Project #1: Moneyball OLS Regression
Michael Pallante

Task 1: Development of Base Model

To begin the process of developing a base model, we first must perform exploratory data analysis of the Moneyball data set. First, we observe the variables in the Moneyball data set. Immediately, we notice some missing values in the columns. In Figure 1.0 below, we further inspect the missing data through a test to see the percent of null observations by each variable.

Figure 1.0: Missing Data Test- Percent of Null Observations by Variable

Variable	Percent of Null Observations
TEAM_BATTING_HBP	91.608084%
TEAM_BASERUN_CS	33.919156%
TEAM_FIELDING_DP	12.565905%
TEAM_BASERUN_SB	5.755712%
TEAM_BATTING_SO	4.481547%
TEAM_PITCHING_SO	4.481547%
TARGET_WINS	0%
TEAM_BATTING_H	0%
TEAM_BATTING_2B	0%
TEAM_BATTING_3B	0%
TEAM_BATTING_HR	0%
TEAM_BATTING_BB	0%
TEAM_PITCHING_H	0%
TEAM_PITCHING_HR	0%
TEAM_PITCHING_BB	0%
TEAM_FIELDING_E	0%

Next, we explore the relationships between each respective predictor variable and the response variable, TARGET_WINS. In Figure 1.1 below, we can see each predictor variable’s correlation with TARGET_WINS, measured by their Pearson correlation coefficient.

Figure 1.1: Predictor Variable Correlation with the Response Variable (Y), TARGET_WINS

Variable	Pearson Correlation Coefficient
TEAM_BATTING_H	0.38877
TEAM_BATTING_2B	0.28910
TEAM_BATTING_BB	0.23256
TEAM_PITCHING_HR	0.18901
TEAM_BATTING_HR	0.17615

TEAM_BATTING_3B	0.14261
TEAM_BASERUN_SB	0.13514
TEAM_PITCHING_BB	0.12417
TEAM_BATTING_HBP	0.07350
TEAM_BASERUN_CS	0.02240
TEAM_BATTING_SO	-0.03175
TEAM_FIELDING_DP	-0.03485
TEAM_PITCHING_SO	-0.07844
TEAM_PITCHING_H	-0.10994
TEAM_FIELDING_E	-0.17648

Unfortunately, we do not see any variable that has a particularly strong correlation to our response variable, TARGET_WINS. We will need to keep working with the data set to narrow down better predictors. At least, it can be determined that we can safely remove two entire variables from our analysis right away. Those two variables, TEAM_BATTING_HBP and TEAM_BASERUN_CS, were removed from the data set because they each had over 15% of missing values and both variables also showed a particularly insignificant correlation to our response variable. It is a bit too early in the process to decide to remove any other variables. All other missing, null values in the analysis were imputed through the use of regression trees specific to each variable in our analysis. The remaining outliers were converted to null values and then also imputed using variable-specific regression trees. This helped to reduce some of the heavily skewed variables caused by outliers, as well as reduce overly clustered areas of data.

With the information from our EDA and data from our data cleaning, we then create our base regression model to predict our response variable (Y), TARGET_WINS. This base model will use the automated variable selection procedure, forward selection, to obtain our regression model. Our base model can be seen in Figure 1.2 below.

Figure 1.2: Base Model Using Forward Selection

```
Call:
lm(formula = TARGET_WINS ~ (TEAM_BATTING_H + TEAM_BATTING_2B + 
    TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BATTING_SO + 
    TEAM_BASERUN_SB + TEAM_PITCHING_H + TEAM_PITCHING_HR + TEAM_PITCHING_BB + 
    TEAM_PITCHING_SO + TEAM_FIELDING_E + TEAM_FIELDING_DP + INDEX) - 
    INDEX, data = mb)

Residuals:
    Min      1Q  Median      3Q     Max 
-74.999  -8.218   0.311   8.248  63.150 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)      35.6047943  5.5020669   6.471 1.19e-10
TEAM_BATTING_H    0.0270001  0.0047519   5.682 1.50e-08
TEAM_BATTING_2B   0.0095391  0.0092593   1.030 0.303018
TEAM_BATTING_3B   0.1440381  0.0176798   8.147 6.10e-16
TEAM_BATTING_HR   0.0445889  0.0293679   1.518 0.129082
```

```
TEAM_BATTING_BB    0.0300693  0.0080328   3.743 0.000186
TEAM_BATTING_SO   -0.0065104  0.0050589  -1.287 0.198248
TEAM_BASERUN_SB    0.0713793  0.0060738  11.752  < 2e-16
TEAM_PITCHING_H    0.0004355  0.0028192   0.154 0.877237
TEAM_PITCHING_HR   0.0390828  0.0254229   1.537 0.124357
TEAM_PITCHING_BB -0.0080000  0.0070657  -1.132 0.257660
TEAM_PITCHING_SO -0.0049178  0.0042408  -1.160 0.246320
TEAM_FIELDING_E  -0.0577376  0.0051611 -11.187  < 2e-16
TEAM_FIELDING_DP -0.0931839  0.0133377  -6.986 3.69e-12

(Intercept)      ***
TEAM_BATTING_H   ***
TEAM_BATTING_2B
TEAM_BATTING_3B  ***
TEAM_BATTING_HR
TEAM_BATTING_BB  ***
TEAM_BATTING_SO
TEAM_BASERUN_SB  ***
TEAM_PITCHING_H
TEAM_PITCHING_HR
TEAM_PITCHING_BB
TEAM_PITCHING_SO
TEAM_FIELDING_E  ***
TEAM_FIELDING_DP ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.27 on 2262 degrees of freedom
Multiple R-squared:  0.2942,	Adjusted R-squared:  0.2902
F-statistic: 72.54 on 13 and 2262 DF,  p-value: < 2.2e-16
```

The result of the forward selection procedure was that the base model used all of the predictor variables we included in our data set (after our EDA and data cleaning). We were given a set of variables to work with, but we would definitely like to simplify the model as much as we can. From our initial base model, we are able to write out the following mathematical expression for this model:

TARGET_WINS = 35.6048 + 0.027(TEAM_BATTING_H) + 0.0095(TEAM_BATTING_2B) + 0.144(TEAM_BATTING_3B) + 0.0446(TEAM_BATTING_HR) + 0.0301(TEAM_BATTING_BB) – 0.0065(TEAM_BATTING_SO) + 0.0714(TEAM_BASERUN_SB) + 0.0004(TEAM_PITCHING_H) + 0.0391(TEAM_PITCHING_HR) – 0.008(TEAM_PITCHING_BB) – 0.0049(TEAM_PITCHING_SO) – 0.0577(TEAM_FIELDING_E) – 0.0932(TEAM_FIELDING_DP)

While it is easy to tell right away that there are too many unnecessary predictor variables in our model, we can still interpret what the overall equation and the coefficients mean. If we multiply the parameter estimates/coefficients by the mean of each variable, then add the intercept, it should equal the mean TARGET_WINS, which is just about 81 wins (80.79086 to be precise). In this case, the model works because the TARGET_WIN value we get from completing the equation is equal to 80.77192, (approximately 81 wins as well). The adjusted r-squared value of our model is 0.2902 is not bad, but we might be able to improve the fit of our model with some more refinement and simplification.

Task 2: Base Model Simplification and Refinement

This task is designed for the improvement of our base model. It was clear from our first model that we have too many predictor variables. Our first job is to focus in on each of our predictor variables and decide which ones are truly predictive of our win total in baseball. If possible, we ideally would like to trim this specific model down to between 1 to 6 predictor variables because we know from testing our data set that the improvement in our adjusted r-squared value diminishes with the inclusion of more than 6 predictor variables. To be able to trim variables from our base model, it is important to consider the overall logic of baseball, which is to score runs and prevent runs from being scored on you. We know from our initial correlation test that our test nearly aligns with our logic, although it seems to favor batting/scoring runs over preventing runs. We also can see from our initial parameter estimates that some variables are absolutely more statistically significant than others. With these thoughts in mind, we can evaluate our set of predictor variables and narrow them down to the ones we would like to use in our refined model, which we will be referring to as our parsimony model from here on.

Our data set and our previous correlations show that the model would favor offensive categories. It is clear that we need to include the TEAM_BATTING_H, TEAM_BATTING_3B, and TEAM_BATTING_HR need to be included in our model because they each have a higher correlation to TARGET_WINS than most of our variables and are also statistically significant based on their parameter estimates. TEAM_BATTING_2B has a high correlation with TARGET_WINS but a seemingly low statistical significance, so it is up to us whether to include it in our model. We will include it in this case because of the high correlation and also because it is a statistic that should seem to be more significant because it theoretically has a positive impact on wins. TEAM_BATTING_BB and TEAM_BASERUN_SB also have a higher correlation and are statistically significant and should also be included. That gives us a total of 6 predictor variables to this point.

We will not include any variables with a parameter estimate that does not make logical sense. For instance, TEAM_PITCHING_HR, or home runs allowed, should not have a positive parameter estimate because allowing home runs theoretically has a negative impact on wins. The same logic goes for TEAM_PITCHING_H, or hits allowed, because that also has a negative impact on wins. These two variables will not be included. The remaining variables have negative parameter estimates and also have a negative impact on wins, so they will also not be included. Those variables include TEAM_BATTING_SO, TEAM_PITCHING_HR, TEAM_PITCHING_BB, TEAM_PITCHING_SO, TEAM_FIELDING_E, and TEAM_FIELDING_DP. That leaves us with the same 6 predictor variables, mentioned in the previous paragraph, going into our parsimony model. Our parsimony model can be seen in Figure 2.0 below.

Figure 2.0: Parsimony Model

```
Call:
lm(formula = TARGET_WINS ~ TEAM_BATTING_H + TEAM_BATTING_2B + 
    TEAM_BATTING_3B + TEAM_BATTING_HR + TEAM_BATTING_BB + TEAM_BASERUN_SB, 
    data = mb)

Residuals:
    Min      1Q  Median      3Q     Max 
-72.983  -8.752   0.512   8.883  73.893 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)     3.567960   3.977181   0.897   0.3698    
TEAM_BATTING_H  0.029914   0.003498   8.552  < 2e-16 ***
TEAM_BATTING_2B 0.017374   0.009063   1.917   0.0554 .  
TEAM_BATTING_3B 0.106487   0.017401   6.120 1.10e-09 ***
TEAM_BATTING_HR 0.069798   0.007896   8.840  < 2e-16 ***
TEAM_BATTING_BB 0.023143   0.002913   7.945 3.02e-15 ***
TEAM_BASERUN_SB 0.038093   0.004586   8.307  < 2e-16 ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.82 on 2269 degrees of freedom
Multiple R-squared:  0.2327,	Adjusted R-squared:  0.2307 
F-statistic: 114.7 on 6 and 2269 DF,  p-value: < 2.2e-16
```

From our parsimony model, we are able to write out the following mathematical expression for this model:

TARGET_WINS = 3.568 + 0.0299(TEAM_BATTING_H) + 0.0174(TEAM_BATTING_2B) + 0.1065(TEAM_BATTING_3B) + 0.0698(TEAM_BATTING_HR) + 0.0231(TEAM_BATTING_BB) + 0.0381(TEAM_BASERUN_SB)

This model is also interpreted the same way as the previous model and for the model to work, the TARGET_WINS total must be approximately equal to 81 wins. In this case, the model does work because the total equals 80.75669 wins, which is approximately 81 wins. The differences between this model and our base model are that there are obviously less predictive variables included in the model and also all of the parameter estimates are positive values. The adjusted r-squared value of 0.2307 is actually less than our base model, which was a change that I did not expect. In becoming simpler, the model fit got slightly worse, but not to the point where the model fit is not fine.

Task 3: Contextual Modeling Approach

For this task, we created a revised Moneyball data set with all of our predictor variables converted to reflect production per-game averages of each variable. To create these predictor variables, we divided all of our variables by 162 (the number of games in a season). We then needed to create flag variables for each of our new per-game predictor variables. To create these flag variables, the MLB 2017 per game averages for teams in the league were taken into account. The team with the highest per game average for each variable was the number that was used to determine the values consistent with the modern era, as well as the values out of range of the modern era. All values that exceeded the highest 2017 per game average for each statistic were listed as “out of the range of modern era” and all other values were listed as “consistent with modern era.” We will make a model that we refer to as the contextual model, which was fit with the records that were listed out of the range of modern era. This model can be seen in Figure 3.0 below. In Figure 3.1 below, we found the mean of all other records, which were the records that were consistent with the modern era, by their respective variables.

Figure 3.0: Contextual Model for Values Out of Range of Modern Era (Modern=1)

```
Call:
lm(formula = mbpg$twpg ~ (mbpg$m.bhpg == 1) + (mbpg$m.b2bpg ==
    1) + (mbpg$m.b3bpg == 1) + (mbpg$m.bhrpg == 1) + (mbpg$m.bbbpg ==
    1) + (mbpg$m.bhbppg == 1) + (mbpg$m.bsopg == 1) + (mbpg$m.brsbpg ==
    1) + (mbpg$m.brcspg == 1) + (mbpg$m.fepg == 1) + (mbpg$m.fdppg ==
    1) + (mbpg$m.pbbpg == 1) + (mbpg$m.phpg == 1) + (mbpg$m.phrpg ==
    1) + (mbpg$m.psopg == 1))

Residuals:
     Min       1Q   Median       3Q      Max
-24.6302  -7.5942   0.1067   6.4721  25.3698

Coefficients: (2 not defined because of singularities)
                       Estimate Std. Error t value
(Intercept)              79.630      1.068  74.583
mbpg$m.bhpg == 1TRUE      8.868      2.657   3.338
mbpg$m.b2bpg == 1TRUE    -1.091      5.703  -0.191
mbpg$m.b3bpg == 1TRUE    -1.132      4.115  -0.275
mbpg$m.bhrpg == 1TRUE    -9.773      6.330  -1.544
mbpg$m.bbbpg == 1TRUE     5.830      5.841   0.998
mbpg$m.bhbppg == 1TRUE       NA         NA      NA
mbpg$m.bsopg == 1TRUE     3.263      1.952   1.672
mbpg$m.brsbpg == 1TRUE   10.491      3.492   3.005
mbpg$m.brcspg == 1TRUE   -3.563      2.024  -1.760
mbpg$m.fepg == 1TRUE    -10.643      2.512  -4.236
mbpg$m.fdppg == 1TRUE    -4.925      3.463  -1.422
mbpg$m.pbbpg == 1TRUE     9.067      4.970   1.824
mbpg$m.phpg == 1TRUE     -2.960      5.136  -0.576
mbpg$m.phrpg == 1TRUE     9.142     12.154   0.752
mbpg$m.psopg == 1TRUE        NA         NA      NA
                       Pr(>|t|)
(Intercept)             < 2e-16 ***
mbpg$m.bhpg == 1TRUE    0.00103 **
mbpg$m.b2bpg == 1TRUE   0.84858
mbpg$m.b3bpg == 1TRUE   0.78355
mbpg$m.bhrpg == 1TRUE   0.12441
mbpg$m.bbbpg == 1TRUE   0.31954
mbpg$m.bhbppg == 1TRUE       NA
mbpg$m.bsopg == 1TRUE   0.09632 .
mbpg$m.brsbpg == 1TRUE  0.00305 **
mbpg$m.brcspg == 1TRUE  0.08013 .
mbpg$m.fepg == 1TRUE   3.65e-05 ***
mbpg$m.fdppg == 1TRUE   0.15677
mbpg$m.pbbpg == 1TRUE   0.06978 .
mbpg$m.phpg == 1TRUE    0.56512
mbpg$m.phrpg == 1TRUE   0.45294
mbpg$m.psopg == 1TRUE        NA
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 10.34 on 177 degrees of freedom
  (2085 observations deleted due to missingness)
Multiple R-squared:  0.3212,   Adjusted R-squared:  0.2713
F-statistic: 6.441 on 13 and 177 DF,  p-value: 6.192e-10
```

Figure 3.1: Mean Values Consistent with Modern Era (Modern=0)

Variable	Mean Value
mbpg$m.bhpg (Batting H per game)	0.7891037
mbpg$m.b2bpg (Batting 2B per game)	0.9907733
mbpg$m.b3bpg (Batting 3B per game)	0.5158172
mbpg$m.bhrpg (Batting HR per game)	0.9964851
mbpg$m.bbbpg (Batting BB per game)	0.9450791
mbpg$m.bhbppg (Batting HBP per game)	NA
mbpg$m.bsopg (Batting SO per game	NA
mbpg$m.brsbpg (Baserun SB per game)	NA
mbpg$m.brcspg (Baserun CS per game)	NA
mbpg$m.fepg (Fielding E per game)	0.2728471
mbpg$m.fdppg (Fielding DP per game)	NA
mbpg$m.pbbpg (Pitching BB per game)	0.8356766
mbpg$m.phpg (Pitching H per game)	0.7003515
mbpg$m.phrpg (Pitching HR per game)	0.9947276
mbpg$m.psopg (Pitching SO per game)	NA

All records that appeared as N/A were because those records were almost 0 due to having so many values that fell in the opposite category or contained a lot of missing data. From our contextual model, we are able to write out the following mathematical expression for this model:

TARGET_WINS = 79.630 + 8.868(BATTING_H per game) – 1.091(BATTING_2B per game) – 1.132(BATTING_3B per game) – 9.773(BATTING_HR per game) + 5.830(BATTING_BB per game) + 3.263(BATTING_SO per game) + 10.491(BASERUN_SB per game) – 3.563(BASERUN_CS per game) – 10.643(FIELDING_E per game) – 4.925(FIELDING_DP per game) – 2.960(PITCHING_H per game) + 9.142(PITCHING_HR per game) + 9.067(PITCHING_BB per game)

The parameter estimates came out as values above 1 or below 1 unlike the last 2 models. The parameter estimates also did not make theoretical sense for many of the predictor variables. This model is also interpreted the same way as the previous two model and for the model to work, the TARGET_WINS total must be approximately equal to 81 wins. Despite the issues we have with some of the parameter estimates, the model still works because the total equals 80.75669 wins, which is approximately 81 wins. The adjusted r-squared value for this model is 0.2713, which is an improvement over the parsimony model and still less than the base model.

Task 4: Exporting and Comparing Model Performance

This task serves as a way to summarize the prediction results for the 5 models we built for this project. These prediction results are in terms of average absolute deviation. A perfect fitting model has an average absolute deviation of 0. Our prediction results can be seen in Figure 4.0 below.

Figure 4.0: Prediction Results for the 5 Models in Terms of Average Absolute Deviation

Model	Average Absolute Deviation
Y_MEAN	12.80327
Y_RANDOM	38.1371
Y_BASE	10.4904
Y_PARSIMONY	10.75724
Y_CONTEXTUAL	99.04842

Our results varied in relation to the benchmarks of a perfect model, random model, and a mean (constant term only) model. None of our models were perfect but the base and parsimony models were not terribly far off. As mentioned earlier, our contextual model came out funky and odd and performed poorly in comparison to all other listed models. This could be due to human error, of course. Our base and parsimony models did, however outperform both the mean and random models. The mean model was the closest in relation to our base and parsimony models. The takeaway here is that it is hard to achieve a perfect model, but taking the necessary steps to the quality execution of different model approaches can lead to models that perform well and mostly accurately.

Task 5: You Are Up to Bat

For this task, I decided to fit a model that incorporated parts of the other models we created. For this model, we will be using the data set we used for the base model. I also used the same predictor variables as the parsimony model because I believe they correlate most to winning in baseball. Those predictor variables are also the major basis of the Moneyball book and movie, as “getting on base” was viewed as the variable that would translate into wins for the Oakland Athletics. Our predictor variables all have to do with getting on base, and the result of getting on base is being in position to score runs, and it turn, scoring runs gives you a better chance to outscore your opponent and win games. I also wanted to see how this specific model would react to the incorporation of the per-game production variables, so I translated our predictor variables into per-game production variables similar to the ones we touched on in our contextual model. Here is a breakdown of the 6 predictor variables used in this model: TEAM_BATTING_H_PG (team batting hits per game), TEAM_BATTING_2B_PG (team batting doubles per game), TEAM_BATTING_3B_PG (team batting triples per game), TEAM_BATTING_HR_PG (team batting home runs per game), TEAM_BATTING_BB_PG (team batting walks per game), and TEAM_BASERUN_SB_PG (team baserunning stolen bases per game). I used the same procedures as the procedures in the base model for all missing values, outliers, null values, and correlations. I will be using adjusted r-squared as the criteria for model performance. I will say that I would rather have a slightly underperforming model that makes more sense and is simpler to understand and interpret. This model can be seen in Figure 5.0 below.

Figure 5.0: You are Up to Bat

```
Call:
lm(formula = TARGET_WINS ~ TEAM_BATTING_H_PG + TEAM_BATTING_2B_PG + 
    TEAM_BATTING_3B_PG + TEAM_BATTING_HR_PG + TEAM_BATTING_BB_PG + 
    TEAM_BASERUN_SB_PG)

Residuals:
    Min      1Q  Median      3Q     Max 
-72.983  -8.752   0.512   8.883  73.893 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          3.5680     3.9772   0.897   0.3698
TEAM_BATTING_H_PG    4.8461     0.5667   8.552  < 2e-16
TEAM_BATTING_2B_PG   2.8145     1.4683   1.917   0.0554
TEAM_BATTING_3B_PG  17.2510     2.8189   6.120 1.10e-09
TEAM_BATTING_HR_PG  11.3073     1.2791   8.840  < 2e-16
TEAM_BATTING_BB_PG   3.7492     0.4719   7.945 3.02e-15
TEAM_BASERUN_SB_PG   6.1710     0.7429   8.307  < 2e-16

(Intercept)           
TEAM_BATTING_H_PG  ***
TEAM_BATTING_2B_PG .  
TEAM_BATTING_3B_PG ***
TEAM_BATTING_HR_PG ***
TEAM_BATTING_BB_PG ***
TEAM_BASERUN_SB_PG ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.82 on 2269 degrees of freedom
Multiple R-squared:  0.2327,	Adjusted R-squared:  0.2307 
F-statistic: 114.7 on 6 and 2269 DF,  p-value: < 2.2e-16
```

We are able to write out the following mathematical expression for this model:

TARGET_WINS = 3.568 + 4.8461(TEAM_BATTING_H_PG) + 2.8145(TEAM_BATTING_2B_PG) + 17.2510(TEAM_BATTING_3B_PG) + 11.3073(TEAM_BATTING_HR_PG) + 3.7492(TEAM_BATTING_BB_PG) + 6.1710(TEAM_BASERUN_SB_PG)

For this model to work, our TARGET_WINS total must be equal to the same number from our previous models, which was approximately 81 wins. In this case, the TARGET_WINS total is 80.79086, which is approximately 81 wins, and therefore this model works. The adjusted r-squared is 0.2307 so it is equal to our parsimony model and less than our base model and contextual model.