Bingo Bonus – GLM of champion model was attempted as was Weka for selection and imputation, and program (partially) re-created in Python. Attempted - possible 60 pts; but wasn't able to complete Python program.

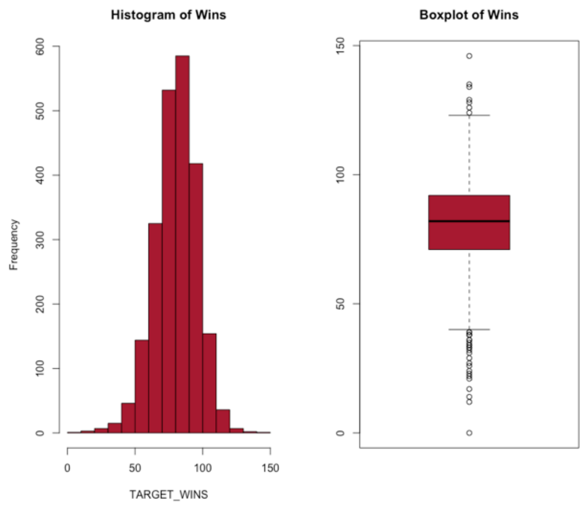
Introduction

The goal of this exercise is to predict the number of wins a baseball team can expect given a set of batting and pitching metrics. The data were collected between 1871 and 2006. Each record represents a single team in a single year. The data have been adjusted, where needed, to conform to a 162-game season. Using strictly OLS regression, model selection, and the data preparations of our choice, we are tasked with delivering the best model we can devise. A scoring program, which can be run independently is also provided for use in evaluating our final model.

Data Exploration

The data set is made up of close to 2300 observations of 17 metrics. The data contain a number of missing values, which means either those rows need to be dropped, or the missing values need to be filled in somehow. Since only 191 of the records are complete, we will be filling in the missing values. As seen in Figure 1, variable we want to predict is close to being normally distributed (skewness approximately -.4), which means the data meet the required assumption of normal distribution. There are numerous outliers in the data; data transforms will be done on most of the predictor variables, to see if normalization of those data has a positive effect on model building.

Figure 1



Scatterplots of the data (Figures 2-4) show there are correlations between wins and several batting metrics, while running and pitching metrics show no particular correlation to wins. We also see some of the batting metrics seem to be correlated with each other, like home runs and walks, so we will need to watch out for that collinearity in the data used when building our models.

Figure 2 - Batting Correlations

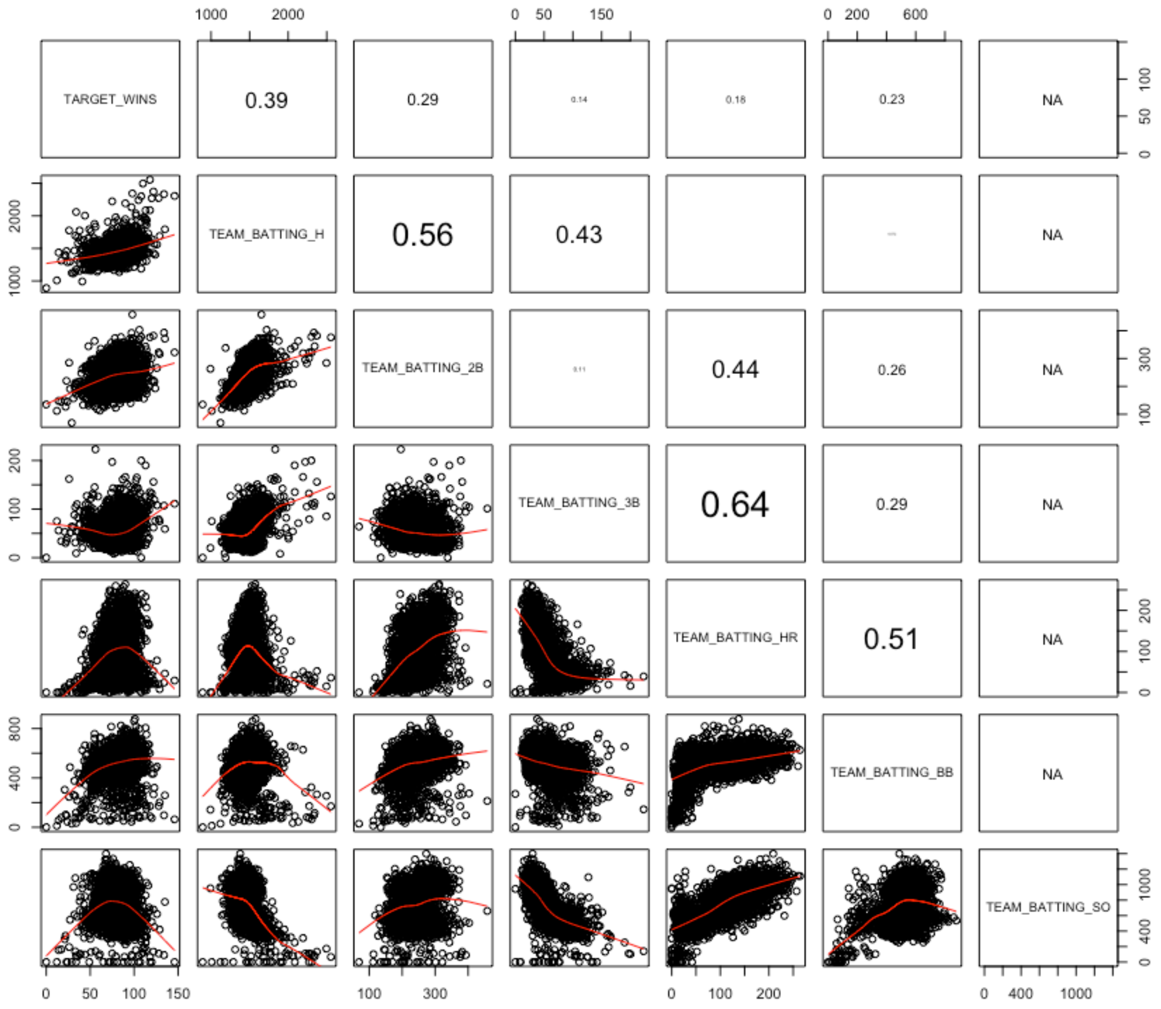


Figure 3 – Pitching Correlations

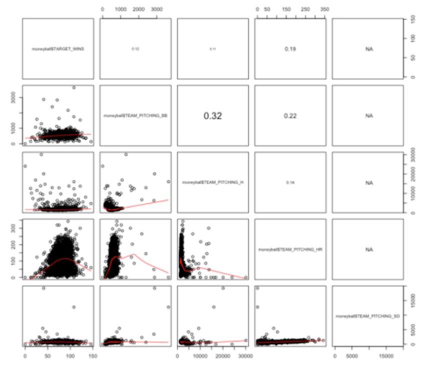
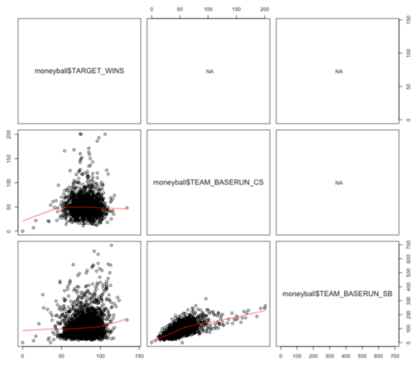


Figure 4 - Baserunning Correlations



Data Preparation

Given the number of missing values, dropping incomplete records was not an option, so imputation was performed for the missing data. Common methods for imputation are filling in the missing values with the mean, median or mode of the values that are present. Mean and median were tried; mean gave the best results and was used the final model. Per the class discussion Q&A forum, observations are not to be dropped, which meant the “-Inf” values created by taking the log of a 0-valued variable had to be handled. This was done by setting any “-Inf” values to zero after the log transformations.

Since the target variable has a skewness of ~-.4, which is very near the +/- .5 limit for being considered normally distributed, transformations of the target variable had the potential to be helpful. I experimented with taking both the log and the square root of the target; ultimately, the log method was selected. Adding flags to indicate if a variable was imputed was also tried, to see if there were useful relationships in the patterns of missing data; flags failed to show significant value for this exercise.

Model Construction and Selection

Numerous models were created and evaluated. Specific models were created to directly compare the results of using mean versus median as the method for imputing missing data. Other models were built to evaluate the different combinations of variables, and the differences in target variable transformations. For each model, the summary data, the VIF information, AIC and MSE values are written. While work was attempted to move in a logical progression, there was an exploratory element to creating then testing the different models to see what seemed to be working best. Finally, the primary measures: adjusted-R2, AIC and MSE were written out for all the models to facilitate comparisons. Table 1 is the sorted version of that output.

Table 1 – Metrics for models in the R Program
** indicates a model from the instructor's sample code*

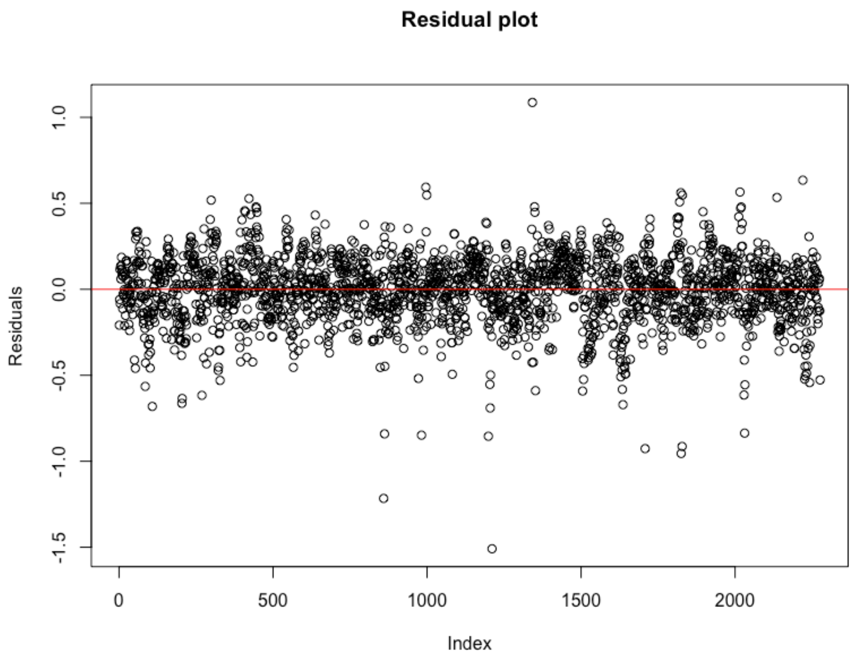
formula name	***adj_r2***	***AIC***	***MSE***
tam_all_var	0.46	-1441.54	0.03
rfe_model	0.34	-1008.58	0.04
tam_log_step	0.34	-1001.78	0.04
weka_vars_log	0.22	-623.28	0.04
tam_sqrt_step	0.33	5227.17	0.58
weka_vars_sqrt	0.22	5563.54	0.67
stepwise_flag	0.41	17832.96	144.93
*stepwise**	0.31	18174.45	169.73
*model3**	0.29	18174.45	169.73
weka_m5p	0.31	18187.20	170.83
tam_sub1	0.30	18200.16	172.26
tam_sub_log	0.30	18200.16	172.26
tam_term	0.30	18209.12	172.64
weka_RF	0.30	18211.90	173.00
tam_mean	0.30	18217.91	173.46
tam_drop_VIF	0.30	18217.91	173.46
tam_med	0.29	18239.96	175.15
tam_step_1	0.28	18287.07	178.97
weka_no_trans	0.22	18454.76	193.50
tam_term2	0.21	18500.36	196.38

The first 4 models are in contention for selection, all have low AIC and MSE values and comparative high adjusted-R^2 values. The first model has the highest adjusted-R^2, and that might seem compelling, but we know from our reading that when comparing model performance on training data, we want to use MSE and AIC/BIC as the measures of predictive power, not adjusted-R^2.

Looking at the models' specifics, we see that "tam_all_var" uses 19 possible explanatory variables, "rfe_model" uses 5, and "tam_log_step" uses 9. For the sake of interpretability, I am eliminating "tam_all_var" from consideration for the final model. A number of the variables in that model are the log values of the basic metrics; log values are poorly understood by most people, and are an unnecessary complexity in this effort.

The stepwise model which included the log variables as part of its formula (tam_step_log) has basically the same results as the model based on RFE variable selection. However, the RFE model has fewer variables which makes it easier to explain and interpret. As a final check, I evaluated the residuals for the candidate "rfe_model", as shown in Figure 5.

Figure 5



The residuals are roughly centered around the 0.0 line in red. There is no obvious pattern to the residuals, which means that there isn't some important predictor missing from the model and therefore leaking into the residuals. So, I am going to select my "rfe_model" model as the champion. It is given by the formula:

log_wins_predicted <- 1.67153122 +

*-0.00049086 * number of walks by batters +*

*0.00075988 * number of base hits +*

*0.30981277 * log (number of walks by batters) +*

*-0.00003232 * number of hits allowed by pitchers +*

*-0.00012413 * number of fielding errors*

Coefficients for the formula make sense intuitively; walking a batter may prevent a strong hitter from getting a double, or better, so a negative coefficient seems rational for that coefficient. Negative coefficients for pitchers allowing hits, and fielders making errors again, seem reasonable. The coefficient for hits, again seems reasonable, as we know from the movie "*Moneyball*", getting base hits is important. The log of the number of walks is more obscure, and can be thought of as minor tweak. For example, if a team has 590 walks for the season, the log(walks) is equal to 6.38. The formula subtracts .2896 for the walks, then adds back 1.971. If a team has fewer walks, say, 74, then .0363 is subtracted for the walks and 22.92 gets added back. The log seems to adjust for how often a team has a batter walked.

Conclusion

When dealing with data which has a skewed target variable, it is useful to transform the variable then fit the models. In this case, the log transformation was used, but other transformations are possible. Simpler models can yield results which work as well as more complex models. Since the AIC value of a model penalizes complexity, it is a valuable way to compare models.

In this case, given the (comparatively) small size of the data set, models were compared based only on their performance on training data; splitting into training and test sets is problematic on small data sets. A fairly simple model, using only 5 out of a possible 27 variables yielded results nearly as good as models using all the variables. Given the collinearity in the data, eliminating some features made sense. Using recursive feature elimination (RFE) to select the variables to use ultimately gave a model with good AIC and MSE values and an acceptable adjusted-R^2 value. This model has the added advantage of being easier to explain due to its simplicity, and has coefficients that make sense, at least to me as a non-baseball person. The simple, RFE model is being submitted as my champion model.

Bingo-Bonus Write up

1) Using GLM instead of LM I re-fit the model and got 0 differences in coefficients or AIC/MSE values. See section at the end of R code for full details. Coefficient (rounded for the sake of printing) are:

```
>round(coef(modelbb),5)

    (Intercept)   TEAM_BATTING_BB    TEAM_BATTING_H log_TEAM_BATTING_BB   TEAM_PITCHING_H   TEAM_FIELDING_E

      1.67153        -0.00049          0.00076          0.30981         -0.00003         -0.00012

> round(coef(glm_mod),5)

    (Intercept)   TEAM_BATTING_BB    TEAM_BATTING_H log_TEAM_BATTING_BB   TEAM_PITCHING_H   TEAM_FIELDING_E

      1.67153        -0.00049          0.00076          0.30981         -0.00003         -0.00012
```

2) Decision Trees – I used RandomForest for imputation of values, see the Bonus Bingo section of the R code, as well as using Trees in Weka for selection. I tried Random Forest and M5P tree routines for feature selection. These map to the weka_RF and weka_m5p models in the main body of the R code. I included them as part of the collection of models to evaluate.

3) Recreation in Python – got as far as I could. Please see included Williams_bonus_bingo.py file.