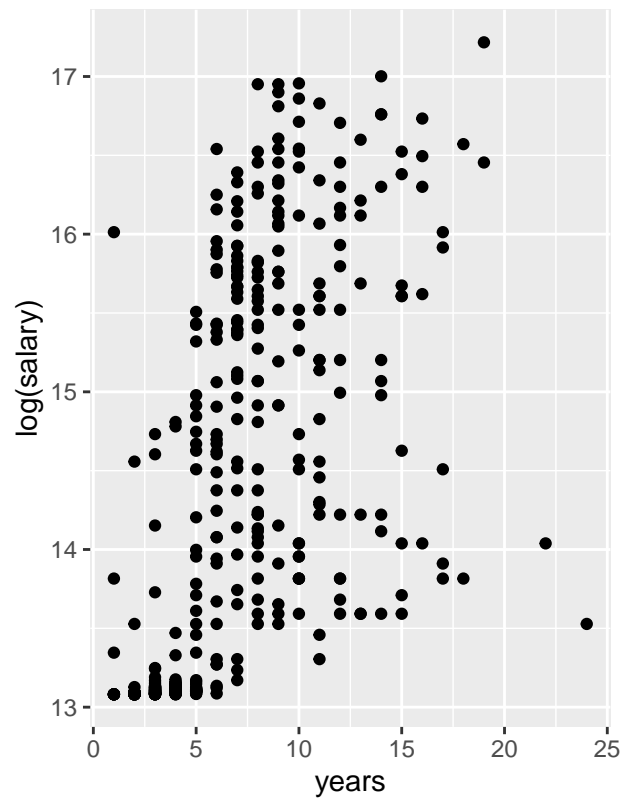
cab.plot2 = ggplot(data = new.baseball, aes(x = log(CAB), y = log(salary))) +
  geom_point() +
  ggtitle(label = "Log(Career At-Bats) vs Log(Salary)")
```

```
grid.arrange(cab.plot1, cab.plot2, nrow = 1)
```

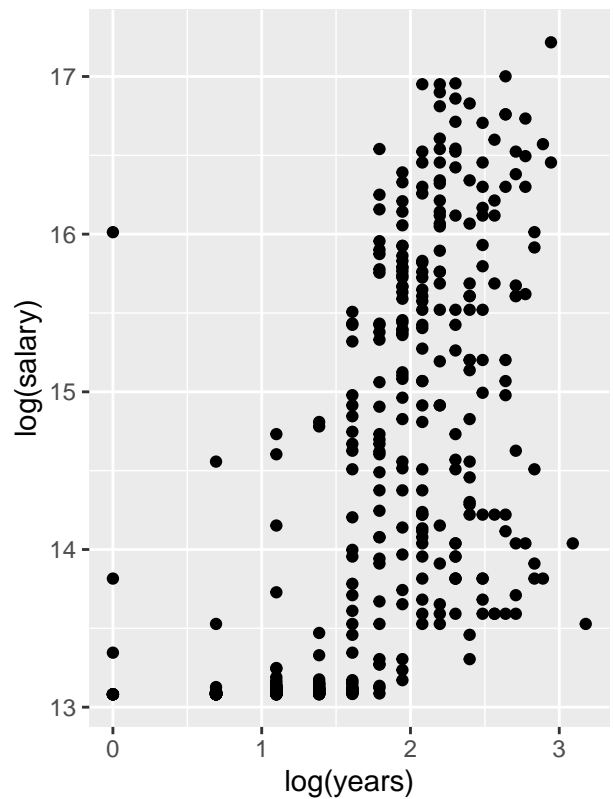


```
yrs.plot1 = ggplot(data = new.baseball, aes(x = years, y = log(salary))) +  
  geom_point() +  
  ggtitle(label = "Years in Majors vs Log(Salary)")  
  
yrs.plot2 = ggplot(data = new.baseball, aes(x = log(years), y = log(salary))) +  
  geom_point() +  
  ggtitle(label = "Log(Years in Majors) vs Log(Salary)")  
  
grid.arrange(yrs.plot1, yrs.plot2, nrow = 1)
```

Years in Majors vs Log(Salary)



Log(Years in Majors) vs Log(Salary)



```
new.baseball$log.CAB      = log(new.baseball$CAB)
new.baseball$log.years   = log(new.baseball$years)
new.baseball$log.salary  = log(new.baseball$salary)
new.baseball = new.baseball %>% dplyr::select(-c(CAB, years, salary))
```

Data Analysis

1)

For the first of the project, I will be fitting a simple model that predicts $\log(\text{salary})$ from the **dummy variables** for years in majors and $\log(\text{career runs})$, allowing for an interaction between the feature variables.

```
dat1 = new.baseball %>% dplyr::select(log.salary, CR, neg.sal, neg.cont)
simple.model = lm(log.salary ~ log(1+CR)*(neg.cont + neg.sal), data = dat1)
summary(simple.model)
```

Call:

```
lm(formula = log.salary ~ log(1 + CR) * (neg.cont + neg.sal),
    data = dat1)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.86728	-0.31778	-0.05134	0.37901	2.75941

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	12.88064	0.34241	37.617	< 2e-16 ***
log(1 + CR)	0.08743	0.09318	0.938	0.3486
neg.cont	-3.96431	0.53493	-7.411	7.10e-13 ***
neg.sal	-1.06871	0.48652	-2.197	0.0286 *
log(1 + CR):neg.cont	0.94036	0.11511	8.169	3.78e-15 ***
log(1 + CR):neg.sal	0.24940	0.11878	2.100	0.0364 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6914 on 415 degrees of freedom

Multiple R-squared: 0.7008, Adjusted R-squared: 0.6971

F-statistic: 194.4 on 5 and 415 DF, p-value: < 2.2e-16

The summary output of this simple model presents some interesting insights. All but $\log(1 + \text{CR})$ have a statistically significant t-statistic and the omnibus F-statistic is statistically significant.

Since the p-value of the t-test statistic main effect $\log(1 + \text{CR})$ is not significant whereas the interaction terms involving $\log(1 + \text{CR})$ are statistically significant, I wanted to test another hypothesis whether $\log(1 + \text{CR}) = \log(1 + \text{CR}): \text{neg.cont} = \log(1 + \text{CR}): \text{neg.sal} = 0$

```
subset.model = lm(log.salary ~ neg.cont + neg.sal, data = dat1)
summary(subset.model)
```

Call:

```
lm(formula = log.salary ~ neg.cont + neg.sal, data = dat1)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.0404	-0.2851	-0.1085	0.5602	2.8227

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	13.1900	0.1168	112.932	<2e-16 ***
neg.cont	1.9371	0.1303	14.867	<2e-16 ***
neg.sal	0.1787	0.1388	1.288	0.198

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.874 on 418 degrees of freedom
Multiple R-squared: 0.5183, Adjusted R-squared: 0.516
F-statistic: 224.9 on 2 and 418 DF, p-value: < 2.2e-16

```

r1 = 0.7008
r0 = 0.5183
df1 = 421-5-1
df2 = 2

f.stat = df1/df2 * (r1-r0)/(1-r1)
pf(f.stat, df1 = df1, df2 = df2)

```

[1] 0.9921303

The p-value of the F-statistic corresponding to the null hypothesis specified above is approximately 1, which means we fail to reject the null and perhaps the coefficients for the 3 feature variables should be 0 (and thus omitted from the model)

2)

Although I have fitted the simple model above, I want to check for any outliers, high leverage points, and influential observations for further evaluation of the simple model. All criteria in determining such observations will be in line with what Fox suggests using.

```
hat.vals = hatvalues(simple.model)
stud.res = studres(simple.model)
cook.dis = cooks.distance(simple.model)

measures = tibble(Hat.Values = hat.vals,
                  Studentized.Residuals = stud.res,
                  Cooks.Distance = cook.dis)
```

First, I'd like to take a look at the **high leverage** points, which are observations with explanatory variables markedly different from that of the average. In terms of numerical cutoffs for diagnostic statistics, *hat values exceeding twice the average hat value $(k+1)/n$ are noteworthy.*

```
h.3 = 3*length(simple.model$coefficients)/nrow(new.baseball)
high.leverage = measures[hat.vals > h.3,]
high.leverage %>% head(n = 5)
```

```
# A tibble: 5 x 3
  Hat.Values Studentized.Residuals Cooks.Distance
    <dbl>             <dbl>             <dbl>
1  0.0477             1.87             0.0289
2  0.0486             0.525            0.00235
3  0.0565             0.0299            0.00000898
4  0.0428            -0.305            0.000695
5  0.0733             0.0664            0.0000582
```

There appears to be **17** data points which have a relatively high leverage.

In addition to high leverage points, I'll analyze discrepant observations to detect outliers within the data through utilizing **studentized residuals** with a numerical cutoff of `|t-test statistic| > 2`

```
outliers = measures[abs(stud.res) > 2,]
outliers %>% head(n = 5)
```

```
# A tibble: 5 x 3
  Hat.Values Studentized.Residuals Cooks.Distance
    <dbl>             <dbl>             <dbl>
1  0.0258             2.41             0.0254
2  0.00468            -2.18            0.00370
3  0.0274             4.12             0.0768
4  0.00804            -3.01            0.0120
5  0.00476            -2.54            0.00510
```

There are **26** observations which are determined to be outliers.

Although I have determined observations that have high leverage or are outliers, what I am most concerned about are the subset of these points which have an influence on the determined coefficients of the model. Such points greatly alter the predictive capability of the simple model and thus cannot be overlooked.

Through recommendation by Fox, the criterion I will be using to determine highly influential points is $D_i > 4/(n-k-1)$

```
cook.cutoff = 4/(nrow(new.baseball)-length(simple.model$coefficients))
influential.points = measures[cook.dis > cook.cutoff,]
influential.points %>% arrange(desc(Cooks.Distance))
```

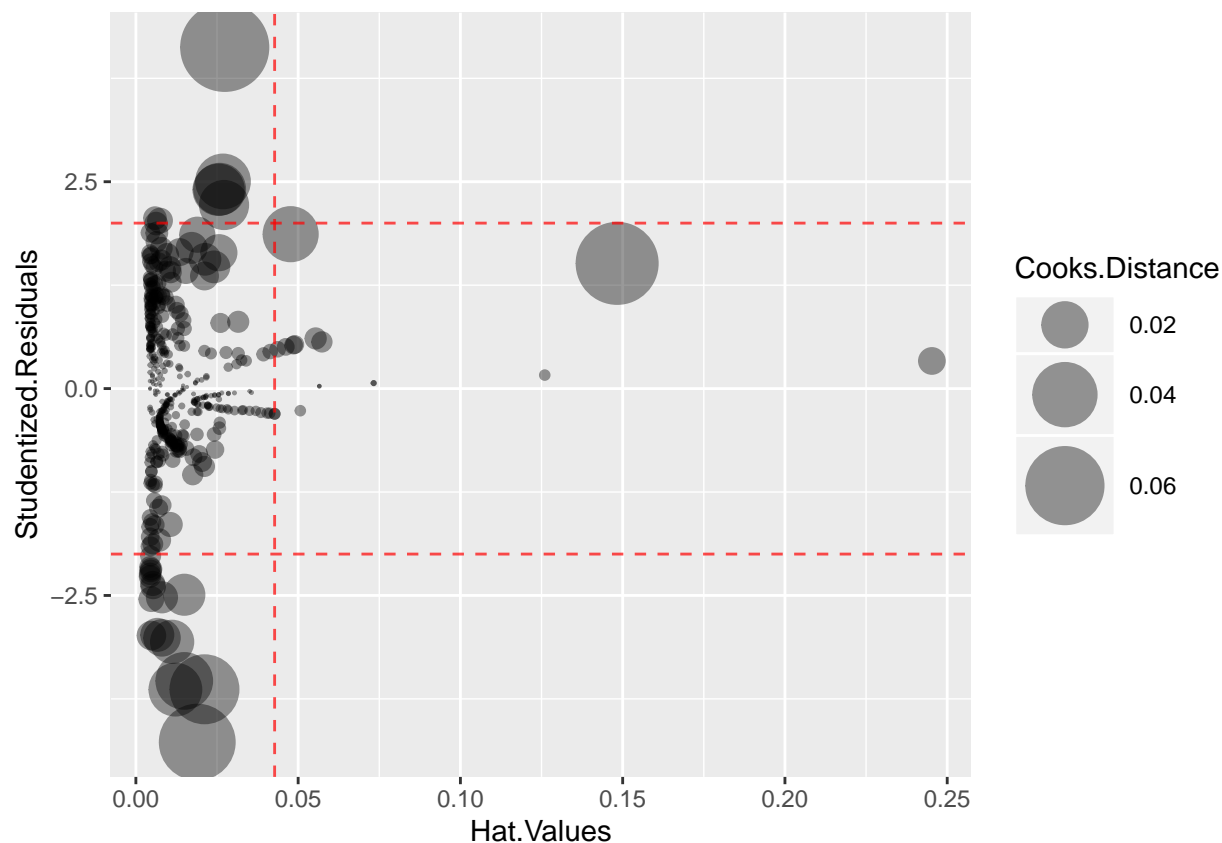
```
# A tibble: 16 x 3
  Hat.Values Studentized.Residuals Cooks.Distance
    <dbl>          <dbl>          <dbl>
1  0.0274          4.12          0.0768
2  0.148           1.51          0.0663
3  0.0189         -4.27          0.0563
4  0.0211         -3.63          0.0462
5  0.0149         -3.53          0.0306
6  0.0477          1.87          0.0289
7  0.0268          2.50          0.0285
8  0.0122         -3.64          0.0264
9  0.0258          2.41          0.0254
10 0.0255          2.40          0.0249
11 0.0272          2.22          0.0227
12 0.0112         -3.06          0.0173
13 0.0149         -2.49          0.0155
14 0.00804        -3.01          0.0120
15 0.0256          1.64          0.0118
16 0.0189          1.86          0.0111
```

It appears there are **16** influential points within the dataset. I'm not very surprised as players' baseball data is incredibly varied and prone to uniquely performing individuals, thus causing there to be influential observations.

To better grasp the idea behind the information produced above, the following is a plot of the **hat** values representing the leverage with relation to the **studentized residuals**. Each circle represents an observation with it's area proportional to it's calculated Cook's Distance.

Note: The horizontal line represents 3 times the average hat value and the 2 vertical lines mark t-test statistics of -2 and 2.

```
ggplot(aes(x = Hat.Values, y = Studentized.Residuals, size = Cooks.Distance), data = measures) +
  geom_point(alpha=0.4) + scale_size(range = c(0, 15)) +
  geom_vline(xintercept = 3*6/421, color = 'red', alpha = .7, linetype = "dashed") +
  geom_hline(yintercept = -2, color='red', alpha = .7, linetype = "dashed") +
  geom_hline(yintercept = 2, color='red', alpha = .7, linetype = "dashed")
```



Lastly, I'd like to take a look at these influential points and observe the feature values.

```
new.baseball[cook.dis > cook.cutoff,]
```

```
# A tibble: 16 x 44
```

	POSC	POSCF	POSMI	G.x	GS	InnOuts	PO	A	E	DP	G.y	AB
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0	0	1	153	147	3953	289	398	8	96	153	607
2	0	0	0	154	149	4030	297	8	6	1	155	560
3	0	0	0	104	102	2702	221	9	3	0	129	487
4	0	0	0	74	56	1523	85	91	6	7	113	278
5	0	0	0	13	12	269	86	3	0	7	60	89
6	0	0	0	131	131	3383	207	7	4	0	135	518
7	0	0	0	93	76	1953	131	2	0	0	130	384
8	0	0	0	70	55	1358	81	3	1	2	94	233
9	0	0	0	50	49	1239	37	81	8	7	74	273
10	0	0	0	107	98	2399	156	1	1	0	130	394
11	0	0	1	158	157	4176	227	434	12	93	158	593
12	0	0	0	105	103	2562	73	208	13	14	108	396
13	0	1	0	125	122	3220	339	6	4	3	128	516
14	0	0	0	4	4	81	35	2	1	4	58	163
15	0	0	1	55	37	1063	58	103	3	19	60	153
16	0	0	0	1	0	0	0	0	0	0	78	2

```
# ... with 32 more variables: R <dbl>, H <dbl>, X2B <dbl>, X3B <dbl>, HR <dbl>,
# RBI <dbl>, SB <dbl>, CS <dbl>, BB <dbl>, SO <dbl>, IBB <dbl>, HBP <dbl>,
# SH <dbl>, SF <dbl>, GIDP <dbl>, CH <dbl>, CHR <dbl>, CR <dbl>, CRBI <dbl>,
```

```
# CBB <dbl>, AVG <dbl>, OBP <dbl>, CAB.avg <dbl>, CH.avg <dbl>,  
# CHR.avg <dbl>, CR.avg <dbl>, CRBI.avg <dbl>, neg.cont <dbl>, neg.sal <dbl>,  
# log.CAB <dbl>, log.years <dbl>, log.salary <dbl>
```

Through manual observation of the feature values of these influential points it appears that these observations have much higher or lower values for certain features such as `G.x`, `InnOuts`, `P0` compared to those of non-influential points. The table of encompassing Cook's Distance with Leverage Hat Value and Studentized Residuals do a much better job providing insight into why a point is influential, however.

3)

```
all.fit = lm(log.salary ~ ., data = new.baseball)
summary(all.fit)
```

Call:

```
lm(formula = log.salary ~ ., data = new.baseball)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.63762	-0.31246	0.06423	0.31947	1.51943

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	14.2702482	0.5240182	27.232	< 2e-16	***
POSC	-0.0426637	0.1518434	-0.281	0.778886	
POSCF	-0.0268705	0.1494435	-0.180	0.857403	
POSMI	0.0780726	0.1027617	0.760	0.447882	
G.x	0.0039647	0.0046887	0.846	0.398321	
GS	-0.0062069	0.0109901	-0.565	0.572567	
InnOuts	0.0001452	0.0004900	0.296	0.767175	
PO	0.0002185	0.0003197	0.683	0.494768	
A	0.0003503	0.0008990	0.390	0.697023	
E	0.0026430	0.0104617	0.253	0.800687	
DP	-0.0042697	0.0029331	-1.456	0.146306	
G.y	-0.0118007	0.0033350	-3.538	0.000453	***
AB	0.0065267	0.0016068	4.062	5.92e-05	***
R	-0.0093742	0.0063621	-1.473	0.141465	
H	-0.0065391	0.0048988	-1.335	0.182729	
X2B	-0.0063059	0.0070018	-0.901	0.368367	
X3B	0.0006723	0.0184037	0.037	0.970877	
HR	-0.0102874	0.0121474	-0.847	0.397599	
RBI	0.0039993	0.0057956	0.690	0.490580	
SB	-0.0013153	0.0058889	-0.223	0.823382	
CS	-0.0276728	0.0185481	-1.492	0.136549	
BB	0.0056713	0.0039142	1.449	0.148192	
SO	-0.0012819	0.0019363	-0.662	0.508337	
IBB	0.0093704	0.0136217	0.688	0.491937	
HBP	0.0036525	0.0110282	0.331	0.740680	
SH	0.0093553	0.0161609	0.579	0.563012	
SF	0.0014463	0.0204969	0.071	0.943783	
GIDP	-0.0086423	0.0093884	-0.921	0.357889	
CH	0.0021802	0.0010424	2.091	0.037157	*
CHR	0.0076434	0.0038205	2.001	0.046147	*
CR	-0.0022325	0.0016753	-1.333	0.183480	
CRBI	-0.0026771	0.0018308	-1.462	0.144502	
CBB	-0.0011839	0.0005733	-2.065	0.039609	*
AVG	0.5156185	3.3598388	0.153	0.878113	
OBP	-0.0025942	0.0291738	-0.089	0.929190	
CAB.avg	-0.0042031	0.0030973	-1.357	0.175589	
CH.avg	-0.0133749	0.0133111	-1.005	0.315641	
CHR.avg	-0.0709547	0.0361205	-1.964	0.050219	.
CR.avg	0.0534385	0.0151076	3.537	0.000455	***

CRBI.avg	0.0469204	0.0166307	2.821	0.005036	**
neg.cont	1.2416258	0.2315305	5.363	1.43e-07	***
neg.sal	0.0541843	0.1522766	0.356	0.722168	
log.CAB	-0.3434284	0.1713702	-2.004	0.045783	*
log.years	0.2802810	0.2404271	1.166	0.244447	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5807 on 377 degrees of freedom
Multiple R-squared: 0.8082, Adjusted R-squared: 0.7863
F-statistic: 36.95 on 43 and 377 DF, p-value: < 2.2e-16

From former analysis I am aware of collinearity within the explanatory variables and hence inflated standard errors. However, it appears several of the coefficients are statistically significant and that the omnibus F-statistic is also statistically significant. In addition, the adjusted R-squared value is a bit lower than the R-squared of the model. This all leans toward the argument of the model requiring the explanatory variables but with possible variable selection that may improve the model's regression capability.

4)

Next, I'll be finding the best **10** models for each model size using **forward selection**. I won't output the results of the summary as they are a bit extensive and in anyways I will select 5 specific models from all of these models in the next part.

```
best10 = regsubsets(log.salary ~ .,
                    data    = new.baseball,
                    nvmax   = 43,
                    nbest   = 10,
                    method  = "forward")
# summary(best10)
```


5)

```
lowest5.bic = sort(summary(best10)$bic)[1:5]
top5.bic.loc = order(summary(best10)$bic)[1:5]
top5.models = summary(best10)$which[top5.bic.loc,]
models = c()

for(i in 1:nrow(top5.models)){
  cat(sprintf("Model %d:", i))
  cat(sprintf("\nBIC: %f", lowest5.bic[i]))

  models = c(models, list(names(which(top5.models[i,]))[-1]))
  cat(sprintf("\nFeatures: %s", paste(models[[i]], collapse = ", ")))
  cat("\n\n")
}
```

Model 1:

BIC: -586.225215

Features: GS, G.y, AB, R, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years

Model 2:

BIC: -583.405937

Features: GS, R, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years

Model 3:

BIC: -582.502074

Features: G.x, GS, G.y, AB, R, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years

Model 4:

BIC: -582.402615

Features: GS, G.y, AB, R, X2B, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years

Model 5:

BIC: -582.248168

Features: GS, G.y, AB, R, H, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years

6)

Using BIC as a criterion may not be sufficient and hence I will use 10-fold cross validation to re-rank the 5 models determined above. Rather than hard-code the k-fold procedure, I will be using built-in functions available in the `caret` package.

The output will show the features of each model for clarity as well as the previously determined BIC value and the newly determined mean-MSE (through 10-fold CV).

```
# 10-fold cross validation setup
train.control = trainControl(method = "cv", number = 10)
mean.mse = c()

for (i in 1:5){
  model.to.train.test = as.formula(paste("log.salary ~",
                                          paste(models[[i]], collapse=" + ")))
  model.trained.tested = train(model.to.train.test,
                              data = new.baseball,
                              method = "lm",
                              trControl = train.control)
  mean.mse = c(mean.mse, model.trained.tested$results[[2]]**2)

  cat(sprintf("Model %d:\nBIC: %f\nFeature: %s\n--> MSE: %f\n\n",
              i,
              lowest5.bic[i],
              paste(models[[i]], collapse = ", "),
              mean.mse[i]))
}
```

```
Model 1:
BIC: -586.225215
Feature: GS, G.y, AB, R, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
--> MSE: 0.344688

Model 2:
BIC: -583.405937
Feature: GS, R, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
--> MSE: 0.356297

Model 3:
BIC: -582.502074
Feature: G.x, GS, G.y, AB, R, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
--> MSE: 0.341484

Model 4:
BIC: -582.402615
Feature: GS, G.y, AB, R, X2B, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
--> MSE: 0.343067

Model 5:
BIC: -582.248168
Feature: GS, G.y, AB, R, H, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
--> MSE: 0.349388
```

From the values above, it is evident the following is the model with the lowest mean-MSE:

```

i = which.min(mean.mse)
cat(sprintf("Model %d:\nBIC: %f\nFeature: %s\n--> MSE: %f",
           i,
           lowest5.bic[i],
           paste(models[[i]], collapse = ", "),
           mean.mse[i]))

```

Model 3:

BIC: -582.502074

Feature: G.x, GS, G.y, AB, R, BB, CH, CBB, CAB.avg, CR.avg, CRBI.avg, neg.cont, log.CAB, log.years
 --> MSE: 0.341484

Through many repetitions it appears that the choice of model changes (stemming from the variability in how data is partitioned in 10-fold CV). It appears that it isn't always the case the model with lowest BIC has the lowest mean-MSE.

7)

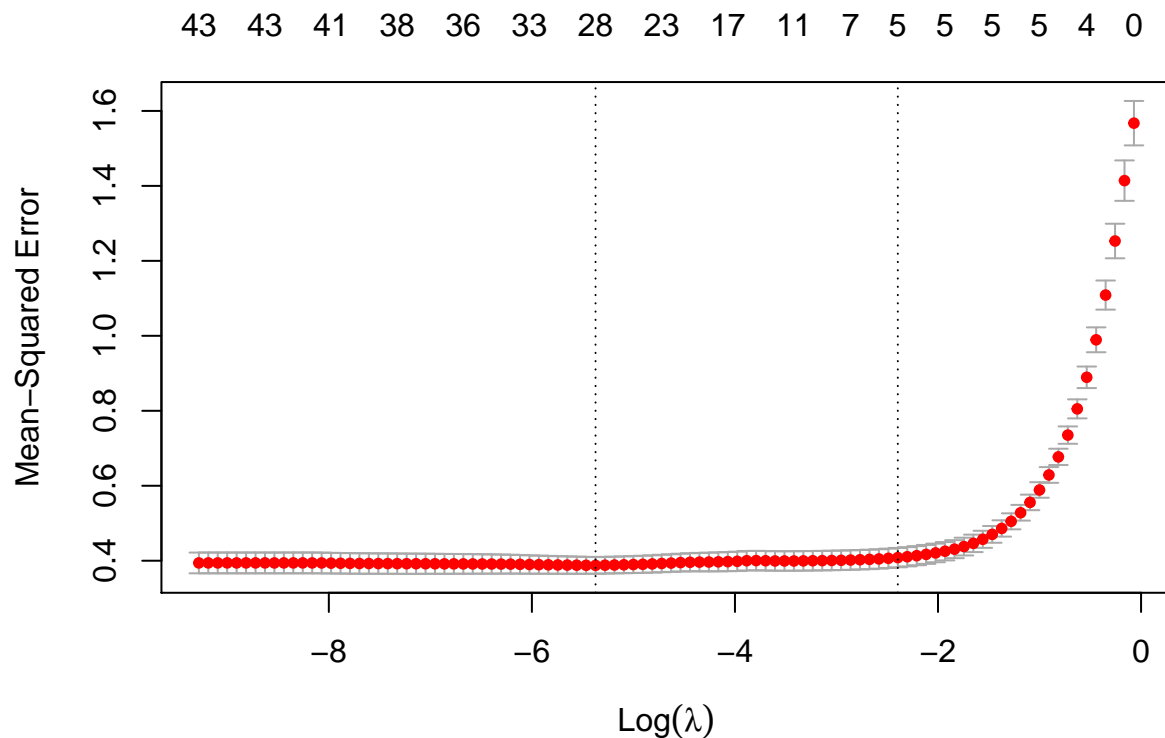
In determining the value of the hyperparameter `lambda` to use for the LASSO model (original feature variables), I will be using 10-fold cross validation with an average MSE for the criterion.

```
X = new.baseball %>% dplyr::select(-c(log.salary)) %>% as.matrix()
y = new.baseball$log.salary

lasso.model = glmnet(x = X, y = y, alpha = 1)
cv.lasso.model = cv.glmnet(x = X, y = y, alpha = 1, nfolds = 10)
```

The following is the plot of the log of the hyperparameter `lambda` with respect to the resulting average MSE.

```
plot(cv.lasso.model)
```



The left-most dashed vertical line in the plot represents the `lambda` corresponding to the minimum MSE whereas the right-most dashed vertical line represents the `lambda` 1 standard error away from the min `lambda`.

I believe the data intrinsically contains a lot of collinear explanatory variables and would like to reduce the number of features used in model prediction. I believe the `lambda` corresponding to the minimum MSE would be suitable as it not only performs variable selection through utilizing just 28 feature variables for regression rather than all 43, it also has the lowest MSE out of all other possible `lambda` hyperparameter values and thus seems to be the best choice in prediction.

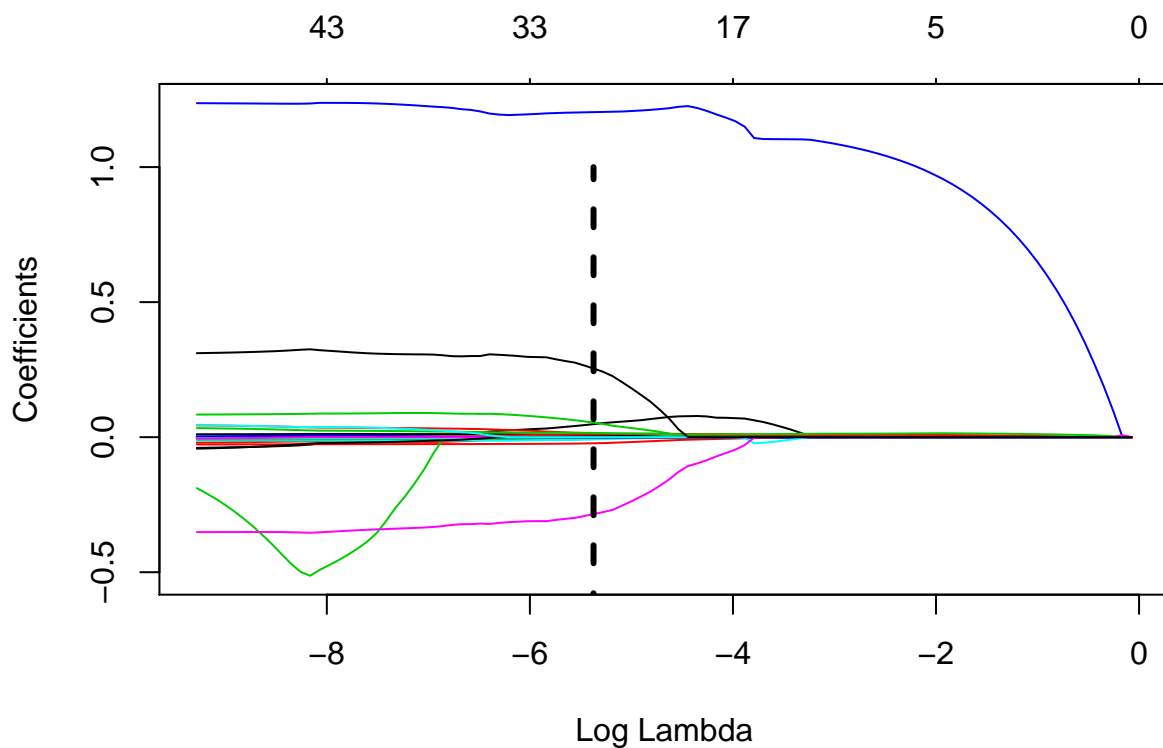
The value of `lambda` I will use for the LASSO model is the following:

```
best.lasso.lambda = cv.lasso.model$lambda.min
best.lasso.lambda
```

```
[1] 0.00464043
```

The following is a visualization of how different values of lambda influence the coefficient values of the regression model:

```
plot(lasso.model, xvar = "lambda", )
lines(c(log(best.lasso.lambda), log(best.lasso.lambda)), c(-1, 1), lty = "dashed", lwd = 3)
```



Lastly, using the hyperparameter value of lambda specified above, I'd like to analyze which coefficients are retained through implicit variable selection of LASSO.

```
best.lasso.coefs = predict(lasso.model, type = 'coefficients', s = best.lasso.lambda)
best.lasso.coefs
```

```
44 x 1 sparse Matrix of class "dgCMatrix"
```

```
      1
(Intercept) 1.392036e+01
POSC        4.819060e-02
POSCF       .
POSMI       5.389061e-02
G.x         7.148762e-04
```

GS	.
InnOuts	8.253821e-05
PO	2.248210e-05
A	.
E	.
DP	-1.477442e-03
G.y	-6.109311e-03
AB	1.353649e-03
R	.
H	.
X2B	-2.861447e-03
X3B	.
HR	-8.452375e-03
RBI	1.503657e-03
SB	-2.788658e-04
CS	-2.260273e-02
BB	4.007321e-03
SO	.
IBB	8.545359e-03
HBP	1.690402e-03
SH	7.327374e-03
SF	1.045290e-02
GIDP	.
CH	1.384542e-05
CHR	9.327663e-04
CR	8.514916e-04
CRBI	.
CBB	-1.026886e-03
AVG	.
OBP	.
CAB.avg	-3.898353e-04
CH.avg	.
CHR.avg	.
CR.avg	1.469175e-02
CRBI.avg	1.560951e-02
neg.cont	1.203511e+00
neg.sal	.
log.CAB	-2.850154e-01
log.years	2.542452e-01

From analyzing the output summary of the LASSO model coefficients, it appears 28 explanatory variables have non-zero coefficients.

I chose the `lambda` value corresponding to the minimum MSE value for the LASSO model, which is the following value:

```
best.lasso.lambda
```

```
[1] 0.00464043
```

In the previous part, I found the top 5 models with the lowest BIC and then used 10-fold cross validation with a mean-MSE criterion to find the best model with the lowest MSE. I'd like compare the that result with the model I determined using LASSO.

```
min(cv.lasso.model$cvm) <= mean.mse
```

```
[1] FALSE FALSE FALSE FALSE FALSE
```

The expression above compared the MSE of the LASSO model to the other 5 models but it turns out LASSO produces a regression model with a higher MSE compared to all.