Segmenting for a Social Entertainment App

MKTG6310-090 Su2019 Assignment 2

Assignment Team 3: Tianyu Xu, Sakshi Vig, Justin Peterson, Richard Mohlman, Steve Lucero, Kris Clegg

R Setup

In the course of our analysis we leveraged the following R packages and libraries:

```
library(tidyverse)
library(missForest)
library(caret)
library(NbClust)
library(car)
library(corrplot)
library(cluster)
library(mclust)
library(flexmix)
library(InformationValue)
```

The App Happy Challenge

General Consensus at App Happy, is that there is a market for a new social entertainment app. However, App Happy currently only operates Apps in the B2B analytics category and don't yet have a product in the consumer entertainment app category.

Below we will perform a number of market segmentation analyses to explore this possible market opportunity and inform potential new product strategy and tactics including:

- descriptive post hoc segmentation analysis (a partitioning clustering method (k-means), or a hierarchical clustering method)
- predictive post hoc segmentation analysis (finite mixture regression, also called clusterwise regression, or latent class regression)

Additionally, to help App Happy better understand which respondents are using only free apps, we will build a model to predict whether new customers will only use free apps based on demographic variables.

Part 1: Exploratory Data Analysis

App Happy hired the Consumer Spy Corporation (CSC) to survey consumers in the entertainment app market. CSC collected data from a sample of consumers, and provided App Happy with a dataset of their responses. The survey questionnaire was based on preliminary qualitative research that included focus groups and one-on-one interviews. The data collected by CSC are the base data we use to complete the below analyses.

Recoding Likert Responses

We opt to reverse selected numerical values so that larger numbers indicate greater agreement.

Question 24

```
appHappyNums$q24r1 <- recode(appHappyNums$q24r1, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r2 <- recode(appHappyNums$q24r2, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r3 <- recode(appHappyNums$q24r3, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r4 <- recode(appHappyNums$q24r4, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r5 <- recode(appHappyNums$q24r5, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r6 <- recode(appHappyNums$q24r6, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r7 <- recode(appHappyNums$q24r7, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r8 <- recode(appHappyNums$q24r8, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r9 <- recode(appHappyNums$q24r9, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r10 <- recode(appHappyNums$q24r10, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r11 <- recode(appHappyNums$q24r11, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q24r12 <- recode(appHappyNums$q24r12, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
```

Question 25

```
appHappyNums$q25r1 <- recode(appHappyNums$q25r1, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r2 <- recode(appHappyNums$q25r2, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r3 <- recode(appHappyNums$q25r3, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r4 <- recode(appHappyNums$q25r4, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r5 <- recode(appHappyNums$q25r5, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r6 <- recode(appHappyNums$q25r6, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r7 <- recode(appHappyNums$q25r7, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r8 <- recode(appHappyNums$q25r8, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r9 <- recode(appHappyNums$q25r9, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r10 <- recode(appHappyNums$q25r10, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r11 <- recode(appHappyNums$q25r11, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q25r12 <- recode(appHappyNums$q25r12, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
```

Question 26

```
appHappyNums$q26r3 <- recode(appHappyNums$q26r3, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r4 <- recode(appHappyNums$q26r4, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r5 <- recode(appHappyNums$q26r5, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r6 <- recode(appHappyNums$q26r6, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r7 <- recode(appHappyNums$q26r7, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r8 <- recode(appHappyNums$q26r8, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r9 <- recode(appHappyNums$q26r9, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r10 <- recode(appHappyNums$q26r10, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r11 <- recode(appHappyNums$q26r11, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r12 <- recode(appHappyNums$q26r12, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r13 <- recode(appHappyNums$q26r13, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r14 <- recode(appHappyNums$q26r14, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r15 <- recode(appHappyNums$q26r15, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r16 <- recode(appHappyNums$q26r16, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r17 <- recode(appHappyNums$q26r17, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
appHappyNums$q26r18 <- recode(appHappyNums$q26r18, "1=6; 2=5; 3=4; 4=3; 5=2; 6=1")
```

Dealing with NAs

Calculating the number of records/columns prior to cleaning NAs.

```
cat('Number of records (rows, observations):', nrow(appHappyNums), '\n')
```

```
## Number of records (rows, observations): 1663
```

```
cat('Number of variables (columns):', ncol(appHappyNums), '\n')
```

```
## Number of variables (columns): 89
```

Determining how many NAs exist.

```
cat('Count of NAs by column:\n')
```

```
## Count of NAs by column:
```

```
naCounts <- appHappyNums %>% is.na %>% colSums
naCounts[naCounts>0]
```

```
## q5r1  q12  q57
##  493   20   91
```

Drop observations w/ NA in q12 (What % of Respondent's App Collection were Free To Download)

```
appHappyNums <- appHappyNums[!is.na(appHappyNums$q12),]
```

Confirming q12 NAs have been removed:

```
cat('Count of NAs by column:\n')
```

```
## Count of NAs by column:
```

```
naCounts <- appHappyNums %>% is.na %>% colSums
naCounts[naCounts>0]
```

```
## q5r1  q57
##  474   91
```

Confirm number of rows was reduced by number of NAs in q12 and number of columns matches original.

```
cat('Number of records (rows, observations):', nrow(appHappyNums), '\n')
```

```
## Number of records (rows, observations): 1643
```

```
cat('Number of variables (columns):', ncol(appHappyNums), '\n')
```

```
## Number of variables (columns): 89
```

Impute values for NAs in q57 (Gender)

```
set.seed(123)
appHappyNums[, -1] <- lapply(appHappyNums[, -1], factor) #casting all non-ID columns to factors for imp
appHappyNumsImp <- appHappyNums %>% dplyr::select(-c(q5r1)) # add all but columns with NAs that we'll k
appHappyNums <- appHappyNums %>% dplyr::select(c(caseID,q5r1)) # save columns w/ NAs for joining back t
appHappyNumsImp <- missForest(appHappyNumsImp)$ximp # impute missing data in q57
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
```

```
appHappyNums <- merge(appHappyNums, appHappyNumsImp, by="caseID") # join data back together
appHappyNums[, -1] <- lapply(appHappyNums[, -1], as.integer) #casting all non-ID columns back to numeri
```

Confirm q57 NAs are gone

```
cat('Count of NAs by column:\n')
```

```
## Count of NAs by column:
```

```
naCounts <- appHappyNums %>% is.na %>% colSums
naCounts[naCounts>0]
```

```
## q5r1
##  474
```

Drop Entire q5r1 Column

NAs in q5r1 are expected and are simply respondents that did not select "other" for q4.

```
appHappyNums <- appHappyNums %>% dplyr::select(-c(q5r1))
```

Confirm number of rows matches post-q12 NA drop and number of columns is reduced by 1

```
cat('Number of records (rows, observations):', nrow(appHappyNums), '\n')
```

```
## Number of records (rows, observations): 1643
```

```
cat('Number of variables (columns):', ncol(appHappyNums), '\n')
```

```
## Number of variables (columns): 88
```

Confirm no NAs remain

```
cat('Count of NAs by column:\n')
```

```
## Count of NAs by column:
```

```
naCounts <- appHappyNums %>% is.na %>% colSums
naCounts[naCounts>0]
```

```
## named numeric(0)
```

Dependant Variable Creation

Creating a new variable to use to predict whether individuals are likely to only use free apps.

```
appHappyNums$onlyFreeApps <- as.numeric(appHappyNums$q12==6)
table(appHappyNums$q12, appHappyNums$onlyFreeApps) # confirm new binary variable generated
```

```
##
##       0   1
##   1  23   0
##   2 185   0
##   3 287   0
##   4 388   0
##   5 414   0
##   6   0 346
```

Confirm number of columns is increased by 1

```
cat('Number of records (rows, observations):', nrow(appHappyNums), '\n')
```

```
## Number of records (rows, observations): 1643
```

```
cat('Number of variables (columns):', ncol(appHappyNums), '\n')
```

```
## Number of variables (columns): 89
```

Train/Test Split of the Data (80/20)

Splitting the data for onlyFreeApps model training and testing.

```
set.seed(123)
appHappyNumsTrain <- appHappyNums %>% dplyr::sample_frac(.8)
appHappyNumsTest  <- dplyr::anti_join(appHappyNums, appHappyNumsTrain, by = 'caseID')
```

```
cat('Number of rows in train dataframe:   ',nrow(appHappyNumsTrain),'\n')
```

```
## Number of rows in train dataframe:    1314
```

```
cat('Number of rows in test dataframe:     ',nrow(appHappyNumsTest),'\n')
```

```
## Number of rows in test dataframe:      329
```

```
cat('Number of rows in original dataframe:',nrow(appHappyNums))
```

```
## Number of rows in original dataframe: 1643
```

Part 2: Post Hoc Descriptive Segmentation Analysis and Profiling

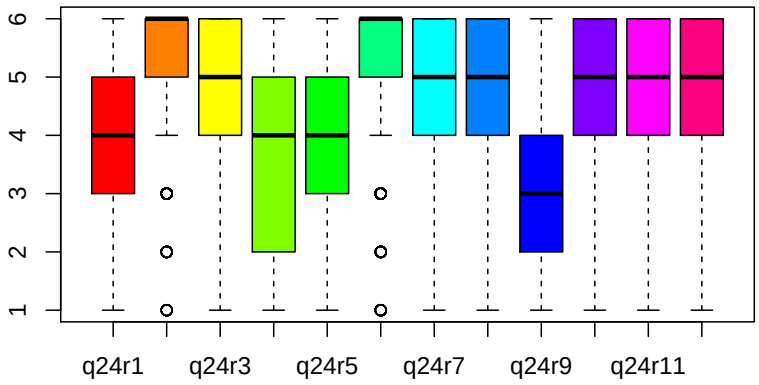
App Happy wants to segment the market based on customers' *attitudes*. Questionnaire items q24, q25, and q26 measure various attitudes. We will begin by creating new datasets to evaluate these questionnaire items.

```
#Create new datasets to evaluate Q24, Q25, and Q26
appHappyNu_q24 <- appHappyNums[,c('q24r1','q24r2','q24r3','q24r4','q24r5','q24r6','q24r7','q24r8','q24r
appHappyNu_q25 <- appHappyNums[,c('q25r1','q25r2','q25r3','q25r4','q25r5','q25r6','q25r7','q25r8','q25r
appHappyNu_q26 <- appHappyNums[,c('q26r3','q26r4','q26r5','q26r6','q26r7','q26r8','q26r9','q26r10','q26
appHappyNu_q24_q25 <- appHappyNums[,c('q24r1','q24r2','q24r3','q24r4','q24r5','q24r6','q24r7','q24r8','
appHappyNu_q24_q26 <- appHappyNums[,c('q24r1','q24r2','q24r3','q24r4','q24r5','q24r6','q24r7','q24r8','
appHappyNu_q25_q26 <- appHappyNums[,c('q25r1','q25r2','q25r3','q25r4','q25r5','q25r6','q25r7','q25r8','
```

Distribution Analysis

Create boxplots to find which questions have high variation/low variation, as well as see differences in average response by question type. Also create the correlation plots comparing each question. We have a plot for q24 to provide a different view of our distribution analysis.

```
boxplot(appHappyNu_q24, col=rainbow(length(unique(appHappyNu_q24))))
```



Boxplot Observations

- Evalutaing the boxplots we can see which questions have very little variance compared to the rest of the questions.
- Questions that jump out immediately with little variance are q24r2, q24r6, q25r8, q25r11, q26r9, q26r15, and q26r17. These responses were all almost exlusively in agreement to the questions.
- From the boxplots we can also see that the majority of questions people agreed with. There are a few that people generally did not agree with, like q24r9, q25r6, q26r11. The rest of the responses seemed to have a pretty good spread of responses across all possibilities.

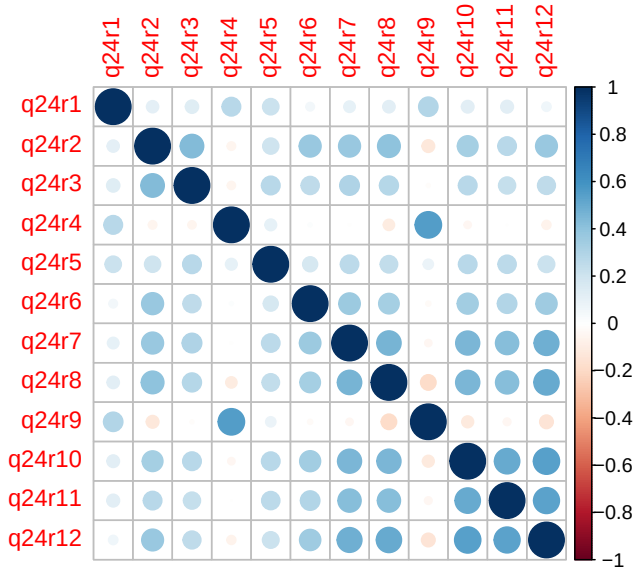
Correlation Analysis

Next we will evaluate the correlation plots to evaluate correlationsions of responses from the same question, and comparing groups of questions.

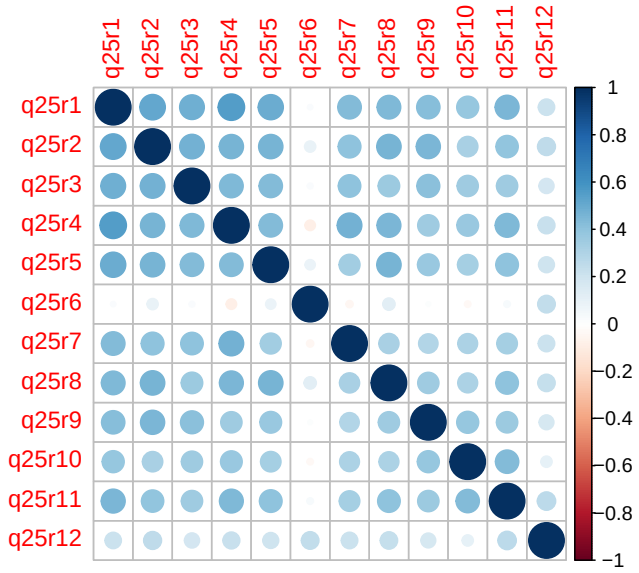
First, we evaluate correlation between sets of responses within each question:

```
M1 <- cor(appHappyNu_q24) # get correlations
M2 <- cor(appHappyNu_q25) # get correlations
M3 <- cor(appHappyNu_q26) # get correlations
```

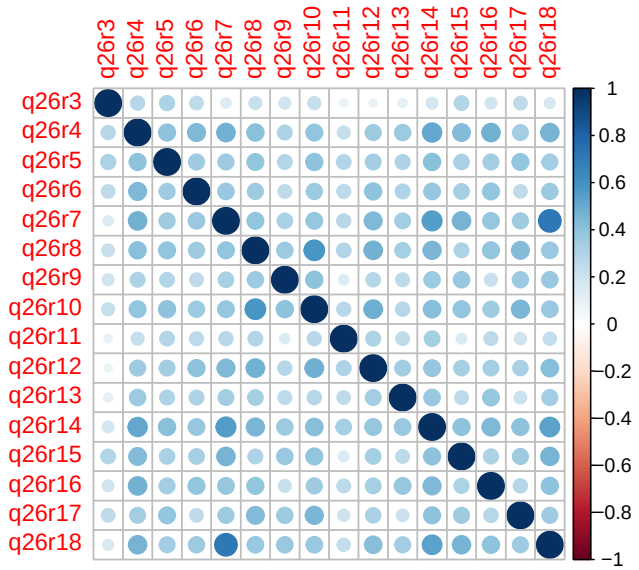
```
corrplot(M1, method = "circle") # plot matrix
```



```
corrplot(M2, method = "circle") # plot matrix
```




```
corrplot(M3, method = "circle") # plot matrix
```



We also took a look at the between question correlation matrices to see if any correlation patterns emerged there. We did not find anything significant and found that the between response correlations were stronger. As a result, we did not include those correlation matrices.

Observations

- Within Q24, r9 and r4 were most strongly positively correlated with one another, while most other items generally had smaller, yet still positive correlations with one another. r12 seemed to have the most possitive correlations with several other questions.
- Within Q25, all items except r6 and r12 were strongly positively correlated with one another.
- Within Q26, all items were positively correlated with one another, with r7 and r18 most strongly positively correlated with one another.

- Correlations appear to be stronger when comparing responses within questions as opposed to correlation across question sets q24, q25 & q26.

K-means Cluster Analysis

Next we are going to perform a k-means clustering analysis. First we need to find out the ideal number of clusters to use for the K-Means Cluster. We decided to use two methods, the **elbow method** and **NBClust** to determine what the optimal number of clusters are.

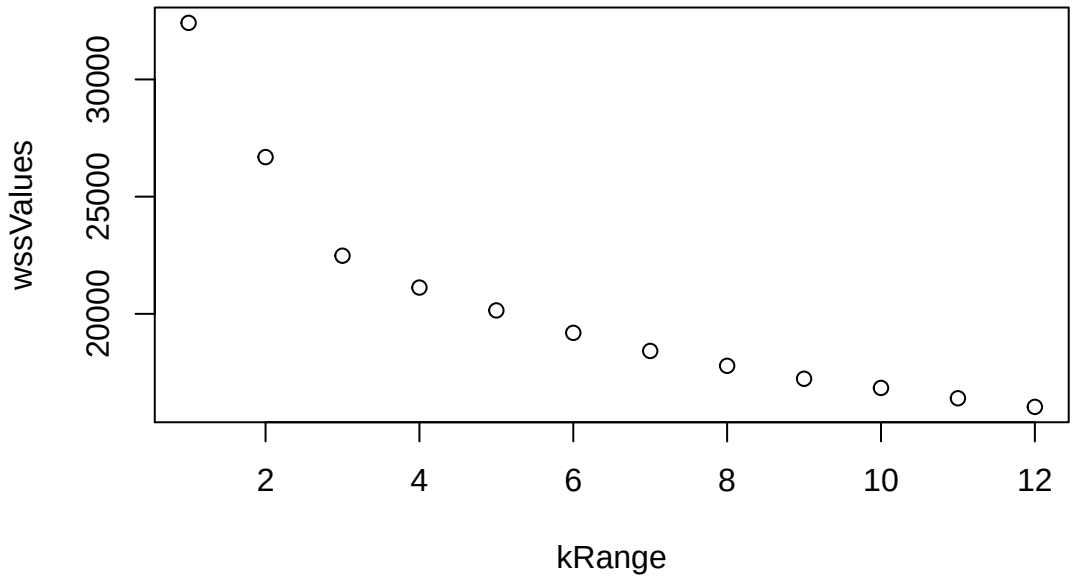
Elbow Method

The 'elbow method' can be used to determine the optimal number of clusters. We want to look for the **elbow** or **knee** of the graph (where the curve flattens out).

```
wss <- function(k){
                  kmeans(appHappyNu_q24,k,nstart=10)$tot.withinss
                  }

kRange <- 1:12
wssValues <- unlist(lapply(kRange, wss))

# plot where x-axis is kRange and y-axis is wssValues
plot(x=kRange,y=wssValues)
```



* Elbow appears to be between **2** and **3**

NBClust Method

```
# K Means clustering method using NBClust
numClusts <- NbClust(appHappyNu_q24,
                     min.nc = 2,
                     max.nc = 12,
                     method = "kmeans",
                     index = "gap")
numClusts$Best.nc
```

```
## Number_clusters     Value_Index
##          2.0000         -0.9638
```

- NBClust method recommended using 2 clusters.

  We decided to proceed and use 3 clusters.

Create Clusters

```
# Set the seed to be able to have others get the same results as us using a random process
set.seed(5)
clusters <- kmeans(appHappyNu_q24,
                   3,
                   nstart=15)
clusters$size # number of records assigned to each cluster
```

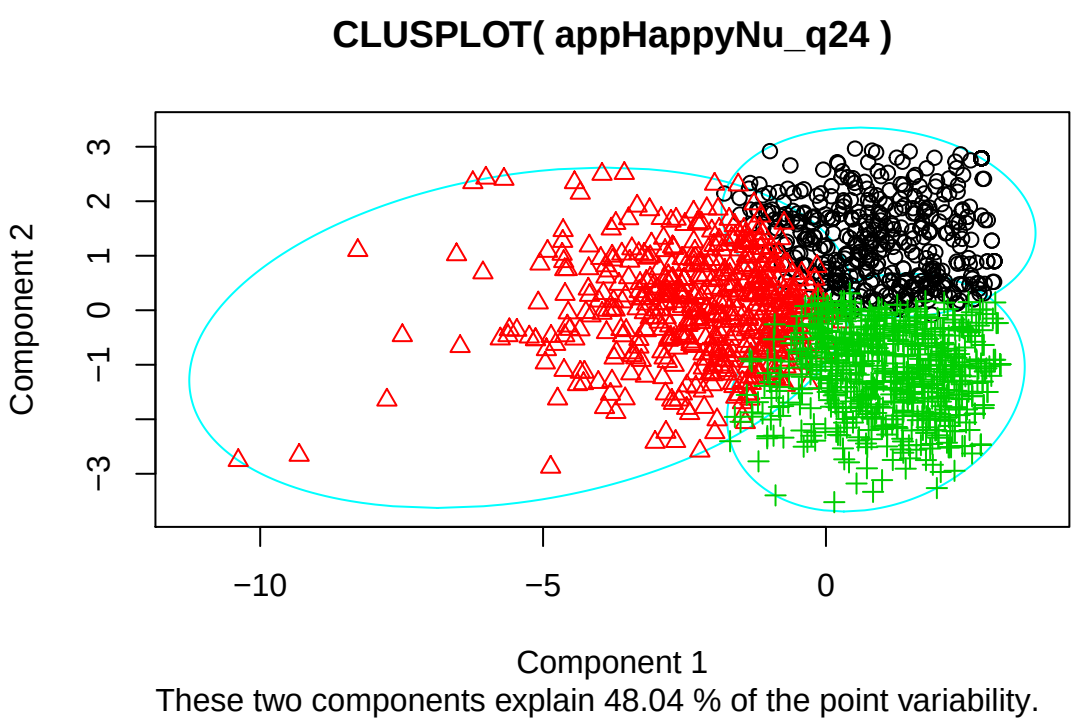
```
## [1] 480 549 614
```

Cluster Evaluation

Visualization

First we graph out the clusters to get a visual representation.

```
clusplot(appHappyNu_q24,clus=clusters$cluster,col.p=clusters$cluster)
```



Average responses for each cluster.

```
#let us find the cluster centers
clusters$centers
```

```
##      q24r1    q24r2    q24r3    q24r4    q24r5    q24r6    q24r7    q24r8
## 1 5.010417 5.595833 5.360417 5.060417 4.881250 5.556250 5.397917 5.337500
## 2 3.455373 4.632058 3.830601 3.930783 3.316940 4.544627 4.074681 4.211293
## 3 3.242671 5.675896 5.197068 2.561889 4.164495 5.540717 5.379479 5.485342
##      q24r9   q24r10   q24r11   q24r12
## 1 4.487500 5.395833 5.150000 5.375000
## 2 3.520947 4.089253 3.579235 4.209472
## 3 1.851792 5.379479 5.001629 5.496743
```

Cluster Demographics

Last we'll take a look at each cluster to see if any cluster has an unequal amount of gender, race, income, etc.

Education: Question 48 Response Distribution by Cluster

```
table(appHappyNums$q48,clusters$cluster)
```

```
##
##       1   2   3
##   1  13  31  18
##   2  77  94  56
##   3 136 157 199
##   4 182 162 241
##   5  29  38  37
##   6  43  67  63
```

Marital Status: Question 49 Response Distribution by Cluster

```
table(appHappyNums$q49,clusters$cluster)
```

```
##
##       1   2   3
##   1 183 220 270
##   2 203 204 195
##   3  69  70  90
##   4  25  55  59
```

Race: Question 54 Response Distribution by Cluster

```
table(appHappyNums$q54,clusters$cluster)
```

```
##
##       1   2   3
##   1 321 384 502
##   2  54  56  45
##   3  39  46  18
##   4   7  11   5
##   5   3  10   3
##   6  56  42  41
```

Hispanic: Question 55 Response Distribution by Cluster

```
table(appHappyNums$q55,clusters$cluster)
```

```
##
##       1   2   3
##   1 105  82  65
##   2 375 467 549
```

Household Income: Question 56 Response Distribution by Cluster

```
table(appHappyNums$q56,clusters$cluster)
```

```
##
##       1  2  3
##   1  26 38 27
##   2  16 27 12
##   3  19 21 17
##   4  43 37 36
##   5  48 56 63
##   6  54 46 67
##   7  51 49 61
##   8  28 43 57
##   9  48 48 51
##   10 30 31 41
##   11 19 32 31
##   12 49 38 51
##   13 16 18 35
##   14 33 65 65
```

Gender: Question 57 Response Distribution by Cluster

```
table(appHappyNums$q57,clusters$cluster)
```

```
##
##       1   2   3
##   1 224 270 264
##   2 256 279 350
```

Our Observations:

- The disproportions that jump out are cluster #2 has an unequal proportion of White respondants, with very few Asian respondents. Cluster #2 seems to be the least diverse cluster with the fewest Hispanic respondents despite being the largest cluster in terms of size.
- Cluster #1 has about half the respondants of people making $150K or more than the other two groups do.
- Cluster #2 appears to be heavily weighted female, while the other two clusters are more balanced.

Profiling

Looking at the cluster features helps us distinguish the different viewpoints of each cluster/group. Comparing how each cluster responded on allows us to profile each cluster to see their differences. Here are the profiles of each cluster:

- Cluster 1 - **Tech Savvy Group**, focus on important technological developments. They feel that there's too much technology or information today, focus on just the important technology. Best to market them then new and important technologies.

- Cluster 2 - **Tech Users Group**, doesn't keep up on important technological developments as much as cluster 1. They use technology in all forms and want all the technology that can be produced. Best to market older revamped technologies.
- Cluster 3 - **Not Tech Savvy Group**, this cluster feels less confident about using technology and it is not as big of a part of their life as cluster 2 and 3. Probably not the best group to market new technologies to, but rather the easily adoptable technologies.

Conclusion:

This cluster analysis allows us to target specific users for the new app. Specifically by coming up with a profile, we learn what is important to each cluster/group.

For the purposes of marketing this new app, **App Happy should likely focus its efforts on cluster #2**, as those people will buy in quickly and use it often. However, they will need to convince those people why they need to keep using their app instead of adopting other technologies.

As a secondary market strategy, they can also **target cluster #1** with a different focus on why this new app is different and ground breaking. Try convincing this group why this tech is different and they will buy in if the argument is sound enough.

Part 3: Probability of Using Only Free Apps

In order to estimate whether a respondent **only** uses free apps, we created a new binary dependent variable called `onlyFreeApps` that has a value of 1 if the value of variable q12 equals 6, and is 0 if q12 equals 1,2,3,4 or 5.

During EDA we randomly split the data into a model estimation training sample and a model test sample (80% training & 20% test).

We use the demographic variables in the survey data (q1, q48-q53, q55-q57) as predictor variables in both a Probit and Logit Regession Model. We omit variable q54 because attitude towards the use of Free Apps is subjective and it does not show any pattern with the race of the customer, therefore, q54 does not have significance in a model. There is also legal concerns with developing marketing strategies based off race.

Binary Logit Regression Model

Building a Binary Logit Regression Model to predict 'onlyFreeApps' value based on demographic data.

Predictor Selection

```
logitModel <- glm(onlyFreeApps ~ q1+q48+q49+q50r1+q50r2+q50r3+q50r4+q50r5+q55+q56+q57,
                  data=appHappyNumsTrain,
                  family=binomial(link=logit))
summary(logitModel)
```

```
##
## Call:
## glm(formula = onlyFreeApps ~ q1 + q48 + q49 + q50r1 + q50r2 +
##     q50r3 + q50r4 + q50r5 + q55 + q56 + q57, family = binomial(link = logit),
##     data = appHappyNumsTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1433  -0.7340  -0.5948  -0.4210   2.3552
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.666966   1.280699  -1.302 0.193051
## q1           0.009671   0.038348   0.252 0.800891
## q48         -0.149888   0.062831  -2.386 0.017053 *
## q49         -0.024142   0.075784  -0.319 0.750060
## q50r1        0.199868   0.276847   0.722 0.470329
## q50r2        0.308683   0.259025   1.192 0.233374
## q50r3        0.040216   0.221594   0.181 0.855986
## q50r4        0.043999   0.232159   0.190 0.849684
## q50r5        0.356495   0.273969   1.301 0.193182
## q55          0.601054   0.217988   2.757 0.005828 **
## q56         -0.090827   0.021526  -4.219 2.45e-05 ***
## q57         -0.485746   0.141700  -3.428 0.000608 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1369.2  on 1313  degrees of freedom
## Residual deviance: 1306.8  on 1302  degrees of freedom
## AIC: 1330.8
##
## Number of Fisher Scoring iterations: 4
```

```
logitModel <- glm(onlyFreeApps ~ q48+q55+q56+q57,
                  data=appHappyNumsTrain,
                  family=binomial(link=logit))
summary(logitModel)
```

```
##
## Call:
## glm(formula = onlyFreeApps ~ q48 + q55 + q56 + q57, family = binomial(link = logit),
##     data = appHappyNumsTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1648  -0.7354  -0.6017  -0.4152   2.4146
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.47404    0.49029  -0.967 0.333619
## q48         -0.15063    0.06180  -2.437 0.014796 *
## q55          0.59870    0.21553   2.778 0.005473 **
## q56         -0.08550    0.02001  -4.273 1.92e-05 ***
## q57         -0.51691    0.13776  -3.752 0.000175 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1369.2  on 1313  degrees of freedom
## Residual deviance: 1309.9  on 1309  degrees of freedom
## AIC: 1319.9
```

```
##
## Number of Fisher Scoring iterations: 4
```

Standard p-value threshold for statistical significance is .05, therefore, we will build our model using only predictors that meet that threshold: q48, q55, q56 & q57.

Additionally, the smaller the AIC value, the better the fit of a model and general consensus is that movement in AIC of 5+ points is meaningful. We are able to reduce the AIC for a model with all predictors from 1330.8 to 1319.9 in a model with only 4 predictors.

Predicting on the Training Data Set

For a binary logistic regression, we need to pay close attention to the cutoff value used against the resulting probability output. We leverage the `InformationValue` package to select the optimal cutoff value.

```
logitModelTrainActuals <- appHappyNumsTrain$onlyFreeApps
logitModelTrainPreds <- predict(logitModel, newdata=appHappyNumsTrain, type="response")
optCut <- optimalCutoff(logitModelTrainActuals, logitModelTrainPreds, optimiseFor = "misclasserror")
logitModelTrainPreds <- as.numeric(logitModelTrainPreds>=optCut)
cat('Optimal Cutoff Value:',optCut)
```

```
## Optimal Cutoff Value: 0.4625809
```

Predictive Accuracy of the Logit Model on the Training Data:

```
table(logitModelTrainPreds)
```

```
## logitModelTrainPreds
##    0    1
## 1312    2
```

```
cat('\n')
```

```
table(logitModelTrainActuals)
```

```
## logitModelTrainActuals
##    0    1
## 1031  283
```

```
cat('\nLogit Model Accuracy:',100*sum(logitModelTrainPreds==logitModelTrainActuals,na.rm=TRUE)/length(l
```

```
##
## Logit Model Accuracy: 78.46271
```

```
cat('\n')
```

```
confusionMatrix(logitModelTrainActuals,logitModelTrainPreds)
```

```
##      0   1
## 0 1030 282
## 1    1   1
```

```
cat('\nType I Error Rate:',confusionMatrix(logitModelTrainActuals,logitModelTrainPreds)[2,1]/length(log
```

```
##
## Type I Error Rate: 0.000761035
```

Predicting on the Test Data Set

```
logitModelTestActuals <- appHappyNumsTest$onlyFreeApps
logitModelTestPreds <- predict(logitModel, newdata=appHappyNumsTest, type="response")
logitModelTestPreds <- as.numeric(logitModelTestPreds>=optCut)
cat('Optimal Cutoff Value:',optCut)
```

```
## Optimal Cutoff Value: 0.4625809
```

Predictive Accuracy of the Logit Model on the Test Data:

```
table(logitModelTestPreds)
```

```
## logitModelTestPreds
##   0   1
## 326   3
```

```
cat('\n')
```

```
table(logitModelTestActuals)
```

```
## logitModelTestActuals
##   0   1
## 266  63
```

```
cat('\nLogit Model Accuracy:',100*sum(logitModelTestPreds==logitModelTestActuals,na.rm=TRUE)/length(logi
```

```
##
## Logit Model Accuracy: 80.54711
```

```
cat('\n')
```

```
confusionMatrix(logitModelTestActuals,logitModelTestPreds)
```

```
##     0  1
## 0 264 62
## 1   2  1
```

```
cat('\nType I Error Rate:',confusionMatrix(logitModelTestActuals,logitModelTestPreds)[2,1]/length(logitM
```

```
##
## Type I Error Rate: 0.006079027
```

Difference between predictive accuracy perecentage of both models

The predictive accuracy percentage for both the training and test data sets is 78.4%. The model appears to have the same predictive accuracy on any data whether its the data the model was trained on or out-of-sample data for the same survey. We also note that the Type I Error rate on both Training and Test data sets is at or near 0%.

Respondents only using free apps are taken as a dependent variable is a function of q48, q55, q56 & q57 as independent variables. This means the relation depends on:

1. (q48) Marital status
2. (q55) Hispanic Ethnicity
3. (q56) Household Income
4. (q57) Gender

Our Logit Model give us the following inferences:

- Married respondents are less likely to spend on apps
- Hispanics are less likely to spend on apps

- As household income increases, respondents are more likely to spend on apps
- Females are more likely to spend on apps

Important to note that this model was created simply to determine factors that drive spend on apps, not necessarily the quantity of spend (i.e. the industry term known as 'whales' aka high-spenders).

Part 4: Post Hoc Predictive Segmentation Analysis

```
appNum <- appHappyNums
dim(appNum)
```

```
## [1] 1643   89
```

Because we want to know how their attitudes predict whether they pay for apps or not, we select column: q24, q25, q26 and q12:

```
appNumS <- appNum %>% select(q24r1:q24r12,q25r1:q25r12,q26r3:q26r17,q26r18,q12,onlyFreeApps)
dim(appNumS)
```

```
## [1] 1643   42
```

Finite Mixture Binary Logit: Search for "best" clustering

Let's search for a solution that has from 1 to 6 clusters. Note how `onlyFreeApps` is expressed as the dependent variable in the model formula: `cbind(onlyFreeApps, 1 - onlyFreeApps)`.

```
binLogSearch=stepFlexmix(cbind(onlyFreeApps, 1 - onlyFreeApps) ~ 1, data = appNumS,
model = FLXMRglmfix(family = "binomial",
fixed=~q24r1 + q24r2 + q24r3 + q24r4 + q24r5 + q24r6 + q24r7 + q24r8 + q24r9 + q24r10 + q24r11 + q24r12
k = 1:6, nrep = 3)
```

```
## 1 : * * *
## 2 : * * *
## 3 : * * *
## 4 : * * *
## 5 : * * *
## 6 : * * *
```

Now we can see which cluster would best to fit data. As we can see below, the 2 clustering has highest loglikelihood, so we choose 2 clustering to make the segements.

```
binLogSearch
```

```
##
## Call:
## stepFlexmix(cbind(onlyFreeApps, 1 - onlyFreeApps) ~ 1, data = appNumS,
##     model = FLXMRglmfix(family = "binomial", fixed = ~q24r1 +
##         q24r2 + q24r3 + q24r4 + q24r5 + q24r6 + q24r7 + q24r8 +
##         q24r9 + q24r10 + q24r11 + q24r12 + q25r1 + q25r2 +
##         q25r3 + q25r4 + q25r5 + q25r6 + q25r7 + q25r8 + q25r9 +
##         q25r10 + q25r11 + q25r12 + q26r3 + q26r4 + q26r5 +
##         q26r6 + q26r7 + q26r8 + q26r9 + q26r10 + q26r11 +
##         q26r12 + q26r13 + q26r14 + q26r15 + q26r16 + q26r17 +
##         q26r18), k = 1:6, nrep = 3)
##
##   iter converged k k0    logLik      AIC      BIC      ICL
```

```
## 1    2      TRUE 1  1 -710.1563 1502.313 1723.888 1723.888
## 2    3      TRUE 2  2 -710.1561 1506.312 1738.696 3974.499
## 3    3      TRUE 3  3 -710.1539 1510.308 1753.500 5188.543
## 4    3      TRUE 4  4 -710.1529 1514.306 1768.307 6161.758
## 5    3      TRUE 5  5 -710.1523 1518.305 1783.114 6874.149
## 6    3      TRUE 6  6 -710.1534 1522.307 1797.925 7407.094
```

Now we focus on the two-components model's results:

```
binLogModk1=stepFlexmix(cbind(onlyFreeApps, 1 - onlyFreeApps) ~ 1, data = appNumS,
model = FLXMRglmfix(family = "binomial",
fixed=~q24r1 + q24r2 + q24r3 + q24r4 + q24r5 + q24r6 + q24r7 + q24r8 + q24r9 + q24r10 + q24r11 + q24r12
k = 2, nrep = 3)
```

```
## 2 : * * *
```

```
summary(refit(binLogModk1))
```

```
## Warning in sqrt(diag(z@vcov)[indices]): NaNs produced
```

```
## $Comp.1
##               Estimate Std. Error z value  Pr(>|z|)
## q24r1        0.1209789  0.0461296  2.6226 0.0087265 **
## q24r2        0.1170067  0.0819762  1.4273 0.1534862
## q24r3        0.0478619  0.0523574  0.9141 0.3606446
## q24r4       -0.0165242  0.0494359 -0.3343 0.7381870
## q24r5        0.0587553  0.0496648  1.1830 0.2367948
## q24r6       -0.0547130  0.0674421 -0.8113 0.4172177
## q24r7       -0.2577491  0.0686314 -3.7556 0.0001730 ***
## q24r8        0.0600414  0.0769734  0.7800 0.4353747
## q24r9       -0.0109521  0.0505073 -0.2168 0.8283318
## q24r10      -0.2373032  0.0728078 -3.2593 0.0011168 **
## q24r11      -0.0567235  0.0637706 -0.8895 0.3737381
## q24r12      -0.0784072  0.0811306 -0.9664 0.3338281
## q25r1       -0.0423049  0.0759985 -0.5567 0.5777636
## q25r2        0.0093520  0.0670513  0.1395 0.8890749
## q25r3        0.0620229  0.0735544  0.8432 0.3991027
## q25r4        0.0061927  0.0760862  0.0814 0.9351316
## q25r5       -0.0016744  0.0719835 -0.0233 0.9814425
## q25r6       -0.0519798  0.0497507 -1.0448 0.2961122
## q25r7        0.1162480  0.0745166  1.5600 0.1187530
## q25r8       -0.1193887  0.0627283 -1.9033 0.0570056 .
## q25r9       -0.0081125  0.0665630 -0.1219 0.9029963
## q25r10      -0.0050432  0.0649655 -0.0776 0.9381235
## q25r11       0.0059144  0.0674356  0.0877 0.9301121
## q25r12       0.0442825  0.0530648  0.8345 0.4039998
## q26r3        0.1803479  0.0603850  2.9866 0.0028207 **
## q26r18      -0.1615793  0.0463106 -3.4890 0.0004848 ***
## (Intercept) -0.2871927  0.6960904 -0.4126 0.6799147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $Comp.2
##               Estimate Std. Error z value  Pr(>|z|)
## q24r1        0.1209789  0.0461296  2.6226 0.0087265 **
## q24r2        0.1170067  0.0819762  1.4273 0.1534862
```

```
## q24r3        0.0478619  0.0523574  0.9141 0.3606446
## q24r4       -0.0165242  0.0494359 -0.3343 0.7381870
## q24r5        0.0587553  0.0496648  1.1830 0.2367948
## q24r6       -0.0547130  0.0674421 -0.8113 0.4172177
## q24r7       -0.2577491  0.0686314 -3.7556 0.0001730 ***
## q24r8        0.0600414  0.0769734  0.7800 0.4353747
## q24r9       -0.0109521  0.0505073 -0.2168 0.8283318
## q24r10      -0.2373032  0.0728078 -3.2593 0.0011168 **
## q24r11      -0.0567235  0.0637706 -0.8895 0.3737381
## q24r12      -0.0784072  0.0811306 -0.9664 0.3338281
## q25r1       -0.0423049  0.0759985 -0.5567 0.5777636
## q25r2        0.0093520  0.0670513  0.1395 0.8890749
## q25r3        0.0620229  0.0735544  0.8432 0.3991027
## q25r4        0.0061927  0.0760862  0.0814 0.9351316
## q25r5       -0.0016744  0.0719835 -0.0233 0.9814425
## q25r6       -0.0519798  0.0497507 -1.0448 0.2961122
## q25r7        0.1162480  0.0745166  1.5600 0.1187530
## q25r8       -0.1193887  0.0627283 -1.9033 0.0570056 .
## q25r9       -0.0081125  0.0665630 -0.1219 0.9029963
## q25r10      -0.0050432  0.0649655 -0.0776 0.9381235
## q25r11       0.0059144  0.0674356  0.0877 0.9301121
## q25r12       0.0442825  0.0530648  0.8345 0.4039998
## q26r3        0.1803479  0.0603850  2.9866 0.0028207 **
## q26r18      -0.1615793  0.0463106 -3.4890 0.0004848 ***
## (Intercept) -0.2105062  0.2981993 -0.7059 0.4802350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat('\nAIC:', AIC(binLogModk1))
```

```
##
## AIC: 1647.304
```

```
cat('\nlog likelihood:',logLik(binLogModk1))
```

```
##
## log likelihood: -794.6518
```

According to the results below, one of the segments has 50.01% prior probability and another segement has total 49.99% prior probability. They also have -726.3169 log likelihood with AIC 1718.634. One of the difference for both segements is the size, one segment has 1457 objects while another has only 186 objects. The ratio for first clustering close to 1 means that the first clustering more tight than the another one.

```
summary(binLogModk1)
```

```
##
## Call:
## stepFlexmix(cbind(onlyFreeApps, 1 - onlyFreeApps) ~ 1, data = appNumS,
##     model = FLXMRglmfix(family = "binomial", fixed = ~q24r1 +
##         q24r2 + q24r3 + q24r4 + q24r5 + q24r6 + q24r7 + q24r8 +
##         q24r9 + q24r10 + q24r11 + q24r12 + q25r1 + q25r2 +
##         q25r3 + q25r4 + q25r5 + q25r6 + q25r7 + q25r8 + q25r9 +
##         q25r10 + q25r11 + q25r12 + q26r3 + q26r18), k = 2,
##     nrep = 3)
##
##        prior size post>0 ratio
## Comp.1 0.506 1300   1643 0.791
```

```
## Comp.2 0.494  343   1643 0.209
##
## 'log Lik.' -794.6518 (df=29)
## AIC: 1647.304   BIC: 1804.028
```

Conclusions

In conclusion. The optimum number of segments the mclust model produces is 2. This is consistent with the flexmix model, so we presume that the accuracy is high. The likelihood and BIC are very low, so the likelihood that this model is close to the truth is high. Because of this we are unable to calculate any differences between segments since all observations are in the same segment.