FTW: Documentation

What did the group set out to do?

The purpose of our project was to create a generalizable model that would be able to accurately predict a PGA player's net worth in future seasons. The primary benefit of this model would be to create good bets on a player's net earnings.

What was done?

About the Data

The first step in this process was to find pre-existing data that could be used to predict a player's net worth. We found a dataset on Kaggle that included a player's net worth (in dollars) along with several descriptive statistics about the player's game. Below is an index of the performance statistics provided in the Kaggle dataset, and the definition of each of the statistics:

1. Season (categorical, units = none): indicates the year in which the other statistics were observed
2. PuttingAverage (quantitative, units = none): indicates the average number of putts made per round
3. DrivingDistance (quantitative, units = yards): indicates the distance of the average drive off the tee
4. DrivingAccuracy (quantitative, units = none): indicates the percentage of drives that hit the fairway, taking on a value of 0 to 100
5. ScoringAvg (quantitative, units = strokes): indicates the average number of strokes a player will need to get the ball in the hole
6. Scrambling (quantitative, units = none): indicates the percent of times that a player is able to "scramble" and make par when off the green
7. OnePutts (quantitative, units = count): indicates the number of one-putts that a player has in all holes in all rounds for a season
8. TwoPutts (quantitative, units = count): indicates the number of two-putts that a player has in all holes in all rounds for a given season
9. ThreePutts (quantitative, units = count): indicates the number of three-putts that a player has in all holes in all rounds for a given season
10. ProximityToHole (quantitative, units = inches): indicates the average distance to the hole, in inches, when an approach shot from any distance hits the green
11. GreensFringeInReg (quantitative, units = none): indicates the percent of approach shots that land on the green or on the fringe
12. SandSave (quantitative, units = none): indicates the percent of successful scrambles that occur from a bunker
13. **OfficialMoney** (quantitative, units = dollars): indicates the net worth of an individual player in the PGA tour

FTW: Documentation

Data Wrangling Techniques

We made the decision to use symbolic regression to create our model. The next part of our project, therefore, dealt with data wrangling.

Data wrangling, formally defined, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics (Wikipedia). Data wrangling is an important step to any regression-based project, because the data provided is almost *never* immediately ready for analysis.

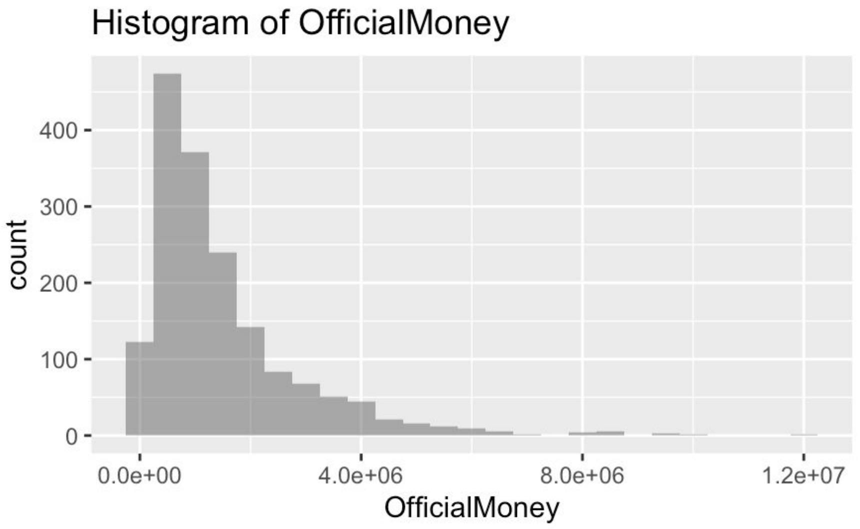
We had to make a series of changes to our data in order to get it ready for regression. Some of the changes that we needed to make (a non-exhaustive list) are:

1. Transform inputted values from strings to doubles, while properly coercing NA values
2. Transforming missing/deleted values and handling abnormal cases
3. Transforming our response variable to logOfficialMoney (the scale of OfficialMoney was too large for good analysis)
4. Swapping elements into appropriate positions to run symbolic regression properly

Decision to Transform the Response Variable

Before running a regression, there are several conditions that need to be checked, one of which is making sure that the outcome variable (in this case, OfficialMoney) is **normally distributed.** Practically, this means that the distribution of OfficialMoney should be unimodal and symmetric *before* we continue with our analysis.

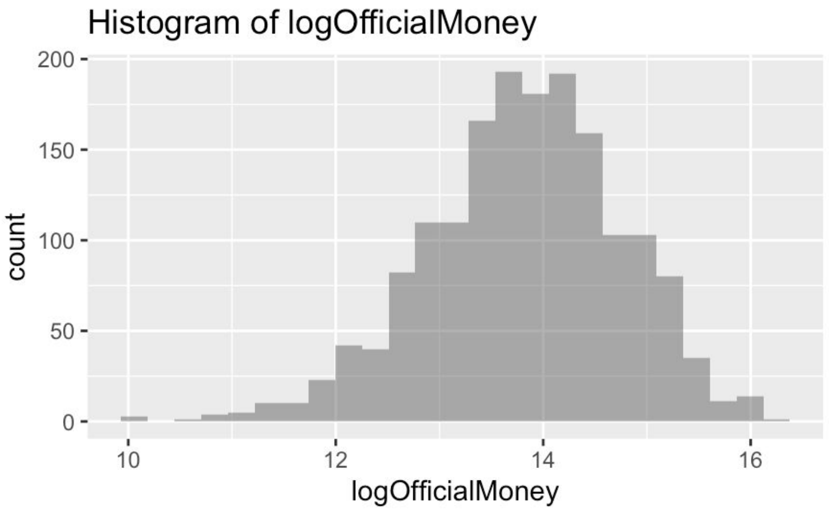
We first looked at the distribution of OfficialMoney through a histogram. We have included the histogram of OfficialMoney below:



FTW: Documentation

We noticed that this distribution is clearly not Normal; in fact, it is extremely right skewed. The right skewness of the distribution makes sense since the best players on the Tour make significantly greater than most people on the Tour. Thus, this variable would benefit from a transformation.

We tried a series of different transformations with our variable and figuring out ways to make the distribution of the variable more Normal. Ultimately, we noticed that taking the logarithm of the player's money was an effective way to make the distribution Normal:



As evidenced by the histogram above, the distribution is clearly more symmetric and is unimodal with a mean/median value of approximately e^14 = $1,202,604.28.

Symbolic Regression: The Details

Eliminating Superfluous Variables

We removed variables from the regression that, from experience, we knew would be insignificant to the model, thus decreasing the complexity of the model. In the end, we kept 9 different variables. The table below shows which variables were represented in our code and their notation in our regression model:

Value in Dataset	Representation in Symbolic Regression
Driving Accuracy	x1
Putting Average	x2
Scrambling	x3
Driving Distance	x4
Sand Saves	x5
Putting Average	x6

FTW: Documentation

Greens/Fringe in Regulation	x7
Proximity to Hole	x8
Scoring Average	x9

Testing/Training Set

Our dataset consisted of nearly 2,000 observations after we finished transforming the data. We decided that 25% of the data should be used for training and 75% of the data should be used for testing.

However, we couldn't simply set aside the first 100 observations since our data was ordered chronologically. In order for the regression to work, we needed to make sure that our data was *randomly sampled across all class years.* We thus generated a random sample with each element having p_selection=0.25 for the training data and p_selection=0.75 for the testing data. While it is possible that there were some overlaps between the testing and training data, we argue that the overlap is not significant enough to decrease the validity of our training set.

Method of Symbolic Regression

We decided to use tree-based symbolic regression, adapted from Lee Spector's code for tree-based symbolic regression. Most features of the regression were the same as from Spector's original code, including mutation, crossover, and the overall process of evolution.

Changes that were made included:

1. We adapted the error function to take in 9 different parameters (corresponding to the 9 variables that we decided to include in the regression)
2. We changed the tournament selection size to 5. After testing a variety of options (including lexicase), we found that tournament selection with a tournament size of 5 was most effective at selecting the best individuals from each generation.
3. We added a few constraints to our evolve method:
    a. Generation constraint: We made the program stop running at 40 generations. We found that after 40 generations the gains made to the program quality plateaued and that the code started becoming more complex and difficult to understand.
    b. Error constraint: Rather than have the program go to an error of 0, we let the evolution stop if the error was less than 250. Although, obviously, the lower the error the better on our training data, it is not essential for the predictions to be entirely accurate for the model to be reasonable. As a result, we decided that 250 was a "reasonable" place to stop our evolution.

FTW: Documentation

What were the results?

We found that our program did a ***reasonably good job at predicting middle-of-the-pack players,*** while we simultaneously found that our program did a ***poor job at predicting the income of top-tier or lower-tier players.*** Please check the results folder for three examples of program outputs that saves our list as a csv-file. Most of the errors were <1 on a log scale, except for the top-tier individuals as we mentioned. (Apologies that the names are all separated as commas- it doesn't detract from the overall analysis so we decided to not worry about it). Also, look at the presentation for examples of "good individuals" that we found in our data. As we can see, the "good players" were both very close to the median value of 1.2 million when applying the antilog transformation!

An important pattern we picked up on when transforming our equations from postfix to infix was that many of the "scrambling" parameters appeared more centrally in the program (GreensFringeInReg, SandSave), suggesting that these parameters in particular are important for determining a professional player's projected earnings.

It was interesting to find that our model did well on the players in the middle of the pack but poorly on "extreme individuals." We hypothesize that this is partially a result of tournament selection: the fittest programs did very well on a majority of individuals quicker. In this case, the distribution of logOfficialMoney showed that the majority of the players were at the middle of the pack. With lexicase selection, which focuses more on specialization rather than overall error, it is possible that we might have found a function that was more effective at predicting individuals at either end of the extremums.

Future Directions?

The preliminary results of our program were, indeed, quite encouraging, as we found that it works reasonably well on middle-of-the-pack players. Money is an incredibly hard attribute to predict, and to be able to find a function that does a reasonable job of predicting this is an achievement in its own sense.

However, there is certainly room for improvement. Here are a few potential directions that this project could head in the future.

1. Use propel to make a different version of our regression system, and implement uniform mutation through that in an attempt to find a more efficient program
2. Implement a lexicase-tournament selection combo method to try and capture the more extreme values in our data
3. Try removing more of the predictors to see if a simpler model comes out
4. Actually implement size-push to get simpler programs and reduce the amount of plateauing in the evolution around 40 generations