Clustering Gambling Behavior

Introduction

The similarities and differences in the behaviours of different people have long been of interest, particularly in psychology and other social science fields. Understanding human behaviour in particular contexts can help us to make informed decisions. Consider a poker game - understanding why players raise, call, and fold in various situations can provide a competitive advantage.

Along these lines, we will focus on the behavior of **online gamblers** from a platform called Bustabit. There are a few basic rules for playing a game of Bustabit:

1. You bet a certain amount of money (in Bits, which is 1 / 1,000,000th of a Bitcoin), and you win if you cash out before the game **busts**.
2. The multiplier value calculates your win at the moment you cashed out. For example, if you bet 100 and the value was 2.50x when you cashed out, you win 250. In addition, a percentage Bonus per game is multiplied with your bet and summed to give your final Profit in a winning game. Assuming a Bonus of 1%, your Profit for this round would be (100 x 2.5) + (100 x .01) - 100 = 151
3. The multiplier increases as time goes on, but if you wait too long to cash out, you may bust and lose your money.
4. Lastly, the house maintains slight advantages because everyone plays busts in 1 out of every 100 games.

Our goal will be to define relevant **groups** or **clusters** of Bustabit users to identify what patterns and behaviours of gambling persist.

The dataset

We will be using data from bustabit with 40000 games by over 4000 different players, for a total of 50000 rows (one game played by one player). The data includes the following variables:

- **Id** - Unique identifier for a particular row (game result for one player)
- **GameID** - Unique identifier for a particular game
- **Username** - Unique identifier for a particular player
- **Bet** - The number of Bits (1 / 1,000,000th of a Bitcoin) bet by the player in this game
- **CashedOut** - The multiplier at which this particular player cashed out
- **Bonus** - The bonus award (in per cent) awarded to this player for the game
- **Profit** - The amount this player won in the game, calculated as (Bet *CashedOut) + (Bet* Bonus) – Bet
- **BustedAt** - The multiplier value at which this game busted
- **PlayDate** - The date and time at which this game took place

```{r}
# Load the relevant package
library(tidyverse)
library(stats)
library(GGally)

# Load the data
bustabit <- read.csv("bustabit.csv")
```

```
# Find the highest multiplier (BustedAt value) achieved in a game
bustabit %>%
  arrange(desc(BustedAt)) %>%
  slice(1)
```

Id <int>	GameID <int>	Username <fctr>	Bet <int>	CashedOut <dbl>	Bonus <dbl>	Profit <dbl>	BustedAt <dbl>	PlayDate <fctr>
19029273	3395044	Shadowshot	130	2	2.77	133.6	251025.1	2016-11-29T00:03:05Z

The Bustabit data provides us with many features, but to better quantify player behavior, we need to derive some more variables. Currently, we have a Profit column that tells us the amount won in that game, but no indication of how much was lost if the player busted, and no indicator variable quantifying whether the game itself was a win or loss overall. Hence, we will derive or modify the following variables:

- **CashedOut** - If the value for CashedOut is NA, we will set it to be 0.01 greater than the BustedAt value to signify that the user failed to cash out before busting
- **Profit** - If the value for Profit is NA, we will set it to be zero to indicate no gain for the player in that game
- **Losses** - If the new value for Profit is zero, we will set this to be the amount the player lost in that game, otherwise we will set it to zero. This value should always be *zero or negative*
- **GameWon** - If the user made a profit in this game, the value should be 1, and 0 otherwise
- **GameLost** If the user had a loss in this game, the value should be 1, and 0 otherwise

```{r}
# Create the new feature variables
bustabit_features <- bustabit %>%
  mutate(CashedOut = ifelse(is.na(CashedOut), BustedAt + 0.01, CashedOut),
         Profit = ifelse(is.na(Profit), 0, Profit),
         Losses = ifelse(Profit==0, -1*Bet, 0),
         GameWon = ifelse(Profit==0, 0, 1),
         GameLost = ifelse(Profit==0, 1, 0))

# Look at the first five rows if the feature data
head(bustabit_features, 5)
```

	Id <int>	GameID <int>	Username <fctr>	Bet <int>	CashedOut <dbl>	Bonus <dbl>	Profit <dbl>	BustedAt <dbl>	PlayDate <fctr>
1	14196549	3366002	papai	5	1.20	0.0	1.00	8.24	2016-11-20T19:44:19Z
2	10676217	3343882	znay22	3	1.41	NA	0.00	1.40	2016-11-14T14:21:50Z
3	15577107	3374646	rrrrrrrr	4	1.33	3.0	1.44	3.15	2016-11-23T06:39:15Z
4	25732127	3429241	sanya1206	10	1.64	NA	0.00	1.63	2016-12-08T18:13:55Z
5	17995432	3389174	ADM	50	1.50	1.4	25.70	2.29	2016-11-27T08:14:48Z

Empirical Analysis

The primary task at hand is to cluster Bustabit **players** by their respective gambling habits. Right now, however, we have features at the per-game level. The features we've derived would be great if we were interested in clustering properties of the games themselves - we know things about the BustedAt multiplier, the time the game took place, and lots more. But to better quantify player behavior, we must group the data by a player (Username) to begin thinking about the relationship and similarity between groups of players. Some per-player features we will create are:

- **AverageCashedOut** - The average multiplier at which the player cashes out
- **AverageBet** - The average bet made by the player
- **TotalProfit** - The total profits over time for the player
- **TotalLosses** - The total losses over time for the player
- **GamesWon** - The total number of individual games the player won
- **GamesLost** - The total number of individual games the player lost

With these variables, we will be able to potentially group similar users based on their typical Bustabit gambling behavior.

```{r}
# Group by players to create per-player summary statistics
bustabit_clus <- bustabit_features %>%
  group_by(Username) %>%
  summarize(AverageCashedOut = mean(CashedOut),
            AverageBet = mean(Bet),
            TotalProfit = sum(Profit),
            TotalLosses = sum(Losses),
            GamesWon = sum(GameWon),
            GamesLost = sum(GameLost))

# View the first five rows of the data
head(bustabit_clus, n = 5)
```

Username <fctr>	AverageCashedOut <dbl>	AverageBet <dbl>	TotalProfit <dbl>	TotalLosses <dbl>	GamesWon <dbl>	GamesLost <dbl>
_caramba_tm_	1.7000	1.333333	3.13	0	3	0
Dear	1.6625	215.000000	0.00	-860	0	4
_lsx	1.2020	6282.000000	3544.56	-2000	4	1
noBap	6.5800	4.000000	0.00	-4	0	1
_TechDeck	1.1900	6.000000	0.00	-6	0	1

The variables are on very different **scales** right now. For example, AverageBet is in bits (1/1000000 of a Bitcoin), AverageCashedOut is a multiplier, and GamesLostand GamesWon are counts. As a result, we would like to **normalize** the variables such that across clustering algorithms, they will have approximately equal weighting.

One thing to think about is that we may want a particular numeric variable to maintain a higher weight in many cases. This could occur if there is some prior knowledge regarding, for example, which variable might be most important in terms of defining similar Bustabit behavior. In this case, without that prior knowledge, we will forego the weighting of variables and scale everything. We are going to use **mean-sd** standardization to scale the data. Note that this is also known as a **Z-score**.

Note that we could compute the Z-scores by using the base R function scale(), but we're going to write our own function in order to get the practice.

```{r}
# Create the mean-sd standardization function
mean_sd_standard <- function(x) {
  (x-mean(x)) / sd(x)
}

# Apply the function to each numeric variables in the clustering
bustabit_standardized <- bustabit_clus %>%
```

```
  mutate_if(is.numeric, mean_sd_standard)

# Summarize our standardized data
summary(bustabit_standardized)
```

```
       Username    AverageCashedOut     AverageBet       TotalProfit        TotalLosses          GamesWon
_caramba_tm_:   1   Min.   :-0.76289   Min.   :-0.1773   Min.   :-0.09052   Min.   :-41.84541   Min.   :-0.4320
_Dear_      :   1   1st Qu.:-0.28157   1st Qu.:-0.1765   1st Qu.:-0.09050   1st Qu.:  0.09837   1st Qu.:-0.3696
_lsx        :   1   Median :-0.18056   Median :-0.1711   Median :-0.08974   Median :  0.10847   Median :-0.3071
_noBap_     :   1   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.0000
_TechDeck   :   1   3rd Qu.: 0.02752   3rd Qu.:-0.1384   3rd Qu.:-0.08183   3rd Qu.:  0.10916   3rd Qu.:-0.1196
-____--     :   1   Max.   :41.72651   Max.   :24.9971   Max.   :40.73652   Max.   :  0.10916   Max.   :13.2534
(Other)     :4143
  GamesLost         cluster
Min.   :-0.41356   1:3863
1st Qu.:-0.41356   2:  17
Median :-0.33306   3:  16
Mean   : 0.00000   4:  94
3rd Qu.:-0.09156   5: 159
Max.   :19.30911
```

With standardized data of per-player features, we are now ready to use K means clustering to cluster the players based on their online gambling behavior. Without prior knowledge, it is often difficult to know an appropriate choice for the number of clusters. We will begin by choosing five. This choice is somewhat arbitrary but represents an excellent initial compromise between choosing too many clusters (which reduces the interpretability of the final results) and choosing too few clusters (which may not capture the specific behaviors effectively). Feel free to play around with other choices for the number of clusters and see what you get instead!

One subtlety to note - because the K means algorithm uses a random start, we will set a random seed first to ensure the results are reproducible.

```{r}
# Set the seed
set.seed(100)

# Cluster the players using kmeans with five clusters
cluster_solution <- kmeans(bustabit_standardized[,-1], centers = 5)

# Store the cluster assignments back into the clustering data frame object
bustabit_clus$cluster <- factor(cluster_solution$cluster)

# Look at the distribution of cluster assignments
table(bustabit_clus$cluster)
```

```
   1    2    3    4    5
3863   17   16   94  159
```

We have a clustering assignment that maps every Bustabit gambler to one of five different groups. To begin assessing the quality and distinctiveness of these groups, we will look at group averages for each cluster across the original variables in our clustering dataset. This will, for example, allow us to see which cluster tends to make the largest bets, which cluster tends to win the most games, and which cluster

tends to lose the most money. This will provide us with our first clear indication as to whether the behaviors of the groups appear distinctive!

```{r}
# Group by the cluster assignment and calculate averages
bustabit_clus_avg <- bustabit_clus %>%
  group_by(cluster) %>%
  summarize_if(is.numeric, mean)

# View the resulting table
bustabit_clus_avg
```

cluster <fctr>	AverageCashedOut <dbl>	AverageBet <dbl>	TotalProfit <dbl>	TotalLosses <dbl>	GamesWon <dbl>	GamesLost <dbl>
1	1.699412	3896.128	4906.6041	-4985.9886	4.2195185	2.772457
2	27.448235	1278.257	619.4041	-581.2941	0.7058824	1.529412
3	2.470024	298945.662	1198191.1631	-1056062.1875	10.5625000	8.062500
4	1.300588	367.926	11467.1474	-8653.7340	89.3085106	29.244681
5	2.538025	1337.694	30732.2461	-31276.5660	23.9559748	48.433962

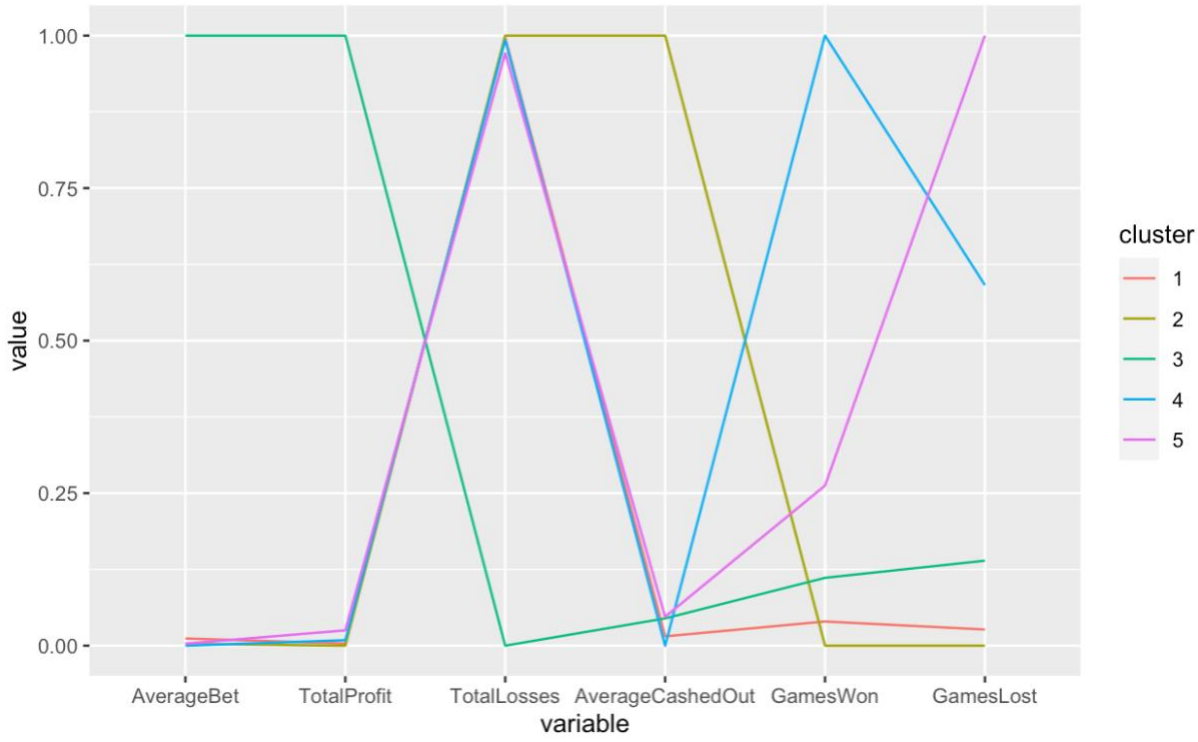
We can already learn a bit about our cluster groupings by looking at the previous table. We can see a group that makes huge bets, a group that tends to cash out at very high multiplier values, and a group that has played many games of Bustabit. We can visualize these group differences graphically using a Parallel Coordinate Plot or PCP. To do so, we will introduce one more kind of scaling: min-max scaling, which forces each variable to fall between 0 and 1.

Other choices of scaling, such as the Z-score method from before, can work effectively as well. However, min-max scaling has the advantage of **interpretability**- a value of 1 for a particular variable indicates that the cluster has the highest value compared to all other clusters, and a value of 0 indicates that it has the lowest. This can help make relative comparisons between the clusters more clear.

```{r}
# Create the min-max scaling function
min_max_standard <- function(x) {
  (x-min(x)) / (max(x) - min(x))
}

# Apply this function to each numeric variable in the bustabit_clus_avg object
bustabit_avg_minmax <- bustabit_clus_avg %>%
  mutate_if(is.numeric, min_max_standard)

# Create a parallel coordinate plot of the values
ggparcoord(bustabit_avg_minmax, columns = 2:ncol(bustabit_avg_minmax),
        groupColumn = "cluster", scale = "globalminmax", order = "skewness")
```



One issue with plots like the previous is that they get more unwieldy as we continue to add variables. One way to solve this is to use the Principal Components of a dataset to reduce the dimensionality to aid visualization. Essentially, this is a two-stage process:

1. We extract the principal components to reduce the dimensionality of the dataset so that we can produce a scatterplot in two dimensions that capture the underlying structure of the higher-dimensional data.
2. We then produce a scatterplot of each observation (in this case, each player) across the two Principal Components and color according to their cluster assignment to visualize the separation of the clusters.

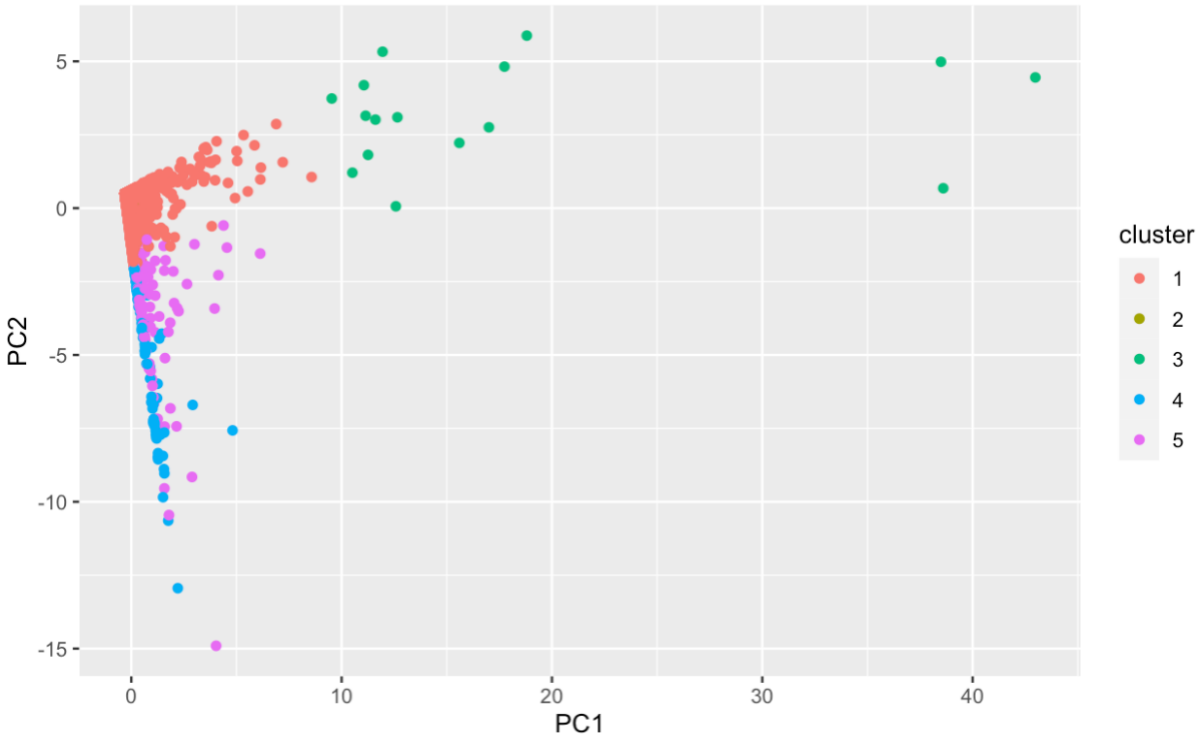
This plot provides exciting information in terms of the similarity of any two players. You will see that players who fall close to the boundaries of clusters might be the ones that exhibit the gambling behavior of a couple of different clusters. After you produce your plot, try to determine which clusters seem to be the most "different." Also, try playing around with different data projections, such as PC3 vs. PC2, or PC3 vs. PC1, to see if you can find one that better differentiates the groups.

```{r}
# Calculate the principal components of the standardized data
my_pc <- as.data.frame(prcomp(bustabit_standardized[,-1])$x)

# Store the cluster assignments in the new data frame
my_pc$cluster <- bustabit_clus$cluster

# Use the ggplot() to plot PC2 vs PC1, and color by the cluster assignment
p1 <- ggplot(data = my_pc, aes(x = PC1, y = PC2, color = cluster)) + geom_point()

p1
```



Conclusion

Though most of the statistical and programmatical work has been completed, the essential part of cluster analysis is to interpret the resulting clusters. This often is the most desired aspect of the research by clients, who are hoping to use the results of your analysis to inform better business decision-making and actionable items. As a final step, we'll use the parallel coordinate plot and cluster means table to interpret the Bustabit gambling user groups! Roughly speaking, we can breakdown the groups as follows:

Risky Commoners:
These users seem to be a step above the Cautious Commoners in their Bustabit gambling habits, making larger average bets, and playing a larger number of games on the site. As a result, though they have about the same number of average games won as the Risk Takers, they have a significantly higher number of games lost.

High Rollers:
High bets are the name of the game for this group. They bet large sums of money in each game, although they tend to cash out at lower multipliers and thus play the game more conservatively, particularly compared to the Risk-Takers. Interestingly, these users have also, on average, earned net positive earnings from their games played.

Risk Takers:
These users have played only a couple of games on average. Still, their average cashed-out value is significantly higher than the other clusters, indicating that they tend to wait for the multiplier to increase to large values before cashing out.

Cautious Commoners:
This is the largest of the five clusters and might be described as the more casual Bustabit players. They've played the fewest number of games overall and tend to make more conservative bets in general.

Strategic Addicts:

These users play a lot of games on Bustabit, but tend to keep their bets under control. As a result, they've made on average a net positive earnings from the site, despite having the most games played. They seem to maintain a strategy (or an automated script/bot) that works to earn them money.

```{r}
# Assign cluster names to clusters 1 through 5 in order
cluster_names <- c(
   "Risky Commoners",
   "High Rollers",
   "Risk Takers",
   "Cautious Commoners",
   "Strategic Addicts"
)

# Append the cluster names to the cluster means table
bustabit_clus_avg_names <- bustabit_clus_avg %>%
  cbind(Name = cluster_names)

# View the cluster means table with your appended cluster names
bustabit_clus_avg_names
```

cluster <fctr>	AverageCashedOut <dbl>	AverageBet <dbl>	TotalProfit <dbl>	TotalLosses <dbl>	GamesWon <dbl>	GamesLost <dbl>	Name <fctr>
1	1.699412	3896.128	4906.6041	-4985.9886	4.2195185	2.772457	Risky Commoners
2	27.448235	1278.257	619.4041	-581.2941	0.7058824	1.529412	High Rollers
3	2.470024	298945.662	1198191.1631	-1056062.1875	10.5625000	8.062500	Risk Takers
4	1.300588	367.926	11467.1474	-8653.7340	89.3085106	29.244681	Cautious Commoners
5	2.538025	1337.694	30732.2461	-31276.5660	23.9559748	48.433962	Strategic Addicts

Source:

- https://www.bustabit.com