Modeling NHL Player Contracts using their Prior Season's Stats

James Braun & Sandy Wu

STAT 350

Harsha Perera

Fall 2019

Introduction:

During the 2019 NHL[1] off-season, 91 NHL forwards[2] signed contracts. We recorded the length and total value of these contracts.[3] In addition, we made note of the forwards' individual hockey statistics from the previous season.[4] Our goal was to model the cap hit[5] of these contracts using the statistics we collected. Our response variable, Y, represented the annual salary in millions of American dollars. Our ten candidate regressors were:

X_1 = Goals X_2 = Assists X_3 = Shots on Goal X_4 = Age X_5 = Plus/Minus[6]
X_6 = Time on Ice in minutes X_7 = Games Played X_8 = Indicator Variable of Position[7]
X_9 = Penalty Minutes X_{10} = Length of Contract in Years

	V1 <fctr>	y <fctr>	x1 <fctr>	x2 <fctr>	x3 <fctr>	x4 <fctr>	x5 <fctr>	x6 <fctr>	x7 <fctr>	x8 <fctr>	x9 <fctr>	x10 <fctr>
1	Adam Erne	1.05	7	13	70	24	10	686	65	0	40	1
2	Adrian Kempe	2	12	16	118	22	-10	1175	81	0	50	3
3	Alex Chiasson	2.15	22	16	123	28	-1	1239	73	0	32	2
4	Alex Iafallo	2.425	15	18	148	25	-17	1380	82	1	22	2
5	Alexander Kerfoot	3.5	15	27	116	24	-9	1161	78	1	38	4
6	Anders Lee	7	28	23	204	28	20	1401	82	1	58	7

This is a sample of the data we collected.

We decided to only model the contracts of NHL forwards instead of the contracts of all NHL players because hockey organizations might be looking for different strengths in players of different positions. For example, goals scored might be an important stat for forwards, but not for goalies. One also might notice how length of contract is a regressor variable even though it isn't an on-ice statistic that can be determined absolutely before a contract is signed. We still included it in our analysis for three reasons. First, many times the length of a contract is settled *in principle* weeks or months before a contract it signed and that information is often leaked to the public. Second, even if the number of years isn't known before signing, contracts can only be a maximum of eight years long. So, it would not be cumbersome to try every possibility. Third, we simply believed that contract length was a very good predictor of the average salary.

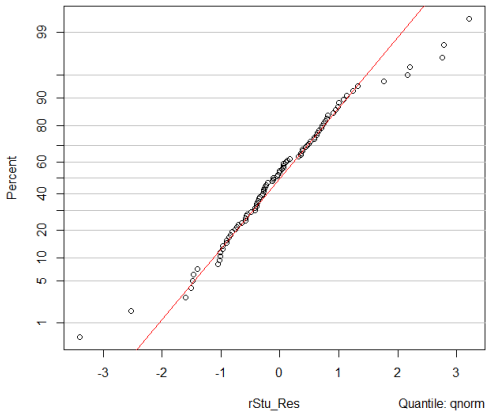
[1] National Hockey League
[2] A hockey player whose position is either left-wing (LW), right-wing (RW), or centre (C)
[3] Contract Information via: https://www.capfriendly.com/
[4] NHL Stats via: https://www.hockey-reference.com/
[5] Also referred to as average annual value (AAV) of the contract, or the yearly salary
[6] Number of times the player's team scored a goal when the player was on the ice minus the number of times the opposing team scored when the player was on the ice
[7] 1 if they were a centre, 0 if they were a winger

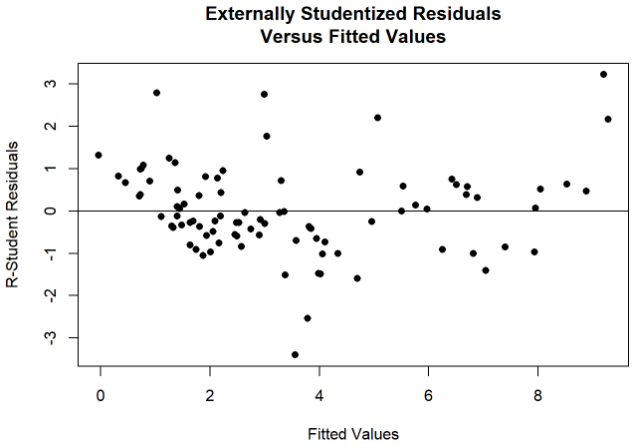
Analysis/Methods:

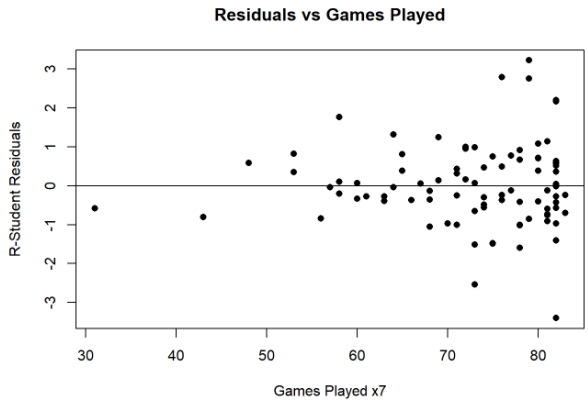
We begun by creating a linear model that contained all ten candidate regressor variables. The resulting equation was:

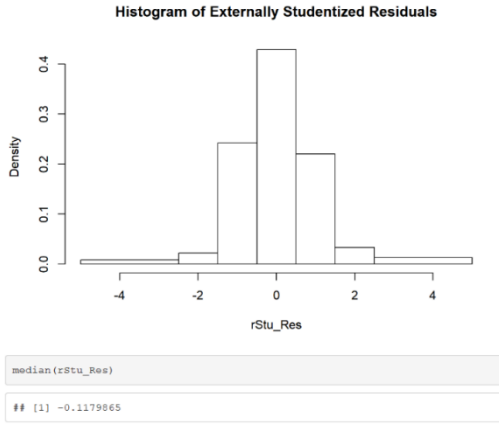
$$\hat{y} = -0.787 + 0.0664x_1 + 0.0685x_2 + 0.0095x_3 + 0.0499x_4 - 0.0108x_5 + 0.00112x_6 - 0.0445x_7 - 0.134x_8 + 0.00338x_9 + 0.394x_{10}$$

We then decided to test the assumptions and produced the following graphs:



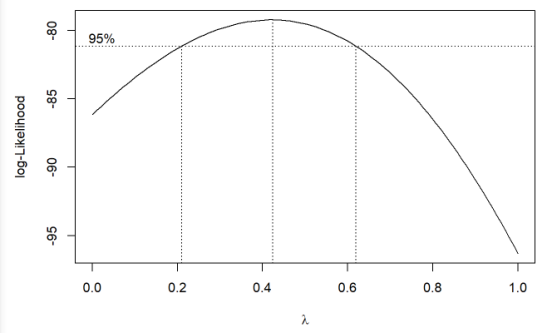






We constructed many R-Student residual plots for our full model. We noticed that for our normal probability plot (seen in the top-left), the distribution of residuals is slightly light-tailed. A transformation on the response variable may be helpful. Our plot of the residuals against fitted values (top-right) is satisfactory as there are no obvious patterns. We then plotted the residuals against all ten regressors in ten sepearate plots. There were no major patterns for any our variables except for the plot with games played (x_7) (bottom-left). The games played plot shows an increasing funnel pattern. This means that as the number of games played increases, so does the variance. This again tells us that a transformation may be required. The median of the histogram (bottom-right) of externally studentized residuals is -0.118, which is less than 0. This means that our model tends to slightly overestimate each contract's salary.

We then decided to use the Box Cox method to determine a λ for a suitable power transformation.

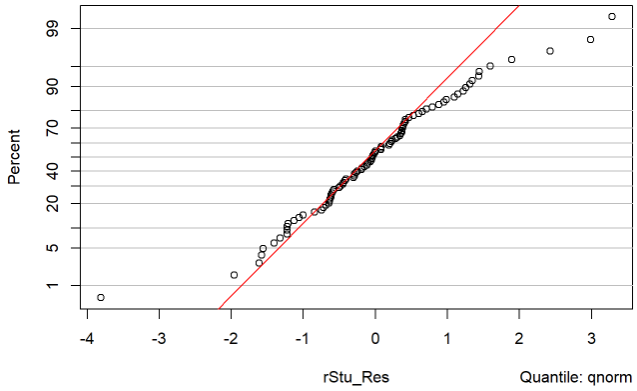


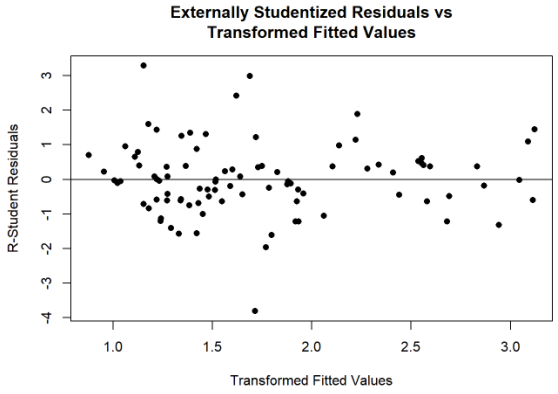
Looking at the results, we can see that λ = 0.5 is within the 95% confidence interval. Therefore, we used a square root transformation to get $y' = \sqrt{y}$.

After fitting a new model using the transformed y, we got the following equation:

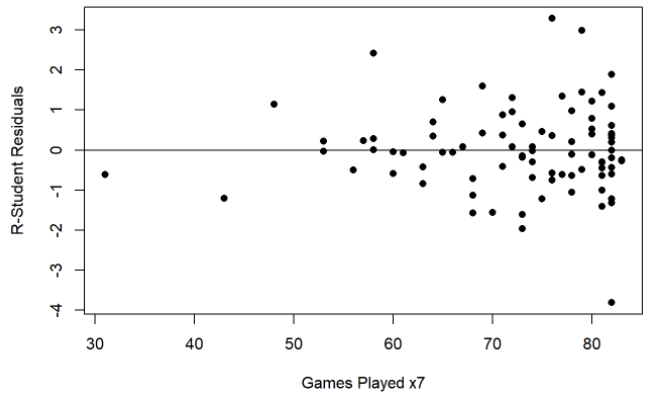
$$\hat{y} = 0.578 + 0.0193x_1 + 0.0126x_2 + 0.0022x_3 + 0.0129x_4 - 0.0029x_5 + 0.0004x_6 - 0.0104x_7 - 0.056x_8 + 0.0007x_9 + 0.102x_{10}$$

We then looked at the same four residual plots as before for comparison's sake.

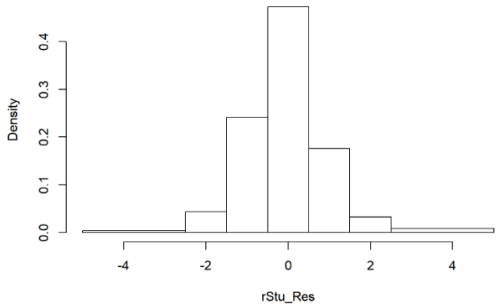




Residuals vs Games Played



Histogram of Externally Studentized Residuals




```
median(rStu_Res)

## [1] -0.03849216
```

Looking at the results, we can see that our normal probability plot looks slightly worse than before; it looks like the residual distribution is more light-tailed. However, the residuals vs fitted values and residuals vs games played plots look roughly the same. In addition, according to the histogram of residuals, the median of our externally studentized residuals decreased in absolute value to -0.0385. Our model now tends to overestimate cap hit slightly less than before. Overall, our transformation did not have a substantial impact on the residuals, but we still feel that it was worth it.

Following the transformation, we moved onto variable selection. We decided to use the old-school methods of forward, backward, and stepwise variable selection to see which variables were significant in terms of predicting the contracts' cap hit.

We obtained the following results:

Forward:
```
Step:  AIC=-267.51
yprime ~ x1 + x10 + x2 + x5 + x3 + x7 + x4
```

Backward:
```
Step:  AIC=-267.41
yprime ~ x1 + x2 + x3 + x4 + x6 + x7 + x10
```

Stepwise:
```
Step:  AIC=-267.51
yprime ~ x1 + x10 + x2 + x5 + x3 + x7 + x4
```

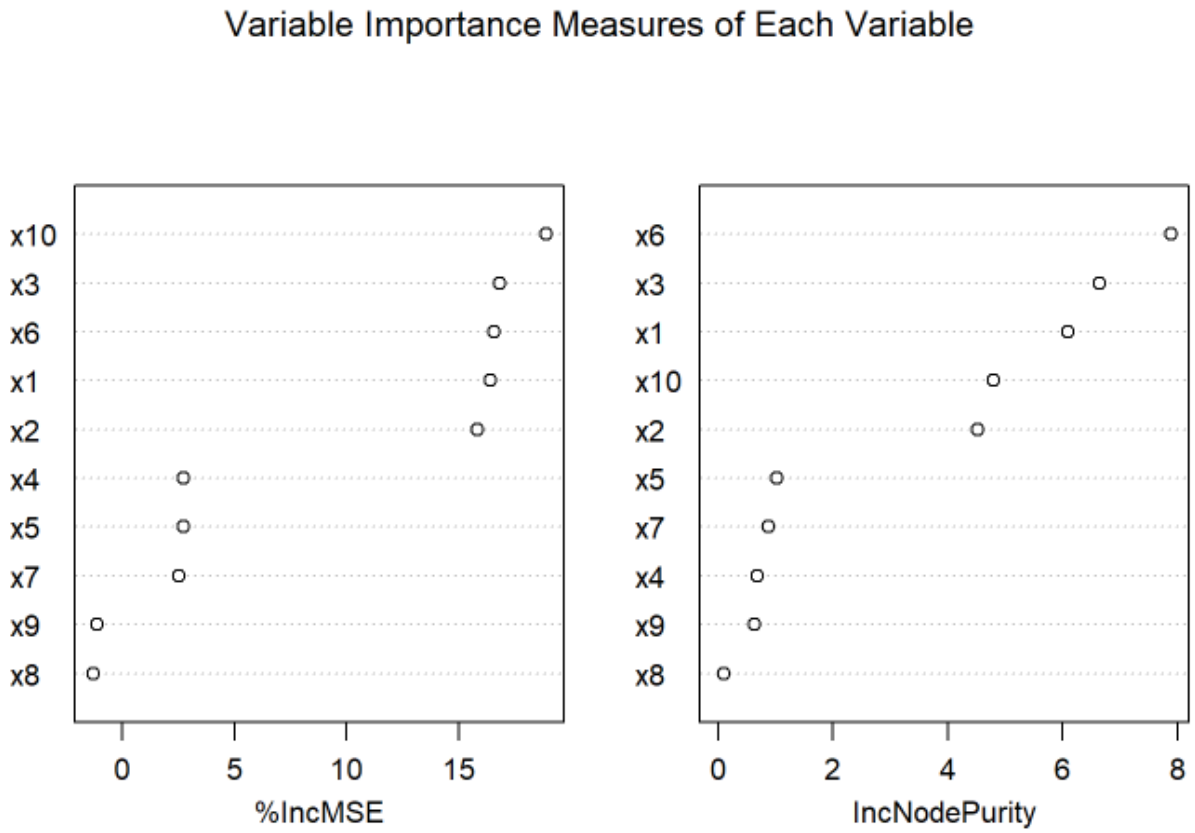
All three techniques determined that x_1, x_2, x_3, x_4, x_7, and x_{10} were significant. In addition, forward and stepwise selection found x_5 to be significant while backward selection found x_6 to be significant. No technique selected x_8 or x_9. One last thing to note was that all three techniques resulted in virtually identical AIC values, either -267.51 or -267.41.

We then employed a more modern approach to selecting variables by using a random forest. Random forests are an improvement on decisions tree because they reduce the variance without increasing the bias. They do this by "averaging multiple deep decision trees, trained on different parts of the same training set".[8] One way random forests reduce the variance is by using bootstrap aggregating (aka bagging). This is when you select m random samples of size n with replacement from the training set (which has n observations). In our dataset, n = 91. Since sampling is done with replacement, not every observation appears in every sample. Roughly 2/3 of the observations appear in any given sample and 1/3 don't. If an observation doesn't appear, it is considered to be "out-of-bag" (aka OOB). Using each sample once, you fit m trees. By default, m = 500. Then, for each observation, you pass it through every tree where the observation is considered to be OOB. Then, you average the predicted values to get the OOB prediction. For example, if we created 5 trees, and hockey player Bob was used in creating trees 2 and 3, then we would pass him through trees 1, 4, and 5 to get three predictions which we would then average to get the OOB prediction for Bob. Finally, for regression we determine the average squared difference between the observed values and their OOB predictions. This is the OOB MSE. There is one extra step with random forests: feature bagging. This is where at each node, rather than considering every variable to split it, you only consider some of them. For regression, you consider roughly one third of variables. This is done because if one variable is really good at predicting observations, then many trees will start the same way and thus be correlated.

[8] https://en.wikipedia.org/wiki/Random_forest

Moving onto variable importance, there are two ways to measure it in random forests. For the first way, you begin by calculating the OOB MSE for each tree. Then, for each variable, permute its values and again calculate the OOB MSE for each tree. If a variable is important, then permuting its values will cause the MSE to increase dramatically. For example, if Bob's goal total changed from 40 to 14, we would expect a random forest model to predict that he would make a lot less money and therefore the prediction would be further from the true salary. Finally, take the difference between the two MSEs (permuted and non-permuted), average over all the trees, and normalize by the standard deviation of the differences. Looking at the standardized values for each variable, a larger number indicates a more important variable. For the second way to measure variable importance, you determine how much, on average, the OOB residual sum of squares decreases when you split on that variable. In other words, this is how much the node purity increases. Again, a larger number indicates a more important variable.

We generated the following variable importance plots:



From these two plots, we saw that variables x_1, x_2, x_3, x_6, and x_{10} are significant.

We then took a step back and summarized our variable selection results. Variables x_1, x_2, x_3, and x_{10} appeared in all four variable selection techniques. Variables x_4, x_5, x_6, and x_7 were chosen by at least one technique. Variables x_8 and x_9 were never selected.

Therefore, we decided to perform All Possible Subset Regression, with x_1, x_2, x_3, and x_{10} already in every model, and then add all possible combinations of x_4, x_6, and x_7. We made the executive decision to not include x_5 because, according to hockey data experts, plus/minus is the worst statistic in hockey at evaluating talent. Garret Hohl, co-founder and Chief Technical Officer at HockeyData Inc, even said that plus/minus "could even be in contention for just the worst statistic in sport."[9]

```
##   # Regs p Regressors   SSres     Rsq Rsq_adj   MSres       Cp   PRESS
## 1      4 5       None 4.70769 0.86344 0.85709 0.05474 15.84768 5.38317
## 2      5 6         x4 4.45773 0.87069 0.86309 0.05244 12.70547   5.338
## 3      5 6         x6 4.70196 0.86361 0.85559 0.05532  17.7298 5.48098
## 4      5 6         x7 4.38067 0.87293 0.86546 0.05154 11.12017 5.12693
## 5      6 7      x4,x6 4.43637 0.87131 0.86212 0.05281 14.26602 5.40119
## 6      6 7      x4,x7 4.16657 0.87914 0.87051  0.0496  8.7156 5.07512
## 7      6 7      x6,x7 4.16601 0.87916 0.87052  0.0496 8.70421 4.99407
## 8      7 8   x4,x6,x7 4.04103 0.88278  0.8729 0.04869 8.13304 5.02106
```

This is a table summarizing the results of the All Possible Subset Regression. Judging by Mallow's Cp Statistic, we have our three finalist models.

To try to determine which of the three models was best, we looked at their VIF values.

```
vif(m2_47)
```

```
##       x1       x2       x3       x4       x7      x10
## 3.541912 2.306068 4.048849 1.139699 1.444963 1.600799
```

```
vif(m2_67)
```

```
##       x1       x2       x3       x6       x7      x10
## 3.531177 2.862633 4.627263 9.349973 2.868712 1.742900
```

```
vif(m3_467)
```

```
##       x1       x2       x3       x4       x6       x7      x10
##  3.608250  2.937499  5.257259  1.223972 10.041344  3.035665  1.748973
```

From this output, we can see that only the model with x_4 and x_7 had no issue with multicollinearity. We felt that choosing this model was worth not having the absolute lowest Cp value.

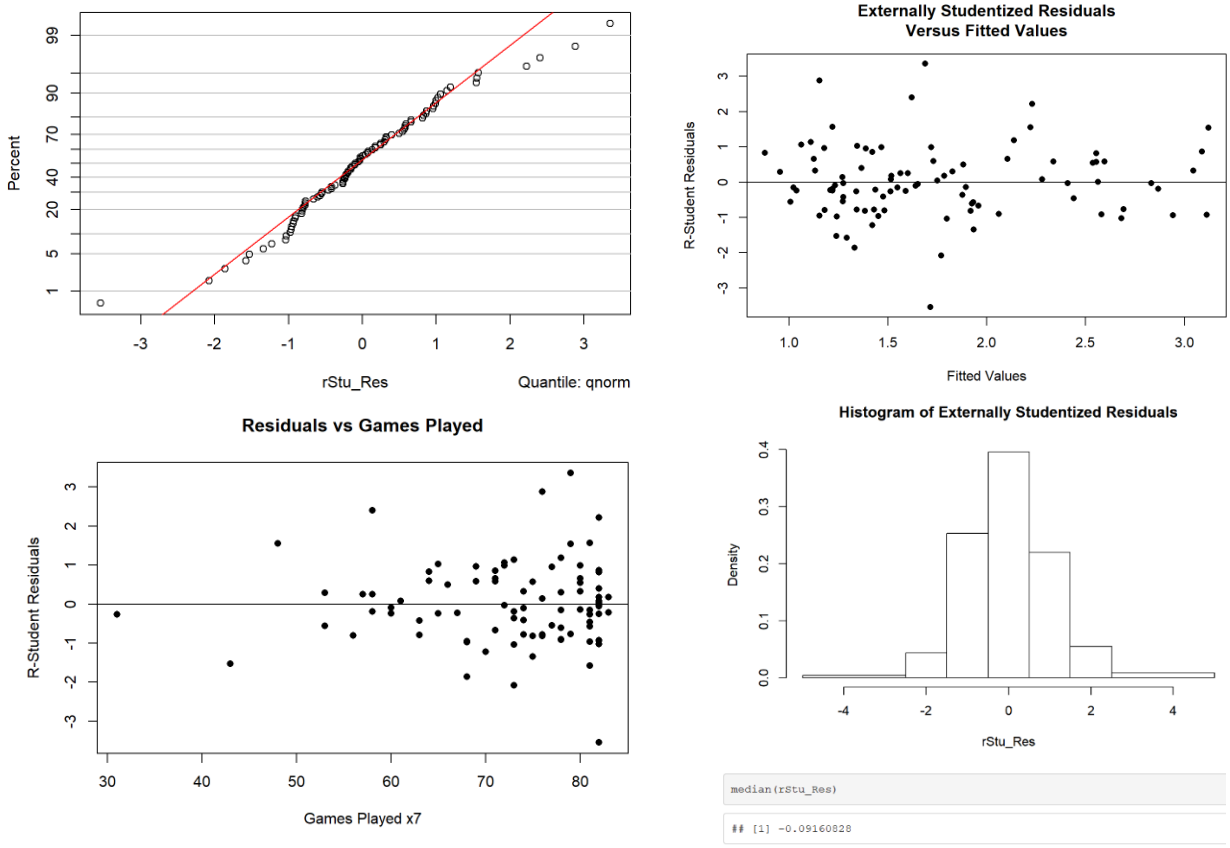
[9] https://hockey-graphs.com/2016/11/01/behind-the-numbers-why-plusminus-is-the-worst-statistic-in-hockey-and-should-be-abolished/

For the final model, we obtained the final model equation:

$$\hat{y} = 0.508 + 0.0187x_1 + 0.0137x_2 + 0.0035x_3 + 0.0148x_4 - 0.0068x_7 + 0.105x_{10}$$

Variables x_1, x_2, x_3, x_4, x_7, and x_{10} were selected for the final model.

For comparison's sake, we looked at the same four residual plots one last time:



We can see that all the plots look generally the same as before. The tails of our normal probability plot look better than in our previous models; the residual distribution is less light-tailed. There are again no problems with our plot of the residuals against the fitted values. The plot of residuals vs games played remained the same. The histogram of residuals is better than our full model, but slightly worse than our transformed model. The median is now -0.0916. This is not a major problem since the median is relatively close to 0, but we are still overestimating our salaries by a small amount.

	Player	G	A	S	AGE	GP	LENGTH	CAP.HIT
1	Alex DeBrincat	41	35	220	21	82	3	6.400000
2	Brayden Schenn	17	37	159	28	72	8	6.500000
3	Nico Hischier	17	30	160	20	69	7	7.250000
4	Austin Watson	7	9	64	27	37	3	1.500000
5	Nick Schmaltz	7	18	62	23	40	7	5.850000
6	Michael Raffl	6	12	65	30	67	2	1.600000
7	Mark Stone	33	40	199	26	77	8	9.500000
8	Jakob Silfverberg	24	19	163	28	73	5	5.250000
9	Eric Staal	22	30	215	34	81	2	3.250000
10	Frank Vatrano	24	15	208	24	81	3	2.533333
11	Auston Matthews	37	36	251	21	68	5	11.634000
12	Jordan Martinook	15	10	146	26	82	2	2.000000
13	Teuvo Teravainen	21	55	167	24	82	5	5.400000
14	Marcus Sorensen	17	13	100	26	80	2	1.500000

To validate our model, we found new data to test on. 14 NHL forwards signed contracts either just before the off-season started or just after the off-season ended. We used our final model to predict these players' cap hit.

Our model produced the following results:

$$new\ data\ R^2_{prediction} = 0.8317$$

$$PRESS\ R^2_{prediction} = 0.8528$$

$$Final\ Model\ R^2 = 0.8791$$

Our model explained roughly 83% of the variation in the test data. This is slightly less than the R^2 prediction value based on the PRESS statistic of 0.85. It is also less than the final model's R^2 value of 0.88. However, we believe that our model is still quite good at predicting.

$$MS_{Res} = 0.0496$$

$$Average\ Squared\ Prediction\ Error = 0.0799$$

Comparing the residual mean square error of the final model to the average squared prediction error for the new contracts, we can see some degradation of performance. However, it is not severe.

```
## df.final
##  7 41 54 70 87
##  6  7  7  5  7
```

These are the points we considered to be influential. We selected these points because they exceeded the influential cut-offs for five or more measures.

```
##         7        41        54        57        70        87
##  2.881883 -2.077919 -3.543969  2.406732  2.221208  3.353936
```

Comparing our influential points to our outlier results, we can see that all five influential points were also in our outlier results. In addition, observation 57 was also considered an outlier even though it was only influential in 2 measures. These six outlier points were selected because the absolute value of their R-student residual was greater than 2. Note that a negative residual means our model overpredicted their salary and a positive residual means our model underpredicted their salary.

Observation 7: Andre Burakovsky | Observation 41: Joe Thornton | Observation 54: Kevin Labanc | Observation 57: Marcus Johansson | Observation 70: Patrik Laine | Observation 87: Wayne Simmonds

Next, we tried to determine why our model failed to accurately predict these players' (the outliers') salaries.[10]

We could not determine exactly why Andre Burakovsky's salary was so high. However, we came up with three possibilities. First, he could excel in stats not collected for our model, like hits or shots blocked. Second, he could have certain "intangibles" that can't be measured quantitatively. For example, he could be a hard worker or a great presence in the locker room. Third, his new team, the Colorado Avalanche, could have simply overpaid.

Joe Thornton was an outlier simply because of how old he is. He was the only forward over the age of 35 who signed a contract and he was 40. Our final model says that salary increases with age. This might only be true up to a certain point before it starts to decrease as age increases.

When we first saw Kevin Labanc's contract, it just seemed like a massive steal. We were not alone in thinking that.[11] Many hockey analysts before us have tried to explain why his contract was so small. We believe TSN hockey analyst Travis Yost gave good insight into the reason.[12] Basically, San Jose has been close to the salary cap ceiling for a while, and Labanc did them a favour by signing such a cheap contract. However, if he has another great year, then he will make considerably more money in his next contract.

Marcus Johansson was injured for a sizeable portion of last season. We decided to extrapolate his statistics as if he played in all 82 games. Re-predicting his salary gave a new estimate of $3.27 million. This is greater than our original estimate of $2.60 million. However, it is still off from his actual salary of $4.5 million.

Patrik Laine and Wayne Simmonds both had off-years last season. Therefore, we wanted to see what would happen if we used Laine's 2017-18 stats and Simmonds' 2016-17 stats to make new predictions. We obtained new salaries of $5.83 million for Laine and $4.57 million for Simmonds. This is much closer to their actual salaries of $6.75 million and $5 million respectively. Especially compared to the original estimates of $4.64 million and $2.43 million.

[10] Player Photographs via: http://www.hockeydb.com/
[11] https://www.bardown.com/hockey-fans-are-perplexed-over-kevin-labanc-s-new-contract-1.1334934
[12] https://www.tsn.ca/on-kevin-labanc-s-confounding-contract-1.1335215

Conclusion:

Looking back at our final model, we are very satisfied with our results. After the transformation, the final model had no egregious violations of the assumptions. In addition, we were able to successfully employ both old and new variable selection techniques to find the truly important regressors. Also, we were able to utilize both our own knowledge of the game of hockey and the knowledge of hockey experts to make pragmatic decisions during the model fitting process. Our final model had no issues with multicollinearity and no hockey statistics we regressed with are difficult to track. Furthermore, we were able to use new data to validate our final model and our results show that it is quite good at predicting new contracts. Finally, we were able to gain a good understanding about the limitations of our model by thoroughly analyzing the outliers. However, there are still improvements we could implement to make our model even better.

We came up with four potential improvements for our model. First, we could make stats per game instead of season totals. This could improve prediction performance of players who were injured or did not play the whole season. It would also allow us to get rid of the GP regressor and its non-constant-variance-indicating plot. However, players who play in very few games (i.e. less than 25) could have misleading results. Second, we could use a weighted average of the past few seasons' results instead of just the previous year's. This could better predict contracts of players who had an off-year or two. Third, we could gather more contract data of veteran (aged 35+) NHL players. That would probably make age a quadratic regressor rather than linear. But hopefully it would more accurately depict the deterioration of ability and salary that actually associates aging. The problem with this suggestion is the fact that very few old NHL players sign contracts. Fourth, we could use more advanced stats such as Corsi, PDO, or Point Shares. These are all relatively new and slightly complicated statistics, but they are becoming very popular in the hockey analytics community.

APPENDIX

Resources Used:

- Contract Information:
  - https://www.capfriendly.com/
- NHL Stats
  - https://www.hockey-reference.com/
- Player Photographs:
  - http://www.hockeydb.com/
- Articles Used:
  - https://en.wikipedia.org/wiki/Random_forest
  - https://hockey-graphs.com/2016/11/01/behind-the-numbers-why-plusminus-is-the-worst-statistic-in-hockey-and-should-be-abolished/
  - https://www.bardown.com/hockey-fans-are-perplexed-over-kevin-labanc-s-new-contract-1.1334934
  - https://www.tsn.ca/on-kevin-labanc-s-confounding-contract-1.1335215
- 'randomForest' R package documentation:
  - https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

Pre-Setup

```
library(e1071)
library(alr3)
library(qpcR)
library(MASS)
library(leaps)
library(randomForest)
library(car)
library(CombMSC)
library(olsrr)

NHLstats = read.csv("NHL_STATS.csv")
NHLContracts = read.csv("NHL_con_fin.csv")
new.NHL = read.csv("new_nhl.csv")
names = NHLstats$Player
fixnames = gsub("\\\\.*", "", names)
NHLstats$Player = fixnames
NHLmerged = merge(NHLstats, NHLContracts, by="Player")
NHLmergedfin = NHLmerged[,c(1,6,9,10,11,13,14,19,21,32,37,39)]
NHLmergedfin$CAP.HIT = NHLmergedfin$CAP.HIT / 1000000
NHLmergedfin$is.centre = as.numeric(NHLmergedfin$Pos == 'C')
head(NHLmergedfin)
```

```
##              Player Pos GP  G  A X... PIM   S  TOI AGE LENGTH CAP.HIT
## 1         Adam Erne  LW 65  7 13   10  40  70  686  24      1   1.050
## 2      Adrian Kempe  LW 81 12 16  -10  50 118 1175  22      3   2.000
## 3     Alex Chiasson  RW 73 22 16   -1  32 123 1239  28      2   2.150
## 4      Alex Iafallo   C 82 15 18  -17  22 148 1380  25      2   2.425
## 5 Alexander Kerfoot   C 78 15 27   -9  38 116 1161  24      4   3.500
## 6        Anders Lee   C 82 28 23   20  58 204 1401  28      7   7.000
##   is.centre
## 1         0
## 2         0
## 3         0
## 4         1
## 5         1
## 6         1
```

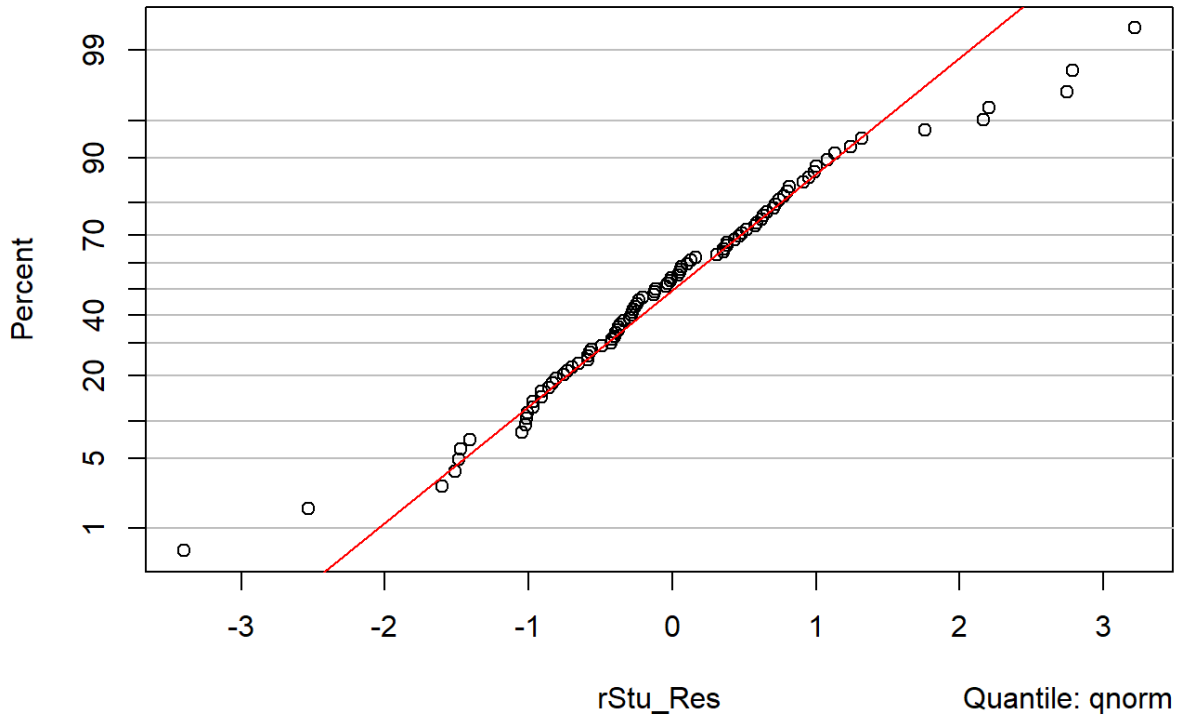
Initial Fitting of the model

```
y  = NHLmergedfin$CAP.HIT
x1 = NHLmergedfin$G
x2 = NHLmergedfin$A
x3 = NHLmergedfin$S
x4 = NHLmergedfin$AGE
x5 = NHLmergedfin$X...
x6 = NHLmergedfin$TOI
x7 = NHLmergedfin$GP
x8 = NHLmergedfin$is.centre
x9 = NHLmergedfin$PIM
x10 = NHLmergedfin$LENGTH


full.NHL.lm = lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10)
summary(full.NHL.lm)
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.56015 -0.48545 -0.09838  0.47051  2.44167
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.787031   1.169536  -0.673  0.50292
## x1           0.066429   0.020382   3.259  0.00164 **
## x2           0.068535   0.013477   5.085 2.37e-06 ***
## x3           0.009488   0.004361   2.176  0.03252 *
## x4           0.049936   0.029109   1.715  0.09013 .
## x5          -0.010831   0.009392  -1.153  0.25226
## x6           0.001119   0.001204   0.930  0.35540
## x7          -0.044524   0.017624  -2.526  0.01350 *
## x8          -0.133934   0.194065  -0.690  0.49210
## x9           0.003384   0.005143   0.658  0.51241
## x10          0.394170   0.061470   6.412 9.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8728 on 80 degrees of freedom
## Multiple R-squared:  0.8896, Adjusted R-squared:  0.8758
## F-statistic: 64.48 on 10 and 80 DF,  p-value: < 2.2e-16
anova(full.NHL.lm)
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## x1         1 359.66  359.66 472.1439 < 2.2e-16 ***
## x2         1  71.01   71.01  93.2207 4.558e-15 ***
## x3         1   8.12    8.12  10.6542  0.001617 **
## x4         1   4.12    4.12   5.4020  0.022651 *
## x5         1   1.69    1.69   2.2128  0.140798
## x6         1   0.04    0.04   0.0542  0.816473
## x7         1  14.58   14.58  19.1386 3.632e-05 ***
```

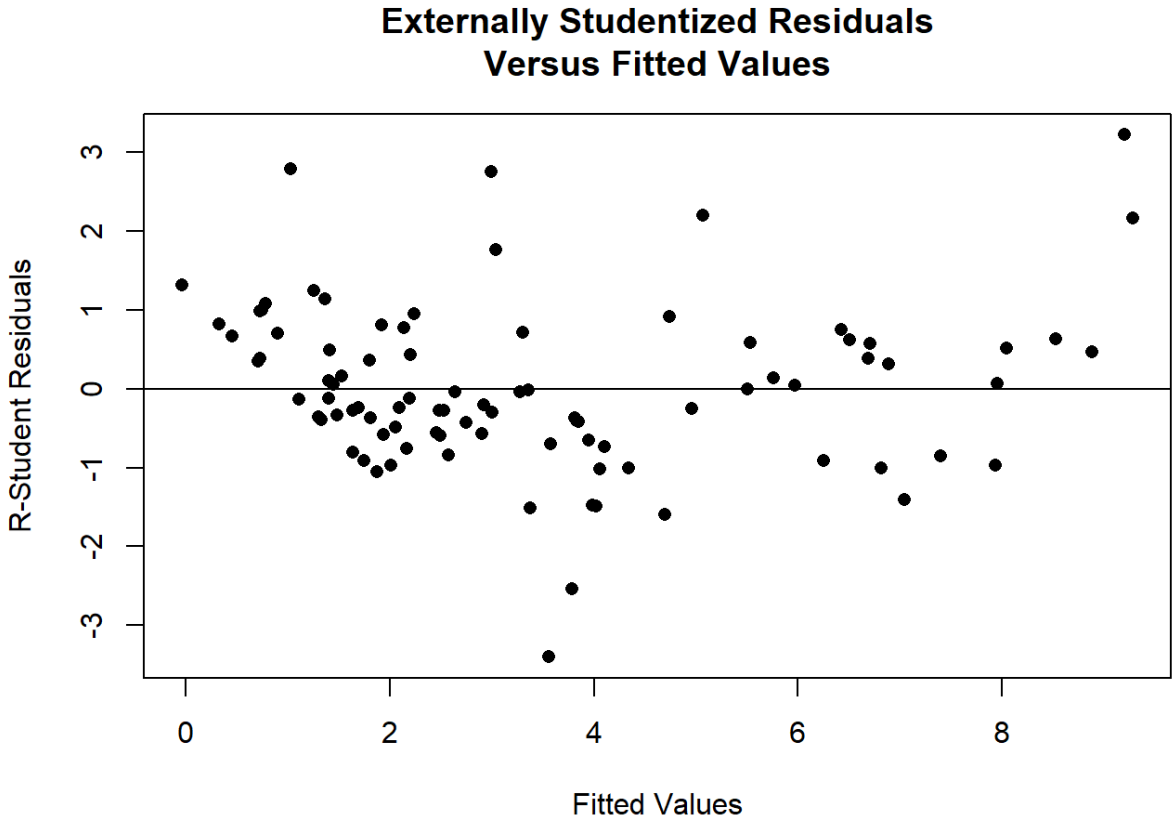
```
## x8          1   0.20    0.20   0.2654  0.607885
## x9          1   0.41    0.41   0.5443  0.462812
## x10         1  31.32   31.32  41.1187 9.311e-09 ***
## Residuals 80  60.94    0.76
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
vif(full.NHL.lm)
##        x1        x2        x3        x4        x5        x6        x7
##  3.912559  3.445137  5.591727  1.241136  1.449791 11.885830  3.680352
##        x8        x9       x10
##  1.124633  1.155142  1.815281
PRESS(full.NHL.lm)
## [1] 85.67509
```

$$\hat{y} = -0.787 + 0.0664x_1 + 0.0685x_2 + 0.0095x_3 + 0.0499x_4 - 0.0108x_5 + 0.00112x_6 - 0.0445x_7 - 0.134x_8 + 0.00338x_9 + 0.394x_{10}$$

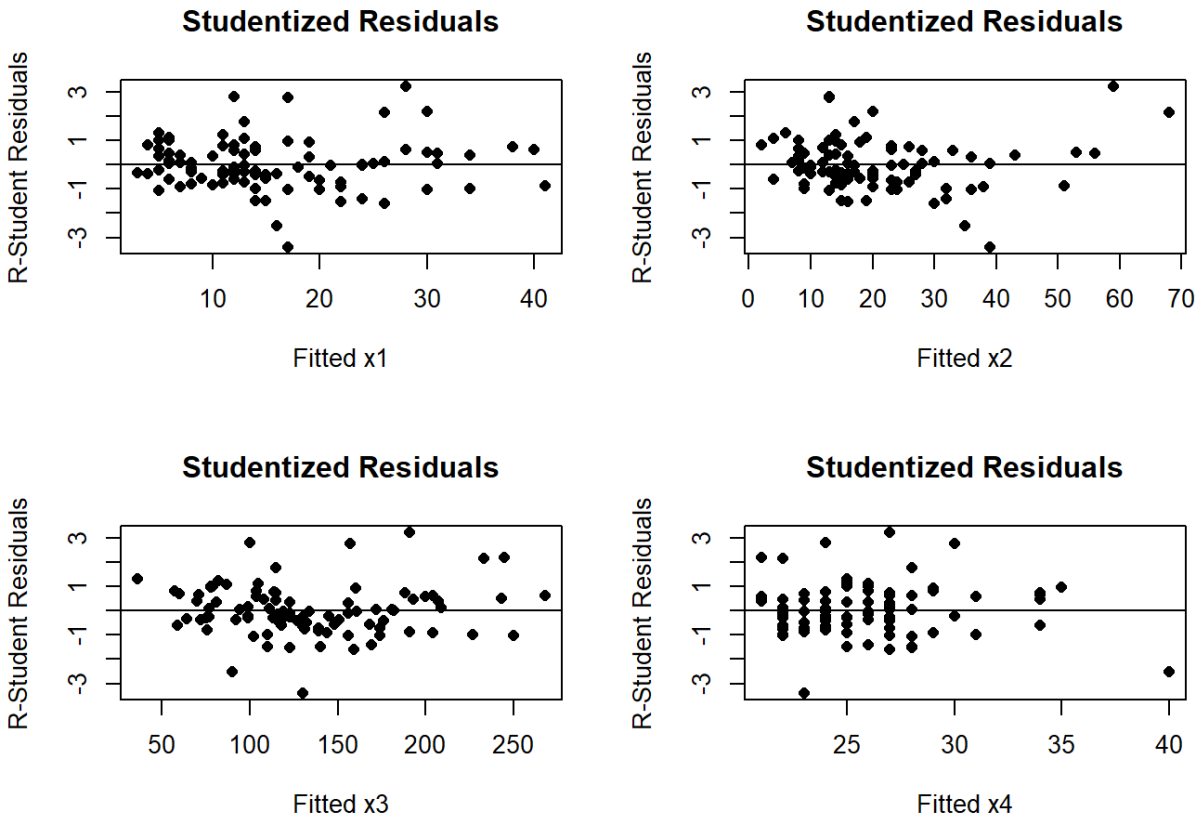
```
rStu_Res = rstudent(full.NHL.lm)
e1071::probplot(rStu_Res, qnorm, xlab='R-Student Residuals', ylab='Percent')
```



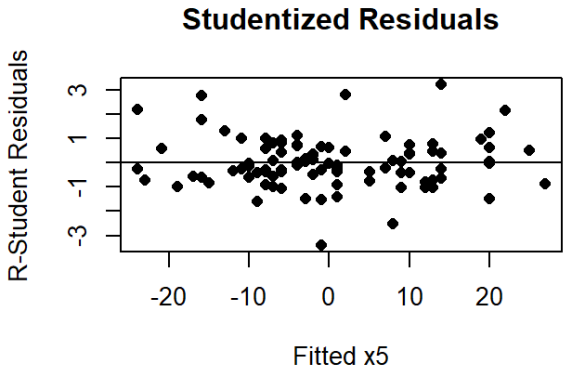
```
yhat = full.NHL.lm$fitted.values
plot(yhat, rStu_Res, ylab='R-Student Residuals', xlab='Fitted Values', main='
Externally Studentized Residuals\nVersus Fitted Values', pch=16)
abline(h=0)
```

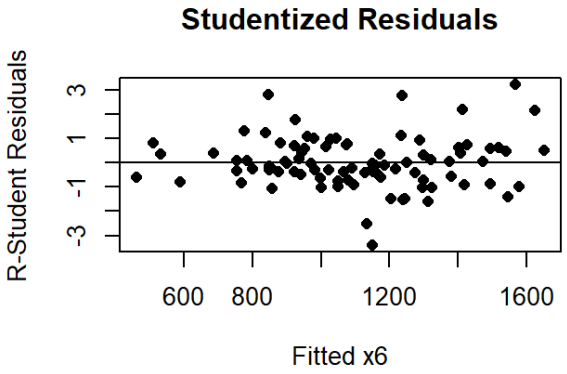


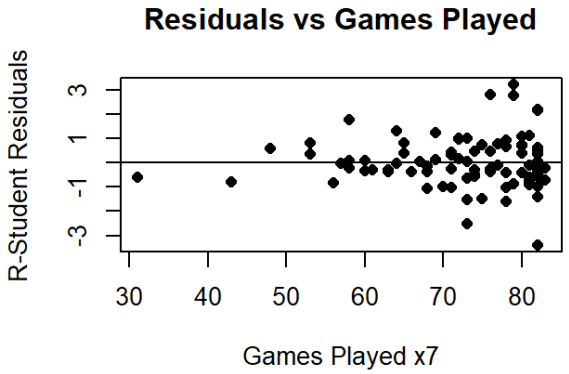
```
par(mfrow=c(2,2))
plot(x1, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x1', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x2, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x2', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x3, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x3', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x4, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x4', main='Studen
tized Residuals', pch=16)
abline(h=0)
```

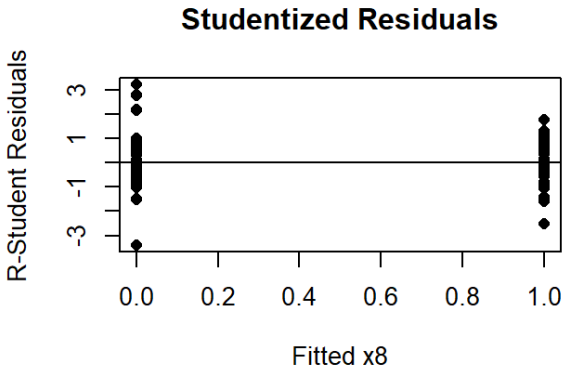


```
plot(x5, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x5', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x6, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x6', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x7, rStu_Res, ylab='R-Student Residuals', xlab='Games Played x7', main='
Residuals vs Games Played', pch=16)
abline(h=0)
plot(x8, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x8', main='Studen
tized Residuals', pch=16)
abline(h=0)
```

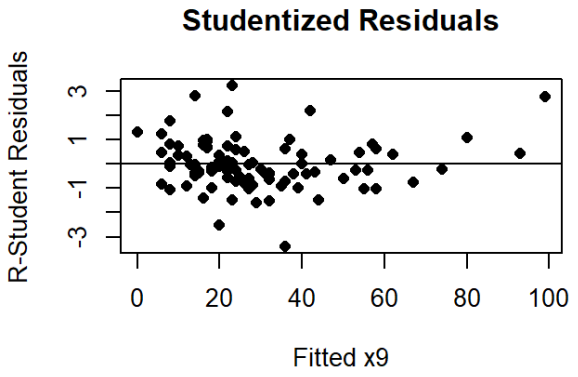


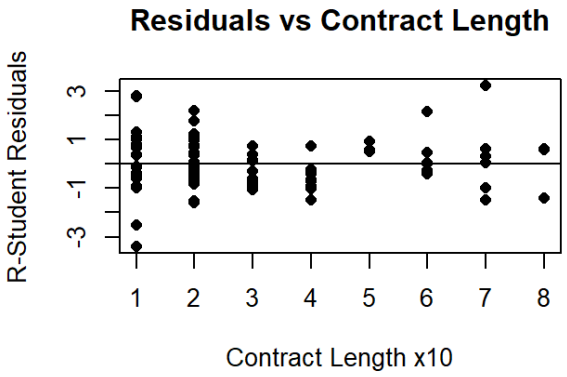


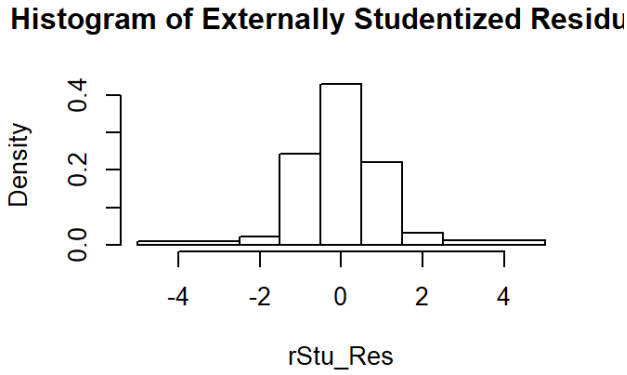




```
plot(x9, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x9', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x10, rStu_Res, ylab='R-Student Residuals', xlab='Contract Length x10', m
ain='Residuals vs Contract Length', pch=16)
abline(h=0)

hist(rStu_Res, main = "Histogram of Externally Studentized Residuals", breaks
=c(-5,-2.5,-1.5,-0.5,0.5,1.5,2.5,5))
median(rStu_Res)
## [1] -0.1179865
```

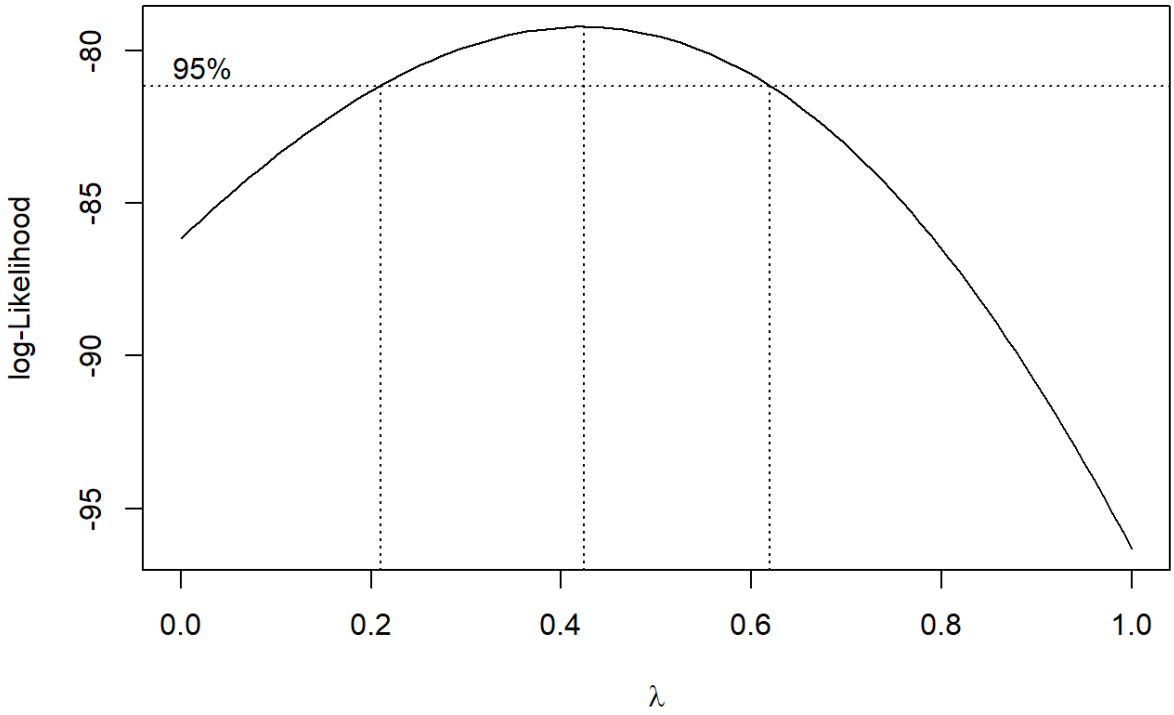






Box Cox Selection and Transformed Model

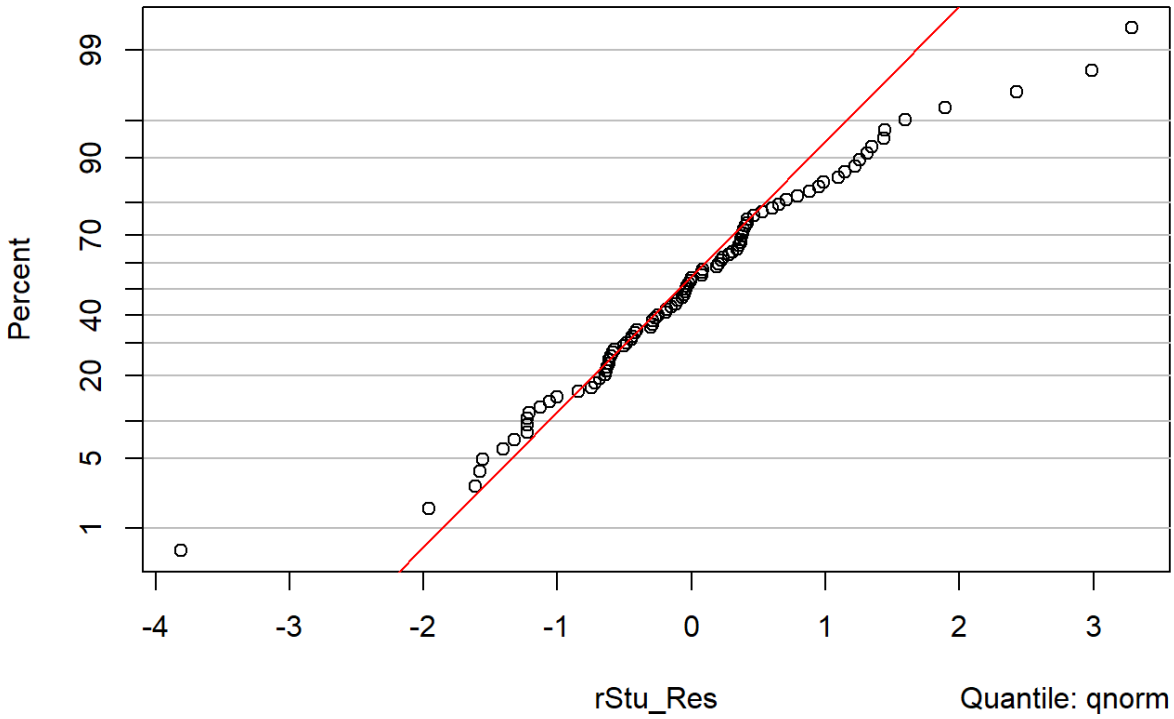
```
boxcox(full.NHL.lm, lambda = seq(0,1,1/10))
```



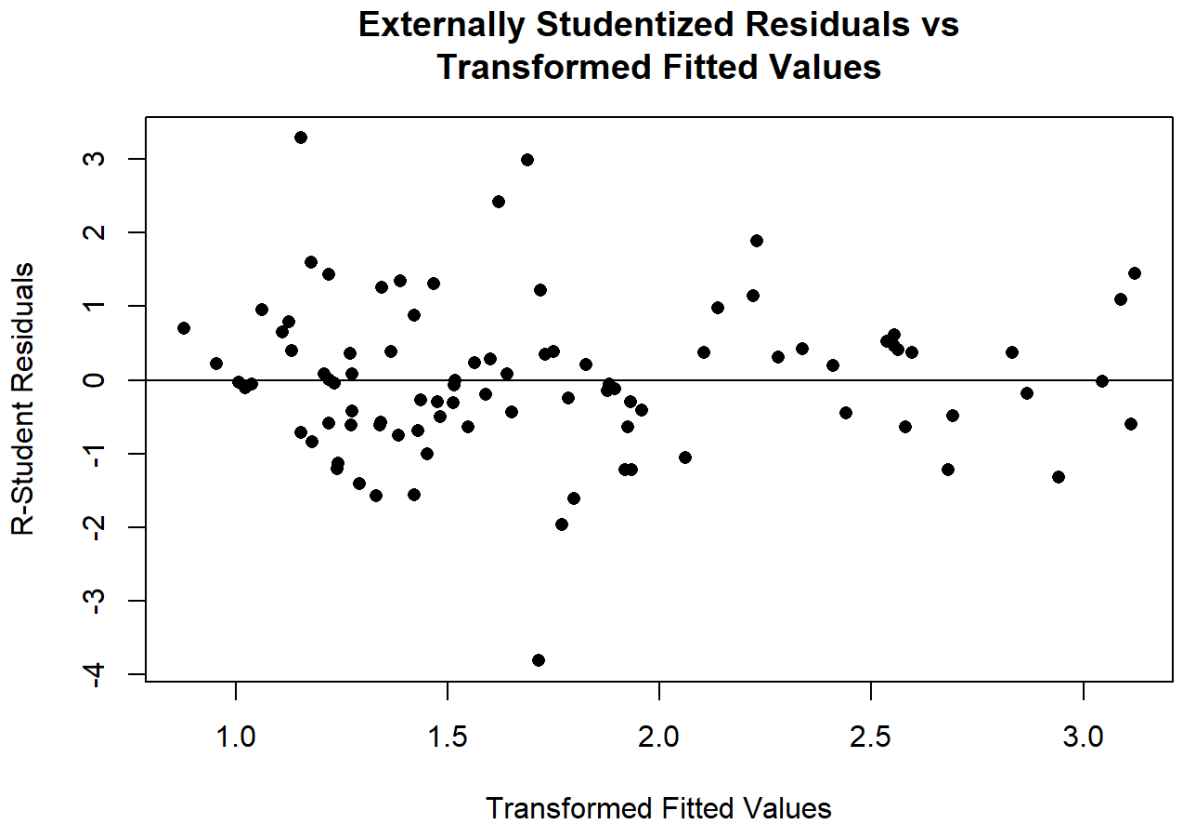

```
yprime = sqrt(y)

full.trans.lm = lm(yprime~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10)
summary(full.trans.lm)
##
## Call:
## lm(formula = yprime ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 +
##     x9 + x10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71348 -0.12029 -0.00825  0.08592  0.64913
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5778671  0.2954380   1.956 0.053960 .
## x1           0.0192812  0.0051488   3.745 0.000339 ***
## x2           0.0126257  0.0034045   3.709 0.000383 ***
## x3           0.0022107  0.0011016   2.007 0.048142 *
## x4           0.0128955  0.0073533   1.754 0.083313 .
## x5          -0.0029303  0.0023726  -1.235 0.220423
## x6           0.0003949  0.0003042   1.298 0.197956
## x7          -0.0104001  0.0044521  -2.336 0.021995 *
```

```
## x8          -0.0557226  0.0490230  -1.137 0.259074
## x9           0.0006955  0.0012992   0.535 0.593910
## x10          0.1021907  0.0155280   6.581 4.47e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2205 on 80 degrees of freedom
## Multiple R-squared:  0.8872, Adjusted R-squared:  0.8731
## F-statistic: 62.92 on 10 and 80 DF,  p-value: < 2.2e-16
anova(full.trans.lm)
## Analysis of Variance Table
##
## Response: yprime
##           Df  Sum Sq Mean Sq  F value    Pr(>F)
## x1         1 23.2031 23.2031 477.3402 < 2.2e-16 ***
## x2         1  3.3109  3.3109  68.1134 2.579e-12 ***
## x3         1  0.6279  0.6279  12.9182 0.0005608 ***
## x4         1  0.2851  0.2851   5.8658 0.0176990 *
## x5         1  0.1133  0.1133   2.3318 0.1306954
## x6         1  0.0134  0.0134   0.2766 0.6003663
## x7         1  0.8651  0.8651  17.7973 6.431e-05 ***
## x8         1  0.0429  0.0429   0.8823 0.3503868
## x9         1  0.0185  0.0185   0.3809 0.5388791
## x10        1  2.1053  2.1053  43.3101 4.473e-09 ***
## Residuals 80  3.8887  0.0486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
vif(full.trans.lm)
##        x1        x2        x3        x4        x5        x6        x7
##  3.912559  3.445137  5.591727  1.241136  1.449791 11.885830  3.680352
##        x8        x9       x10
##  1.124633  1.155142  1.815281
PRESS(full.trans.lm)
## [1] 5.279585
```

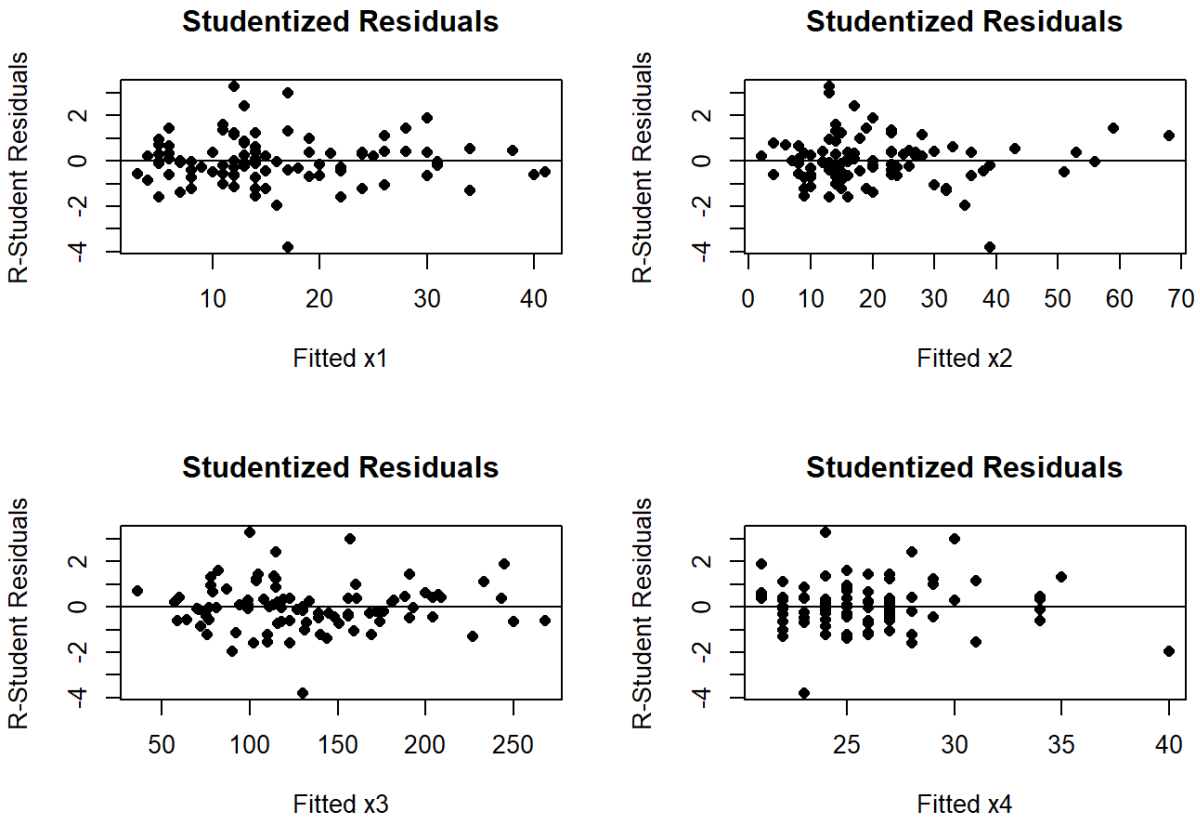
```
rStu_Res = rstudent(full.trans.lm)
e1071::probplot(rStu_Res, qnorm, xlab='R-Student Residuals', ylab='Percent')
```



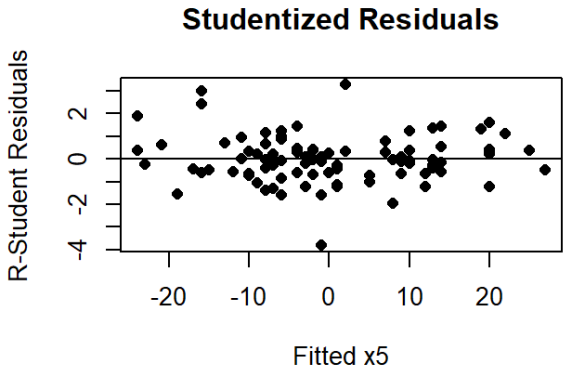
```
yhat = full.trans.lm$fitted.values
plot(yhat, rStu_Res, ylab='R-Student Residuals', xlab='Transformed Fitted Val
ues', main='Externally Studentized Residuals vs\nTransformed Fitted Values',
pch=16)
abline(h=0)
```

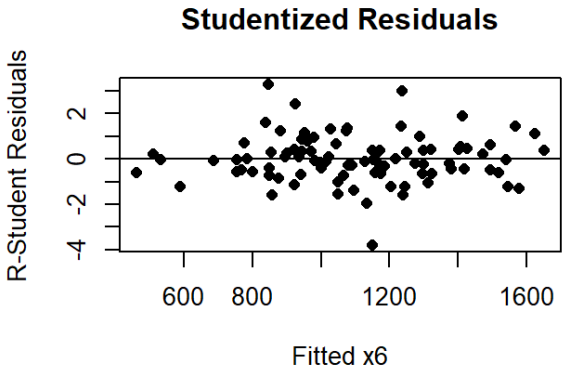


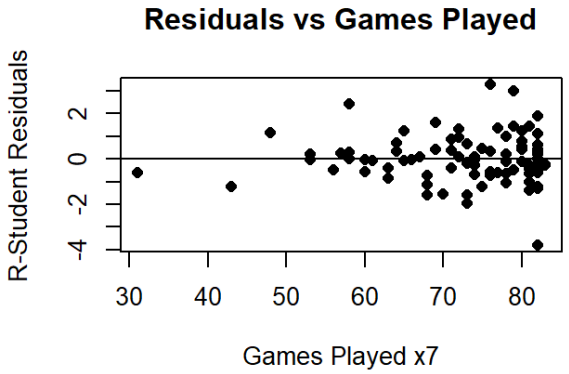
```
par(mfrow=c(2,2))
plot(x1, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x1', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x2, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x2', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x3, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x3', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x4, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x4', main='Studen
tized Residuals', pch=16)
abline(h=0)
```

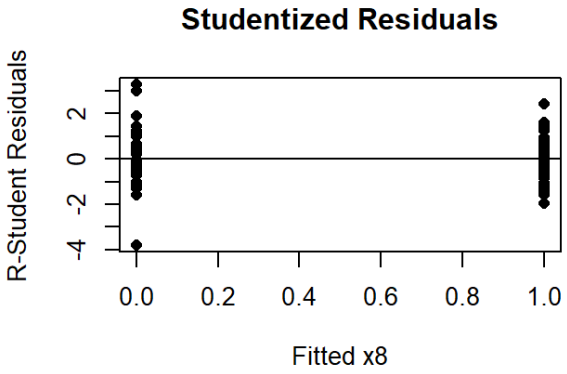


```
plot(x5, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x5', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x6, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x6', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x7, rStu_Res, ylab='R-Student Residuals', xlab='Games Played x7', main='
Residuals vs Games Played', pch=16)
abline(h=0)
plot(x8, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x8', main='Studen
tized Residuals', pch=16)
abline(h=0)
```

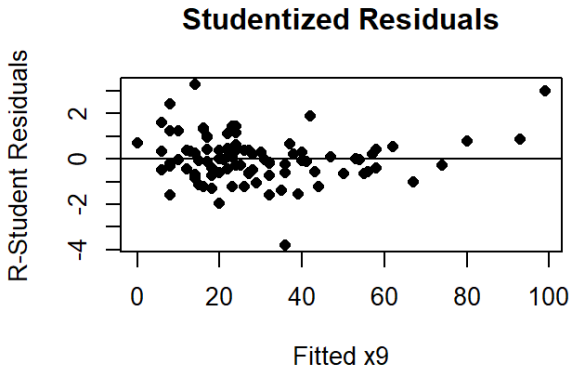


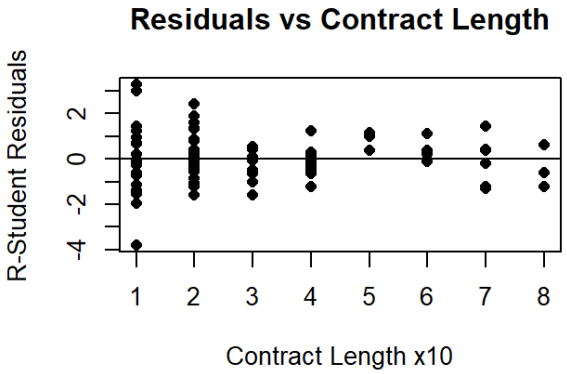


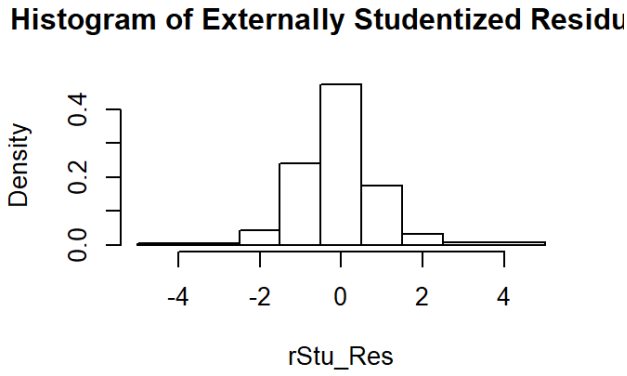




```
plot(x9, rStu_Res, ylab='R-Student Residuals', xlab='Fitted x9', main='Studen
tized Residuals', pch=16)
abline(h=0)
plot(x10, rStu_Res, ylab='R-Student Residuals', xlab='Contract Length x10', m
ain='Residuals vs Contract Length', pch=16)
abline(h=0)


hist(rStu_Res, main = "Histogram of Externally Studentized Residuals", breaks
=c(-5,-2.5,-1.5,-0.5,0.5,1.5,2.5,5))
median(rStu_Res)
## [1] -0.03849216
```







$$\hat{y}=0.578+0.0193x_1+0.0126x_2+0.0022x_3+0.0129x_4-0.0029x_5+0.0004x_6-0.0104x_7-0.056x_8+0.0007x_9+0.102x_{10}$$

Forward/Backward/Stepwise Selection

```
NHL.full.df = cbind(yprime,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
NHL.full.df = as.data.frame(NHL.full.df)
head(NHL.full.df)
##     yprime x1 x2  x3 x4  x5   x6 x7 x8 x9 x10
## 1 1.024695  7 13  70 24  10  686 65  0 40   1
## 2 1.414214 12 16 118 22 -10 1175 81  0 50   3
## 3 1.466288 22 16 123 28  -1 1239 73  0 32   2
## 4 1.557241 15 18 148 25 -17 1380 82  1 22   2
## 5 1.870829 15 27 116 24  -9 1161 78  1 38   4
## 6 2.645751 28 23 204 28  20 1401 82  1 58   7
null.trans.lm = lm(yprime~1, data=NHL.full.df)

forwNHL = stepNHL = step(null.trans.lm, data=NHL.full.df, scope = list(lower=
null.trans.lm, upper=full.trans.lm), direction = "forward")
## Start:  AIC=-86.33
## yprime ~ 1
##
##        Df Sum of Sq    RSS      AIC
## + x1    1   23.2031 11.271 -186.062
## + x3    1   21.9452 12.529 -176.434
## + x2    1   21.8372 12.637 -175.653
## + x6    1   21.4057 13.069 -172.598
## + x10   1   19.9023 14.572 -162.689
## + x7    1    4.4521 30.022  -96.912
## + x5    1    1.7907 32.684  -89.182
## <none>               34.474  -86.328
## + x8    1    0.4823 33.992  -85.610
## + x4    1    0.3655 34.109  -85.298
## + x9    1    0.0923 34.382  -84.572
##
## Step:  AIC=-186.06
## yprime ~ x1
##
##        Df Sum of Sq     RSS     AIC
## + x10   1    4.7460  6.5253 -233.80
## + x2    1    3.3109  7.9604 -215.71
## + x6    1    1.9686  9.3028 -201.53
## + x3    1    1.6767  9.5946 -198.72
## <none>              11.2713 -186.06
## + x8    1    0.0551 11.2162 -184.51
## + x9    1    0.0280 11.2433 -184.29
## + x5    1    0.0250 11.2463 -184.26
## + x7    1    0.0113 11.2601 -184.15
## + x4    1    0.0012 11.2702 -184.07
##
## Step:  AIC=-233.8
## yprime ~ x1 + x10
##
##        Df Sum of Sq    RSS     AIC
## + x2    1   1.49804 5.0273 -255.53
## + x3    1   0.72624 5.7990 -242.54
## + x6    1   0.51556 6.0097 -239.29
## <none>              6.5253 -233.80
## + x5    1   0.12391 6.4014 -233.55
## + x8    1   0.11829 6.4070 -233.47
```

```
## + x7    1   0.04591 6.4794 -232.44
## + x4    1   0.02616 6.4991 -232.17
## + x9    1   0.00324 6.5221 -231.85
##
## Step:  AIC=-255.53
## yprime ~ x1 + x10 + x2
##
##        Df Sum of Sq    RSS     AIC
## + x5    1   0.32490 4.7024 -259.61
## + x3    1   0.31957 4.7077 -259.51
## + x7    1   0.12635 4.9009 -255.85
## <none>              5.0273 -255.53
## + x4    1   0.09708 4.9302 -255.31
## + x8    1   0.07318 4.9541 -254.87
## + x6    1   0.05115 4.9761 -254.47
## + x9    1   0.00476 5.0225 -253.62
##
## Step:  AIC=-259.61
## yprime ~ x1 + x10 + x2 + x5
##
##        Df Sum of Sq    RSS     AIC
## + x3    1  0.197697 4.5047 -261.52
## + x4    1  0.119774 4.5826 -259.96
## + x8    1  0.108980 4.5934 -259.75
## <none>              4.7024 -259.61
## + x7    1  0.097648 4.6047 -259.52
## + x6    1  0.010145 4.6922 -257.81
## + x9    1  0.007219 4.6951 -257.75
##
## Step:  AIC=-261.52
## yprime ~ x1 + x10 + x2 + x5 + x3
##
##        Df Sum of Sq    RSS     AIC
## + x7    1  0.253753 4.2509 -264.80
## + x4    1  0.245580 4.2591 -264.62
## <none>              4.5047 -261.52
## + x8    1  0.065963 4.4387 -260.87
## + x6    1  0.017751 4.4869 -259.88
## + x9    1  0.000004 4.5047 -259.52
##
## Step:  AIC=-264.8
## yprime ~ x1 + x10 + x2 + x5 + x3 + x7
##
##        Df Sum of Sq    RSS     AIC
## + x4    1  0.214562 4.0363 -267.51
## + x6    1  0.129902 4.1210 -265.62
## <none>              4.2509 -264.80
## + x8    1  0.046086 4.2048 -263.79
## + x9    1  0.008194 4.2427 -262.98
##
## Step:  AIC=-267.51
## yprime ~ x1 + x10 + x2 + x5 + x3 + x7 + x4
##
##        Df Sum of Sq    RSS     AIC
## <none>              4.0363 -267.51
## + x6    1  0.057675 3.9787 -266.82
## + x8    1  0.057401 3.9789 -266.82
```

```
## + x9    1  0.018183 4.0182 -265.92
backNHL = step(full.trans.lm, data=NHL.full.df, direction = "backward")
## Start:  AIC=-264.9
## yprime ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10
##
##        Df Sum of Sq    RSS     AIC
## - x9    1   0.01393 3.9027 -266.58
## - x8    1   0.06280 3.9515 -265.44
## - x5    1   0.07415 3.9629 -265.18
## - x6    1   0.08192 3.9707 -265.00
## <none>              3.8887 -264.90
## - x4    1   0.14949 4.0382 -263.47
## - x3    1   0.19577 4.0845 -262.43
## - x7    1   0.26526 4.1540 -260.90
## - x2    1   0.66854 4.5573 -252.47
## - x1    1   0.68167 4.5704 -252.20
## - x10   1   2.10527 5.9940 -227.53
##
## Step:  AIC=-266.58
## yprime ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x10
##
##        Df Sum of Sq    RSS     AIC
## - x5    1   0.07547 3.9781 -266.83
## - x8    1   0.07601 3.9787 -266.82
## - x6    1   0.07628 3.9789 -266.82
## <none>              3.9027 -266.58
## - x4    1   0.14484 4.0475 -265.26
## - x3    1   0.20747 4.1101 -263.86
## - x7    1   0.25169 4.1544 -262.89
## - x2    1   0.65678 4.5595 -254.42
## - x1    1   0.69294 4.5956 -253.70
## - x10   1   2.10985 6.0125 -229.25
##
## Step:  AIC=-266.83
## yprime ~ x1 + x2 + x3 + x4 + x6 + x7 + x8 + x10
##
##        Df Sum of Sq    RSS     AIC
## - x8    1   0.06289 4.0410 -267.41
## <none>              3.9781 -266.83
## - x4    1   0.12539 4.1035 -266.01
## - x6    1   0.15549 4.1336 -265.35
## - x3    1   0.23288 4.2110 -263.66
## - x7    1   0.42101 4.3991 -259.68
## - x2    1   0.58171 4.5598 -256.42
## - x1    1   0.61977 4.5979 -255.66
## - x10   1   2.03438 6.0125 -231.25
##
## Step:  AIC=-267.41
## yprime ~ x1 + x2 + x3 + x4 + x6 + x7 + x10
##
##        Df Sum of Sq    RSS     AIC
## <none>              4.0410 -267.41
## - x4    1   0.12498 4.1660 -266.63
## - x6    1   0.12554 4.1666 -266.62
## - x3    1   0.29917 4.3402 -262.91
## - x7    1   0.39534 4.4364 -260.91
## - x1    1   0.61569 4.6567 -256.50
```

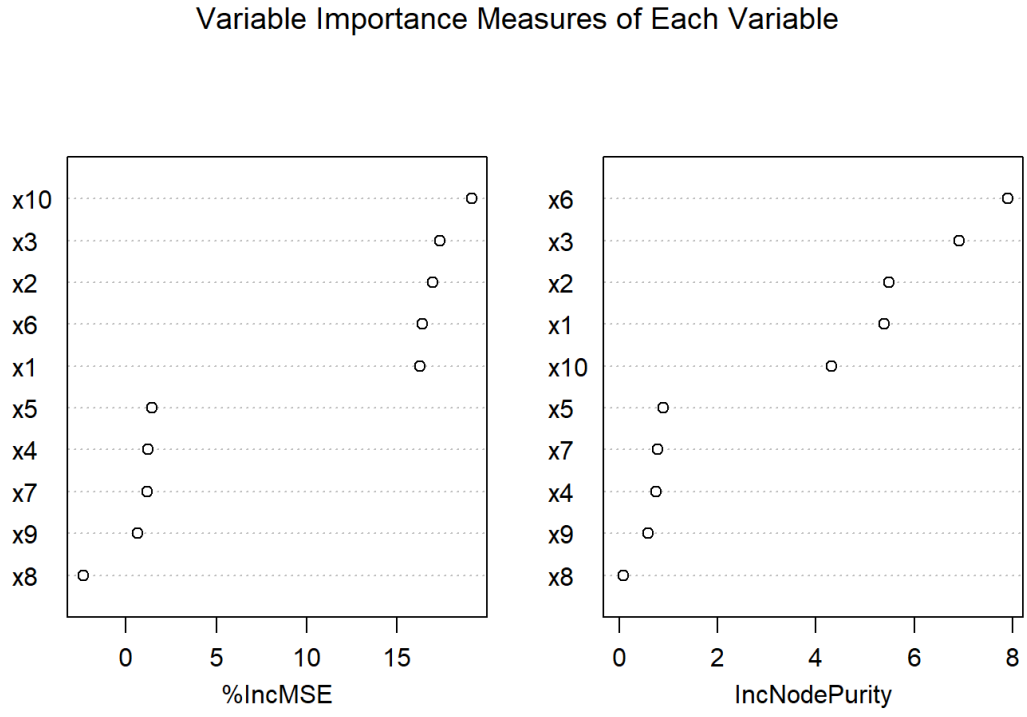
```
## - x2    1   0.63018 4.6712 -256.22
## - x10   1   2.01467 6.0557 -232.60
stepNHL = step(null.trans.lm, data=NHL.full.df, scope = list(upper=full.trans
.lm), direction = "both")
## Start:  AIC=-86.33
## yprime ~ 1
##
##        Df Sum of Sq    RSS      AIC
## + x1    1   23.2031 11.271 -186.062
## + x3    1   21.9452 12.529 -176.434
## + x2    1   21.8372 12.637 -175.653
## + x6    1   21.4057 13.069 -172.598
## + x10   1   19.9023 14.572 -162.689
## + x7    1    4.4521 30.022  -96.912
## + x5    1    1.7907 32.684  -89.182
## <none>               34.474  -86.328
## + x8    1    0.4823 33.992  -85.610
## + x4    1    0.3655 34.109  -85.298
## + x9    1    0.0923 34.382  -84.572
##
## Step:  AIC=-186.06
## yprime ~ x1
##
##        Df Sum of Sq    RSS      AIC
## + x10   1    4.7460  6.525 -233.801
## + x2    1    3.3109  7.960 -215.711
## + x6    1    1.9686  9.303 -201.530
## + x3    1    1.6767  9.595 -198.719
## <none>              11.271 -186.062
## + x8    1    0.0551 11.216 -184.509
## + x9    1    0.0280 11.243 -184.289
## + x5    1    0.0250 11.246 -184.265
## + x7    1    0.0113 11.260 -184.153
## + x4    1    0.0012 11.270 -184.072
## - x1    1   23.2031 34.474  -86.328
##
## Step:  AIC=-233.8
## yprime ~ x1 + x10
##
##        Df Sum of Sq     RSS     AIC
## + x2    1    1.4980  5.0273 -255.53
## + x3    1    0.7262  5.7990 -242.54
## + x6    1    0.5156  6.0097 -239.29
## <none>               6.5253 -233.80
## + x5    1    0.1239  6.4014 -233.55
## + x8    1    0.1183  6.4070 -233.47
## + x7    1    0.0459  6.4794 -232.44
## + x4    1    0.0262  6.4991 -232.17
## + x9    1    0.0032  6.5221 -231.85
## - x10   1    4.7460 11.2713 -186.06
## - x1    1    8.0469 14.5722 -162.69
##
## Step:  AIC=-255.53
## yprime ~ x1 + x10 + x2
##
##        Df Sum of Sq    RSS     AIC
## + x5    1   0.32490 4.7024 -259.61
```

```
## + x3    1   0.31957 4.7077 -259.51
## + x7    1   0.12635 4.9009 -255.85
## <none>              5.0273 -255.53
## + x4    1   0.09708 4.9302 -255.31
## + x8    1   0.07318 4.9541 -254.87
## + x6    1   0.05115 4.9761 -254.47
## + x9    1   0.00476 5.0225 -253.62
## - x2    1   1.49804 6.5253 -233.80
## - x1    1   2.69424 7.7215 -218.48
## - x10   1   2.93314 7.9604 -215.71
##
## Step:  AIC=-259.61
## yprime ~ x1 + x10 + x2 + x5
##
##        Df Sum of Sq    RSS     AIC
## + x3    1   0.19770 4.5047 -261.52
## + x4    1   0.11977 4.5826 -259.96
## + x8    1   0.10898 4.5934 -259.75
## <none>              4.7024 -259.61
## + x7    1   0.09765 4.6047 -259.52
## + x6    1   0.01014 4.6922 -257.81
## + x9    1   0.00722 4.6951 -257.75
## - x5    1   0.32490 5.0273 -255.53
## - x2    1   1.69902 6.4014 -233.55
## - x1    1   2.84071 7.5431 -218.61
## - x10   1   2.99627 7.6986 -216.75
##
## Step:  AIC=-261.52
## yprime ~ x1 + x10 + x2 + x5 + x3
##
##        Df Sum of Sq    RSS     AIC
## + x7    1   0.25375 4.2509 -264.80
## + x4    1   0.24558 4.2591 -264.62
## <none>              4.5047 -261.52
## + x8    1   0.06596 4.4387 -260.87
## + x6    1   0.01775 4.4869 -259.88
## - x3    1   0.19770 4.7024 -259.61
## + x9    1   0.00000 4.5047 -259.52
## - x5    1   0.20303 4.7077 -259.51
## - x1    1   1.02969 5.5343 -244.79
## - x2    1   1.25359 5.7582 -241.18
## - x10   1   2.71142 7.2161 -220.64
##
## Step:  AIC=-264.8
## yprime ~ x1 + x10 + x2 + x5 + x3 + x7
##
##        Df Sum of Sq    RSS     AIC
## + x4    1   0.21456 4.0363 -267.51
## + x6    1   0.12990 4.1210 -265.62
## <none>              4.2509 -264.80
## - x5    1   0.12976 4.3807 -264.06
## + x8    1   0.04609 4.2048 -263.79
## + x9    1   0.00819 4.2427 -262.98
## - x7    1   0.25375 4.5047 -261.52
## - x3    1   0.35380 4.6047 -259.52
## - x1    1   1.00239 5.2533 -247.53
## - x2    1   1.24467 5.4956 -243.43
```

```
## - x10   1   2.63273 6.8836 -222.94
##
## Step:  AIC=-267.51
## yprime ~ x1 + x10 + x2 + x5 + x3 + x7 + x4
##
##        Df Sum of Sq    RSS     AIC
## <none>              4.0363 -267.51
## + x6    1   0.05768 3.9787 -266.82
## + x8    1   0.05740 3.9789 -266.82
## - x5    1   0.13022 4.1666 -266.62
## + x9    1   0.01818 4.0182 -265.92
## - x4    1   0.21456 4.2509 -264.80
## - x7    1   0.22274 4.2591 -264.62
## - x3    1   0.47872 4.5151 -259.31
## - x1    1   0.80626 4.8426 -252.94
## - x2    1   1.28773 5.3241 -244.31
## - x10   1   2.60370 6.6401 -224.21
```

Random Forests

```
set.seed(350)
NHLrf = randomForest(yprime ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, importance = TR
UE)

NHLrf
##
## Call:
##  randomForest(formula = yprime ~ x1 + x2 + x3 + x4 + x5 + x6 +      x7 + x
8 + x9 + x10, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.07323807
##                     % Var explained: 80.67
head(NHLrf$predicted)
##        1        2        3        4        5        6
## 1.180301 1.525715 1.742995 1.962161 1.848168 2.594620
head(NHLrf$rsq)
## [1] 0.4549282 0.4906070 0.5994709 0.6623834 0.7026964 0.7119744
head(NHLrf$mse)
## [1] 0.2064951 0.1929786 0.1517365 0.1279028 0.1126306 0.1091157
mean((NHLrf$predicted-yprime)^2)
## [1] 0.07323807
mean(NHLrf$mse)
## [1] 0.07721765
mean(NHLrf$rsq)
## [1] 0.7961736
varImpPlot(NHLrf, main="Variable Importance Measures of Each Variable")
```



All Possible Subset Regression

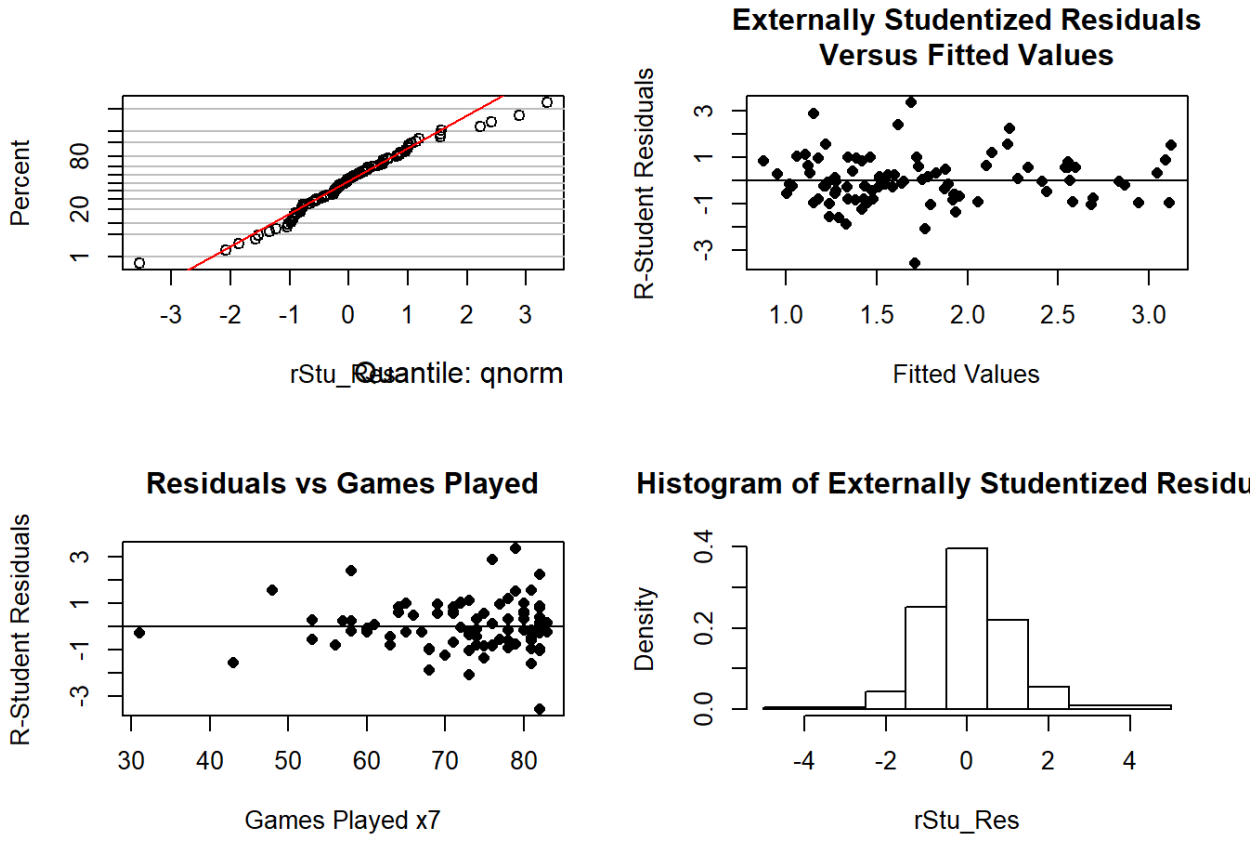
```
m      = lm(yprime~x1+x2+x3+x10)
m1_4   = lm(yprime~x1+x2+x3+x4+x10)
m1_6   = lm(yprime~x1+x2+x3+x6+x10)
m1_7   = lm(yprime~x1+x2+x3+x7+x10)
m2_46  = lm(yprime~x1+x2+x3+x4+x6+x10)
m2_47  = lm(yprime~x1+x2+x3+x4+x7+x10)
m2_67  = lm(yprime~x1+x2+x3+x6+x7+x10)
m3_467 = lm(yprime~x1+x2+x3+x4+x6+x7+x10)
regM_list = list(m=m, m1_4=m1_4, m1_6=m1_6, m1_7=m1_7, m2_46=m2_46, m2_47=m2_
47, m2_67=m2_67, m3_467=m3_467)
SSres   = rep(NA, length(regM_list))
Rsq     = rep(NA, length(regM_list))
Rsq_adj = rep(NA, length(regM_list))
MSres   = rep(NA, length(regM_list))
cp      = rep(NA, length(regM_list))
PRESS   = rep(NA, length(regM_list))
for (i in 1:length(regM_list))
{
  mdl = regM_list[[i]]
  SSres[i]   = anova(mdl)$`Sum Sq`[length(anova(mdl)$`Sum Sq`)]
  Rsq[i]     = summary(mdl)$r.squared
  Rsq_adj[i] = summary(mdl)$adj.r.squared
  MSres[i]   = anova(mdl)$`Mean Sq`[length(anova(mdl)$`Sum Sq`)]
  cp[i]      = Cp(mdl, S2=summary(full.trans.lm)$sigma^2)
  PRESS[i]   = PRESS(mdl)
}
num_reg = c(4,5,5,5,6,6,6,7)
p       = c(5,6,6,6,7,7,7,8)
regr    = c('None', 'x4', 'x6', 'x7', 'x4,x6', 'x4,x7', 'x6,x7', 'x4,x6,x7')
mdl_sel_data = data.frame(cbind(num_reg, p, regr, round(SSres,5), round(Rsq,5
), round(Rsq_adj,5), round(MSres,5), round(cp,5), round(PRESS,5)))
names(mdl_sel_data) = c("# Regs", "p", "Regressors", "SSres", "Rsq", "Rsq_adj
", "MSres", "Cp", "PRESS")
mdl_sel_data
##   # Regs p Regressors   SSres     Rsq Rsq_adj   MSres       Cp   PRESS
## 1      4 5       None 4.70769 0.86344 0.85709 0.05474 15.84768 5.38317
## 2      5 6         x4 4.45773 0.87069 0.86309 0.05244 12.70547   5.338
## 3      5 6         x6 4.70196 0.86361 0.85559 0.05532  17.7298 5.48098
## 4      5 6         x7 4.38067 0.87293 0.86546 0.05154 11.12017 5.12693
## 5      6 7      x4,x6 4.43637 0.87131 0.86212 0.05281 14.26602 5.40119
## 6      6 7      x4,x7 4.16657 0.87914 0.87051  0.0496   8.7156 5.07512
## 7      6 7      x6,x7 4.16601 0.87916 0.87052  0.0496  8.70421 4.99407
## 8      7 8   x4,x6,x7 4.04103 0.88278  0.8729 0.04869  8.13304 5.02106
vif(m2_47)
##       x1       x2       x3       x4       x7      x10
## 3.541912 2.306068 4.048849 1.139699 1.444963 1.600799
vif(m2_67)
##       x1       x2       x3       x6       x7      x10
## 3.531177 2.862633 4.627263 9.349973 2.868712 1.742900
vif(m3_467)
##        x1        x2        x3        x4        x6        x7       x10
##  3.608250  2.937499  5.257259  1.223972 10.041344  3.035665  1.748973
```

Final Model

```
last.nhl.lm = lm(yprime~x1+x2+x3+x4+x7+x10)
summary(last.nhl.lm)
##
## Call:
## lm(formula = yprime ~ x1 + x2 + x3 + x4 + x7 + x10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69557 -0.14947 -0.02009  0.12446  0.67792
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5083102  0.2798946   1.816 0.072926 .
## x1           0.0186747  0.0049486   3.774 0.000299 ***
## x2           0.0136612  0.0028137   4.855 5.49e-06 ***
## x3           0.0034728  0.0009469   3.668 0.000428 ***
## x4           0.0147884  0.0071180   2.078 0.040800 *
## x7          -0.0068274  0.0028180  -2.423 0.017550 *
## x10          0.1052544  0.0147300   7.146 3.02e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2227 on 84 degrees of freedom
## Multiple R-squared:  0.8791, Adjusted R-squared:  0.8705
## F-statistic: 101.8 on 6 and 84 DF,  p-value: < 2.2e-16
anova(last.nhl.lm)
## Analysis of Variance Table
##
## Response: yprime
##           Df  Sum Sq Mean Sq  F value    Pr(>F)
## x1         1 23.2031 23.2031 467.7862 < 2.2e-16 ***
## x2         1  3.3109  3.3109  66.7501 2.768e-12 ***
## x3         1  0.6279  0.6279  12.6597 0.0006174 ***
## x4         1  0.2851  0.2851   5.7484 0.0187202 *
## x7         1  0.3481  0.3481   7.0182 0.0096377 **
## x10        1  2.5326  2.5326  51.0591 3.017e-10 ***
## Residuals 84  4.1666  0.0496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$$\hat{y}=0.508+0.0187x_1+0.0137x_2+0.0035x_3+0.0148x_4-0.0068x_7+0.105x_{10}$$

```
par(mfrow=c(2,2))

rStu_Res = rstudent(last.nhl.lm)

e1071::probplot(rStu_Res, qnorm, xlab='R-Student Residuals', ylab='Percent')
#mostly straiht but there are outliers

yhat = full.trans.lm$fitted.values

plot(yhat, rStu_Res, ylab='R-Student Residuals', xlab='Fitted Values', main='
Externally Studentized Residuals\nVersus Fitted Values', pch=16)

abline(h=0)

plot(x7, rStu_Res, ylab='R-Student Residuals', xlab='Games Played x7', main='
Residuals vs Games Played', pch=16)

abline(h=0)

hist(rStu_Res, main = "Histogram of Externally Studentized Residuals", breaks
=c(-5,-2.5,-1.5,-0.5,0.5,1.5,2.5,5))
```




```
median(rStu_Res)
```

```
## [1] -0.09160828
```

Influential Observations Analysis

```
 (p = length(last.nhl.lm$coefficients))
## [1] 7
(n = nrow(NHLmergedfin))
## [1] 91
infM   = influence.measures(last.nhl.lm)
infMat = data.frame(infM$infmat)

h            = infMat$hat
D            = infMat$cook.d
DFFITS       = infMat$dffit
DFBETA_INCPT = infMat$dfb.1_
DFBETA_x1    = infMat$dfb.x1
DFBETA_x2    = infMat$dfb.x2
DFBETA_x3    = infMat$dfb.x3
DFBETA_x4    = infMat$dfb.x4
DFBETA_x7    = infMat$dfb.x7
DFBETA_x10   = infMat$dfb.x10
COVRATIO     = infMat$cov.r

(infM_DLdata = data.frame(cbind(h=h, D=D, DFFITS=DFFITS, DFBETA_INCPT=DFBETA_
INCPT, DFBETA_x1=DFBETA_x1, DFBETA_x2=DFBETA_x2, DFBETA_x3=DFBETA_x3, DFBETA_
x4=DFBETA_x4, DFBETA_x7=DFBETA_x7, DFBETA_x10=DFBETA_x10, COVRATIO=COVRATIO))
)
##              h             D       DFFITS  DFBETA_INCPT     DFBETA_x1
## 1   0.04281919 3.559590e-04 -0.049635508 -0.0291212225 -0.0052269988
## 2   0.04332593 1.637141e-04 -0.033655561 -0.0044233085  0.0012526916
## 3   0.04999528 8.061453e-03 -0.237653854  0.0129543562 -0.1860487883
## 4   0.03073337 1.232881e-05 -0.009234557  0.0032263803  0.0018842280
## 5   0.03957318 5.595940e-04  0.062248779  0.0048451893  0.0019608202
## 6   0.09916699 2.716241e-07  0.001370669 -0.0005379852  0.0001596256
## 7   0.03678157 4.168154e-02  0.563156527  0.1011135777  0.1560056317
## 8   0.02851276 5.723883e-04 -0.062971926 -0.0256739365 -0.0362619047
## 9   0.03085403 4.215317e-03  0.171700992  0.0712205123  0.0603708494
## 10  0.06069608 6.329721e-04 -0.066193910 -0.0078774502 -0.0153998277
## 11  0.05109527 4.636590e-03 -0.179726271 -0.0670796273 -0.1034293732
## 12  0.14601010 5.719175e-02  0.637902306 -0.1115102571 -0.0846629271
## 13  0.08002235 8.106400e-04 -0.074908507  0.0200363524  0.0522028686
## 14  0.07667176 2.256109e-04 -0.039507328  0.0128301539  0.0050419098
## 15  0.21196005 2.284872e-02 -0.398953295 -0.0857173015 -0.2779793383
## 16  0.03490510 1.718968e-03 -0.109255695  0.0390650074 -0.0553340094
## 17  0.06278487 8.384399e-04  0.076192283  0.0499779353  0.0056545468
## 18  0.07782965 4.073567e-03  0.168194199  0.0734827633  0.0150053799
## 19  0.03736873 8.988722e-04  0.078925403 -0.0270224483 -0.0310141427
## 20  0.04639602 8.819801e-06 -0.007810541  0.0034356788 -0.0008919545
## 21  0.08239801 7.882239e-04  0.073863882 -0.0014575401 -0.0404408922
## 22  0.07367426 5.030987e-03  0.187034928 -0.0064891136  0.0866038035
## 23  0.06862159 7.219642e-03  0.224381337  0.1032369590  0.0712291073
## 24  0.16578978 1.892194e-02  0.363212702 -0.0132494233 -0.2360198029
## 25  0.07194058 4.820612e-03  0.183074455  0.0739028522 -0.0236562997
## 26  0.03365070 5.189687e-03  0.190648215 -0.0038612784  0.0089705412
## 27  0.11199826 3.195237e-02 -0.475152936 -0.0215067748 -0.0516398951
## 28  0.25108585 3.428182e-03 -0.154051327 -0.0464894131 -0.0026416824
## 29  0.03275901 4.437188e-03  0.176151407  0.0093413430 -0.0467580056
## 30  0.04642157 4.414406e-03 -0.175400858 -0.0980488791  0.0520877261
## 31  0.05244866 1.181895e-02 -0.288493819  0.1109748602 -0.0572066177
```

```
## 32 0.09612904 2.657929e-03 -0.135729050  0.0166565830  0.0951786040
## 33 0.07044637 6.556736e-03 -0.213729039  0.0257079899 -0.0536312801
## 34 0.08050104 2.654502e-03 -0.135671723  0.0785113772  0.0636230999
## 35 0.13048791 2.297635e-03  0.126143876  0.0223493531  0.0700652123
## 36 0.03393930 1.267805e-03  0.093783974  0.0587954831  0.0335000252
## 37 0.04963845 1.480645e-05  0.010119964  0.0009228966  0.0056985920
## 38 0.03210794 2.334494e-04 -0.040195016 -0.0182883058  0.0180359626
## 39 0.23208138 3.741992e-02 -0.511389739  0.0969304879 -0.1470187397
## 40 0.20252798 1.175471e-02  0.285689004 -0.0960996706  0.1807131059
## 41 0.29182361 2.445203e-01 -1.333884067  0.7695033251  0.0166135546
## 42 0.06598184 6.247870e-04  0.065761924  0.0297135659 -0.0038383310
## 43 0.07026545 3.816707e-04 -0.051390663 -0.0379636449  0.0106543102
## 44 0.03400538 6.540675e-03  0.214362199 -0.0037791276 -0.0234264253
## 45 0.03706455 5.440728e-03  0.195141621 -0.0794975564 -0.0182800536
## 46 0.05091850 1.083103e-02  0.276037153 -0.1500565140 -0.0283570884
## 47 0.02910260 7.561570e-04 -0.072395469 -0.0266689311 -0.0025866719
## 48 0.02937022 4.112033e-03 -0.169609367 -0.0546746156 -0.0735690558
## 49 0.05133278 5.140043e-03 -0.189303024  0.0357126583  0.0285351925
## 50 0.05788238 5.049948e-04 -0.059120847 -0.0358080156  0.0159515869
## 51 0.02987185 1.633067e-03 -0.106515368 -0.0292599009  0.0002604476
## 52 0.06696283 3.122114e-04  0.046478442 -0.0087078110 -0.0292872566
## 53 0.07604068 3.982604e-03  0.166306577 -0.0108172482 -0.0467776296
## 54 0.11650766 2.079878e-01 -1.286962080 -0.0424364712  0.0004191197
## 55 0.09098723 1.248375e-02 -0.295385850 -0.0346142302 -0.1134445278
## 56 0.04376009 6.063143e-03 -0.205924571 -0.0130958118  0.0551948994
## 57 0.04295127 3.513200e-02  0.509857291  0.1776493349 -0.0091569366
## 58 0.03376305 1.500864e-03 -0.102069901  0.0220246405  0.0041106019
## 59 0.14643591 5.794953e-02  0.642204672  0.1288076086 -0.1253147206
## 60 0.08895082 5.281465e-04 -0.060453788  0.0029751501 -0.0265426534
## 61 0.10488607 5.079193e-03  0.187772325  0.0558799447  0.0954361464
## 62 0.06288669 2.318364e-02  0.406333421 -0.1408311811 -0.2070762144
## 63 0.02309707 1.495532e-03 -0.101975045  0.0148352190  0.0358616274
## 64 0.12795881 2.190434e-03  0.123163996  0.0382175610  0.0209292751
## 65 0.22221997 3.109872e-02  0.465906050 -0.0061041508 -0.1949399530
## 66 0.11263533 4.504041e-04 -0.055823080  0.0398872247  0.0119748425
## 67 0.08015958 3.049103e-02 -0.466079511  0.1416694997  0.3335326521
## 68 0.03763833 4.127511e-06 -0.005343114  0.0010348269  0.0025338179
## 69 0.02553463 3.617168e-05 -0.015818225 -0.0020644771 -0.0021981447
## 70 0.15581135 1.242692e-01  0.954264922  0.0881493052  0.1622430384
## 71 0.06125599 3.296178e-03  0.151310739  0.0960493750  0.1139519032
## 72 0.05794854 5.144491e-05  0.018864042  0.0161938280  0.0064976308
## 73 0.11388841 3.473705e-04  0.049022310 -0.0359754210 -0.0224717499
## 74 0.02776291 2.747992e-03 -0.138421820  0.0673313797  0.0600798975
## 75 0.06052665 2.852213e-03 -0.140715521 -0.0865134873  0.0238543050
## 76 0.04207849 2.103415e-02 -0.389272323  0.0705792773  0.2742362374
## 77 0.08428937 8.624542e-03 -0.245198941 -0.1523444050  0.0790612271
## 78 0.05317657 6.563398e-03 -0.214110444  0.0333095575 -0.1179754472
## 79 0.04389193 3.129539e-04 -0.046538460  0.0124805196  0.0166191781
## 80 0.02459753 2.667247e-03  0.136427662  0.0795255002  0.0291654095
## 81 0.03346853 1.578047e-04  0.033043865 -0.0112262886 -0.0114806694
## 82 0.10669996 1.338299e-05 -0.009621150 -0.0009418278  0.0019787405
## 83 0.09467082 1.237784e-02 -0.294051503 -0.0349035327  0.0212667024
## 84 0.03416622 4.607473e-03 -0.179493867 -0.0186176509  0.0701778553
## 85 0.05513947 5.826447e-05  0.020075593  0.0020077113  0.0034905518
## 86 0.11899508 1.903056e-02  0.364954671 -0.1770922355  0.1520004985
## 87 0.07584670 1.175458e-01  0.960840079 -0.5448677258 -0.0649057695
## 88 0.09179053 1.526814e-02 -0.327033688  0.0746703072 -0.0107016606
```

```
## 89 0.12655548 4.784184e-02 -0.583325631 -0.4869803964 -0.0683899949
## 90 0.04186707 5.301216e-05 -0.019149531 -0.0149378252 -0.0022108347
## 91 0.03842728 6.407085e-03  0.211933408  0.0274090956 -0.0553173826
##        DFBETA_x2     DFBETA_x3     DFBETA_x4     DFBETA_x7    DFBETA_x10
## 1  -0.0119413491  0.0238976775  0.0228209829  0.0044414155  0.0125684834
## 2   0.0044540840  0.0105653513  0.0184867742 -0.0193169734 -0.0059580037
## 3   0.0701938681  0.0938203211 -0.0374089144 -0.0187108358  0.0683035406
## 4   0.0013925049 -0.0024948725 -0.0002784635 -0.0050705770  0.0032883263
## 5   0.0255145225 -0.0387039801 -0.0196870726  0.0266045643  0.0184074592
## 6  -0.0007534770  0.0003395719  0.0004560386  0.0001473975  0.0008031158
## 7  -0.0336132306 -0.2355810672 -0.2165538439  0.2353709117 -0.2300491916
## 8   0.0039599900  0.0379094243  0.0279786616 -0.0010837206 -0.0182069495
## 9   0.0099590759 -0.1178049972 -0.0626553966  0.0183280351 -0.0055311538
## 10  0.0422073619 -0.0109989067  0.0268014656 -0.0183646775  0.0127816574
## 11  0.0565296643  0.0325578746  0.0803069099 -0.0066038796  0.0940658364
## 12  0.4886181155 -0.1130766938  0.1270373836 -0.0040445897  0.1265780926
## 13  0.0005649398 -0.0546197310 -0.0059108948 -0.0158887760  0.0197856106
## 14  0.0147762995  0.0047643722 -0.0059665476 -0.0159386107 -0.0320601621
## 15 -0.1493096103  0.1879586078  0.0919765150  0.0200111371  0.1675508890
## 16  0.0098608349  0.0572244609 -0.0173711895 -0.0571868836 -0.0214160144
## 17 -0.0158351828 -0.0074423847 -0.0149821177 -0.0438573874 -0.0023615555
## 18 -0.0079475674  0.0865903972 -0.0452716543 -0.0962020964 -0.0615087920
## 19  0.0005265010  0.0052971716 -0.0009708117  0.0535433520 -0.0116840080
## 20  0.0020211945 -0.0002954949 -0.0023241300 -0.0021219850 -0.0044982501
## 21  0.0025755637  0.0248181446  0.0329461145 -0.0319192984  0.0319827518
## 22 -0.0942955356 -0.0974475743 -0.0436239799  0.1061867255  0.0157738648
## 23  0.0045280705 -0.1603824323 -0.0831934209  0.0149737000 -0.0049286656
## 24  0.0153069611  0.1377477152 -0.0534150532  0.0264184003  0.2322600582
## 25  0.0324892030 -0.0609072825 -0.0841572476 -0.0082016348  0.1282561191
## 26  0.0221313913  0.0151831697  0.0855158929 -0.0562175483 -0.0982590200
## 27  0.1602364735  0.1788971066  0.0718703746 -0.0954980519 -0.4225054388
## 28  0.0067380030 -0.0312247116 -0.0561573011  0.1259639442  0.0104596433
## 29  0.0892427532 -0.0450478879 -0.0475186421  0.0721808574 -0.0483909347
## 30 -0.0617552317  0.0471771080  0.0668184749  0.0353740160 -0.0104948077
## 31  0.0895988151 -0.0337737193 -0.1829877006 -0.0087550632  0.1046182701
## 32 -0.0168048347 -0.1079460279 -0.0211668400  0.0131437244  0.0590463772
## 33  0.0761295671  0.1242660307  0.0045477467 -0.1000146957 -0.1211522383
## 34 -0.0507130312 -0.0759270151 -0.0829152403 -0.0071121900  0.0229636321
## 35 -0.0074226676 -0.1071908968 -0.0547672579  0.0630952750  0.0041222769
## 36 -0.0311483034 -0.0278773462 -0.0413695114 -0.0335926075  0.0436365512
## 37 -0.0015232191 -0.0021374346 -0.0036091201  0.0031435751 -0.0048904314
## 38 -0.0124279673  0.0019660066  0.0139604544  0.0049471188  0.0015191157
## 39  0.3418571371 -0.1831212646 -0.0973300161  0.0565656204 -0.1944616295
## 40 -0.0667804379 -0.0115364099  0.1506759559 -0.0228973946 -0.0710069241
## 41 -0.6767697322  0.2194022774 -1.0357478620 -0.1958984786  0.4052811721
## 42 -0.0237060830  0.0352760261  0.0011614882 -0.0502749867 -0.0086915936
## 43  0.0153265479 -0.0176049577  0.0240877570  0.0308398285 -0.0014013791
## 44 -0.0174277574 -0.0564306847 -0.0151909922  0.0905045086 -0.0431666030
## 45  0.0422306578 -0.0879770393  0.0360158412  0.1215552746  0.0747904114
## 46 -0.1287679611  0.0804601540  0.1496952122  0.0528949656  0.1404407147
## 47 -0.0115899882  0.0290352505  0.0003797216  0.0178895531 -0.0024004588
## 48  0.0363248322  0.0590651259  0.0268594677  0.0055488275  0.0676593944
## 49  0.0864992254 -0.1122459589 -0.0304685532 -0.0166425154  0.0877990768
## 50 -0.0221488446  0.0165429631  0.0219309604  0.0173925056 -0.0046551744
## 51  0.0149081885 -0.0348467131  0.0519015852 -0.0023171696  0.0267056359
## 52  0.0109892675  0.0257592436 -0.0028887660  0.0125594801 -0.0173992548
## 53  0.0552325819 -0.0039888734  0.0335918958 -0.0281657553  0.1145722405
```

```
## 54 -0.9657229267  0.4441962106  0.2788522625 -0.3932270293  0.7137391057
## 55  0.1228023278 -0.0220180581  0.0801336739  0.0139934804 -0.1277723394
## 56  0.0570383342 -0.0083293206  0.0953394766 -0.1025962626 -0.0375242977
## 57  0.0367205684  0.1271968172  0.1284503485 -0.3883393630 -0.1132773468
## 58  0.0371927301 -0.0212273166 -0.0082112542 -0.0383445865  0.0461594195
## 59  0.2131365426  0.0415583048  0.2250500694 -0.4382592545  0.2017485025
## 60 -0.0084041896  0.0214241216 -0.0117678985  0.0094548504 -0.0285431118
## 61  0.0566296728 -0.0319569780 -0.0733075532 -0.0165908278 -0.0915075715
## 62  0.1595134877  0.0053479979  0.0464208318  0.2305798316 -0.1372223068
## 63  0.0064966231 -0.0546173607 -0.0406614438  0.0317860711 -0.0257724022
## 64  0.0846477059 -0.0361607356 -0.0340285286 -0.0280498306  0.0039996125
## 65  0.3862789239  0.0696339809 -0.0197130829 -0.0321785998 -0.0344177752
## 66 -0.0013021862  0.0035787822 -0.0402893014 -0.0280888199  0.0054434094
## 67 -0.1141701048 -0.2384638439 -0.0595516533 -0.1480578712  0.1858012632
## 68  0.0017727922 -0.0008313465 -0.0010375360 -0.0012722092 -0.0022672152
## 69  0.0032784852  0.0073007844  0.0038657150 -0.0043917877 -0.0092142295
## 70 -0.4473519903  0.5347072792 -0.1872781062 -0.1048925176 -0.3905992355
## 71 -0.0282434367 -0.0574033553 -0.0610802491 -0.0608982460 -0.0389836525
## 72 -0.0047546720 -0.0057782700 -0.0124414862 -0.0085133245  0.0039533830
## 73 -0.0049721561  0.0171695098  0.0401725760  0.0142292767  0.0007283639
## 74  0.0228741817 -0.0473789428 -0.0736639155 -0.0237244597 -0.0575553535
## 75  0.0019872221 -0.0278223011  0.0134831123  0.1013665296  0.0250327129
## 76 -0.0099323179 -0.1265500213 -0.1453505664  0.0139011335 -0.1229511837
## 77  0.0160613495 -0.1392752448  0.0601174162  0.1875043479  0.0371691442
## 78 -0.0476083489  0.0483564034 -0.0355288329 -0.0243397209  0.1281769725
## 79  0.0127574273 -0.0162012830  0.0041584980 -0.0252116197  0.0087130646
## 80 -0.0301390788 -0.0270235934 -0.0854744144 -0.0026180783 -0.0245261389
## 81  0.0060053199  0.0010235840 -0.0001508205  0.0216499858 -0.0091502936
## 82 -0.0053928683 -0.0029284287  0.0019117528  0.0013607184  0.0020060284
## 83  0.0173231327 -0.1967747469  0.0352478674  0.0952272563  0.0820984943
## 84  0.0440740516 -0.0954757308 -0.0152911113  0.0326359502  0.0672452893
## 85 -0.0081548455 -0.0006885386 -0.0078706795  0.0041136400  0.0118538109
## 86 -0.0111619746 -0.1541520293  0.2415129447  0.0847729264 -0.0246120269
## 87 -0.2963187215  0.4675189052  0.5669386609  0.1963065132 -0.4432898311
## 88  0.0254596784  0.0719063760 -0.0246393329 -0.0807824646 -0.2738289353
## 89  0.0349609955  0.0055596529  0.1883321867  0.4943643570 -0.0193023977
## 90 -0.0009101697  0.0066419157  0.0091403380  0.0083025494 -0.0007774818
## 91  0.0586086696 -0.0616089102 -0.0431929755  0.0681985755 -0.0539464209
##     COVRATIO
## 1  1.1308314
## 2  1.1343012
## 3  1.0462264
## 4  1.1216738
## 5  1.1233149
## 6  1.2071561
## 7  0.5791085
## 8  1.1066914
## 9  1.0382196
## 10 1.1511174
## 11 1.0896644
## 12 1.0447418
## 13 1.1756260
## 14 1.1758823
## 15 1.3129943
## 16 1.0959072
## 17 1.1518520
## 18 1.1464231
```

```
## 19 1.1144911
## 20 1.1402350
## 21 1.1790424
## 22 1.1312982
## 23 1.1024418
## 24 1.2328671
## 25 1.1298924
## 26 1.0310557
## 27 1.0546936
## 28 1.4433936
## 29 1.0411198
## 30 1.0814143
## 31 1.0121029
## 32 1.1856704
## 33 1.1120808
## 34 1.1618929
## 35 1.2395143
## 36 1.1021682
## 37 1.1440570
## 38 1.1189175
## 39 1.3169303
## 40 1.3272245
## 41 1.0767030
## 42 1.1582748
## 43 1.1661911
## 44 1.0092406
## 45 1.0394156
## 46 1.0174786
## 47 1.1036656
## 48 1.0345011
## 49 1.0842605
## 50 1.1487359
## 51 1.0867093
## 52 1.1625379
## 53 1.1441189
## 54 0.4590165
## 55 1.1119283
## 56 1.0521793
## 57 0.7085849
## 58 1.0975487
## 59 1.0432087
## 60 1.1898599
## 61 1.1844829
## 62 0.9458231
## 63 1.0727285
## 64 1.2361956
## 65 1.3117486
## 66 1.2229456
## 67 0.9610320
## 68 1.1299086
## 69 1.1150445
## 70 0.8598563
## 71 1.1247011
## 72 1.1537769
## 73 1.2252795
## 74 1.0572032
## 75 1.1279441
```

```
## 76 0.8536540
## 77 1.1241393
## 78 1.0724779
## 79 1.1328537
## 80 1.0478776
## 81 1.1221156
## 82 1.2172579
## 83 1.1206390
## 84 1.0431073
## 85 1.1502380
## 86 1.1363818
## 87 0.4833675
## 88 1.0957410
## 89 1.0241207
## 90 1.1341623
## 91 1.0292854
# Influential points with DFBETAS

# Intercept - 41 87 89
table(abs(infM_DLdata$DFBETA_INCPT) > (2/sqrt(n)))
##
## FALSE  TRUE
##    88     3
(df.int=which(abs(infM_DLdata$DFBETA_INCPT) > (2/sqrt(n))))
## [1] 41 87 89
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_INCPT[df.int]
## [1]  0.7695033 -0.5448677 -0.4869804
# x1 - 15 24 67 76
table(abs(infM_DLdata$DFBETA_x1) > (2/sqrt(n)))
##
## FALSE  TRUE
##    87     4
(df.x1=which(abs(infM_DLdata$DFBETA_x1) > (2/sqrt(n))))
## [1] 15 24 67 76
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x1[df.x1]
## [1] -0.2779793 -0.2360198  0.3335327  0.2742362
# x2 - 12 39 41 54 59 95 70 87
table(abs(infM_DLdata$DFBETA_x2) > (2/sqrt(n)))
##
## FALSE  TRUE
##    83     8
(df.x2=which(abs(infM_DLdata$DFBETA_x2) > (2/sqrt(n))))
## [1] 12 39 41 54 59 65 70 87
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x2[df.x2]
## [1]  0.4886181  0.3418571 -0.6767697 -0.9657229  0.2131365  0.3862789
## [7] -0.4473520 -0.2963187
# x3 - 7 41 54 67 70 87
table(abs(infM_DLdata$DFBETA_x3) > (2/sqrt(n)))
##
## FALSE  TRUE
##    85     6
```

```
(df.x3=which(abs(infM_DLdata$DFBETA_x3) > (2/sqrt(n))))
## [1]  7 41 54 67 70 87
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x3[df.x3]
## [1] -0.2355811  0.2194023  0.4441962 -0.2384638  0.5347073  0.4675189
# x4 - 7 41 54 59 86 87
table(abs(infM_DLdata$DFBETA_x4) > (2/sqrt(n)))
##
## FALSE  TRUE
##    85     6
(df.x4=which(abs(infM_DLdata$DFBETA_x4) > (2/sqrt(n))))
## [1]  7 41 54 59 86 87
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x4[df.x4]
## [1] -0.2165538 -1.0357479  0.2788523  0.2250501  0.2415129  0.5669387
# x7 - 7 54 57 59 62 89
table(abs(infM_DLdata$DFBETA_x7) > (2/sqrt(n)))
##
## FALSE  TRUE
##    85     6
(df.x7=which(abs(infM_DLdata$DFBETA_x7) > (2/sqrt(n))))
## [1]  7 54 57 59 62 89
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x7[df.x7]
## [1]  0.2353709 -0.3932270 -0.3883394 -0.4382593  0.2305798  0.4943644
# x10 - 7 24 27 41 54 70 87 88
table(abs(infM_DLdata$DFBETA_x10) > (2/sqrt(n)))
##
## FALSE  TRUE
##    83     8
(df.x10=which(abs(infM_DLdata$DFBETA_x10) > (2/sqrt(n))))
## [1]  7 24 27 41 54 70 87 88
2/sqrt(n)
## [1] 0.209657
infM_DLdata$DFBETA_x10[df.x10]
## [1] -0.2300492  0.2322601 -0.4225054  0.4052812  0.7137391 -0.3905992
## [7] -0.4432898 -0.2738289
# Influential points with DFFITS - 7 12 41 54 59 70 87 89
table(abs(infM_DLdata$DFFITS) > (2*sqrt(p/n)))
##
## FALSE  TRUE
##    83     8
(df.fit=which(abs(infM_DLdata$DFFITS) > (2*sqrt(p/n))))
## [1]  7 12 41 54 59 70 87 89
2*sqrt(p/n)
## [1] 0.5547002
infM_DLdata$DFFITS[df.fit]
## [1]  0.5631565  0.6379023 -1.3338841 -1.2869621  0.6422047  0.9542649
## [7]  0.9608401 -0.5833256
# Influential points with COVARIANCE RATIO - 7 15 24 28 35 39 40 54 57 64 65
87
ll = 1 - 3*(p/n)
ul = 1 + 3*(p/n)
table(infM_DLdata$COVRATIO<ll | infM_DLdata$COVRATIO>ul)
```

```
##
## FALSE  TRUE
##    79    12
(df.cov=which(infM_DLdata$COVRATIO<ll | infM_DLdata$COVRATIO>ul))
##  [1]  7 15 24 28 35 39 40 54 57 64 65 87
(ll = 1 - 3*(p/n))
## [1] 0.7692308
(ul = 1 + 3*(p/n))
## [1] 1.230769
infM_DLdata$COVRATIO[df.cov]
##  [1] 0.5791085 1.3129943 1.2328671 1.4433936 1.2395143 1.3169303 1.3272245
##  [8] 0.4590165 0.7085849 1.2361956 1.3117486 0.4833675
# Influential points with Cook's Distance - NONE
table(infM_DLdata$D>1)
##
## FALSE
##    91
(df.cook=which(infM_DLdata$D>1))
## integer(0)
# Influential/Leverage points with hat diagonals - 15 24 28 39 40 41 65 70 ar
e leverage,
#but only 41 and 70 seem to be influential since they have large residuals.
table(infM_DLdata$h>2*p/n)
##
## FALSE  TRUE
##    83     8
(df.hat=which(infM_DLdata$h>2*p/n))
## [1] 15 24 28 39 40 41 65 70
2*p/n
## [1] 0.1538462
infM_DLdata$h[df.hat]
## [1] 0.2119600 0.1657898 0.2510859 0.2320814 0.2025280 0.2918236 0.2222200
## [8] 0.1558114
rStu_Res[df.hat]
##         15         24         28         39         40         41
## -0.7692529  0.8147416 -0.2660543 -0.9302281  0.5669030 -2.0779189
##         65         70
##  0.8716361  2.2212077
df.final = c(df.int,df.x1,df.x2,df.x3,df.x4,df.x7,df.x10,df.fit, df.cov, df.c
ook, df.hat)
(df.table = table(df.final)) # This is the table summarizing how many times e
ach observation exceeded a cut-off for the 10 measures.
## df.final
##  7 12 15 24 27 28 35 39 40 41 54 57 59 62 64 65 67 70 76 86 87 88 89
##  6  2  3  4  1  2  1  3  2  7  7  2  4  1  1  3  2  5  1  1  7  1  3
```

Outliers

```
outliers = which(abs(rStu_Res)>2)
rStu_Res[abs(rStu_Res)>2]
##         7        41        54        57        70        87
##  2.881883 -2.077919 -3.543969  2.406732  2.221208  3.353936
NHLmergedfin[outliers,]
##              Player Pos GP  G  A X... PIM   S  TOI AGE LENGTH CAP.HIT
## 7  Andre Burakovsky  LW 76 12 13    2  14 100  846  24      1    3.25
## 41     Joe Thornton   C 73 16 35    8  20  90 1135  40      1    2.00
## 54     Kevin Labanc  RW 82 17 39   -1  36 130 1150  23      1    1.00
## 57 Marcus Johansson   C 58 13 17  -16   8 115  924  28      2    4.50
## 70     Patrik Laine  RW 82 30 20  -24  42 245 1413  21      2    6.75
## 87   Wayne Simmonds  RW 79 17 13  -16  99 157 1238  30      1    5.00
##    is.centre
## 7          0
## 41         1
## 54         0
## 57         1
## 70         0
## 87         0
```

Model Validation

```
names(new.NHL) = c("player","x1","x2","x3","x4","x7","x10","yprime")
new.NHL$yprime = sqrt(new.NHL$yprime)
new.yhat = predict(last.nhl.lm, newdata=new.NHL[,2:8])
new.SST = sum((new.NHL$yprime-mean(new.NHL$yprime))^2)
new.SSE = sum((new.NHL$yprime-new.yhat)^2)
R2.new.pred = 1-new.SSE/new.SST

last.Rsq = summary(last.nhl.lm)$r.squared

last.PRESS = PRESS(last.nhl.lm)
last.SST = sum(anova(last.nhl.lm)[,2])

PRESS.R2.pred = 1-last.PRESS/last.SST

last.MSRes = summary(last.nhl.lm)$sigma^2
new.MSE = mean((new.NHL$yprime-new.yhat)^2)
```

$newdataR2_{prediction}=0.8317$

$PRESSR2_{prediction}=0.8528$

$FinalModelR2=0.8791$

$MS_{Res}=0.0496$

$AverageSquaredPredictionError=0.0799$

Fixing Outliers

```
#Thornton
NHLmergedfin[NHLmergedfin$AGE>35,]
##          Player Pos GP  G  A X... PIM  S  TOI AGE LENGTH CAP.HIT is.centre
## 41 Joe Thornton   C 73 16 35    8  20 90 1135  40      1       2         1

#Johansson
NHLmergedfin[57,c(1,4,5,8,10,3,11)]
##              Player  G  A   S AGE GP LENGTH
## 57 Marcus Johansson 13 17 115  28 58      2
marcus.extra = as.data.frame(cbind(13/58*82,17/58*82,115/58*82,28,82,2))
names(marcus.extra) = c("x1","x2","x3","x4","x7","x10")
marcus.extra
##         x1       x2       x3 x4 x7 x10
## 1 18.37931 24.03448 162.5862 28 82   2
predict(last.nhl.lm, newdata= marcus.extra)^2
##        1
## 3.273374
last.nhl.lm$fitted.values[57]^2
##       57
## 2.596255
NHLmergedfin[57,12]
## [1] 4.5

#Laine
new_laine = data.frame(x1=44, x2=20, x3=245, x4=21, x7=82, x10=2)
predict(last.nhl.lm, newdata = new_laine)^2
##       1
## 5.833575
NHLmergedfin[70,12]
## [1] 6.75

#Simmonds
new_sim = data.frame(x1=31, x2=23, x3=224, x4=28, x7=82, x10=1)
last.nhl.lm$fitted.values[87]^2
##       87
## 2.427828
predict(last.nhl.lm, newdata = new_sim)^2
##        1
## 4.574581
NHLmergedfin[87,12]
## [1] 5
```