DATA 621 Homework 1

Shane Hylton

2022-09-25

Contents

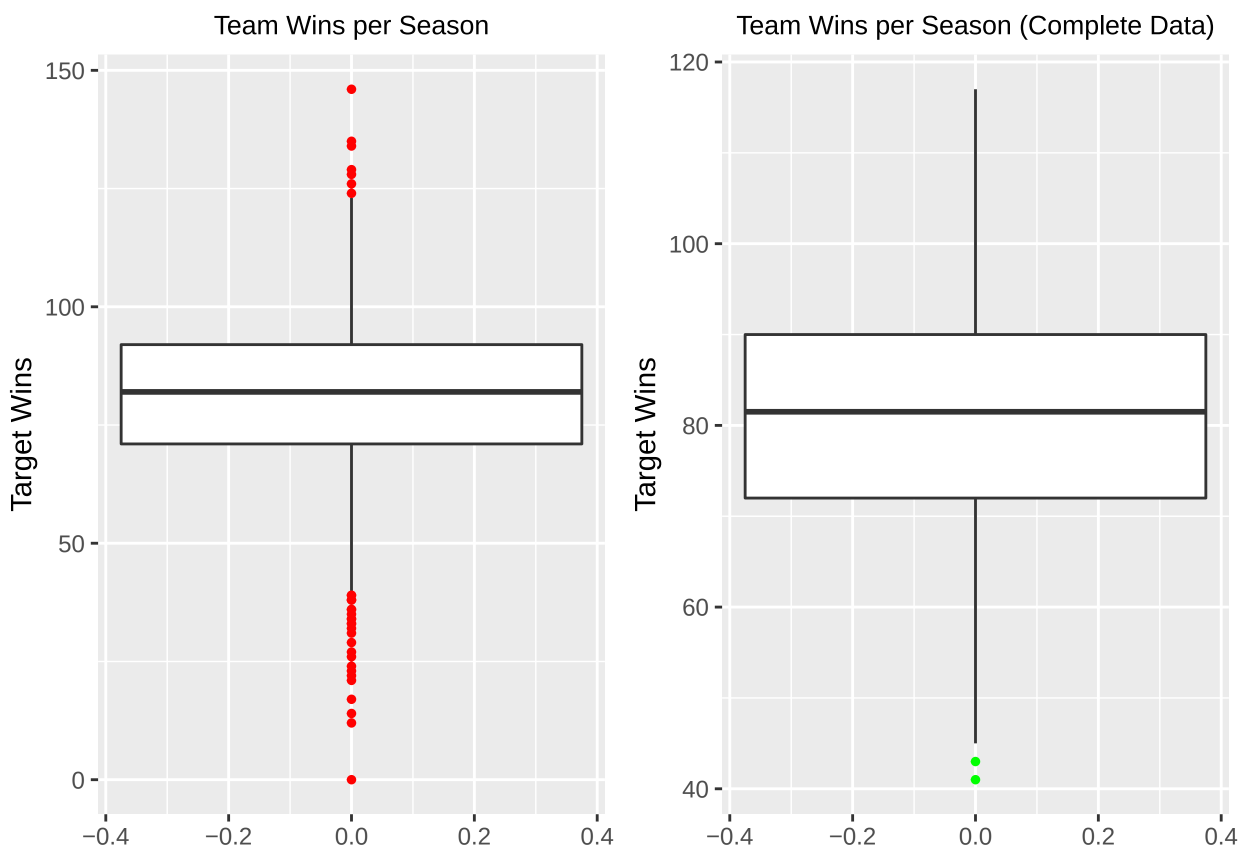
Data Exploration

The training data contains the records and team statistics of 2276 teams between the years of 1871 and 2006. Of those 2276 teams, the median record is 82 wins, and the IQR for wins is (71,92). One can assume that the third quartile is enough wins to make the postseason in any given season. The median number of hits in a season is 1454. There are quite a few columns with missing data. If we isolate only the complete seasons with data in all columns, we are left with only 191 complete cases. If we instead begin by removing the TEAM_BATTING_HBP column and then searching for all complete cases, we will be left with 1486 complete cases. This should provide enough information to draw some conclusions. Without knowing the year of each season, it will be difficult to take into account the changes the game has experienced in the last few decades.

The initial model used for fitting the data showed a strong normal distribution of residuals, as well as a linear normal probability plot. The scatter plot of residuals is evenly distributed above and below the axis. With that being said, the R-Squared is not ideal, so we will need to adjust the model to land on a better fit. Many of the variables are correlated with one another, and the strongest single relationship with team wins is TEAM_BATTING_H, with a correlation of 0.39.

```
##      INDEX         TARGET_WINS     TEAM_BATTING_H TEAM_BATTING_2B
##  Min.   :   1.0   Min.   :  0.00   Min.   : 891   Min.   : 69.0  
##  1st Qu.: 630.8   1st Qu.: 71.00   1st Qu.:1383   1st Qu.:208.0  
##  Median :1270.5   Median : 82.00   Median :1454   Median :238.0  
##  Mean   :1268.5   Mean   : 80.79   Mean   :1469   Mean   :241.2  
##  3rd Qu.:1915.5   3rd Qu.: 92.00   3rd Qu.:1537   3rd Qu.:273.0  
```

```
##  Max.   :2535.0   Max.   :146.00   Max.   :2554   Max.   :458.0  
##                                                                     
##  TEAM_BATTING_3B  TEAM_BATTING_HR  TEAM_BATTING_BB TEAM_BATTING_SO 
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.0   Min.   :   0.0  
##  1st Qu.: 34.00   1st Qu.: 42.00   1st Qu.:451.0   1st Qu.: 548.0  
##  Median : 47.00   Median :102.00   Median :512.0   Median : 750.0  
##  Mean   : 55.25   Mean   : 99.61   Mean   :501.6   Mean   : 735.6  
##  3rd Qu.: 72.00   3rd Qu.:147.00   3rd Qu.:580.0   3rd Qu.: 930.0  
##  Max.   :223.00   Max.   :264.00   Max.   :878.0   Max.   :1399.0  
##                                                    NA's   :102     
##  TEAM_BASERUN_SB TEAM_BASERUN_CS TEAM_BATTING_HBP TEAM_PITCHING_H
##  Min.   :  0.0   Min.   :  0.0   Min.   :29.00    Min.   : 1137  
##  1st Qu.: 66.0   1st Qu.: 38.0   1st Qu.:50.50    1st Qu.: 1419  
##  Median :101.0   Median : 49.0   Median :58.00    Median : 1518  
##  Mean   :124.8   Mean   : 52.8   Mean   :59.36    Mean   : 1779  
##  3rd Qu.:156.0   3rd Qu.: 62.0   3rd Qu.:67.00    3rd Qu.: 1682  
##  Max.   :697.0   Max.   :201.0   Max.   :95.00    Max.   :30132  
##  NA's   :131     NA's   :772     NA's   :2085                    
##  TEAM_PITCHING_HR TEAM_PITCHING_BB TEAM_PITCHING_SO  TEAM_FIELDING_E
##  Min.   :  0.0    Min.   :   0.0   Min.   :    0.0   Min.   :  65.0  
##  1st Qu.: 50.0    1st Qu.: 476.0   1st Qu.:  615.0   1st Qu.: 127.0  
##  Median :107.0    Median : 536.5   Median :  813.5   Median : 159.0  
##  Mean   :105.7    Mean   : 553.0   Mean   :  817.7   Mean   : 246.5  
##  3rd Qu.:150.0    3rd Qu.: 611.0   3rd Qu.:  968.0   3rd Qu.: 249.2  
##  Max.   :343.0    Max.   :3645.0   Max.   :19278.0   Max.   :1898.0  
##                                    NA's   :102                       
##  TEAM_FIELDING_DP
##  Min.   : 52.0   
##  1st Qu.:131.0   
##  Median :149.0   
##  Mean   :146.4   
##  3rd Qu.:164.0   
##  Max.   :228.0   
##  NA's   :286     
```



Data Preparation

Looking at the original dataframe with 2276 entries, one of the main columns that is missing data is the HBP column. Based on the original fit, HBP did not add much to the model, so I will remove that statistic, especially considering how much it is out of the control of the team. Most likely, HBP is not something that can be predicted year over year. It is possible that there is a reasonable means to predict the HBP statistic for an individual, but it is not something that can easily be predicted in this case. I will remove the HBP column entirely because there are only 191 complete entries for the HBP statistic, so it cannot be used as a predictor. The Caught Stealing statistic has 772 NA entries, so I will replace all of the NAs in that statistic with the median. The same will be done for the other few columns that have 800 or fewer NAs.

I created a ratio for Home Runs per Hit to see if there is any value in finding home run frequency for a given team. I also deconstructed the Hits variable, and in its place I left singles, doubles, triples, and home runs. This will remove the collinearity between hits and the existing doubles, triples, and home runs.

Unrealistic data will be removed as well. For instance, one row shows nearly 20,000 strike outs (MLB Record 1450) and some other rows show a prediction of well above 116 wins, which is the MLB record.

Build Models

Model One

For the first model, I will remove unnecessary variables only, and I will seek a collection of variables with p-values of 0.05 or lower. Hits were removed and instead I broke down the hits based on the number of bases gained by each type of hit. I removed hits because there was a concern of colinearity and overlap with the other three types of hit already in the model. I used variables for singles, doubles, triples, and home runs.

With a p-value of 0.227, TEAM_BASERUN_SB, provides very little to the target variable. Doubles are the only other variable that has a p-value greater than 0.05.

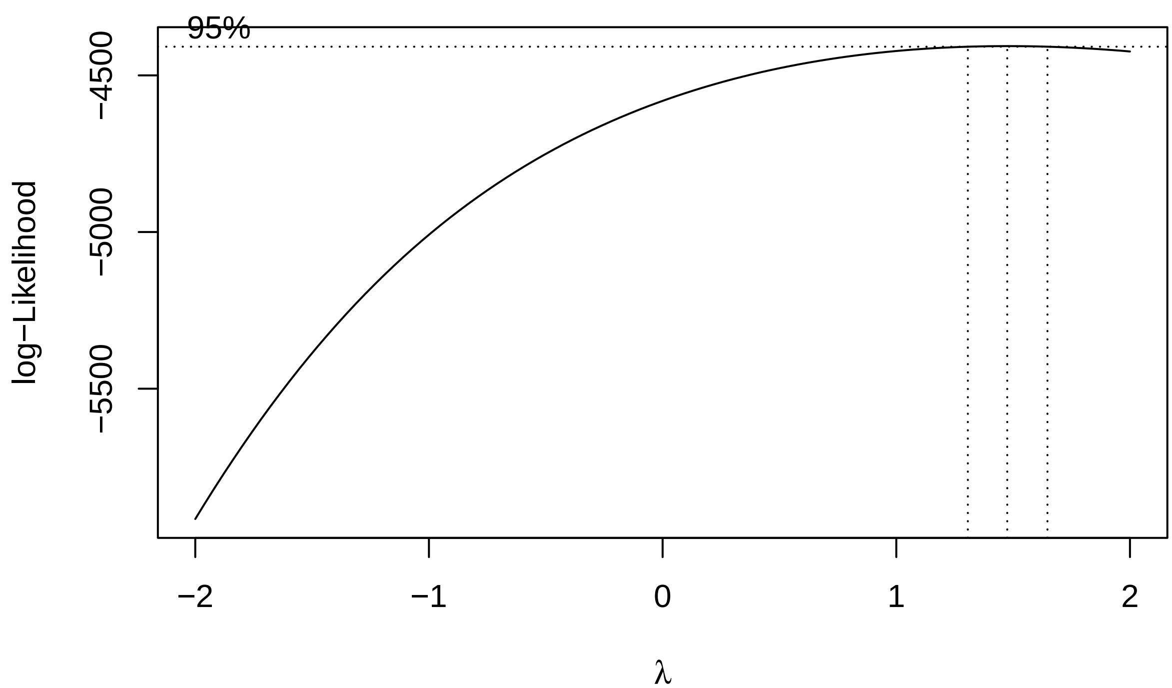
For the original first model, the R-squared is 0.3039, with a standard error of 12.36 on 2207 degrees of freedom, and the fit decent, though the R-squared leaves the door open for error. In the first model, after removing the variables that had p-values above 0.05, the R-squared decreased to 0.3018 with a standard error of 12.37 on 2210 degrees of freedom.

The intercept of 29.44 games provides a baseline number that represents the amount of wins a team should be expected to have, and the expected wins can increase or decrease from that mark. All hit types and walks add to the wins column, while strikeouts detract from wins. Stolen bases add to wins, as do pitching strike outs. Contrary to expectations, fielding double plays detract from a team's expected win total. Errors, allowed walks, and allowed home runs all also detract from expected wins.

Model Two

For the second model, I used the Box-Cox method to find a lambda that is appropriate for the model. The R-squared of that model was 0.3012, and the standard error was 0.411 with 2210 degrees of freedom. While the R-squared is lower than the first model, the standard error is substantially lower.

The baseline expected wins for the Box-Cox model is 42.5. All types of hits add to the wins, as do walks and stolen bases.Strikeouts detract from potential wins. Fielding double plays again detract from the expected win total, which is certainly contrary to expectations. Errors, allowed walks, and allowed home runs all detract from win total.



Model Three

For the third model, I manipulated the predictor variables to come up with a model that contains variables with magnitudes that are more comparable to one another. I started by taking the log of each predictor variable.

The third model was then created using the backwards selection method after taking the log of each variable. The resulting R-squared was 0.2935, and the standard error was 12.45 on 2209 degrees of freedom.

The intercept on this model is not as intuitive, because the coefficients have very high magnitudes. The intercept, or the baseline for wins is -326. One single adds 60 to that tally, so it is very easy to overcome the negative value to start. This model agrees with the other two in the ways that each statistic adds or detracts from win total. Batter strikeouts and baserunners being caught stealing both detract from the expected win total. Errors, Allowed Home Runs, and double plays all detract from win total. Allowed walks actually add to the win total in this model, which is also confusing.

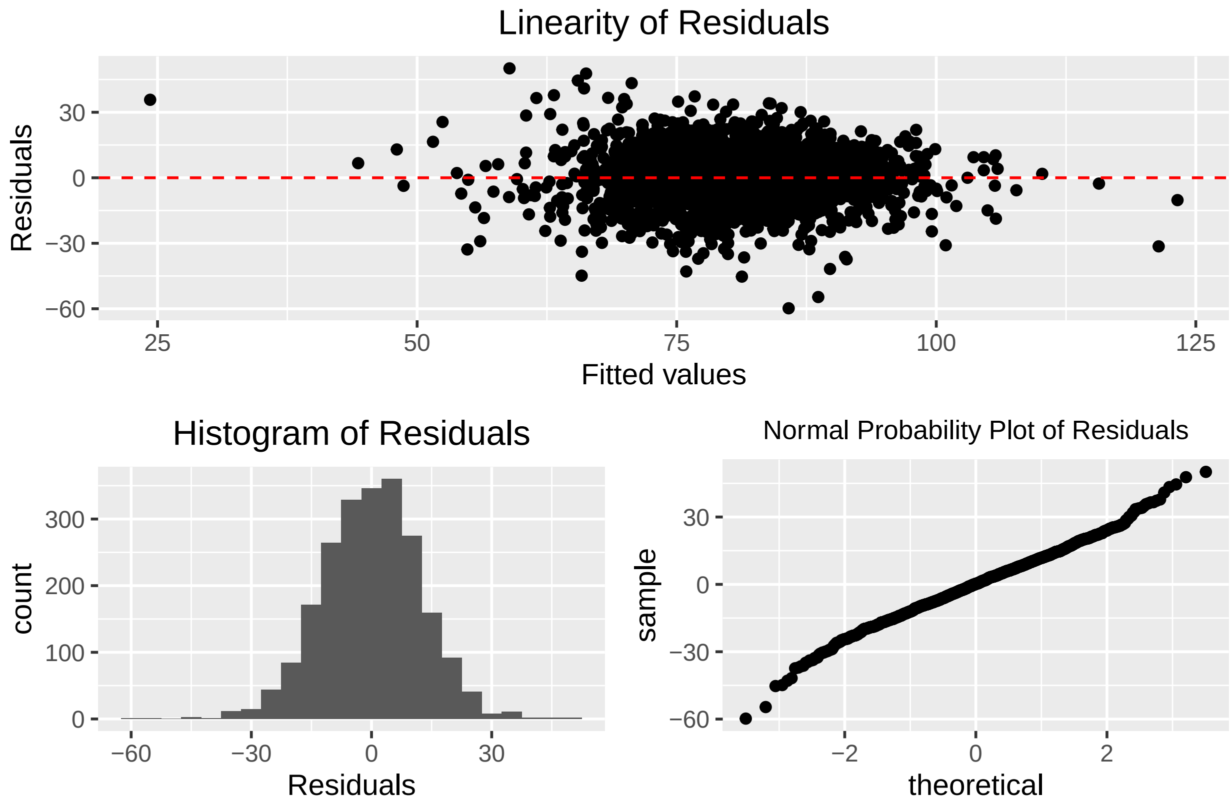
Select Models

In the end, the Box-Cox derived model is the most appealing judging by the R-squared coupled with the residual standard error. While it has the lowest R-squared of the three, it has the lowest standard error, which indicates that the residuals are closely distributed around the population mean. It is worth considering, however, that the Box-Cox model was heavily reshaped, so the standard error is actually not as strong as it may appear.

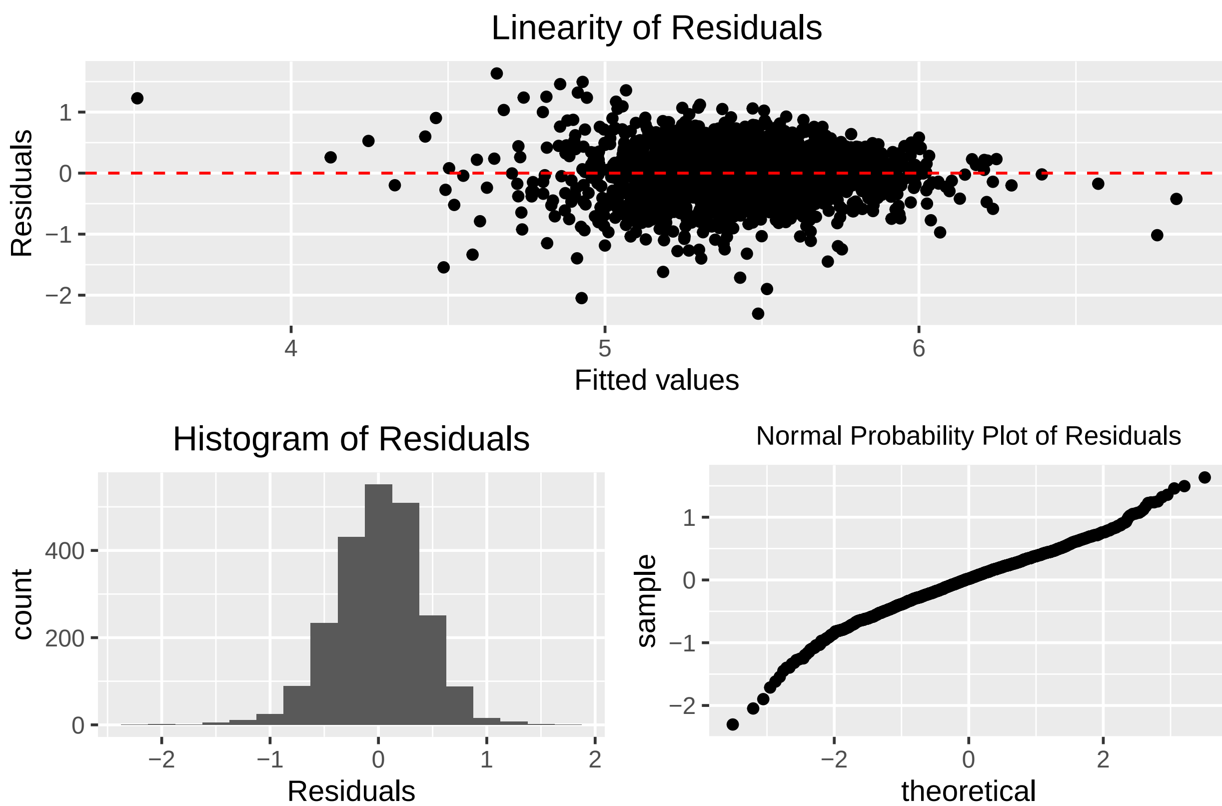
Visually, the Log Transform with Backwards Selection model looks to be the most normally distributed. The Normal Probability Plot looks to be the most linear of the three, and the histogram of the residuals is reasonably normally distributed. The Box-Cox model looks excellent at times, but it begins to falter when the residuals drop to -2. While the log transformed model appears optimal visually, it is the model that makes the least sense intuitively for the linear equation.

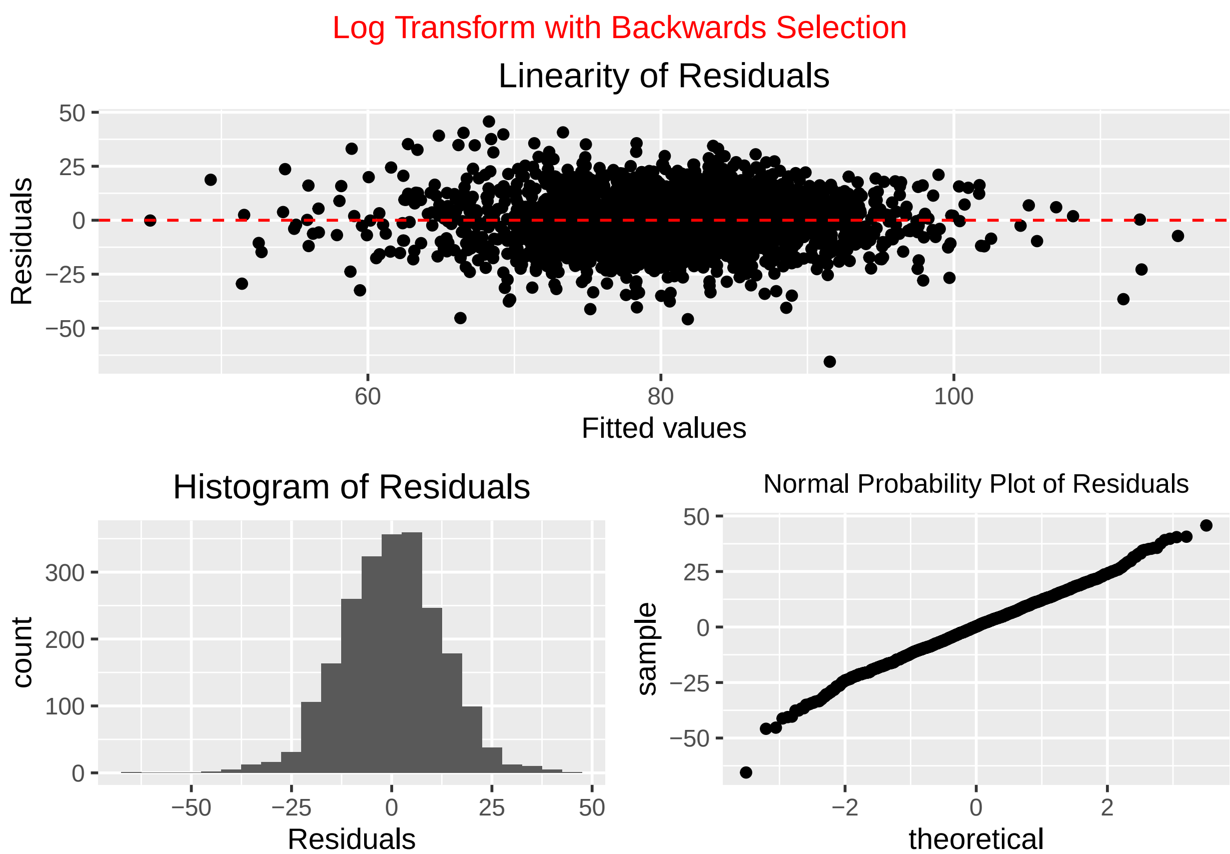
Though the normal probability plot is not as linear as the other two, the Box-Cox transformed model appears to be the most suitable for the prediction.

Backwards Selection



Box−Cox


Linearity of Residuals
Residuals
Fitted values
Histogram of Residuals
count
Residuals
Normal Probability Plot of Residuals
sample
theoretical



Implementation

Box-Cox Model Predictions: I have exported the predicted win totals for the test data to a .csv file for validation. The Box-Cox approach is the model that I place the most trust in of the three models.

```
##          9        10        14        47        60        63        74        83
##   63.16235  66.05107  73.71984  84.06200  85.94800  69.89524  74.73712  75.31305
##         98       120       123       135       138       140       151       153
##   71.92240  72.10583  67.35366  82.10839  83.05294  81.21356  84.59763  76.75974
##        171       184       193       213       217       226       230       241
##   73.26269  77.34222  71.02774  88.47646  80.91712  81.98404  81.53605  71.12581
##        291       294       300       348       350       357       367       368
##   79.36235  84.83239  38.51263  71.59511  81.44810  70.48021  90.33603  85.69557
##        372       382       388       396       398       403       407       410
##   82.91980  84.77386  80.21819  86.86909  75.81775  91.87735  84.66680  91.73063
##        412       414       436       440       476       479       481       501
##   80.24230  90.00926  20.68016 106.45997  90.64881  89.73458  95.46120  75.55795
##        503       506       519       522       550       554       566       578
##   68.51120  77.83048  73.98201  81.91802  76.76453  73.48031  72.92921  77.27315
##        596       599       605       607       614       644       692       699
##   93.57440  74.79651  65.71809  80.07562  87.41220  77.07324  87.42577  84.90542
##        700       716       721       722       729       731       746       763
##   82.94810  97.00215  74.83948  79.64437  79.41967  91.85437  87.11922  69.10090
##        774       776       788       789       792       811       835       837
##   78.28955  90.99223  76.90365  84.35939  84.15820  82.24225  72.32644  76.06597
```

```
##       861       862       863       871       879       887       892       904 
##  80.84853  86.06122  95.28084  72.87643  87.12269  78.23907  82.85926  83.32918 
##       909       925       940       951       976       981       983       984 
##  89.41038  89.59402  77.55566  40.43827  72.51846  84.11699  83.80039  84.83767 
##       989       995      1000      1001      1007      1016      1027      1033 
##  91.29362 104.45900  86.68586  87.67206  78.35979  72.73412  82.39811  85.01872 
##      1070      1081      1084      1098      1150      1160      1169      1172 
##  78.49467  70.45279  54.46746  74.52523  86.78104  57.51937  84.74349  83.60067 
##      1174      1176      1178      1184      1193      1196      1199      1207 
##  93.44683  89.96816  78.89630  75.53131  84.92372  79.66138  71.51192  69.98615 
##      1218      1223      1226      1227      1229      1241      1244      1246 
##  91.53911  68.85925  69.53624  69.41187  70.14011  85.41506  89.67615  76.31274 
##      1248      1249      1253      1261      1305      1314      1323      1328 
##  91.82277  90.47405  83.75979  77.46268  77.00501  87.06963  88.59466  68.56130 
##      1353      1363      1371      1372      1389      1393      1421      1431 
##  74.24017  75.17864  85.06637  79.68199  64.44808  74.27128  90.25171  72.15874 
##      1437      1442      1450      1463      1464      1470      1471      1484 
##  71.96945  72.33254  76.86080  78.19854  77.79217  81.54656  82.31634  78.15410 
##      1495      1507      1514      1526      1549      1552      1556      1564 
## 110.06549  67.79593  76.54196  69.89758  88.97775  62.62044  94.67603  74.04047 
##      1585      1586      1590      1591      1592      1603      1612      1634 
## 108.48193 112.18479  95.22755 107.50649 100.85809  91.80554  83.65769  81.38384 
##      1645      1647      1673      1674      1687      1688      1700      1708 
##  71.89480  78.62040  92.22551  89.88312  79.78728  92.66094  80.98516  74.39351 
##      1713      1717      1721      1730      1737      1748      1749      1763 
##  78.54193  70.64754  74.13225  79.98571  88.03224  89.13335  84.87879  85.39122 
##      1768      1778      1780      1782      1784      1794      1803      1804 
##  61.84158  88.20541  81.31597  48.32000  54.82042 107.21748  69.26910  81.72867 
##      1819      1832      1833      1844      1847      1854      1855      1857 
##  73.08224  73.96290  75.12188  65.78807  76.03674  87.16553  80.97946  85.58538 
##      1864      1865      1869      1880      1881      1882      1894      1896 
##  76.28767  79.63179  74.59296  89.60832  79.01148  85.06655  77.74216  75.66640 
##      1916      1918      1921      1926      1938      1979      1982      1987 
##  76.78880  69.73302 104.38018  91.27429  84.56410  65.47298  69.22399  83.55438 
##      1997      2004      2011      2015      2022      2025      2027      2031 
##  79.30856  92.56781  75.24214  78.82637  77.85116  71.02896  79.25471  71.57751 
##      2036      2066      2073      2087      2092      2125      2148      2162 
##  72.12873  75.99270  79.70030  78.96676  81.48417  65.88686  81.08704  93.61784 
##      2191      2203      2218      2221      2225      2232      2267      2291 
##  80.63685  89.36051  80.04056  74.97087  80.71063  77.97297  88.81714  74.84053 
##      2299      2317      2318      2353      2403      2411      2415      2424 
##  88.71741  85.73469  81.46841  78.57915  62.18962  84.26617  78.58300  83.96403 
##      2441      2464      2465      2472      2481      2487      2500      2501 
##  71.72821  82.70666  79.89430  60.61107  90.67626 683.88316  69.74434  77.86701 
##      2520      2521      2525 
##  80.24422  81.96591  73.87058
```

Appendix: Code

```
#mlb_train <- read.csv("https://raw.githubusercontent.com/st3vejobs/DATA-621/main/MLB/moneyball-trainin

#mlb_test <- read.csv("https://raw.githubusercontent.com/st3vejobs/DATA-621/main/MLB/moneyball-evaluati
```

```
fit <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BAT

#summary(fit)

#nrow(mlb_train)

summary(mlb_train)

mlb_trim <- subset(mlb_train, select =  -c(11))

mlb_trim_sub <- drop_na(mlb_trim)

mlb_trim$TEAM_BATTING_SO <- replace_na(mlb_trim$TEAM_BATTING_SO, median(mlb_trim$TEAM_BATTING_SO, na.rm

mlb_trim$TEAM_BASERUN_SB <- replace_na(mlb_trim$TEAM_BASERUN_SB, median(mlb_trim$TEAM_BASERUN_SB, na.rm

mlb_trim$TEAM_BASERUN_CS <- replace_na(mlb_trim$TEAM_BASERUN_CS, median(mlb_trim$TEAM_BASERUN_CS, na.rm

mlb_trim$TEAM_PITCHING_SO <- as.numeric(mlb_trim$TEAM_PITCHING_SO)

mlb_trim$TEAM_PITCHING_SO <- replace_na(mlb_trim$TEAM_PITCHING_SO, median(mlb_trim$TEAM_PITCHING_SO, na

mlb_trim$TEAM_FIELDING_DP <- replace_na(mlb_trim$TEAM_FIELDING_DP, median(mlb_trim$TEAM_FIELDING_DP, na

#summary(mlb_trim)

cor_mat <- data.frame(cor(mlb_trim))

mlb_trim <- subset(mlb_trim, mlb_trim$TEAM_PITCHING_SO <= 1500)
mlb_trim <- subset(mlb_trim, mlb_trim$TARGET_WINS <= 120)
mlb_trim <- subset(mlb_trim, mlb_trim$TARGET_WINS >= 20)
mlb_trim <- subset(mlb_trim, mlb_trim$TEAM_BATTING_HR > 0)
mlb_trim <- subset(mlb_trim, mlb_trim$TEAM_BATTING_SO > 0)
mlb_trim <- subset(mlb_trim, mlb_trim$TEAM_PITCHING_SO > 0)
mlb_trim$TEAM_BATTING_S <- mlb_trim$TEAM_BATTING_H - mlb_trim$TEAM_BATTING_2B - mlb_trim$TEAM_BATTING_3
```

```
all_box <- ggplot(mlb_train, aes(y=TARGET_WINS))+
  geom_boxplot(outlier.color = 'red', outlier.size = 1)+
  ylab('Target Wins')+
  ggtitle('Team Wins per Season')+
  theme(plot.title = element_text(hjust = 0.5, size = 10))

complete_box <- ggplot(mlb_trim_sub, aes(y=TARGET_WINS))+
  geom_boxplot(outlier.color = 'green', outlier.size = 1)+
  ylab('Target Wins')+
  ggtitle('Team Wins per Season (Complete Data)')+
  theme(plot.title = element_text(hjust = 0.5, size = 10))

ggarrange(all_box,complete_box)
```

```
fit_res_sct <- ggplot(data = fit, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = 'red') +
```

```r
  xlab("Fitted values") +
  ylab("Residuals") +
  ggtitle("Linearity of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

fit_res_hst <- ggplot(data = fit, aes(x = .resid)) +
  geom_histogram(binwidth = 5) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

fit_res_npp <- ggplot(data = fit, aes(sample = .resid)) +
  stat_qq()+
  ggtitle("Normal Probability Plot of Residuals")+
  theme(plot.title = element_text(hjust = 0.5, size = 9.5))

lin_analysis <- ggarrange(fit_res_sct, fit_res_hst, fit_res_npp)

#lin_analysis
```

```r
mlb_trim$HR_H <- mlb_trim$TEAM_BATTING_HR / mlb_trim$TEAM_BATTING_H

fit_2 <- lm(TARGET_WINS ~ TEAM_BATTING_S + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BA

#summary(fit_2)

fit_2_update <- lm(TARGET_WINS ~ TEAM_BATTING_S + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB +

#summary(fit_2_update)
```

```r
mlb_mod <- mlb_trim
#mlb_mod$TEAM_BATTING_2B <- mlb_mod$TEAM_BATTING_2B / 2
#mlb_mod$TEAM_BATTING_3B <- mlb_mod$TEAM_BATTING_3B * .75
#mlb_mod$TEAM_BATTING_HR <- mlb_mod$TEAM_BATTING_HR * 100
#mlb_mod$HR_H <- mlb_mod$HR_H * 3
#mlb_mod$TEAM_FIELDING_E <- mlb_mod$TEAM_FIELDING_E * 3

fit_3 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BA

#summary(fit_3)

bc <- boxcox(fit_3)
lam <- bc$x[which.max(bc$y)]

fit_3_bc <- lm((TARGET_WINS^(lam - 1))/lam ~ TEAM_BATTING_S + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_

#summary(fit_3_bc)
```

```r
mod_ln <- mlb_trim
mod_ln <- subset(mlb_trim, select = -c(1,2,17))
mod_ln$TEAM_BATTING_S <- mod_ln$TEAM_BATTING_H - mod_ln$TEAM_BATTING_2B - mod_ln$TEAM_BATTING_3B - mod_l
mod_ln <- log((mod_ln))
```

```
mod_ln$TARGET_WINS <- mlb_trim$TARGET_WINS
mod_ln$HR_H <- mlb_trim$HR_H

fit_4 <- lm(TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_B

#summary(fit_4)

fit_5 <- lm(TARGET_WINS ~ TEAM_BATTING_S + TEAM_BATTING_2B + TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_B

#summary(fit_5)
```

```
back_res_sct <- ggplot(data = fit_2_update, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = 'red') +
  xlab("Fitted values") +
  ylab("Residuals") +
  ggtitle("Linearity of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

back_res_hst <- ggplot(data = fit_2_update, aes(x = .resid)) +
  geom_histogram(binwidth = 5) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

back_res_npp <- ggplot(data = fit_2_update, aes(sample = .resid)) +
  stat_qq()+
  ggtitle("Normal Probability Plot of Residuals")+
  theme(plot.title = element_text(hjust = 0.5, size = 9.5))

lin_analysis_back <- ggarrange(back_res_sct,ggarrange(back_res_hst, back_res_npp), nrow = 2)

annotate_figure(lin_analysis_back, top = text_grob("Backwards Selection",color = "red", size = 12))

#lin_analysis_back
```

```
bc_res_sct <- ggplot(data = fit_3_bc, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = 'red') +
  xlab("Fitted values") +
  ylab("Residuals") +
  ggtitle("Linearity of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

bc_res_hst <- ggplot(data = fit_3_bc, aes(x = .resid)) +
  geom_histogram(binwidth = .25) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

bc_res_npp <- ggplot(data = fit_3_bc, aes(sample = .resid)) +
  stat_qq()+
  ggtitle("Normal Probability Plot of Residuals")+
```

```
  theme(plot.title = element_text(hjust = 0.5, size = 9.5))

lin_analysis_bc <- ggarrange(bc_res_sct,ggarrange(bc_res_hst, bc_res_npp), nrow = 2)

annotate_figure(lin_analysis_bc, top = text_grob("Box-Cox",color = "red", size = 12))
```

```
log_res_sct <- ggplot(data = fit_5, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = 'red') +
  xlab("Fitted values") +
  ylab("Residuals") +
  ggtitle("Linearity of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

log_res_hst <- ggplot(data = fit_5, aes(x = .resid)) +
  geom_histogram(binwidth = 5) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")+
  theme(plot.title = element_text(hjust = 0.5))

log_res_npp <- ggplot(data = fit_5, aes(sample = .resid)) +
  stat_qq()+
  ggtitle("Normal Probability Plot of Residuals")+
  theme(plot.title = element_text(hjust = 0.5, size = 9.5))

lin_analysis_log <- ggarrange(log_res_sct,ggarrange(log_res_hst, log_res_npp), nrow = 2)

annotate_figure(lin_analysis_log, top = text_grob("Log Transform with Backwards Selection",color = "red

#lin_analysis_back
```

```
mlb_test$TEAM_BATTING_S <- mlb_test$TEAM_BATTING_H - mlb_test$TEAM_BATTING_2B - mlb_test$TEAM_BATTING_3I

mlb_test$HR_H <- mlb_test$TEAM_BATTING_HR / mlb_test$TEAM_BATTING_H

mlb_test_archive <- mlb_test$INDEX
rownames(mlb_test) <- mlb_test$INDEX

mlb_test_log <- subset(mlb_test, select = -c(1, 2, 10,18))

na_median <- function(x) replace(x, is.na(x), median(x, na.rm = TRUE))

mlb_test_log <- replace(mlb_test_log, TRUE, lapply(mlb_test_log, na_median))
mlb_test_test <- mlb_test_log
mlb_test_test$HR_H <- mlb_test$HR_H

mlb_test_log[mlb_test_log == 0] <- 1



mlb_test_log <- log(mlb_test_log)
mlb_test_log$HR_H <- mlb_test$HR_H
```

```
#predict(fit_5, mlb_test_log)

bc_test <- predict(fit_3_bc, mlb_test_test)

bc_test <- (bc_test*lam)^(1/(lam - 1))

bc_test

write.table(bc_test,"box_cox_prediction_mlb.csv",row.names = TRUE, col.names = FALSE, quote = FALSE, sep
```