**IST 687**

Final Project: Poker Game

**Truth Tellers**

Beau Spratt

Nicholas Nguyen

Marianne Garbo

Becca Davis

**Table of Contents**

**Introduction and Background**  **………..…………..………… 2**

**Pluribus ....……..…………..………… 2**

Analysis 1 .…………..………….. 2

Business Questions ………..…….. 3

Cleanup and Preparation ………..…….. 4

Methods of Analysis ………..…….. 4

Conclusion ………..…….. 6

Analysis 2 …………..………….. 6

Business Questions ………..…….. 6

Cleanup and Preparation ………..…….. 6

Methods of Analysis ………..…….. 11

Conclusion ………..…….. 17

**One Dollar Comparison (Kaggle Data) ....……..…………..………… 17**

Business Questions ………..…….. 19

Cleanup and Preparation ………..…….. 19

Methods of Analysis ………..…….. 22

Conclusion ………..…….. 24

**Citations**  **....……..…………..………… 25**

**Appendices**  **....……..…………..………… 25**

Code: Pluribus Analysis 1 ………..…….. 25

Code: Pluribus Analysis 2 ………..…….. 35

Code: One Dollar Comparison Analysis ………..…….. 65

**Introduction and Background**

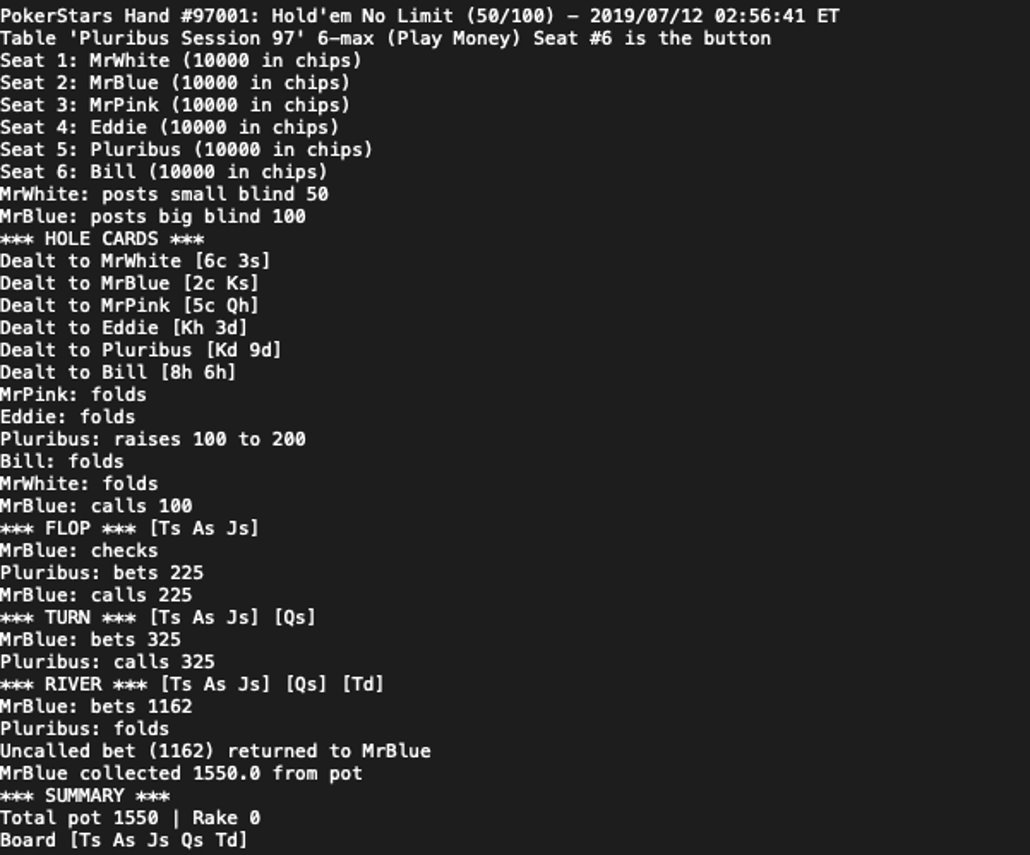
Poker is a card game where players wager on who has the best combination of 5 cards . Bluffing is a strategy successful poker players engage in. It is an act of deception aimed at making your weak hand look stronger than it is with the intent of getting your opponent to fold. The goal of bluffing is to win irrespective of the hand you have been dealt. Here we apply our data science skills to answer questions around bluffing. Previous reports have studied to see if bluffing can be predicted by reviewing body language and facial imagery for “tells”. Attempts included taking thermal images of players faces, scrapping face data from televised poker games, and sensors that detect moisture in players hands

Here, we take a look at historic poker datasets and consider them for bluffing statistics by applying data science. The datasets employed in the exploration came from historical poker hands of human vs human Kaggle.com [1], historical poker hands of human vs an AI (Pluribus) Kevinwang.com [2] and odds data from betandbeat.com [3]. The group considered these large datasets to examine the effectiveness of bluffing.

Pluribus: Analysis 1

**Data Dictionary**

* Dataset [2] Human Vs AI (Pluribus) 491 ((17 attributes and 491 observations (pulled one group of players)).
* Originally a log format:



* **Transformed Fields**

| **Column** | **Description** |
| --- | --- |
| Game | id of the game |
| Winner | name of winner |
| B.Blinder | player who is big blind |
| B=W | was the blinder the winner? |
| Bluff? | was the winning hand a bluff? |
| R.W.Odds | rounded odds of the winner |
| Pot | size of the pot |
| Flop | flop hand |
| Turn | turn hand |
| River | river hand |
| Has Flop | game has a flop? |
| Pluribus | AIs hand |
| Bill | Bill hand |
| Eddie | Eddie hand |
| Budd | Budd hand |
| Gogo | Gogo hand |
| MrWhite | MrWhite hand |

**Business Questions**

Historic porker datasets were examined to answer some questions on the data when it comes to bluffing. Did players bluff more vs an AI or Human? Does the Pot or Flop affect bluffing? Was bluffing a good idea (did they earn chips)? When is the best time to bluff (blind, flop, river)? Who Bluffs the most? Can we predict when a game will be won with a bluff?

**Cleanup and Preparation**

Datasets from the above sources were merged from text and csv files then used to pull in odds and used to create a bluff column for each game in order to answer our questions about the data. The [2] dataset was parsed from a log format and hand data cleaned up for pulling in the correct odds. Unwanted characters were removed. Columns that were not needed to answer the questions we had were dropped. Data was also checked for any unnecessary data, irregular data or outliers.

**Methods of Analysis**

With this new data we ran comparative reports using pivot tables and calculated statistics including wins and bluff wins by player, total hands won by bluffing including when there was a flop or not, who was the big blind, and considering the size of the pot. Finally, we ran a SVM model to see if a bluff could be predicted in the game

**Comparison of Humans and AI:**

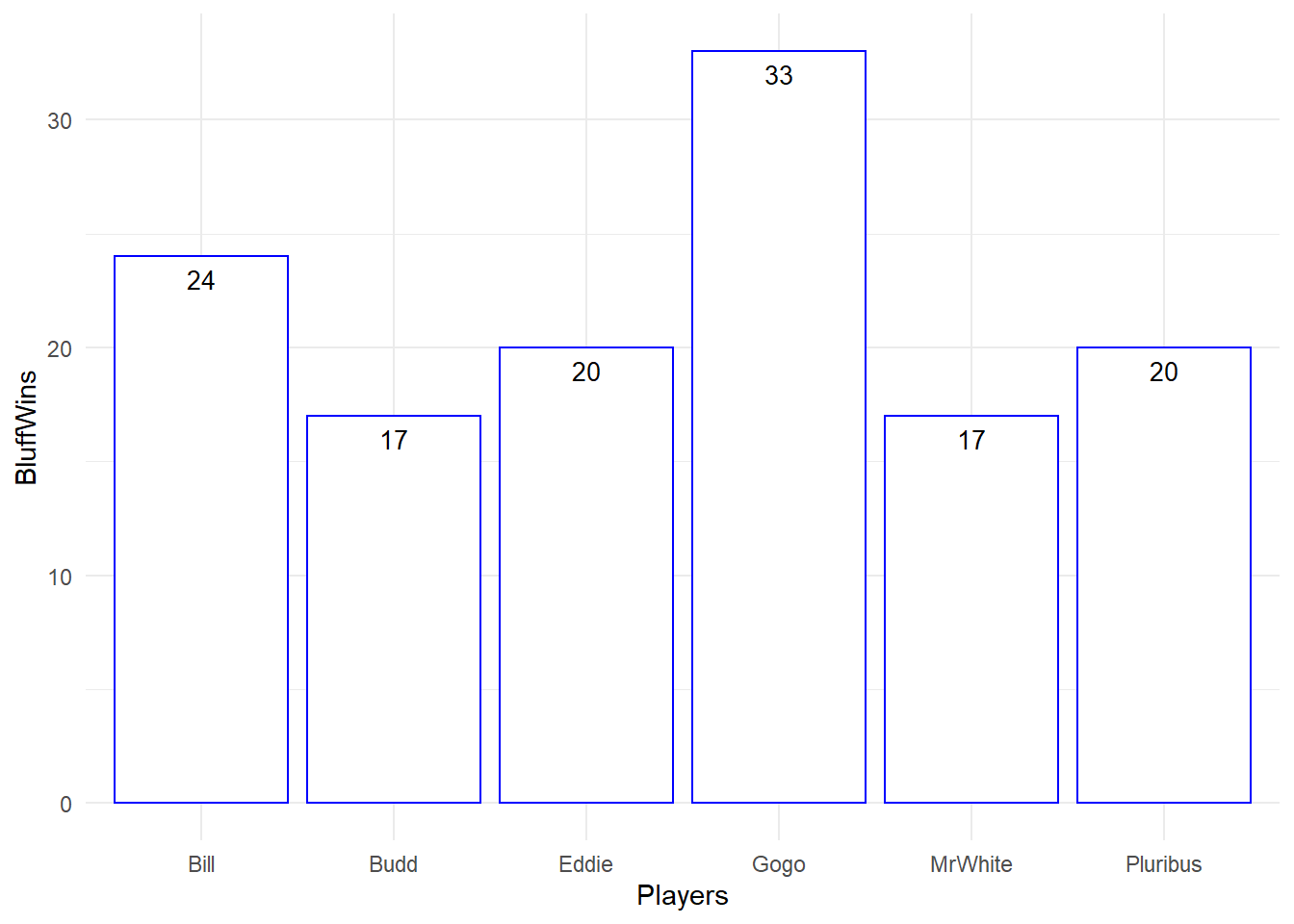
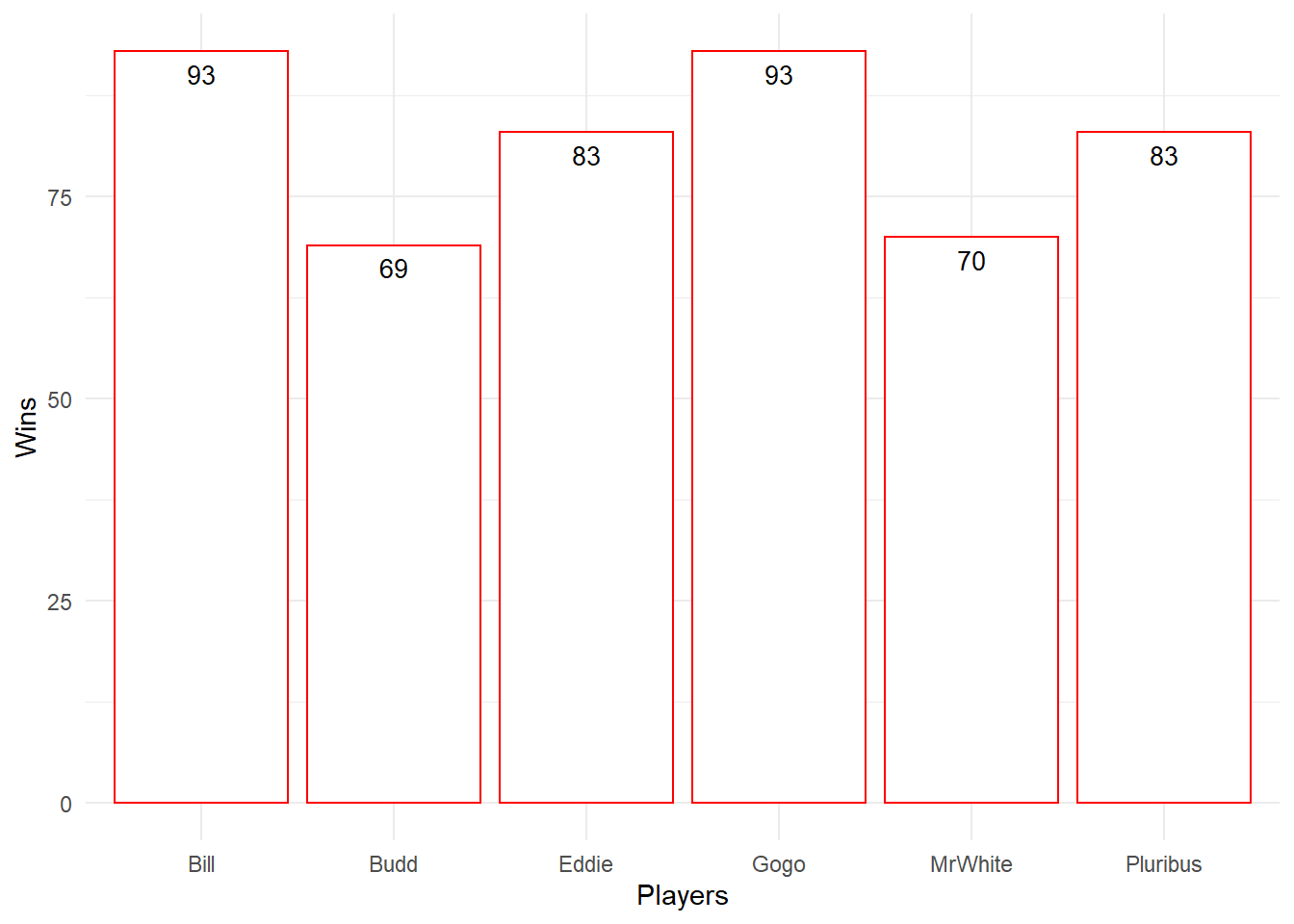
Bluffing occurred more vs humans 62% of the time vs 27% of the time but equal seat dataset analysis is needed since players appear to bluff a lot more when 3 players were in a game vs 6 players. We tried to extrapolate the seats by merging in different odds at different seat counts but the amount of error introduced was too great.

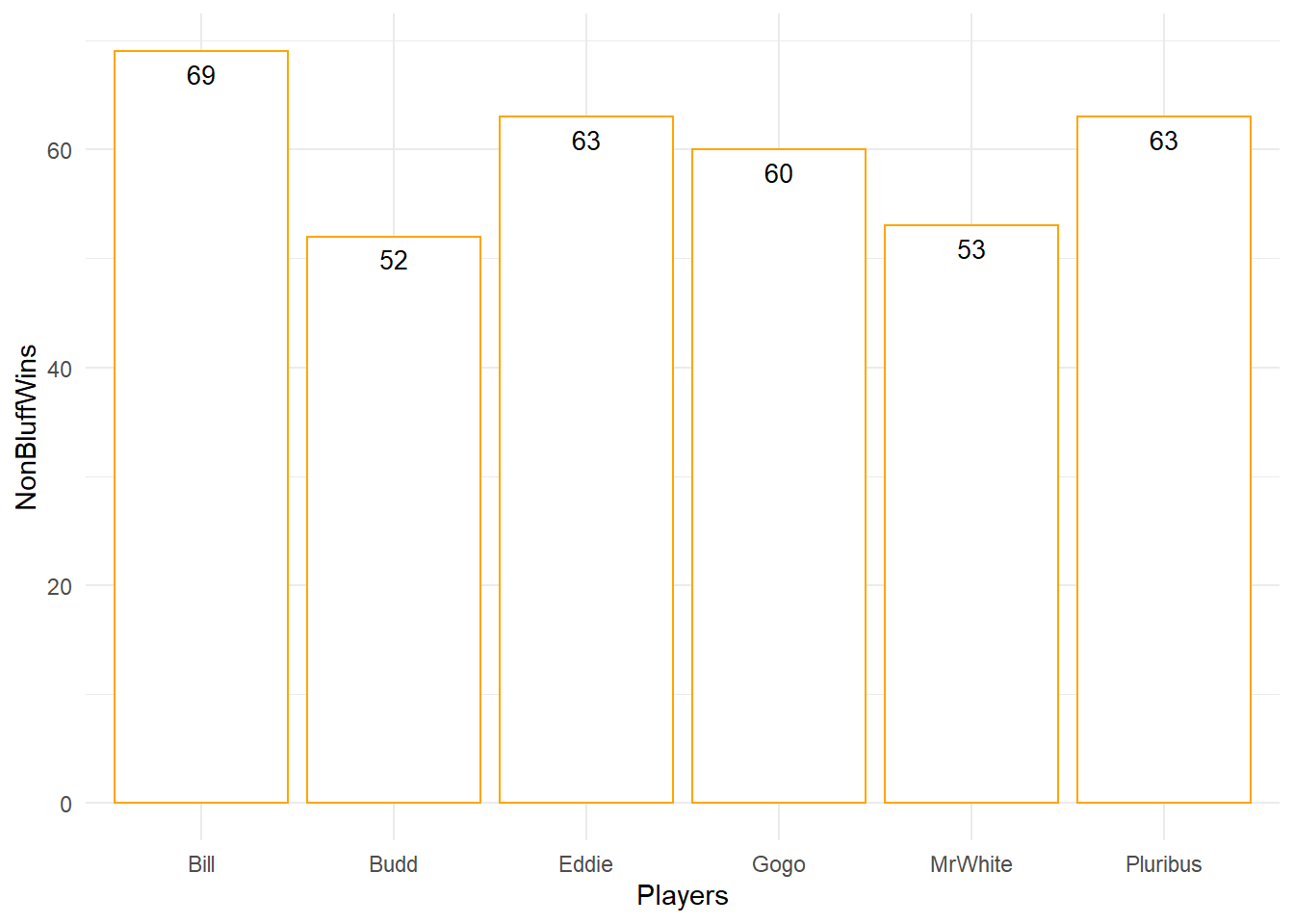
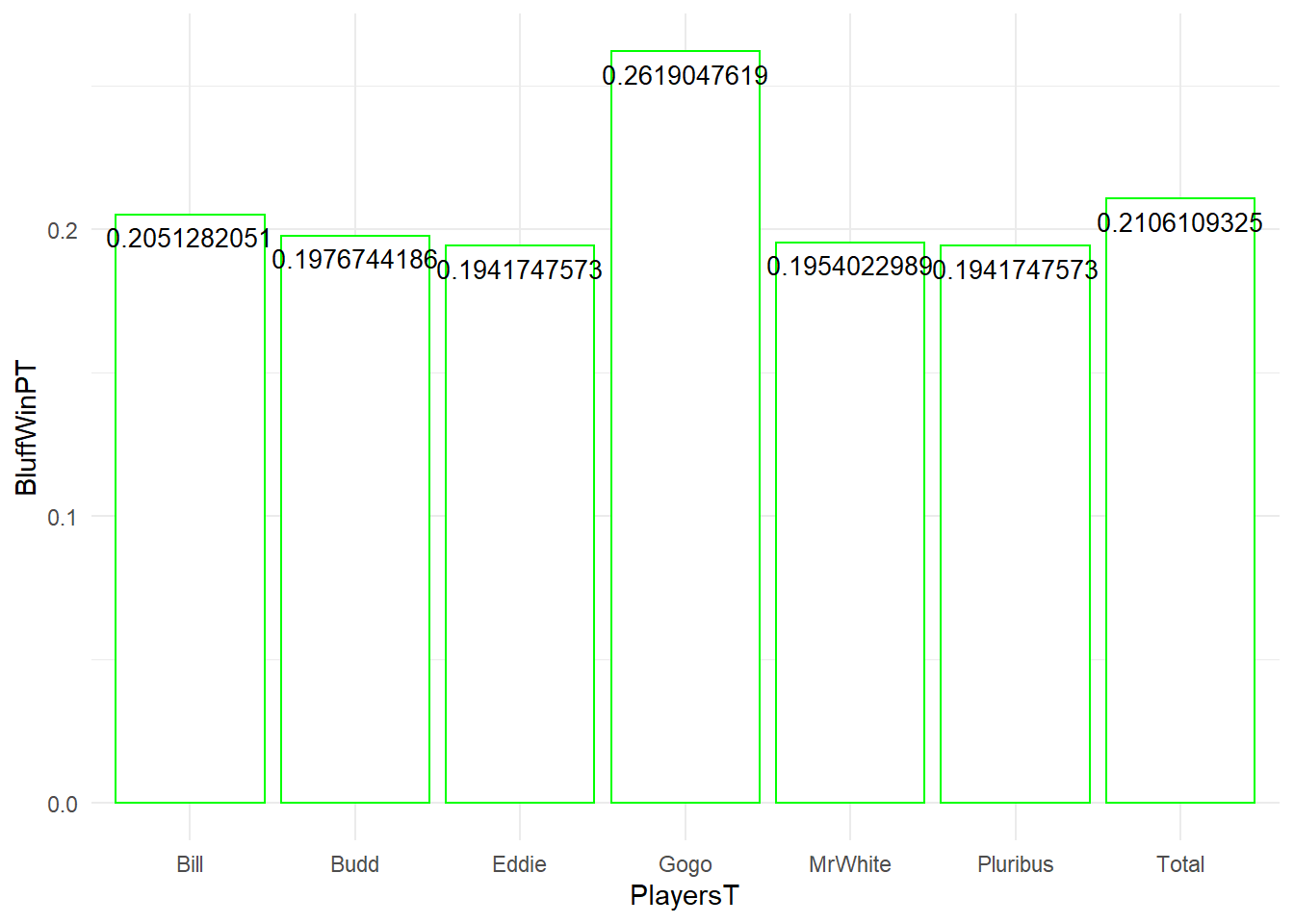
| Bluff? | Hands | HandsAI | Flop | FlopAI | Pot | PotAI |
| --- | --- | --- | --- | --- | --- | --- |
| Yes | 12900 | 131 | 8732 | 59 | 351 | 769 |
| No | 7784 | 360 | 5561 | 201 | 579 | 1704 |

As for the flop data and pot analysis we concluded from the stats above, bluffing occurred less when a flop was shown as which leads to the notion that bluffing occurs more pre-flop (68%/45% of bluffs had a flop vs 71%/56% of non-bluffs had a flop). Pots were 61%/45% smaller on average during bluffing wins leading to increased pot size making players hesitant to bluff.

**Player Results (dataset 2)**

| Player | Bluff Wins | Wins | Buff win % |
| --- | --- | --- | --- |
| Bill | 24 | 93 | 0.2051282051 |
| Eddie | 20 | 83 | 0.1941747573 |
| Budd | 17 | 69 | 0.1976744186 |
| **Gogo** | 33 | 93 | **0.2619047619** |
| MrWhite | 17 | 70 | 0.1954022989 |
| Pluribus | 20 | 83 | 0.1941747573 |
| Total | 131 | 491 | **0.2106109325** |

Next for our player results, Gogo Bluffed the most and tied for most wins which would mean bluffing is a great strategy for Gogo. Since Bill Got the same number of wins as Gogo with less bluffs means Bill likely got better hands. Gogos bluff wins 26% of the time led to superior results. Budd and MrWhite had last in bluffs and wins leading to the notion that they should bluff more or improve their poker face.



**SVM Model (Model 1)**

We ran a model using an SVM algorithm on the Pluribus dataset to see if we could predict when a game would be won by a bluff. We used the following fields for the algorithm: winner, the blind, pot size, flop and Bluff? as the factor field. We use 70% of the dataset for training and 30% for predicting. The accuracy was calculated at 74% running a confusion matrix. This model could be improved with player and flop hand analysis but it's already surprising how powerful machine learning can be.

**SVM Model (Model 2)**

We ran a second model to see if we could predict if a player would bluff again after winning a hand with a bluff. We used the following fields for the algorithm: Total Games, Total Wins, Bluff Wins, Non-Bluff Wins, and Bluff Percentage. The accuracy was calculated using another confusion matrix at only 25%. This could be improved upon on the basis that the data seems inconclusive with a such a low result. Analyzing further data about the probability of bluffing could possibly make this model more effective. Looking at card data could improve this model by including what cards are dealt when a player bluffs.

**Conclusions:**

Bluff analysis like this is useful to poker players for improving their own skills. They can see performance stats to know how much they should bluff and even, with the right algorithms, predict when a game could include a bluff. Indicators like their hand, flop, pot size, table size and who is the blind can inform players when bluffing is more common. Knowing the best and worst combination of hands can help players when they have those cards. Looking at the results we got, it is a real advantage players can have to hone in their knowledge of bluffing via data science. Imagine how powerful a future phone app can be that reads all the available data about the game and suggests to players if they should fold or raise. With the advancement of machine learning and AI, the future for games of skill may need vast countering of AI algorithms and thorough cheat detection.

Pluribus: Analysis 2

**Business Questions**

At what round is best to win and earn the most money? What round do most players win? Which round did players bet/raise/call the most money and which round did they least?

**Cleanup and Preparation**

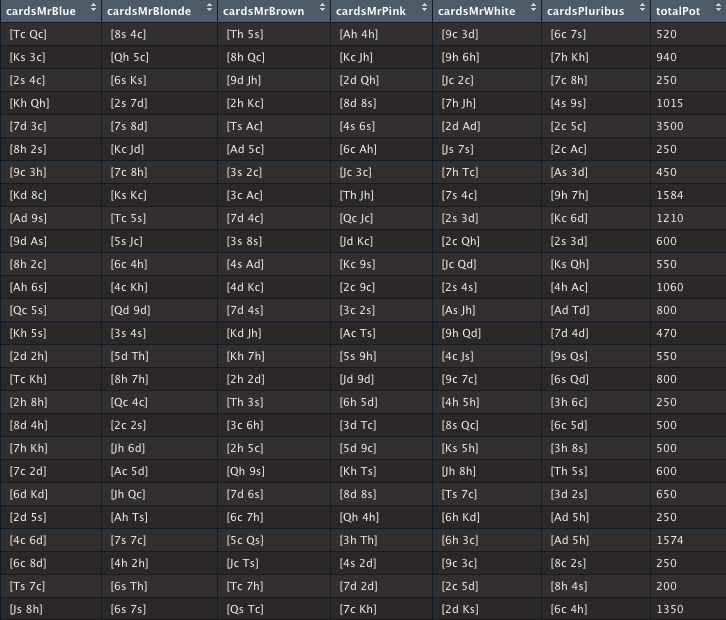
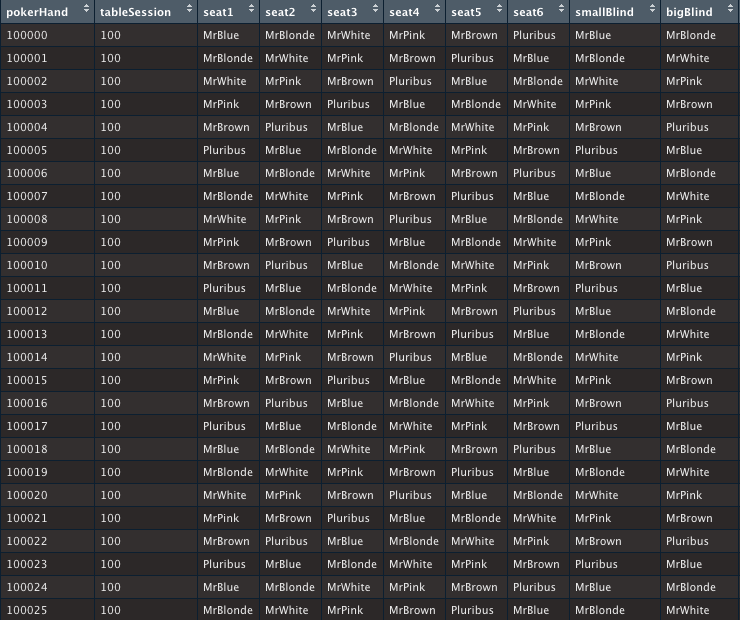
As shown in the first analysis, the game data was sourced from multiple text files. The total number of text files that could be used from KevinWang’s Pluribus work was 92 text files. However, some text files contained a different set of 6 players. In order to have consistency in our analysis, only four Pluribus text files were utilized, which were text files ending in 100,101,102, & 103.

The four text files were read by R and each line is stored into a dataframe called df.



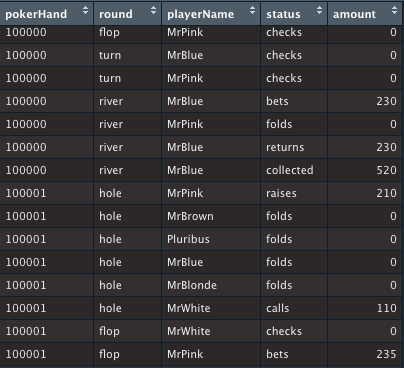
**Pluribus (df)**

Because each game had different outcomes, multiple data frames were needed to be made. So for the first read through of df in R, only lines that were constant in every game was stored in the dataframe, df2 with each row comprising of one poker hand, its table session, the player names for seat1-6, players who were smallBlind & bigBlind for that current game, the cards for all 6 players, and game’s total pot.



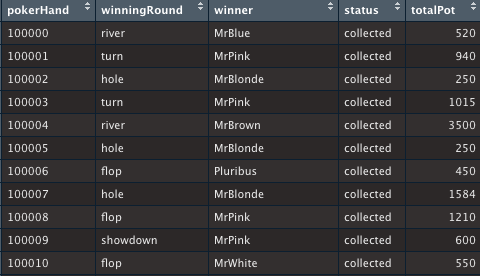
**Pluribus (df2)**

In order to include the historical data of each game, each line was then read by R and for each player move done in each round, it was stored into a dataframe called df3.



**Pluribus (df3)**

Rows that contained the status “collected” meant the player had won the game and collected the pot so by filtering out the rows that contained the status “collected” in df3, another dataframe was made named dfWinner.



**Pluribus (dfWinner)**

In some of the games, there were two players that collected money from the pot. Since most of the data contains games where the pot was not split, the rows containing those games were removed from each dataframe (i.e. pokerHand # 102000). This game will be considered an outlier.

Another method used to verify the correctness of the data munging was to check each column of each data frame for NAs. Only 2 elements in df3 were NA while there were 12 other elements that although contained a string or a value, when viewed using the View(df3) function, was read as NA. I then corrected these elements by manually each adding in their correct values and strings.

After cleaning up the four dataframes, they were then merged to make two datasets called masterData1 and masterData2. These data frames will be used as the primary sources for Pluribus Analysis 2.

**Data Dictionary: masterData1**

masterData1 contains 299 observations and 23 variables. Each row in this dataset is equal to one game while the 23 variables are described as follows:

* pokerHand : a character variable that contains the poker game number (ex. “100000" "100001" "100002" "100003" …)
* tableSession : a character variable that contains the table session (ex. "100" "100" "100" "100" …)
* seat1 : a character variable that contains the player name seated at seat 1 (ex. "MrBlue" "MrBlonde" "MrWhite" "MrPink" …)
* seat2 : a character variable that contains the player name seated at seat 2 (ex. "MrBlonde" "MrWhite" "MrPink" "MrBrown" …)
* seat3 : a character variable that contains the player name seated at seat 3 (ex. "MrWhite" "MrPink" "MrBrown" "Pluribus" …)
* seat4 : a character variable that contains the player name seated at seat 4 (ex. "MrPink" "MrBrown" "Pluribus" "MrBlue" …)
* seat5 : a character variable that contains the player name seated at seat 5 (ex. "MrBrown" "Pluribus" "MrBlue" "MrBlonde" …)
* seat6 : a character variable that contains the player name seated at seat 6 (ex. "Pluribus" "MrBlue" "MrBlonde" "MrWhite" …)
* smallBlind : a character variable that contains the small blind’s player name (ex. "MrBlue" "MrBlonde" "MrWhite" "MrPink" …)
* bigBlind : a character variable that contains the big blind’s player name (ex. "MrBlonde" "MrWhite" "MrPink" "MrBrown" …)
* cardsMrBlue : a character variable that contains MrBlue’s 2 cards (ex. " [Tc Qc]" " [Ks 3c]" " [2s 4c]" " [Kh Qh]" …)
* cardsMrBlonde: a character variable that contains MrBlonde’s 2 cards (ex. " [8s 4c]" " [Qh 5c]" " [6s Ks]" " [2s 7d]" …)
* cardsMrBrown : a character variable that contains MrBrown’s 2 cards (ex. " [Th 5s]" " [8h Qc]" " [9d Jh]" " [2h Kc]" …)
* cardsMrPink : a character variable that contains MrPink’s 2 cards (ex. " [Ah 4h]" " [Kc Jh]" " [2d Qh]" " [8d 8s]" …)
* cardsMrWhite : a character variable that contains MrWhite’s 2 cards (ex. " [9c 3d]" " [9h 6h]" " [Jc 2c]" " [7h Jh]" …)
* cardsPluribus: a character variable that contains Pluribus’s 2 cards (ex. " [6c 7s]" " [7h Kh]" " [7c 8h]" " [4s 9s]" …)
* holePot : a numerical variable that contains the pot accumulated during the hole round (ex. 370 320 250 525 450 …)
* flopPot : a numerical variable that contains the pot accumulated during the flop round (ex. 0 470 0 340 0 …)
* turnPot : a numerical variable that contains the pot accumulated during the turn round (ex. 0 600 0 600 2900 0 0 0 0 0 …)
* riverPot : a numerical variable that contains the pot accumulated during the river round (ex. 230 0 0 0 3500 0 0 0 0 200 …)
* winningRound : a character variable that contains the final round, where a player has won (ex. "river" "turn" "hole" "turn" …)
* winner: a character variable that contains the winning player’s name (ex. "MrBlue" "MrPink" "MrBlonde" "MrPink" …)
* totalPot : a numerical variable that contains the total pot won (ex. 520 940 250 1015 3500 …)

**Data Dictionary: masterData2**

masterData2 contains 3349 observations and 23 variables. Each row is a historical representation of a poker move done by a player in any certain round during the game.

* pokerHand : a character variable that contains the poker game number (ex. “100000" "100001" "100002" "100003" …)
* tableSession : a character variable that contains the table session (ex. "100" "100" "100" "100" …)
* seat1 : a character variable that contains the player name seated at seat 1 (ex. "MrBlue" "MrBlonde" "MrWhite" "MrPink" …)
* seat2 : a character variable that contains the player name seated at seat 2 (ex. "MrBlonde" "MrWhite" "MrPink" "MrBrown" …)
* seat3 : a character variable that contains the player name seated at seat 3 (ex. "MrWhite" "MrPink" "MrBrown" "Pluribus" …)
* seat4 : a character variable that contains the player name seated at seat 4 (ex. "MrPink" "MrBrown" "Pluribus" "MrBlue" …)
* seat5 : a character variable that contains the player name seated at seat 5 (ex. "MrBrown" "Pluribus" "MrBlue" "MrBlonde" …)
* seat6 : a character variable that contains the player name seated at seat 6 (ex. "Pluribus" "MrBlue" "MrBlonde" "MrWhite" …)
* smallBlind : a character variable that contains the small blind’s player name (ex. "MrBlue" "MrBlonde" "MrWhite" "MrPink" …)
* bigBlind : a character variable that contains the big blind’s player name (ex. "MrBlonde" "MrWhite" "MrPink" "MrBrown" …)
* cardsMrBlue : a character variable that contains MrBlue’s 2 cards (ex. " [Tc Qc]" " [Ks 3c]" " [2s 4c]" " [Kh Qh]" …)
* cardsMrBlonde: a character variable that contains MrBlonde’s 2 cards (ex. " [8s 4c]" " [Qh 5c]" " [6s Ks]" " [2s 7d]" …)
* cardsMrBrown : a character variable that contains MrBrown’s 2 cards (ex. " [Th 5s]" " [8h Qc]" " [9d Jh]" " [2h Kc]" …)
* cardsMrPink : a character variable that contains MrPink’s 2 cards (ex. " [Ah 4h]" " [Kc Jh]" " [2d Qh]" " [8d 8s]" …)
* cardsMrWhite : a character variable that contains MrWhite’s 2 cards (ex. " [9c 3d]" " [9h 6h]" " [Jc 2c]" " [7h Jh]" …)
* cardsPluribus: a character variable that contains Pluribus’s 2 cards (ex. " [6c 7s]" " [7h Kh]" " [7c 8h]" " [4s 9s]" …)
* round : a character variable that contains the current round the move was made (ex. "hole" "hole" "hole" "hole" …)
* playerName : a character variable that contains the player name who has made a move (ex. "MrWhite" "MrPink" "MrBrown" "Pluribus" …)
* status : a character variable that contains the status or “move” being made by playerName (ex. "folds" "raises" "folds" "folds" …)
* amount : a numerical value that contains the amount made during the “move” (ex. 0 210 0 0 160 0 0 0 0 0 …)
* winningRound : a character variable that "river" "river" "river" "river" ...
* winner : chr "MrBlue" "MrBlue" "MrBlue" "MrBlue" ...
* totalPot : num 520 520 520 520 520 520 520 520 520 520 ...

**Methods of Analysis**

The main focus in this analysis is what is happening during each round in the game.

With the help of masterData1, first, the sum() function was used to see how many games finished at what round (i.e. sum(masterData1$winningRound == "flop"). The results from those sums were then use to calculate its percentage compared to the entire dataset (rounded to the 2nd decimal place)

| **Round** | **Count** | **Percentage** |
| --- | --- | --- |
| hole | 140 | 46.82% |
| flop | 47 | 15.72% |
| turn | 33 | 11.04% |
| river | 33 | 11.04% |
| showdown | 46 | 15.38% |

Out of the 299 games examined, almost half (49.82%) ended at the hole round.

The next thing to examine is the amount called/raised/bet/returned in each round. To do this, sub-data frames were made from masterData2 by filtering data by its round column. The aggregate() function was used to calculate the min, mean, and max by each status type.

|  | **Call** | | |
| --- | --- | --- | --- |
| **Round** | **mean** | **minimum** | **maximum** |
| hole | 237.06 | 50 | 1100 |
| flop | 453.73 | 100 | 1925 |
| turn | 783.89 | 110 | 2650 |
| river | 1003 | 0 | 6825 |

Looking at the call data table’s maximum values, the highest call amount was done during the river round. The purpose of calling is to match a bet or raise and the table clearly shows that as such. The maximum average mean is $1003, which also happened during the river round.

|  | **Raise** | | |
| --- | --- | --- | --- |
| **Round** | **mean** | **minimum** | **maximum** |
| hole | 471.04 | 200 | 10000 |
| flop | 1043.22 | 450 | 1640 |
| turn | 1956.25 | 1000 | 3800 |
| river | 1587.50 | 500 | 3750 |

In poker, raising the pot not only helps with increasing the pot size but also cuts the game short, if you raise high enough that the opponent gets scared and folds. Based on the table above, we can see that the maximum value raised was $10000, which is the total amount of money each player could have each game.

|  | **Bet** | | |
| --- | --- | --- | --- |
| **Round** | **mean** | **minimum** | **maximum** |
| hole | NA | NA | NA |
| flop | 408.21 | 100 | 1925 |
| turn | 719.05 | 100 | 3174 |
| river | 1119.49 | 0 | 6825 |

In the bet table, the entire hole round there are no values because there is no betting allowed in this round. Only the moves fold, raise, check and call are present in the round. Like the call data, the maximum bet amount happened during the river round.

In masterData1, the summary function was applied onto each pot round. For example, summary(masterData1$holePot) was used to get the data in the first row of the table below.

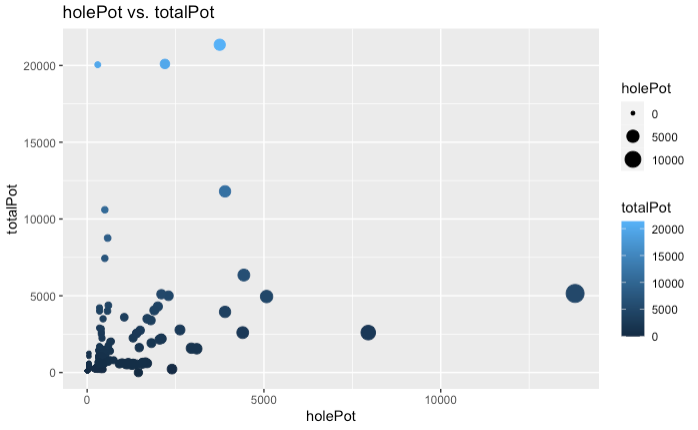
| **Round** | **minimum** | **1st Quartile** | **median** | **mean** | **3rd Quartile** | **maximum** |
| --- | --- | --- | --- | --- | --- | --- |
| hole | 0 | 225 | 350 | 653 | 575 | 13800 |
| flop | 0 | 0 | 0 | 232.80 | 242.5 | 3850 |
| turn | 0 | 0 | 0 | 246.30 | 0 | 5300 |
| river | 0 | 0 | 0 | 421.70 | 0 | 13650 |
| total | 100 | 250 | 500 | 1150 | 1000 | 21350 |

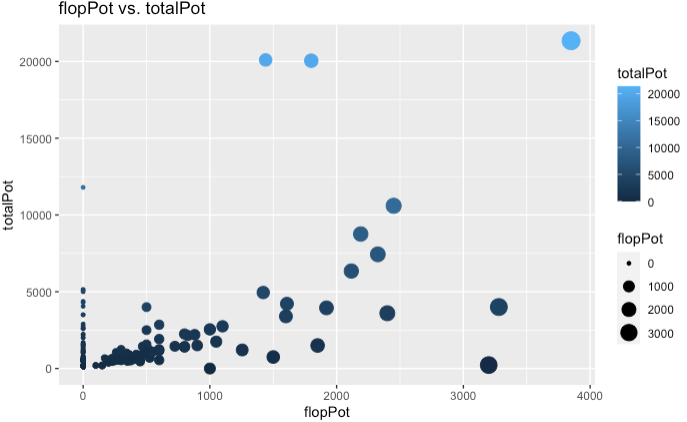
Excluding the total pot, the highest maximum pot in each round is during the hole round at $13,800 while the river comes in second at $13,650. Also when looking at the rounds’ average values. We can see that the hole round has the highest average of $653 and the river round has the second highest of $421.70. From this, we can therefore conclude that the best round to win the most money is during the hole round.

Next, masterData2’s columns: pokerHand, holePot, flopPot, turnPot, riverPot, and totalPot were used as a sub-data frame for linear modeling, which will be used to test how the different types of moves affect the pot size.

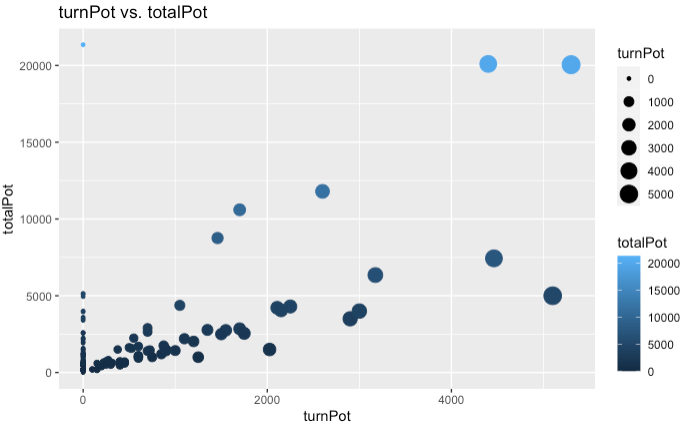
The outcome variable or the dependent variable of the model will be totalPot while the predictors or independent variables of the model will be the different pot sizes per round: holePot, flopPot, turnPot, and riverPot.

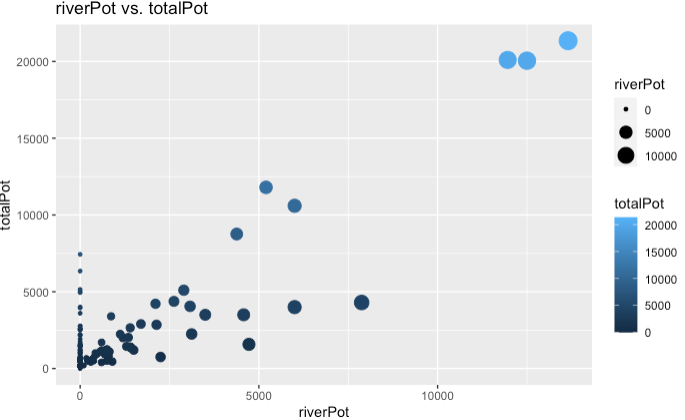
First, multiple bi-variate scatter plots were made for each predictor vs the outcome variable.





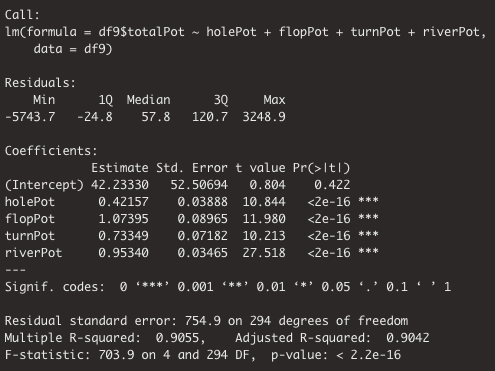
When looking at the 2 plots: holePot vs totalPot and flopPot vs totalPot, there is no significant correlation.





When looking at the two plots: turnPlot vs totalPlot and riverPlot vs totalPot, there is a slight correlation between the 2. There is also a linear trend but it is very faint.

Linear modeling was then use to verify our observations. The lm() function was coded as follows: lm(formula = df9$totalPot ~ holePot + flopPot + turnPot + riverPot, data=df9) and stored into a variable multiReg. Then the summary() was run to see the model’s results.



The adjusted Rsquared is 0.9042, which tells us the results have a variance of 9042%. Based on the summary, the coefficients that are positively significant are as follows: holePot = 0.42, flopPot=1.07, turnPot = 0.73, riverPot = 0.95 and b intercept = 42.23. All the predictors are statistically significant based on their p-values being less then 0.05.

**Conclusion**

Based on our second analysis, the hole round is when most players won and surprisingly, it is also most ideal round to win money. This is when players should exhibit their best bluffing skills in order to win the most money.

For future studies, I recommend trying to apply associative rules with the two datasets: masterData1 and masterData2. It would be interesting to see the following:

* relationship between the different cards dealt to players vs. the different poker moves (call/raise, etc.)
* relationship between the different cards dealt to players vs. the money called/raised/bet per round

One Dollar Comparison (Kaggle Data)

**Data Dictionary**

* Data Set [1] Kaggle: 35 attributes and 102615 observations

| **column** | **type** | **description** |
| --- | --- | --- |
| buyin | character | Amount paid (USD) to play the tournament |
| tourn\_id | integer | Tournament id |
| table | integer | Table number reference in the tournament |
| hand\_id | integer | Hand id |
| date | date | Date of a played hand |
| time | datetime | Time of a played hand |
| table\_size | integer | Maximum number of players per table |
| level | integer | Blinds levels |
| playing | integer | Number of players currently in the table |
| seat | integer | Seat number of each player |
| name | character | Player login |
| initial\_stack | double | Initial stack of each player |
| position | integer | Position of each player |
| action\_pre | character | Preflop actions of each player |
| action\_flop | character | Flop actions of each player |
| action\_turn | character | Turn actions of each player |
| action\_river | character | River actions of each player |
| all\_in | boolean | Informs if a player did an all-in bet (TRUE) or not (FALSE) |
| cards | character | Hand of each player (only available if you see it, obviously) |
| board\_flop | character | Cards on flop |
| board\_turn | character | Cards on turn |
| board\_river | character | Cards on river |
| combination | character | Cards combination of each player |
| pot\_pre | double | Preflop pot size |
| pot\_flop | double | Flop pot size |
| pot\_turn | double | Turn pot size |
| pot\_river | double | River pot size |
| ante | double | Ante paid |
| blinds | double | Blinds paid |
| bet\_pre | double | Bet done on preflop |
| bet\_flop | double | Bet done on flop |
| bet\_turn | double | Bet done on turn |
| bet\_river | double | Bet done on river |
| result | character | Four categories: 1-won, means a player went to showdown and won; 2-lost, means a player went to showdown and lost; 3-gave up, means a player folded at some point; 4-took chips, means a player took chips without going to showdown. |
| balance | double | How much a player won or lost after a hand |

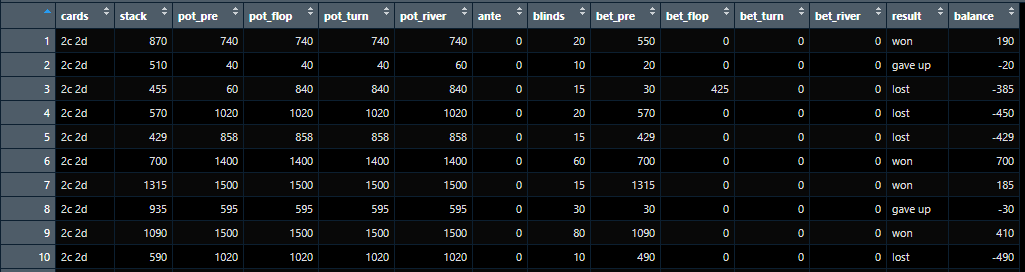
**Business Questions**

What was the percentage of players winning or losing a hand? Which card combinations given to players were the highest and lowest earning? How did players fare when they went all in?

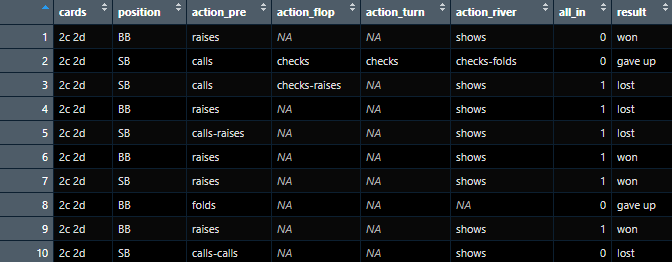
**Cleaning and Organization**

For this data set there was not much that needed to be cleaned up, but some that needed to be changed. In areas where there was no data in, the original set put either ‘X’s’ or ‘--’. In order for everything to be the same and run the code easier, we changed all of the empty cells into ‘NA’. This helped out tremendously as we wanted to create multiple data frames in order to get a clearer idea of how player results turned out. I also had to turn the all-in data into binary 1 and 0. Originally they had True as going all in and False as not going all in. But I changed that to True = 1 and False = 0.

Next, we created a data frame to organize every single hand played and get the bet amounts that they made called Hands. The data frame includes the results of the hand played and the amount of chips they either won or lost. Following that we created a follow up data frame to include the actions that were taken from each player to see if they called, raised, checked or folded called Actions. This will give whoever views this data set an easier way to look at what each player did for each turn of a card.



**Hands (Kaggle DF1)**

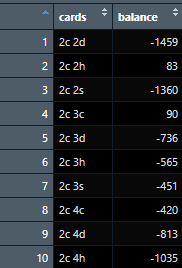
****

**Actions (Kaggle DF2)**

The last data frames created for this data set were all aggregated. We wanted to do this in order to see how each individual card combination fared against each other. This combines all similar cards into 1 and we can get a complete result of how each card did. There are 2 sets of aggregate that would need to be done in order to get a sense of how the combination of cards did. One was to get the average of each card combination and how they fared, and the second was to get the sum balance of each card combination. With these new aggregated data frames, you can look up any card combination to see the average and total earnings.

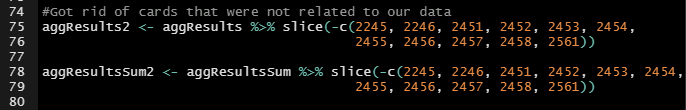
****

**Average Aggregate (Kaggle DF3)**

****

**Sum Aggregate (Kaggle DF4)**

After aggregating the results, we saw that there was data in the “cards” section that we did not want. In order to get the proper analysis we had to get rid of those combinations as follows.



**Getting rid of unnecessary cards**

**Analysis**

The first item we wanted to analyze within this data was the results of each hand and how many times those results happened.

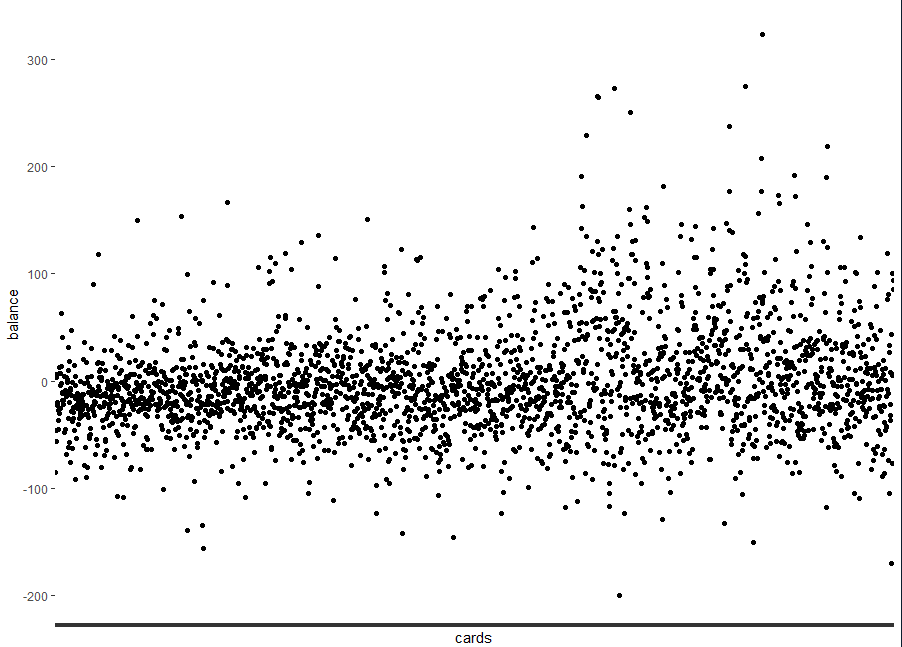
|  | RESULTS | PERCENTAGE |
| --- | --- | --- |
| **Won** | 10,772 | 10.5% |
| **Took Home** | 31,087 | 30.3 |
| **Gave Up** | 50,728 | 49.4 |
| **Lost** | 10,028 | 9.8 |

As you can see there were more times when a player lost overall (Gave Up, Lost) than actually taking any chips at the end of a round.

Next we wanted to see which card combinations were the best and worst to have. The first chart below shows which card combination averaged the lowest and highest earning chips PER HAND. Without surprise a pair of Kings average to be the best combination to start with for any player averaging 323.8 chips won if you got that hand. While surprising though is the lowest average hand being an Ace and 10 (T). With both cards considered to be high, I was surprised to see that this was the lowest average card combination. This could be a sign that while they do not have a pair to start, they have a strong hand but eventually lose out when other players start having better combinations as more cards are turned over. Also that players will continue to bluff with higher cards and no combinations rather than lower starting cards.

| CARDS | AVERAGE |
| --- | --- |
| **Ah Th** | -200 |
| **Ks Kh** | 323.8 |

We add the graph below as well to show the average amount of each card combination possible. The lowest cards that start with 2 are on the left and go all the way to the end on the right are cards that started with Kings or K. Along the Y-Axis balance refers to the amount a player on average won or lost with that card combination. The further along you go on the scatter plot you can see that the higher cards took on average more chips. With a better starting hand players can be more aggressive without having to hesitate about bluffing.



**Balance Scatter Plot**

The following chart shows the lowest and highest combined TOTAL EARNINGS of card combinations. The lowest being a pair of 4s which could be surprising but also unsurprising at the same time. A pair being the one of the lowest winning combinations you can have, can result in not much happening if you decide to keep playing it. Thus resulting in some players losing at the end of the round going in with one of the lowest pairs at. But considering it is still a pair, it is somewhat surprising a non paired hand was not a lower earning than the pair of 4s. This could prove that players are unlikely to bluff and risk their chips with a lower starting hand. The highest earning is no surprise, similar to the average stated above a pair of kings is the highest earning card combination out of all combinations.

| CARDS | TOTAL EARNINGS |
| --- | --- |
| **4d 4s** | -4075 |
| **Ks Kh** | 8095 |

The last analysis we wanted to look into was the All In data. How many times did players go all in and what were the results any time they did? Switching the data to binary for the all\_in column is what made this process easier. We found that players only went all in 14,215 times which was only 13.8% of the hands played.

|  | RESULT | PERCENTAGE |
| --- | --- | --- |
| **Gave Up** | 0 | 0% |
| **Lost** | 4,074 | 28.6% |
| **Took Chips** | 5,994 | 42.1% |
| **Won** | 4,145 | 29.2% |

**All In Data**

**Conclusions**

Players who went all in more than likely came out on top combining Took Chips and Won, 71.3%. Going all-in is a high risk, high reward tactic. It can be considered a great bluff move if the player has nothing and wants to risk all of their chips to get the other player to fold. But if they are bluffing and their opponent calls them, their chances of winning are slim.

After doing a comparison analysis for our Kaggle dataset, we can conclude that having a higher starting hand that already has a combination to win was obviously the best start. Players did not need to bluff and can play a more aggressive style against their counterparts. Although having a great starting hand did not always mean they won, as you can see from the scatter plot that the higher starting hands also had some of the lowest balance.

**Citations**

Dataset Sources:

[1] MURILO GARCIA 2021 <https://www.kaggle.com/datasets/murilogmamaral/online-poker-games> (Historical Human Games)

[2] Kevin Wang 2019 <http://kevinwang.us/lets-analyze-pluribuss-hands/> (AI VS Humans)

[3] [https://betandbeat.com/poker/strategy/preflop/starting-hands/win-percentages/ **(Odds)**](https://betandbeat.com/poker/strategy/preflop/starting-hands/win-percentages/%C2%A0)

Readings:

<https://redsharkpoker.com/poker-bluffing/>

<https://github.com/murilogmamaral/datasetbuilding/>

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.6987&rep=rep1&type=pdf>

<https://link.springer.com/chapter/10.1007/978-3-031-06433-3_34> (CNN to derive bluffs from facial images)

<http://archive.ics.uci.edu/ml/datasets/Poker+Hand>

**Appendices**

Pluribus: Analysis 1

**Poker Model.R** - Run a SVM model using data

#Beau Spratt

#IST687

library(tidyverse)

library(caret)

library(kernlab)

library(ggplot2)

df <- read\_csv("C:/Users/beaud/Downloads/syr/IST687/Final Project/P-HandsX.csv")

#select columns for model and create a factor out of bluff in game

df2 <- data.frame(bluff=as.factor(df$`Bluff?`),

Winner=df$Winner,

BW=df$`B=W`,

Pot=df$Pot,

Flop=df$`Has Flop`) #add flop details?

#set training and test data sets

trainList <- createDataPartition(y=df2$bluff,p=.70, list=FALSE)

trainSet <- df2[trainList,]

testSet <- df2[-trainList,]

#build svm model to predict if the game has a bluff

model.ksvm <- train(bluff~., data=trainSet, method="svmRadial", preProc=c("center","scale"))

svmPred <- predict(model.ksvm, newdata=testSet, type = "raw")

#display accuracy of predictions

confusionMatrix(svmPred,reference = testSet$bluff)

**KaggleFilter.R** - Filter Kaggleset for winners and do reports

#Beau Spratt

#IST687

library(tidyverse)

#load datasets from kaggle and odds

x <- read\_csv("C:/Users/beaud/Downloads/syr/IST687/Final Project/one\_dollar\_spin\_and\_go.csv")

y <- read\_csv("C:/Users/beaud/Downloads/syr/IST687/Final Project/HandQuality.csv")

# filter and count for reports

x %>% count(name, result, sort=TRUE)

wins <- x %>% filter(cards!='--'& result =='took chips'| result =='won')

wins %>% count(name, result, sort=TRUE)

wins %>% count(tourn\_id, pot\_river, sort=TRUE)

library(dplyr)

#run reports and pivot example %>% group\_by(tourn\_id)?

potMean = wins %>%

summarise(mean\_pot = mean(pot\_river),

total\_pot = sum(pot\_river),

.groups = 'dyrop')

View(potMean)

**P-Hands.R** - Format Pluribus dataset hands to match and pull in odds

#Beau Spratt

#IST687

library(tidyverse)

y <- read\_csv("C:/Users/beaud/Downloads/syr/IST687/Final Project/HandQuality2.csv")

z <- read\_csv("C:/Users/beaud/Downloads/syr/IST687/Final Project/Pluribus.csv")

#format hands to match HandQuality Dataset including suited

#checks if suited by substring match

suitedM <- substring(z$MrWhite,3,3)==substring(z$MrWhite,6,6)

z <- cbind(z, suitedM)

#change a logical to string and replace value

z$suitedM<-paste(z$suitedM)

z$suitedM[z$suitedM=='FALSE']<- ''

z$suitedM[z$suitedM=='TRUE']<- 's'

z$MrWhite <- str\_replace\_all(paste(substring(z$MrWhite,2,2),substring(z$MrWhite,5,5))," ","")

z$MrWhite <- gsub(" ", "", paste(z$MrWhite,z$suitedM))

suitedG <- substring(z$Gogo,3,3)==substring(z$Gogo,6,6)

z <- cbind(z, suitedG)

z$suitedG<-paste(z$suitedG)

z$suitedG[z$suitedG=='FALSE']<- ''

z$suitedG[z$suitedG=='TRUE']<- 's'

z$Gogo <- str\_replace\_all(paste(substring(z$Gogo,2,2),substring(z$Gogo,5,5))," ","")

z$Gogo <- gsub(" ", "", paste(z$Gogo,z$suitedG))

suitedBu <- substring(z$Budd,3,3)==substring(z$Budd,6,6)

z <- cbind(z, suitedBu)

z$suitedBu<-paste(z$suitedBu)

z$suitedBu[z$suitedBu=='FALSE']<- ''

z$suitedBu[z$suitedBu=='TRUE']<- 's'

z$Budd <- str\_replace\_all(paste(substring(z$Budd,2,2),substring(z$Budd,5,5))," ","")

z$Budd <- gsub(" ", "", paste(z$Budd,z$suitedBu))

suitedE <- substring(z$Eddie,3,3)==substring(z$Eddie,6,6)

z <- cbind(z, suitedE)

z$suitedE<-paste(z$suitedE)

z$suitedE[z$suitedE=='FALSE']<- ''

z$suitedE[z$suitedE=='TRUE']<- 's'

z$Eddie <- str\_replace\_all(paste(substring(z$Eddie,2,2),substring(z$Eddie,5,5))," ","")

z$Eddie <- gsub(" ", "", paste(z$Eddie,z$suitedE))

suitedBi <- substring(z$Bill,3,3)==substring(z$Bill,6,6)

z <- cbind(z, suitedBi)

z$suitedBi<-paste(z$suitedBi)

z$suitedBi[z$suitedBi=='FALSE']<- ''

z$suitedBi[z$suitedBi=='TRUE']<- 's'

z$Bill <- str\_replace\_all(paste(substring(z$Bill,2,2),substring(z$Bill,5,5))," ","")

z$Bill <- gsub(" ", "", paste(z$Bill,z$suitedBi))

suitedP <- substring(z$Pluribus,3,3)==substring(z$Pluribus,6,6)

z <- cbind(z, suitedP)

z$suitedP<-paste(z$suitedP)

z$suitedP[z$suitedP=='FALSE']<- ''

z$suitedP[z$suitedP=='TRUE']<- 's'

z$Pluribus <- str\_replace\_all(paste(substring(z$Pluribus,2,2),substring(z$Pluribus,5,5))," ","")

z$Pluribus <- gsub(" ", "", paste(z$Pluribus,z$suitedP))

z <- z[,-9:-14]

ordercols <- colnames(z)

#add odds of each hand by merging with by each player (sloppy but works) #left join

dfNew <- merge(z, y, by.x = "MrWhite", by.y = "Hand",all.x=TRUE)

dfNew['Modds'] = dfNew[,12]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- dfNew[,-9]

dfNew <- merge(dfNew, y, by.x = "Gogo", by.y = "Hand",all.x=TRUE)

dfNew['Godds'] = dfNew[,13]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- dfNew[,-10]

dfNew <- merge(dfNew, y, by.x = "Budd", by.y = "Hand",all.x=TRUE)

dfNew['BUodds'] = dfNew[,14]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- dfNew[,-11]

dfNew <- merge(dfNew, y, by.x = "Eddie", by.y = "Hand",all.x=TRUE)

dfNew['Eodds'] = dfNew[,15]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- dfNew[,-12]

dfNew <- merge(dfNew, y, by.x = "Bill", by.y = "Hand",all.x=TRUE)

dfNew['BLodds'] = dfNew[,16]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- dfNew[,-13]

dfNew <- merge(dfNew, y, by.x = "Pluribus", by.y = "Hand",all.x=TRUE)

dfNew['Podds'] = dfNew[,17]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

dfNew <- dfNew[,-14]

#removes dupes

dfNew=unique(dfNew)

write.csv(dfNew,"C:/Users/beaud/Downloads/syr/IST687/Final Project/P-Hands.csv")

**DS1.ipynb** - Pull winners, hand, and pot size from dataset

#Beau Spratt

#IST687

#filter dataset for winners, their hand and pot size

import csv

# create an output file name

outfile2 = 'WinnersDS1.csv'

# open the file

csvfileout = open(outfile2, 'w', newline='')

# create a csv writer for a comman sep file, with quoting as needed

PKwriter = csv.writer(csvfileout, delimiter=',', quoting=csv.QUOTE\_MINIMAL)

# write the header row as a list of column labels

PKwriter.writerow(['hand\_id','Name','Hand', 'Pot'])

#read full hand dataset

infile = r'one\_dollar\_spin\_and\_go.csv'

with open(infile, 'r') as csvfile:

stateReader = csv.DictReader(csvfile)

for row in stateReader:

#filter for rows with hands and winners and write to file

if row['cards'] != '--' and (row['result']=='won' or row['result']=='took chips'):

hand = row['cards'][0]+row['cards'][3]

if row['cards'][1]==row['cards'][4]:

hand = hand+'s'

PKwriter.writerow([row['hand\_id'],row['name'],hand,row['pot\_river']])

csvfileout.close()

print('file output as:',outfile2)

csvfile.close()

**DS1-Odds.ipynb** - Grab and merge in Odds to dataset

#Beau Spratt

#IST687

import pandas as pd

#read winners from DS1 and Hand Quality

csv1 = pd.read\_csv(r'WinnersDS1.csv')

csv2 = pd.read\_csv(r'HandQuality.csv')

#merge them

merged\_data = csv1.merge(csv2,on=["Hand"])

merged\_data.to\_csv("WinnersDS1Odds.csv")

#csvfileout.close()

#print('file output as:',outfile2)

#csvfile.close()

#csvfile2.close()

**Merge.ipynb** - Merge Text Files of log files

#Beau Spratt

#IST687

#combine text log files with poker hands

import csv

import glob

file\_names = glob.glob(r"C:\Users\beaud\Downloads\syr\IST652\Final Project\data\\*.txt")

with open(r"C:\Users\beaud\Downloads\syr\IST652\Final Project\data\Merged.txt", 'w') as outfile:

# Iterate through list

for names in file\_names:

# Open each file in read mode

with open(names) as infile:

# read the data from file1 and

# file2 and write it in file3

outfile.write(infile.read())

# Add '\n' to enter data of file2

# from next line

outfile.write("\n")

**P-HandParser.ipynb** - Converts Hands from log file to csv pulling winner, hands, flop, blinder and odds

#Beau Spratt

#IST687

#read merged poker hands dataset and output gameid, name of winner, hands for each player, flop etc.,

import csv

# first create an output file name

outfile2 = 'Merged.csv'

# open the file

csvfileout = open(outfile2, 'w', newline='')

# create a csv writer for a comman sep file, with quoting as needed

PKwriter = csv.writer(csvfileout, delimiter=',', quoting=csv.QUOTE\_MINIMAL)

# write the header row as a list of column labels

PKwriter.writerow(['Game','Winner','Blinder','Pot','Flop','Turn','River','MrWhite','Gogo','Budd','Eddie','Bill','Pluribus'])

infile = r'C:\Users\beaud\Downloads\syr\IST652\Final Project\data\Merged.txt'

Flop = ''

Turn = ''

River= ''

with open(infile, 'r') as csvfile:

stateReader = csv.reader(csvfile, delimiter='\n')

for line in stateReader:

#make a string for handID

if str(line).find('Hand #')>0:

Hand = str(line)[17:24]

#make a string for Blind Name

if str(line).find(': posts big blind')>0:

Blinder = str(line)[2:str(line).find(': posts big blind')]

#make a string for Flop

if str(line).find('FLOP')>0:

Flop = str(line)[14:25]

#make a string for Turn

if str(line).find('TURN')>0:

Turn = str(line)[25:30]

#make a string for River

if str(line).find('RIVER')>0:

River = str(line)[32:36]

#make a string of each players hand

if str(line).find('Dealt to MrWhite')>0:

MrWhite = str(line)[-9:-2]

if str(line).find('Dealt to Gogo')>0:

Gogo = str(line)[-9:-2]

if str(line).find('Dealt to Budd')>0:

Budd = str(line)[-9:-2]

if str(line).find('Dealt to Eddie')>0:

Eddie = str(line)[-9:-2]

if str(line).find('Dealt to Bill')>0:

Bill = str(line)[-9:-2]

if str(line).find('Dealt to Pluribus')>0:

Pluribus = str(line)[-9:-2]

#make a string of winner

if str(line).find('collected')>0:

print(Hand,',',str(line)[2:str(line).find('collected')],',',Blinder,',',str(line)[str(line).find('collected')+10:str(line).find(' from pot')],',',Flop,',',Turn,',',River,',',MrWhite,',',Gogo,',',Budd,',',Eddie,',',Bill,',',Pluribus)

PKwriter.writerow([Hand,str(line)[2:str(line).find('collected')],Blinder,str(line)[str(line).find('collected')+10:str(line).find(' from pot')],Flop,Turn,River,MrWhite,Gogo,Budd,Eddie,Bill,Pluribus])

Flop = ''

Turn = ''

River= ''

csvfileout.close()

print('file output as:',outfile2)

csvfile.close()

**Reports.ipynb** - Build reports using datasets

#Beau Spratt

#IST687

import csv

import pandas as pd

import numpy as np

#Did players bluff more vs an AI or Human?

#Was bluffing a good idea(did they earn chips)?

#Who bluffed the best?

#Was bluffing more common when there was a flop?

#create pivots for one and spin dataset

csv1 = pd.read\_csv(r'WinnersDS1Odds3.csv')

table = pd.pivot\_table(csv1, index=['Bluff?'], values=['Pot'], aggfunc=np.mean, sort=False)

table2 = pd.pivot\_table(csv1, index=['Bluff?'], values=['hand\_id'],aggfunc='count', sort=False)

table10 = pd.pivot\_table(csv1, index=['Bluff?'], values=['Flop'], aggfunc='count', sort=False)

#create pivots for pluribus dataset

csv2 = pd.read\_csv(r'P-Hands2.csv')

table3 = pd.pivot\_table(csv2, index=['Bluff?'], values=['Pot'], aggfunc=np.mean, sort=False)

table4 = pd.pivot\_table(csv2, index=['Bluff?'], values=['Game'],aggfunc='count', sort=False)

table5 = pd.pivot\_table(csv2, index=['Winner'], values=['Bluff?'],aggfunc=np.sum, sort=False)

table6 = pd.pivot\_table(csv2, index=['Winner'], values=['Bluff?'],aggfunc='count', sort=False)

table11 = pd.pivot\_table(csv2, index=['Bluff?'], values=['Has Flop'], aggfunc='count', sort=False)

#Concatenate for comparison

table9 = pd.concat([table2,table4,table10,table11,table,table3], axis=1)

table9.columns = ['Hands','HandsAI','Flop','FlopAI','Pot', 'PotAI']

print(table9)

table7 = pd.concat([table5,table6], axis=1)

table7.columns = ['Bluffs', 'Total']

print(table7)

Bar Graphs of Pluribus Data

library(ggplot2) library(dplyr) library(imputeTS) library(tidyverse)

Players <- c("Bill", "Eddie", "Budd", "Gogo", "MrWhite", "Pluribus")

BluffWins <- c(24, 20, 17, 33, 17, 20)

Wins <- c(93, 83, 69, 93, 70, 83)

BluffWinP <- c(0.2051282051, 0.1941747573, 0.1976744186, 0.2619047619, 0.1954022989, 0.1941747573)

NonBluffWins <- c(69, 63, 52, 60, 53, 63)

Poker.data <- data.frame(Players, BluffWins, Wins, BluffWinP, NonBluffWins, strainAsFactors=FALSE)

G1 <- ggplot(data=Poker.data, aes(x=Players, y=BluffWins)) + geom\_bar(stat="identity", color="blue", fill="white") + geom\_text(aes(label=BluffWins), vjust=1.6, color="black", size=3.5) + theme\_minimal()

G2 <- ggplot(data=Poker.data, aes(x=Players, y=Wins)) + geom\_bar(stat="identity", color="red", fill="white") + geom\_text(aes(label=Wins), vjust=1.6, color="black", size=3.5) + theme\_minimal()

G6 <- ggplot(data=Total.data, aes(x=PlayersT, y=BluffWinPT)) + geom\_bar(stat="identity", color="green", fill="white") + geom\_text(aes(label=BluffWinPT), vjust=1.6, color="black", size=3.5) + theme\_minimal()

G7 <- ggplot(data=Poker.data, aes(x=Players, y=NonBluffWins)) + geom\_bar(stat="identity", color="orange", fill="white") + geom\_text(aes(label=NonBluffWins), vjust=1.6, color="black", size=3.5) + theme\_minimal()

library(kernlab)

library(caret)

library(ISLR)

library(InformationValue)

data <- Poker.data

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))

train <- data[sample, ]

test <- data[!sample, ]

model <- glm(Wins~NonBluffWins+BluffWins+BluffWinP, data=train)

predicted <- predict(model, test, type="response")

test$Wins <- ifelse(test$Wins=="Yes", 1, 0)

optimal <- optimalCutoff(test$Wins, predicted)[1]

confusionMatrix(test$Wins, predicted)

sensitivity(test$Wins, predicted)

specificity(test$Wins, predicted)

misClassError(test$Wins, predicted, threshold=optimal)

Pluribus: Analysis 2

---

output:

pdf\_document: default

html\_document: default

---

IST 687

Final Project

Pluribus Analysis 2

Marianne Lynne S. Garbo

EXTRACT & CLEAN UP TXTFILES INTO DATASET

#### Read every df[x,2] in every text file.

```{r}

# upload libraries needed

library(RCurl)

library(tidyverse)

library(ggplot2)

library(readr)

library(readtext)

library(readtextgrid)

library(stringr)

library(reader)

library(data.table)

# set director where text files are

setwd("/Users/jaywhitehead/Desktop/IST 687/IST 687 Project/pluribus\_converted\_logs/")

# list all files

file\_list <- list.files(path="/Users/jaywhitehead/Desktop/IST 687/IST 687 Project/pluribus\_converted\_logs/")

# initialize empty vectors

fls <- NULL

lns <- NULL

# for loop to read every file in folder

for (file in file\_list){

temp\_data <- readLines(file)

for(line in temp\_data){

fls <- c(fls, file)

lns <- c(lns, line)

}

}

# store read lines and its source file into a dataframe

df <- data.frame(file=fls, fline=lns)

```

#### Read df for lines that are standard & repetitive

```{r}

# string of vectors

pokerHand <- NULL

tableSession <- NULL

playerNames <- NULL

seat1 <- NULL

seat2 <- NULL

seat3 <- NULL

seat4 <- NULL

seat5 <- NULL

seat6 <- NULL

smallBlind <- NULL

bigBlind <- NULL

cardsMrBlue <- NULL

cardsMrBlonde <- NULL

cardsMrWhite<- NULL

cardsMrPink<- NULL

cardsMrBrown<- NULL

cardsPluribus <- NULL

winner <- NULL

totalPot <- NULL

board <- NULL

# WHILE LOOP

x <- 1

dfSize <- nrow(df)

# (dfSize+1)

while (x != (dfSize+1)){

## Poker Hand #

if(startsWith(df[x,2],"PokerStars Hand #") == TRUE){

# get substring after pattern

handstring<- sub(".\*PokerStars Hand #","", df[x,2])

# get hand number by getting substring before pattern

handNum <- sub(": Hold'em No Limit.\*","",handstring)

# store in a vector

pokerHand <- c(pokerHand,handNum)

}

## Table Session #

else if (startsWith(df[x,2],"Table 'Pluribus Session ") == TRUE){

# get substring after pattern

sessionstring<- sub(".\*Table 'Pluribus Session ","", df[x,2])

# get hand number by getting substring before pattern

sessionNum <- sub("' 6-max.\*","",sessionstring)

# store in a vector

tableSession <- c(tableSession, sessionNum)

}

## Seat #1

else if(startsWith(df[x,2],"Seat 1: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat1string<- sub(".\*Seat 1: ","", df[x,2])

# get hand number by getting substring before pattern

player1Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat1string))

player1Name <- gsub("[()]", "", player1Nam)

# store in vector

seat1<- c(seat1, player1Name)

# store in a vector for player names

playerNames <- c(playerNames, player1Name)

}

## Seat #2

else if(startsWith(df[x,2],"Seat 2: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat2string<- sub(".\*Seat 2: ","", df[x,2])

# get hand number by getting substring before pattern

player2Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat2string))

player2Name <- gsub("[()]", "", player2Nam)

# store in vector

seat2 <- c(seat2, player2Name)

# store in a vector for player names

playerNames <- c(playerNames, player2Name)

}

## Seat #3

else if(startsWith(df[x,2],"Seat 3: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat3string<- sub(".\*Seat 3: ","", df[x,2])

# get hand number by getting substring before pattern

player3Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat3string))

player3Name <- gsub("[()]", "", player3Nam)

# store in vector

seat3 <- c(seat3, player3Name)

# store in a vector for player names

playerNames <- c(playerNames, player3Name)

}

## Seat #4

else if(startsWith(df[x,2],"Seat 4: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat4string<- sub(".\*Seat 4: ","", df[x,2])

# get hand number by getting substring before pattern

player4Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat4string))

player4Name <- gsub("[()]", "", player4Nam)

# store in vector

seat4 <- c(seat4, player4Name)

# store in a vector for player names

playerNames <- c(playerNames, player4Name)

}

## Seat #5

else if(startsWith(df[x,2],"Seat 5: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat5string<- sub(".\*Seat 5: ","", df[x,2])

# get hand number by getting substring before pattern

player5Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat5string))

player5Name <- gsub("[()]", "", player5Nam)

# store in vector

seat5 <- c(seat5, player5Name)

# store in a vector for player names

playerNames <- c(playerNames, player5Name)

}

## Seat #6

else if(startsWith(df[x,2],"Seat 6: ") == TRUE & endsWith(df[x,2]," (10000 in chips)") == TRUE){

# get substring after pattern

seat6string<- sub(".\*Seat 6: ","", df[x,2])

# get hand number by getting substring before pattern

player6Nam <- gsub(" ","",sub("(10000 in chips).\*","",seat6string))

player6Name <- gsub("[()]", "", player6Nam)

# store in vector

seat6 <- c(seat6, player6Name)

# store in a vector for player names

playerNames <- c(playerNames, player6Name)

}

## small blind

else if(grepl("posts small blind ", df[x,2], fixed = TRUE)){

smallBlindString <- sub(": posts small blind.\*","", df[x,2])

smallBlind <- c(smallBlind, smallBlindString)

}

## big blind

else if(grepl("posts big blind ", df[x,2], fixed = TRUE)){

bigBlindString <- sub(": posts big blind.\*","", df[x,2])

bigBlind <- c(bigBlind, bigBlindString)

}

## MrBlue cards

else if(startsWith(df[x,2],"Dealt to MrBlue ") == TRUE) {

cardString <- sub(".\*Dealt to MrBlue","",df[x,2])

cardsMrBlue <- c(cardsMrBlue, cardString)

}

## MrBlonde cards

else if(startsWith(df[x,2],"Dealt to MrBlonde ") == TRUE) {

cardString<- sub(".\*Dealt to MrBlonde","",df[x,2])

cardsMrBlonde <- c(cardsMrBlonde, cardString)

}

## MrWhite cards

else if(startsWith(df[x,2],"Dealt to MrWhite ") == TRUE) {

cardString <- sub(".\*Dealt to MrWhite","",df[x,2])

cardsMrWhite <- c(cardsMrWhite, cardString)

}

## MrPink cards

else if(startsWith(df[x,2],"Dealt to MrPink ") == TRUE) {

cardString <- sub(".\*Dealt to MrPink","",df[x,2])

cardsMrPink <- c(cardsMrPink, cardString)

}

## MrBrown cards

else if(startsWith(df[x,2],"Dealt to MrBrown ") == TRUE) {

cardString <- sub(".\*Dealt to MrBrown","",df[x,2])

cardsMrBrown <- c(cardsMrBrown, cardString)

}

## Pluribus cards

else if(startsWith(df[x,2],"Dealt to Pluribus ") == TRUE) {

cardString <- sub(".\*Dealt to Pluribus","",df[x,2])

cardsPluribus <- c(cardsPluribus, cardString)

}

## Total Pot

else if (startsWith(df[x,2],"Total pot") == TRUE){

totalPotString <- sub("Rake.\*","",sub("\\|", "", gsub(" ","",sub(".\*Total pot ","", df[x,2]))))

totalPot <- c(totalPot,totalPotString)

}

x <- x+1

}

```

#### store in a dataframe, df2 (ONE ROW = ONE GAME; has 303 obs of 17 variables)

```{r}

df2 <- data.frame(pokerHand,tableSession,seat1,seat2,seat3,seat4,seat5,seat6,smallBlind,bigBlind,cardsMrBlue,cardsMrBlonde,cardsMrBrown,cardsMrPink,cardsMrWhite,cardsPluribus, totalPot)

```

#### Run through df again to retrieve data for other columns

```{r}

x <- 1

pokerHand <- NULL

round <- NULL

playerName <- NULL

status <- NULL

amount <- NULL

pokerHand2 <- NULL

holePot <- NULL

flopPot <- NULL

turnPot <- NULL

riverPot <- NULL

handNum <-0

holeCurrentPot <- 0

flopCurrentPot <- 0

turnCurrentPot <- 0

riverCurrentPot <- 0

#(dfSize +1)

while (x != (dfSize +1)) {

## Poker Hand #

if(startsWith(df[x,2],"PokerStars Hand #") == TRUE) {

pokerHand2 <- c(pokerHand2,handNum)

holePot <- c(holePot,holeCurrentPot)

flopPot <- c(flopPot,flopCurrentPot)

turnPot <- c(turnPot,turnCurrentPot)

riverPot <- c(riverPot,riverCurrentPot)

# get substring after pattern

handstring<- sub(".\*PokerStars Hand #","", df[x,2])

# get hand number by getting substring before pattern

handNum <- sub(": Hold'em No Limit.\*","",handstring)

holeCurrentPot <- 0

flopCurrentPot <- 0

turnCurrentPot <- 0

riverCurrentPot <- 0

x <- x+1

next

}

## HOLE ROUND

else if (df[x,2] == "\*\*\* HOLE CARDS \*\*\*"){

x <- x+7

while (startsWith(df[x,2],"\*\*\* SUMMARY \*\*\*") != TRUE && startsWith(df[x,2],"\*\*\* FLOP \*\*\*") != TRUE){

## MrBlue

if(startsWith(df[x,2],"MrBlue: folds") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlue")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlue")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlue: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlue")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlue: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlue")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlue")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

## MrBlonde

else if (startsWith(df[x,2],"MrBlonde: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlonde: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlonde: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

## MrWhite

else if (startsWith(df[x,2],"MrWhite: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrWhite")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrWhite: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrWhite")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrWhite: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrWhite")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrWhite: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrWhite")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrWhite: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrWhite")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

## MrPink

else if (startsWith(df[x,2],"MrPink: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrPink")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrPink: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrPink")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrPink: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrPink")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrPink: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrPink")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrPink: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrPink")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

## MrBrown

else if (startsWith(df[x,2],"MrBrown: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBrown")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrBrown: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBrown")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrBrown: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBrown")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBrown: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBrown")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBrown: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "MrBrown")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

## Pluribus

else if(startsWith(df[x,2],"Pluribus: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "Pluribus")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"Pluribus: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "Pluribus")

status <- c(status,"calls")

amount <- c(amount,callNum)

holeCurrentPot <- holeCurrentPot + callNum

}

else if (startsWith(df[x,2],"Pluribus: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "Pluribus")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

holeCurrentPot <- holeCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "Pluribus: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "Pluribus")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "Pluribus: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, "Pluribus")

status <- c(status,"bets")

amount <- c(amount,betNum)

holeCurrentPot <- holeCurrentPot + betNum

}

# Uncalled bet

else if (startsWith(df[x,2],"Uncalled bet") == TRUE){

name <- sub(".\*returned to ","",df[x,2])

returnNum <- strtoi(sub(" returned to.\*","",sub(".\*Uncalled bet ","", gsub("[()]", "", df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, name)

status <- c(status,"returns")

amount <- c(amount,returnNum)

}

# Collected pot

else if (startsWith(df[x,2],"") == TRUE){

name <- gsub(" ","",sub("collected.\*","",df[x,2]))

collectNum <- strtoi(sub(".0frompot.\*","",sub(".\*collected","",gsub(" ","",df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "hole")

playerName <- c(playerName, name)

status <- c(status,"collected")

amount <- c(amount,collectNum)

}

x<-x+1

}#end-while (HOLE ROUND)

## FLOP ROUND

if(startsWith(df[x,2],"\*\*\* FLOP \*\*\*") == TRUE){

x <- x+1

while (startsWith(df[x,2],"\*\*\* SUMMARY \*\*\*") != TRUE && startsWith(df[x,2],"\*\*\* TURN \*\*\*") != TRUE){

## MrBlue

if(startsWith(df[x,2],"MrBlue: folds") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlue")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlue")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlue: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlue")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlue: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlue")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlue")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

## MrBlonde

else if (startsWith(df[x,2],"MrBlonde: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlonde: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlonde: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

## MrWhite

else if (startsWith(df[x,2],"MrWhite: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrWhite")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrWhite: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrWhite")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrWhite: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrWhite")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrWhite: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrWhite")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrWhite: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrWhite")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

## MrPink

else if (startsWith(df[x,2],"MrPink: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrPink")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrPink: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrPink")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrPink: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrPink")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrPink: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrPink")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrPink: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrPink")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

## MrBrown

else if (startsWith(df[x,2],"MrBrown: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBrown")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrBrown: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBrown")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrBrown: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBrown")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBrown: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBrown")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBrown: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "MrBrown")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

## Pluribus

else if(startsWith(df[x,2],"Pluribus: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "Pluribus")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"Pluribus: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "Pluribus")

status <- c(status,"calls")

amount <- c(amount,callNum)

flopCurrentPot <- flopCurrentPot + callNum

}

else if (startsWith(df[x,2],"Pluribus: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "Pluribus")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

flopCurrentPot <- flopCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "Pluribus: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "Pluribus")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "Pluribus: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, "Pluribus")

status <- c(status,"bets")

amount <- c(amount,betNum)

flopCurrentPot <- flopCurrentPot + betNum

}

# Uncalled bet

else if (startsWith(df[x,2],"Uncalled bet") == TRUE){

name <- sub(".\*returned to ","",df[x,2])

returnNum <- strtoi(sub(" returned to.\*","",sub(".\*Uncalled bet ","", gsub("[()]", "", df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, name)

status <- c(status,"returns")

amount <- c(amount,returnNum)

}

# Collected pot

else if (startsWith(df[x,2],"") == TRUE){

name <- gsub(" ","",sub("collected.\*","",df[x,2]))

collectNum <- strtoi(sub(".0frompot.\*","",sub(".\*collected","",gsub(" ","",df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "flop")

playerName <- c(playerName, name)

status <- c(status,"collected")

amount <- c(amount,collectNum)

}

x<-x+1

}#end-while (FLOP ROUND)

## TURN ROUND

if(startsWith(df[x,2],"\*\*\* TURN \*\*\*") == TRUE){

x <- x+1

while (startsWith(df[x,2],"\*\*\* SUMMARY \*\*\*") != TRUE && startsWith(df[x,2],"\*\*\* RIVER \*\*\*") != TRUE){

## MrBlue

if(startsWith(df[x,2],"MrBlue: folds") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlue")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlue")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlue: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlue")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlue: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlue")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlue")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

## MrBlonde

else if (startsWith(df[x,2],"MrBlonde: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlonde: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlonde: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

## MrWhite

else if (startsWith(df[x,2],"MrWhite: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrWhite")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrWhite: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrWhite")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrWhite: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrWhite")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrWhite: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrWhite")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrWhite: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrWhite")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

## MrPink

else if (startsWith(df[x,2],"MrPink: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrPink")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrPink: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrPink")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrPink: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrPink")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrPink: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrPink")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrPink: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrPink")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

## MrBrown

else if (startsWith(df[x,2],"MrBrown: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBrown")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrBrown: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBrown")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrBrown: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBrown")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBrown: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBrown")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBrown: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "MrBrown")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

## Pluribus

else if(startsWith(df[x,2],"Pluribus: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "Pluribus")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"Pluribus: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "Pluribus")

status <- c(status,"calls")

amount <- c(amount,callNum)

turnCurrentPot <- turnCurrentPot + callNum

}

else if (startsWith(df[x,2],"Pluribus: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "Pluribus")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

turnCurrentPot <- turnCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "Pluribus: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "Pluribus")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "Pluribus: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, "Pluribus")

status <- c(status,"bets")

amount <- c(amount,betNum)

turnCurrentPot <- turnCurrentPot + betNum

}

# Uncalled bet

else if (startsWith(df[x,2],"Uncalled bet") == TRUE){

name <- sub(".\*returned to ","",df[x,2])

returnNum <- strtoi(sub(" returned to.\*","",sub(".\*Uncalled bet ","", gsub("[()]", "", df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, name)

status <- c(status,"returns")

amount <- c(amount,returnNum)

}

# Collected pot

else if (startsWith(df[x,2],"") == TRUE){

name <- gsub(" ","",sub("collected.\*","",df[x,2]))

collectNum <- strtoi(sub(".0frompot.\*","",sub(".\*collected","",gsub(" ","",df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "turn")

playerName <- c(playerName, name)

status <- c(status,"collected")

amount <- c(amount,collectNum)

}

x<-x+1

}#end-while (TURN ROUND)

## RIVER ROUND

if(startsWith(df[x,2],"\*\*\* RIVER \*\*\*") == TRUE){

x <- x+1

while (startsWith(df[x,2],"\*\*\* SUMMARY \*\*\*") != TRUE && startsWith(df[x,2],"\*\*\* SHOWDOWN \*\*\*") != TRUE){

## MrBlue

if(startsWith(df[x,2],"MrBlue: folds") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlue")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlue")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlue: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlue")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlue: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlue")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlue: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlue")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

## MrBlonde

else if (startsWith(df[x,2],"MrBlonde: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2], "MrBlonde: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBlonde: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBlonde: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBlonde")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

## MrWhite

else if (startsWith(df[x,2],"MrWhite: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrWhite")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrWhite: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrWhite")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrWhite: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrWhite")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrWhite: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrWhite")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrWhite: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrWhite")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

## MrPink

else if (startsWith(df[x,2],"MrPink: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrPink")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrPink: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrPink")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrPink: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrPink")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrPink: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrPink")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrPink: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrPink")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

## MrBrown

else if (startsWith(df[x,2],"MrBrown: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBrown")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"MrBrown: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBrown")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2],"MrBrown: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBrown")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "MrBrown: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBrown")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "MrBrown: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "MrBrown")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

## Pluribus

else if(startsWith(df[x,2],"Pluribus: folds") == TRUE) {

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "Pluribus")

status <- c(status,"folds")

amount <- c(amount,0)

}

else if (startsWith(df[x,2],"Pluribus: calls") == TRUE){

callNum <- strtoi(gsub(" ","", sub(".\*calls","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "Pluribus")

status <- c(status,"calls")

amount <- c(amount,callNum)

riverCurrentPot <- riverCurrentPot + callNum

}

else if (startsWith(df[x,2],"Pluribus: raises") == TRUE){

raiseNum <- strtoi(gsub(" ","", sub(".\*to","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "Pluribus")

status <- c(status,"raises")

amount <- c(amount,raiseNum)

riverCurrentPot <- riverCurrentPot + raiseNum

}

else if (startsWith(df[x,2], "Pluribus: checks") == TRUE){

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "Pluribus")

status <- c(status,"checks")

amount <- c(amount,0)

}

else if (startsWith(df[x,2], "Pluribus: bets") == TRUE){

betNum <- strtoi(gsub(" ","", sub(".\*bets","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, "Pluribus")

status <- c(status,"bets")

amount <- c(amount,betNum)

riverCurrentPot <- riverCurrentPot + betNum

}

# Uncalled bet

else if (startsWith(df[x,2],"Uncalled bet") == TRUE){

name <- sub(".\*returned to ","",df[x,2])

returnNum <- strtoi(sub(" returned to.\*","",sub(".\*Uncalled bet ","", gsub("[()]", "", df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, name)

status <- c(status,"returns")

amount <- c(amount,returnNum)

}

# Collected pot

else if (startsWith(df[x,2],"") == TRUE){

name <- gsub(" ","",sub("collected.\*","",df[x,2]))

collectNum <- strtoi(sub(".0frompot.\*","",sub(".\*collected","",gsub(" ","",df[x,2]))))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "river")

playerName <- c(playerName, name)

status <- c(status,"collected")

amount <- c(amount,collectNum)

}

x<-x+1

}#end-while(RIVER ROUND)

## SHOWDOWN

if(startsWith(df[x,2],"\*\*\* SHOWDOWN \*\*\*") == TRUE){

x <- x+2

name <- sub(" collected.\*","",df[x,2])

showNum <- strtoi(sub(".0 from pot.\*","",sub(".\*collected ","",df[x,2])))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "showdown")

playerName <- c(playerName, name)

status <- c(status,"collected")

amount <- c(amount,showNum)

x <- x+5

name <- sub(".\*: ","",sub(" showed.\*","",df[x,2]))

pokerHand <- c(pokerHand,handNum)

round <- c(round, "showdown")

playerName <- c(playerName, name)

status <- c(status,"lost")

amount <- c(amount,0)

}#end-if(SHOWDOWN)

}#end-if(RIVER)

}#end-if(TURN)

}#end-if (FLOP)

x <- x+1

next

}#end-elseif

x <- x+1

}#end-while(ENTIRE DF)

pokerHand2 <- c(pokerHand2,handNum)

holePot <- c(holePot,holeCurrentPot)

flopPot <- c(flopPot,flopCurrentPot)

turnPot <- c(turnPot,turnCurrentPot)

riverPot <- c(riverPot,riverCurrentPot)

```

#### store in new dataframe, df3 (HISTORICAL with 3383 obs of 5 variables)

```{r}

df3 <- data.frame(pokerHand,round,playerName,status,amount)

```

#### store in new dataframe, df4 (pot amount per poker game)

```{r}

df4 <- data.frame(pokerHand2,holePot,flopPot,turnPot,riverPot)

df4 <- df4[-1,]

colnames(df4) <- c("pokerHand","holePot","flopPot","turnPot","riverPot")

```

#### check for games with 2 winners and remove that game from all dataframes, just to standardize analysis.

```{r}

df2 <- df2[-164,]

df3 <- df3[-1868:-1886,]

df4 <- df4[-165,]

df3[729,5] <- 6825

df3[730,5] <- 6825

df3[2099,5] <- 0

df3[2654,5] <- 0

df3[2655,5] <- 275

df3[2721,5] <- 225

df3[2721,3] <- "MrBlue"

df3[2722,3] <- "MrBlonde"

df3[2795,5] <- 200

df3[2796,5] <- 0

df3[3057,5] <- 0

df3[3057,3] <- "MrBlue"

df3[3058,3] <- "MrBlonde"

df3[3132,5] <- 0

df2 <- df2[-92,]

df3 <- df3[-1081:-1094,]

df4 <- df4[-93,]

```

#### filter winners by filtering players who collected money in each game and store into new dataframe, dfWinner (1 row = 1 winner, 304 obvs of 5 variables; Because there are 304 obvs, there is 1 game that has 2 winners, which is game#102000.)

```{r}

dfWinner <- df3

dfWinner <- data.frame(filter(dfWinner,dfWinner$status=="collected"))

# rename dfWinner columns

colnames(dfWinner) <- c("pokerHand","winningRound","winner","status","totalPot")

```

#### merge dfWinner to df2 to create masterData1

```{r}

# merge dfWinner & df2 = masterData1

masterData1 <- merge(df2, df4, by = "pokerHand")

masterData1 <- merge(masterData1, dfWinner, by = "pokerHand")

masterData1 <- masterData1[,c(-17,-24)]

colnames(masterData1)[23] <- "totalPot"

masterData1[279,17]<-13800

masterData1[62,20] <- 13650

masterData1[183,20] <- 7875

masterData1[236,20] <- 11960

masterData1[248,20] <- 12500

```

#### Merge df2 & df3 to create masterData2

```{r}

# merge df2 & df3 = masterData2

masterData2 <- merge(df2,df3, by = "pokerHand")

masterData2 <- merge(masterData2, dfWinner, by = "pokerHand")

masterData2 <- masterData2[,c(-17,-24)]

colnames(masterData2)[23] <- "totalPot"

colnames(masterData2)[19] <- "status"

masterData2[3117,20] <- 10000

```

ANALYSIS:

```{r}

numHoleWins <- sum(masterData1$winningRound == "hole")

numFlopWins <- sum(masterData1$winningRound == "flop")

numTurnWins <- sum(masterData1$winningRound == "turn")

numRiverWins <- sum(masterData1$winningRound == "river")

numShowdownWins <- sum(masterData1$winningRound == "showdown")

numHoleWins

numFlopWins

numTurnWins

numRiverWins

numShowdownWins

# hole=140, flop=47, turn=33, river=33, showdown=46

# 299 = number of pokerHands

percentHoleWins <- numHoleWins/299

percentFlopWins <- numFlopWins/299

percentTurnWins <- numTurnWins/299

percentRiverWins <- numRiverWins/299

percentShowdownWins <- numShowdownWins/299

percentHoleWins

percentFlopWins

percentTurnWins

percentRiverWins

percentShowdownWins

# hole=0.4682274, flop=0.1571906, turn=0.1103679, river=0.1103679, showdown=0.1538462

```

#### 2) summary of different pots per round

```{r}

summary(masterData1$holePot)

summary(masterData1$flopPot)

summary(masterData1$turnPot)

summary(masterData1$riverPot)

summary(masterData1$totalPot)

```

#### sub-data frames from masterData2

```{r}

df5 <- data.frame(filter(masterData2,masterData2$round=="hole"))

df5 <- df5[,19:20]

df6 <- data.frame(filter(masterData2,masterData2$round=="flop"))

df6 <- df6[,19:20]

df7 <- data.frame(filter(masterData2,masterData2$round=="turn"))

df7 <- df7[,19:20]

df8 <- data.frame(filter(masterData2,masterData2$round=="river"))

df8 <- df8[,19:20]

```

#### hole around aggregate

```{r}

aggregate(df5$amount,list(df5$status), FUN=mean)

aggregate(df5$amount,list(df5$status), FUN=min)

aggregate(df5$amount,list(df5$status), FUN=max)

aggregate(df6$amount,list(df6$status), FUN=mean)

aggregate(df6$amount,list(df6$status), FUN=min)

aggregate(df6$amount,list(df6$status), FUN=max)

aggregate(df7$amount,list(df7$status), FUN=mean)

aggregate(df7$amount,list(df7$status), FUN=min)

aggregate(df7$amount,list(df7$status), FUN=max)

aggregate(df8$amount,list(df8$status), FUN=mean)

aggregate(df8$amount,list(df8$status), FUN=min)

aggregate(df8$amount,list(df8$status), FUN=max)

```

#### start linear modeling

```{r}

library(imputeTS)

library(stats)

library(datasets)

df9 <- masterData1

df9 <- df9[,c(1,17,18,19,20,21,22,23)]

df9 <- df9[,-6:-7]

```

#### bi-variate scatterplots

```{r}

# plot points with holePot in the x axis and totalPot in the y axis.

plotholePot<- ggplot(data = df9, aes(x = holePot, y = totalPot)) +

geom\_point(aes(size = holePot, color = totalPot)) +

ggtitle("holePot vs. totalPot")

plotholePot

# plot points with holePot in the x axis and totalPot in the y axis.

plotflopPot<- ggplot(data = df9, aes(x = flopPot, y = totalPot)) +

geom\_point(aes(size = flopPot, color = totalPot)) +

ggtitle("flopPot vs. totalPot")

plotflopPot

# plot points with holePot in the x axis and totalPot in the y axis.

plotturnPot<- ggplot(data = df9, aes(x = turnPot, y = totalPot)) +

geom\_point(aes(size = turnPot, color = totalPot)) +

ggtitle("turnPot vs. totalPot")

plotturnPot

# plot points with holePot in the x axis and totalPot in the y axis.

plotriverPot<- ggplot(data = df9, aes(x = riverPot, y = totalPot)) +

geom\_point(aes(size = riverPot, color = totalPot)) +

ggtitle("riverPot vs. totalPot")

plotriverPot

```

#### LINEAR MODEL

```{r}

multiReg <- lm(formula = df9$totalPot ~ holePot + flopPot + turnPot + riverPot, data=df9)

summary(multiReg)

```

#### Store df, masterData1 and masterData2 into excel files

```{r}

write.csv(df,"/Users/jaywhitehead/Desktop/IST 687/IST 687 Project/pluribus\_converted\_logs/df.csv",row.names=FALSE)

write.csv(masterData1,"/Users/jaywhitehead/Desktop/IST 687/IST 687 Project/pluribus\_converted\_logs/masterData1.csv",row.names=FALSE)

write.csv(masterData2,"/Users/jaywhitehead/Desktop/IST 687/IST 687 Project/pluribus\_converted\_logs/masterData2.csv",row.names=FALSE)

```

One Dollar Spin (Kaggle)

