Solo 1 Assignment: The App Happy Company and the Market for Social Entertainment Apps

Ji Jung

Introduction

App Happy hired Consumer Spy Corporation to undertake a general attitudinal post hoc segmentation analysis with survey data of 1,600 respondents. The segmentation analysis was to identify distinct groups that App Happy could utilize in targeting consumers based on segment and to develop marketing strategies.

Questions from the survey data relating to respondents' technological acceptance, personality characteristics and purchasing behavior were selected as basis variables for the segmentation analysis. These variables were used in applying hierarchical and non-hierarchical clustering methods with various distance metrics. Upon identifying relevant segments, data exploration techniques were then leveraged to identify any demographic or customer behavior characteristics which can distinguish the identified clusters. Finally, a classification model was recommended which should enable App Happy to classify future customers into segments, including those customers whom the company has no survey data.

Data Analysis

The survey data of 1,663 respondents conducted by the Consumer Spy Corporation. The data is stored within a R data file, which contains two R data frames. The first of these has numerically coded survey response data, while the second has response data coded in the character strings of the questionnaire's value labels.

The dictionary of the survey questions suggests it appears the original survey had 57 questions, while the survey subset used for this assessment included 16 questions. Questions in the subset are varied, including those related to respondent demographics, purchasing behavior, and online browsing habits.

Customer Segmentation

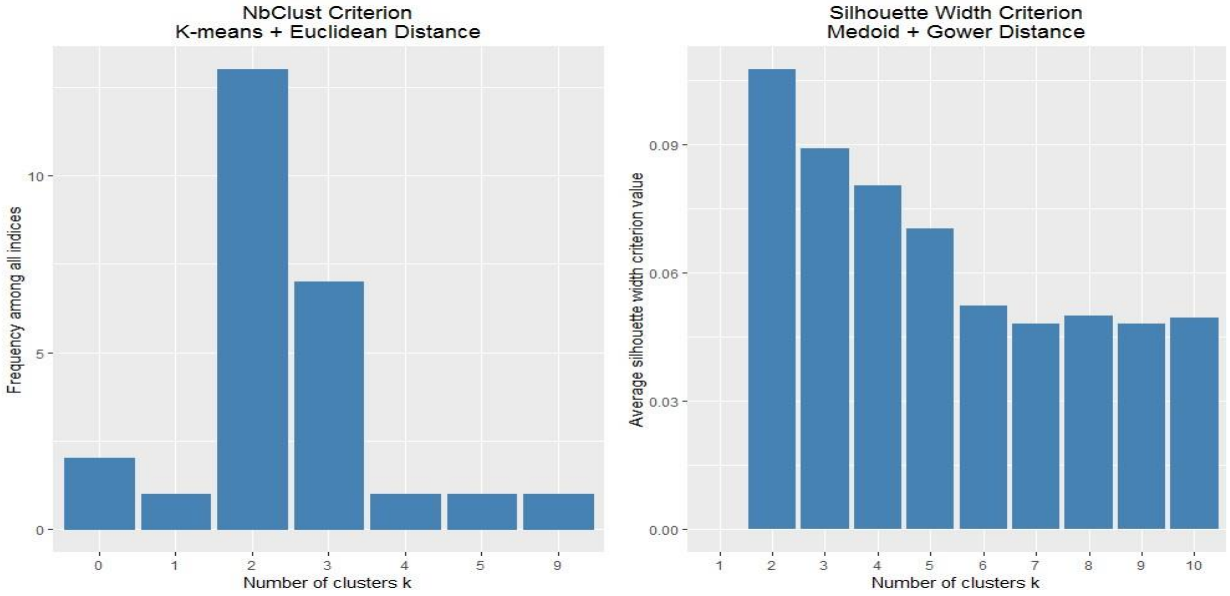
Clustering Method

Clustering algorithms can be separated into two main classes, non-hierarchical and hierarchical. For nonhierarchical clustering, the desired number of clusters is specified in advance to set the number of cluster centers. Each data point is then assigned to its nearest cluster center by minimizing or maximizing a desired criterion, with cluster centroids iteratively recalculated until they remain stable. Under this method, the relationship between clusters remains largely undetermined. The analyst gains a representation of data clusters based on only their predetermined number of centers. In hierarchical clustering, the number of clusters need not be specified in advance. Instead, pairs of clusters are repetitively linked until every data point is included in a hierarchy of cluster relationships. The analyst then has the freedom to assess the full array of cluster options and their hierarchy of relationships using a dendrogram.

For this assessment, both non-hierarchical and hierarchical methods to segment are utilized to the set of attitudinal variables. For non-hierarchical clustering, two R functions are used. The first is the k-means function as part of the stats R package, which is used to perform k-means clustering. The second is the pam function as part of the cluster R package to perform partitioning of k-clusters around medoids. This is a robust form of k-means, particularly to outliers, for it minimizes the sum of dissimilarities instead of a sum of squared distance. Finally, for hierarchical clustering, hcut function is utilized as part of the factoextra R package. ward.D agglomeration technique is then used in the same package to generate dendrograms to aid in deriving an appropriate number of clusters.

Non-hierarchical Clustering

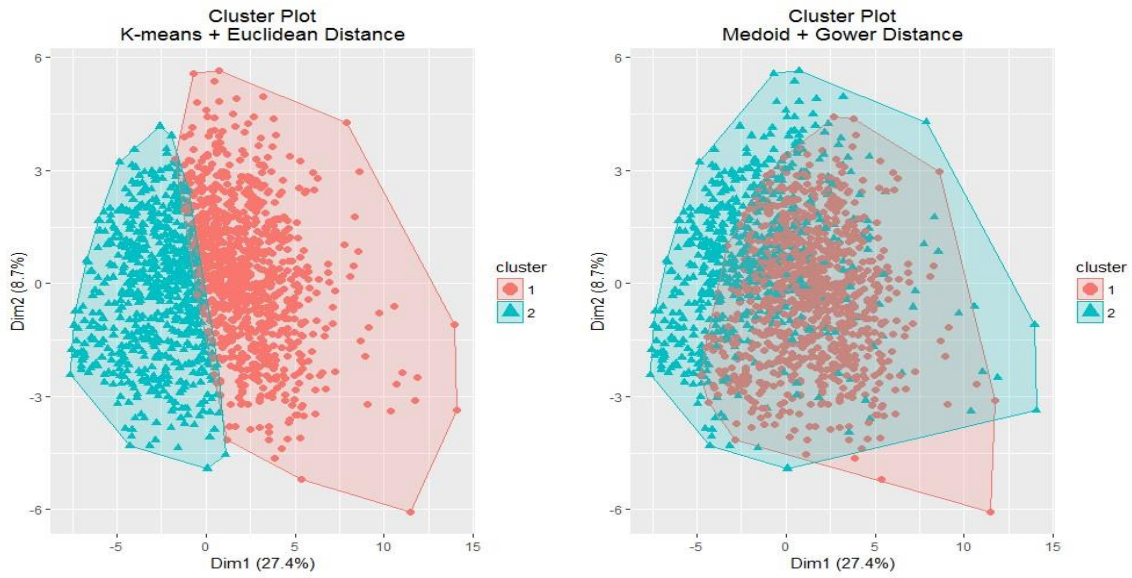
Non-hierarchical clustering methods require the number of clusters be nominated prior to implementation of the clustering routine. There are several criteria available to aid the analyst in determining an appropriate number of clusters prior to employing the clustering technique. The most robust of these can be found in the NbClust. This package provides the ability to employ up to 30 separate cluster criterion measures as a form of index. NbClust function is then applied to the k-means clustering technique using the Euclidean distance metric and show index frequencies for a range of clusters below. NbClust package is unable to accommodate the medoid clustering technique. Instead pamk R package is used to derive a silhouette width criterion to assess the optimal number of clusters under this technique.



Non-hierarchical Cluster Criterion

The use of two clusters is reported to be optimal for both clustering methods, while the use of three clusters is reported to be the next best option.

With the number of clusters pre-determined based on these criteria, each non-hierarchical clustering method can be pursued. Scatter plots highlight the non-hierarchical cluster assignments are based on a two-dimensional reduction of the original set of attitudinal survey responses.

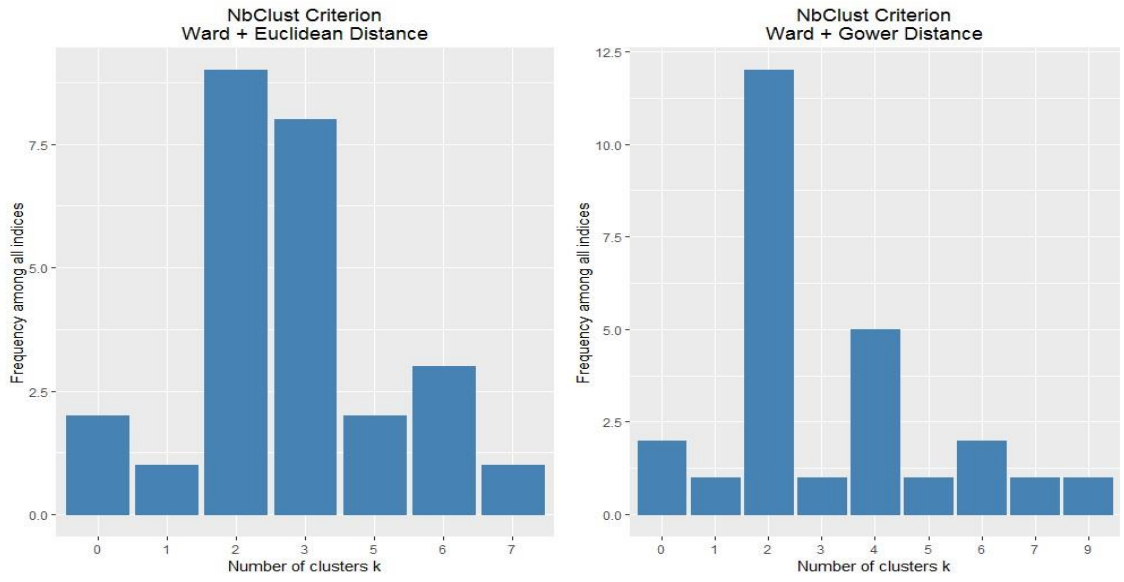


Non-hierarchical Cluster Plots

The cluster boundaries are determined by connecting the most extreme data points for each cluster. The medoid cluster technique is shown to have a much greater overlap of clusters than the k-means technique. Clearly, the overlap of cluster points and therefore the inability to distinguish clusters be a disadvantage for the Medoid cluster technique.

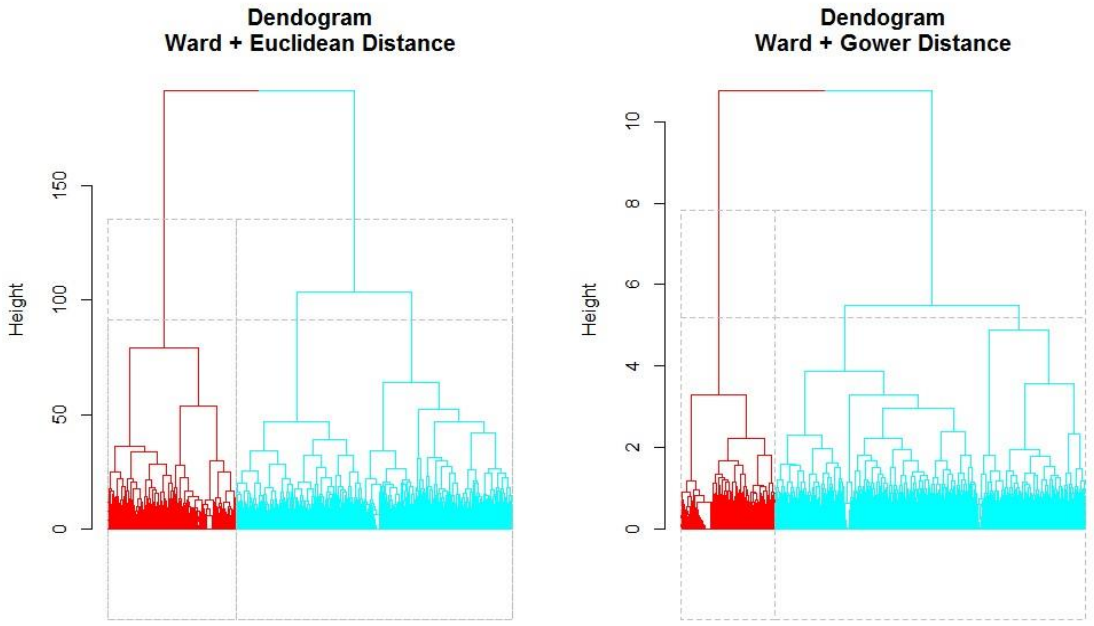
Hierarchical Clustering

For the hierarchical clustering method, a similar process is done to first determine an appropriate number of clusters. A pre-determined number of clusters will aid in interpretation of these plots. NbClust function is to the ward clustering technique using both the Euclidean and Gower distance metric.



Hierarchical Cluster Criterion

Usage of three clusters is reported to be optimal when using the Euclidean distance metric. When using the Gower distance metric, the use of two clusters is reported to be optimal. Hierarchical clustering methods are also able to provide a dendrogram to help assist in determining the number of clusters. Dendrograms for both distance metrics are shown below, colored according to the use of their optimal number of clusters as reported by the NbClust criterion.



Hierarchical Cluster Dendrogram

Determining the optimum cut using a dendrogram is a subjective process. Although there are some rules of thumb available (i.e. clustering from the top-down/bottom-up to include a minimum amount of leaves, or excluding any early made branches with a minimum amount of leaves), the practice is still likely to result in widely varying results between analysts. For the two dendrograms shown above, three clusters being appropriate under both distance metrics. The amount of data points included within each cluster. That is, making a cut after the second Clade when using either distance metric, results in data points being attributed evenly between three clusters. Note that while the NbClust criterion reported three clusters to be optimum when using the Euclidean distance metric, however, four clusters are reported to be inferior when using the Gower distance metric.

Scatter plots which highlight the hierarchical cluster assignments based on a two-dimensional reduction of the original set of attitudinal survey responses.

Hierarchical Cluster Plots

Both are shown to have a large amount of overlap between clusters. Assigning three clusters according to the Euclidean distance metric makes it difficult to distinguish the second cluster from the remaining two clusters. Assigning three clusters according to the Gower distance metric makes it difficult to distinguish the first and third cluster.

Statistical Confirmation

As a final evaluation of the performance of each clustering method, a generalized linear model is applied to each question based on each clustering method, and measured the models Akaike Information Criterion for each. The lower the AIC value, the more superior that clustering method is in predicting each cluster.

An assessment of the AIC value for each question reveals that each clustering method tends to demonstrate low AIC values for questions two and four, as well as high AIC values for questions one, and 56. This is interesting as questions two and four are focused on the purchase behavior of consumer segments, while questions one and 56 are focused on characterizing the demographic features of each consumer segment.

The average AIC value is noted for each clustering method to understand if a particular method is shown to have superior predictive performance. A summary of the results for each is shown in the table below.

	1	2	3	4
Method	Non-Hierarchical	Non-Hierarchical	Hierarchical	Hierarchical
Technique	k-means	Medoid	Ward	Ward
Distance Metric	Euclidean	Gower	Eucildean	Gower
Number of Clusters	2	2	3	3
Mean AIC	4025.706	4131.336	4015.816	4063.146

Segment Profiling

Visual methods are used to evaluate both the demographic and consumer behavior characteristics of the two clusters determined by the selected clustering method. A set of bar plots of the total response count and proportion of responses over each available option are created for each question.

Demographic Characteristics

Several similarities existed over demographic characteristics for both clusters. A similar proportion of respondents from both clusters were either currently married, had no children, or were as likely to be male or female. There were also similarities over the total number of respondents for each cluster. There were significantly more respondents who were younger than 35, college graduates, white, married, had no children, and had a household annual income of $30,000 to $70,000.

Fortunately, clusters were also able to be distinguished by many demographic characteristics. The first cluster for example, was found to contain a greater number of younger respondents, who were more educated, were more often white or Caucasian, or had higher incomes. This cluster was more likely to have been previously married but now separated, or to have older children than those respondents from cluster two who also had children.

Consumer Preferences

There are many similarities over consumer preferences for both clusters. Most respondents owned an iPhone device, used gaming or social networking applications, had more than 10 applications on their chosen device, or acquired at least half of their device applications freely.

In terms of the differences in consumer preferences between clusters, respondents from the first cluster had a greater preference for iPhone and Android devices and less of a preference for iPods and tablet devices. In addition, the first cluster had less of a preference for entertainment or T.V. applications, but a greater bias towards gaming, music and social applications. Finally, respondents from the first cluster were also found to have less applications, more free applications and less likely to visit the websites designated by the survey.

Initial Recommendation

Results from segment profiling indicate that the consumers within the first cluster tend to have less of a preference for entertainment and T.V. applications whilst more of a preference for social media applications. Since App Happy aim to produce a social entertainment application, the results suggest that the social media aspect of the product may have greater appeal to the first cluster, whilst the entertainment aspect of the product may have greater appeal to the second cluster.

The results indicated that consumers within the first cluster prefer using iPhone and Android devices. If App Happy were to target a product at consumers designated by the first cluster, that product would likely have greater adoption if it were made available on these devices. If App Happy were to target a product at consumers designated by the second cluster, that product would likely have greater adoption if it were made available on these devices and advertised on websites designated by the survey.

Predictive Classifier

App Happy has requested that the segmentation results be used to inform a classification model, which can ultimately be applied to future survey data to generate meaningful customer segmentations. Such

classification models can derive predictors from observations that are known, and apply those derived predictors to new observations.

There are many classification models which App Happy could employ to predict customer membership, with each model generally being categorized according to their underlying algorithm. Three possible algorithms include those based on a Decision Tree, Support Vector Machine, or K-Nearest Neighbors methodology. A summary of the strengths and weaknesses of each are shown below.

Decision Tree

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Strengths: Fast training and testing phases. Low memory usage. Simple to understand and interpret. Implicitly performs variable screening or feature selection.

Weaknesses: The tendency to overfit data.

Support Vector Machine

Support Vector Machines are a set of supervised learning methods used for classification, regression and outlier detection. They use a linear hyperplane to separate data point, but can also be used as a non-linear classified with kernels.

Strengths: Low memory usage, as it only needs to store a subset of the data to make predictions. SVM is effective in high dimensional spaces. Versatile, as the kernel allows expert knowledge of the problem to be built into the classifier.

Weaknesses: Slow training and testing phases.

K-Nearest Neighbors

According to scikit learn, supervised neighbors-based learning methods come in two flavors: classification for data with discrete labels, and regression for data with continuous. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these.

Strengths: Fast training phase. Versatile, as it does not make any assumption about the underlying data distribution.

Weaknesses: Slow testing phase. High memory usage, as it requires storing all training data.

Initial Recommendation

It is difficult to recommend a classification model without directly assessing the performance of each method. However, it is fair to say that if the model is intended to be applied to large datasets where CPU/memory constraints may be of issue, a DT or SVM based classifier may be superior. On the other hand, if App Happy intends on capturing data which exhibits varying types, scale and distributions, then a KNeighbors based classifier may be superior.

Conclusion

For this assessment, various combinations of clustering techniques and distance metrics are used to segment survey data according to three basis variables. Combinations involved the use of hierarchical and non-hierarchical clustering methods under k-means, medoid and ward techniques, as well as the use of Euclidean and Gower distance metrics. While there were some noted concerns assuming the data to be of continuous type, results showed the k-means clustering technique with Euclidean distance metric to be the most favorable. This was largely due to this methods ability to distinguish between clusters.

Respondents from both clusters were found to share many demographic and behavioral characteristics, many unique biases were also identified. Clusters could be distinguished according to respondent age, ethnicity, education and incomes. With respect to consumer attitudes, clusters could be distinguished according to their preferences for type of electronic device, use of applications and propensity to purchase rather than freely download applications. Most notably, the first cluster was found to have less of a preference for entertainment and T.V. applications, more of a preference for social media applications, a greater preference for using iPhone and Android devices, and less of a tendency to browse those websites designated by the survey

There are many classification models which App Happy could employ to predict customer membership, with each model generally being categorized according to their underlying algorithm. Three possible algorithms include those based on a Decision Tree, Support Vector Machine, or K-Nearest Neighbors methodology. This assessment presented a summary of advantages and disadvantages associated with each, a recommendation of classification model would benefit from a direct assessment using the supplied data.

Appendix for Codes

```
for(package in c('cluster', 'factoextra', 'fpc', 'NbClust',
                 'reshape', 'plyr',
                 'ggplot2', 'scales', 'grid', 'gridExtra')) {
  if(!require(package, character.only=TRUE)) {
    install.packages(package)
    library(package, character.only=TRUE)
  }
}

rm(package)

#############################################################################
# Preliminaries
#############################################################################

#---------------------------------------------------------------------
# Data prep
#---------------------------------------------------------------------

# Load the dataset
rm(list=ls())
load('apphappyData.RData')
df_appnum.raw <- apphappy.3.num.frame
df_applab.raw <- apphappy.3.labs.frame

# Subset the data for attitudinal variables
colstart <- which(colnames(df_appnum.raw) == 'q24r1')
colend <- which(colnames(df_appnum.raw) == 'q26r17')
df_appnum.att = df_appnum.raw[,c(colstart:colend)]
df_applab.att = df_applab.raw[,c(colstart:colend)]
```

```
# Subset the data for nonattitudinal variables
df_appnum.nonatt = df_appnum.raw[,c(-colstart:-colend)]
df_applab.nonatt = df_applab.raw[,c(-colstart:-colend)]

# Count NA rows for attitudinal variables
inclna <- nrow(df_appnum.att)
exclna <- nrow(na.omit(df_appnum.att))
print(paste0("Basis observations: ", inclna - exclna))

# Count NA rows for non-attitudinal variables
inclna <- nrow(df_appnum.nonatt)
exclna <- nrow(na.omit(df_appnum.nonatt))
print(paste0("Non-basis observations: ", inclna - exclna))

# Count NA rows for original dataset
inclna <- nrow(df_appnum.raw)
exclna <- nrow(na.omit(df_appnum.raw))
print(paste0("Dataset observations: ", inclna - exclna))

rm(apphappy.4.num.frame, apphappy.4.labs.frame, colstart, colend)



##############################################################################
# Non-hierarchical Clustering
##############################################################################

#-----------------------------------------------------------------------
# Determine number of clusters
#-----------------------------------------------------------------------

# Method: kmeans, Distance: euclidean
nbc.eu <- NbClust(df_appnum.att,
```

```
            min.nc=2, max.nc=10, distance='euclidean',
            method='kmeans', index='all')
table(nc$Best.n[1,])

plot1 <- fviz_nbclust(nbc.eu) +
 ggtitle('NbClust Criterion\nK-means + Euclidean Distance')

rm(nbc.eu)



# Method: medoid, Distance: euclidean
pamk.eu <- pamk(df_appnum.att, diss=FALSE, krange=1:10,
          criterion='asw', critout=TRUE)

df_pamk <- melt(pamk.eu$crit)
df_pamk$index <- rownames(df_pamk)

plot2 <- ggplot(df_pamk, aes(x=index, y=value)) +
  geom_bar(stat='identity', fill='steelblue') +
  scale_x_discrete(limits=c(1:10)) +
 ggtitle('Silhouette Width Criterion\n') +
 labs(x='Number of clusters k', y='Average silhouette width criterion value')

rm(pamk.eu, df_pamk)



# Method: medoid, Distance: gower
dist.daisy <- daisy(df_applab.att, metric='gower', stand=FALSE)
pamk.gow <- pamk(dist.daisy, diss=TRUE, krange=1:10,
            criterion='asw', critout=TRUE)

df_pamk <- melt(pamk.gow$crit)
```

```
df_pamk$index <- rownames(df_pamk)

plot3 <- ggplot(df_pamk, aes(x=index, y=value)) +
 geom_bar(stat='identity', fill='steelblue') +
 scale_x_discrete(limits=c(1:10)) +
 ggtitle('Silhouette Width Criterion\nMedoid + Gower Distance') +
 labs(x='Number of clusters k', y='Average silhouette width criterion value')

rm(dist.daisy, pamk.gow, df_pamk)

rm(plot1, plot2, plot3)

#---------------------------------------------------------------------------
# Clustering analysis
#---------------------------------------------------------------------------

# Method: kmeans, Distance: euclidean
k = 2 # Set cluster center count according to no. clust criterion

set.seed(1)
res.km <- kmeans(df_appnum.att, centers=k)

plot1 <- fviz_cluster(res.km, df_appnum.att,
                show.clust.cent=TRUE, geom='point',
                title='Cluster Plot\nK-means + Euclidean Distance')


# Method: medoid, Distance: euclidean
k = 2 # Set cluster center count according to no. clust criterion

set.seed(1)
res.pam1 <- pam(df_appnum.att, k=k, diss=FALSE,
```

```
            metric='euclidean',
            stand=FALSE, cluster.only=FALSE)

plot2 <- fviz_cluster(res.pam1,
                show.clust.cent=TRUE, geom='point',
                title='Cluster Plot\nMedoid + Euclidean Distance')



# Method: medoid, Distance: gower
k = 2 # Set cluster center count according to no. clust criterion

set.seed(1)
dist.daisy <- daisy(df_applab.att, metric='gower', stand=FALSE)
res.pam2 <- pam(dist.daisy, k=k, diss=TRUE,
             metric='euclidean',
             stand=FALSE, cluster.only=FALSE)

plot3 <- fviz_cluster(list(data=df_appnum.att,
                     cluster=res.pam2$cluster),
                 show.clust.cent=TRUE, geom='point',
                 title='Cluster Plot\nMedoid + Gower Distance')

rm(dist.daisy)
rm(plot1, plot2, plot3)
rm(k)



################################################################################
# Hierarchical Clustering
################################################################################

#-------------------------------------------------------------------
```

```
# Determine number of clusters
#-------------------------------------------------------------------------

# Method: ward.D, Distance: euclidean
nbc.eu <- NbClust(df_appnum.att, diss=NULL,
                  min.nc=2, max.nc=10, distance='euclidean',
                  method='ward.D2', index='all')
table(nbc.eu$Best.n[1,])

plot1 <- fviz_nbclust(nbc.eu) +
  ggtitle('NbClust Criterion\nWard + Euclidean Distance')


# Method: ward.D, Distance: manhattan
nbc.man <- NbClust(df_appnum.att, diss=NULL,
                   min.nc=2, max.nc=10, distance='manhattan',
                   method='ward.D2', index='all')
table(nbc.man$Best.n[1,])

plot2 <- fviz_nbclust(nbc.eu) +
 ggtitle('NbClust Criterion\nWard + Manhattan Distance')


# Method: ward.D, Distance: gower
#dist.daisy <- daisy(df_applab.att, metric='gower', stand=FALSE)
nbc.gow <- NbClust(df_appnum.att, diss=dist.daisy,
                   min.nc=2, max.nc=10, distance=NULL,
                  method='ward.D2', index='all')
table(nbc.gow$Best.n[1,])

plot3 <- fviz_nbclust(nbc.gow) +
  ggtitle('NbClust Criterion\nWard + Gower Distance')
```

```
rm(plot1, plot2, plot3)

#---------------------------------------------------------------------------
# Clustering analysis
#---------------------------------------------------------------------------

# Method: ward.D, Distance: euclidean
k = 3 # Color dendogram according to no. clust criterion

set.seed(1)
res.hcut1 <- hcut(df_appnum.att, k=k, isdiss=FALSE,
                  hc_func='hclust', hc_metric='euclidean',
                  hc_method='ward.D2', stand=FALSE)

plot1a <- fviz_silhouette(res.hcut1) +
 scale_y_continuous(limits = c(-0.25, 0.5)) +
 ggtitle('Silhouette Plot\nWard + Euclidean Distance') +
 theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())

plot1b <- fviz_cluster(res.hcut1, df_appnum.att,
                       show.clust.cent=TRUE, geom='point',
                       title='Cluster Plot\nWard + Euclidean Distance')


# Method: ward.D, Distance: manhattan
k = 3 # Color dendogram according to no. clust criterion

set.seed(1)
res.hcut2 <- hcut(df_appnum.att, k=k, isdiss=FALSE,
                  hc_func='hclust', hc_metric='manhattan',
                  hc_method='ward.D2', stand=FALSE)
```

```
plot2a <- fviz_silhouette(res.hcut2) +
 scale_y_continuous(limits = c(-0.25, 0.5)) +
  ggtitle('Silhouette Plot\nWard + Manhattan Distance') +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())

plot2b <- fviz_cluster(res.hcut2, df_appnum.att,
                  show.clust.cent=TRUE, geom='point',
                  title='Cluster Plot\nWard + Manhattan Distance')

# Method: ward.D, Distance: gower
k = 2 # Color dendogram according to no. clust criterion

set.seed(1)
dist.daisy <- daisy(df_applab.att, metric='gower', stand=FALSE)
res.hcut3 <- hcut(dist.daisy, k=k, isdiss=TRUE,
                hc_func='hclust', hc_metric='euclidean',
                hc_method='ward.D2', stand=FALSE)

plot3a <- fviz_silhouette(res.hcut3) +
  scale_y_continuous(limits = c(-0.25, 0.5)) +
  ggtitle('Silhouette Plot\nWard + Gower Distance') +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())

plot3b <- fviz_cluster(res.hcut3, df_appnum.att,
                    show.clust.cent=TRUE, geom='point',
                    title='Cluster Plot\nWard + Gower Distance')

rm(dist.daisy)

# Method: ward.D, Distance: gower
k = 3 # Re-run for four clusters based on interpretation of dendogram
```

```
set.seed(1)
dist.daisy <- daisy(df_applab.att, metric='gower', stand=FALSE)
res.hcut3 <- hcut(dist.daisy, k=k, isdiss=TRUE,
                  hc_func='hclust', hc_metric='euclidean',
                  hc_method='ward.D2', stand=FALSE)

plot3a <- fviz_silhouette(res.hcut3) +
 scale_y_continuous(limits = c(-0.25, 0.5)) +
 ggtitle('Silhouette Plot\nWard + Gower Distance') +
 theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())

plot3b <- fviz_cluster(res.hcut3, df_appnum.att,
                       show.clust.cent=TRUE, geom='point',
                       title='Cluster Plot\nWard + Gower Distance')

rm(dist.daisy)

png(filename='images/hier_sil.png',
    width = 1000, height = 600, res = 100)

grid.arrange(plot1a, plot3a, ncol=2)
grid.arrange(plot1a, plot2a, plot3a, ncol=3)

rm(plot1b, plot2b, plot3b)
rm(k)

################################################################################
# Statistical Confirmation
################################################################################

ls_nm <- list('res.km', 'res.pam2', 'res.hcut1', 'res.hcut3')
```

```
ls_df <- list(res.km, res.pam2, res.hcut1, res.hcut3)
apply.glm.f <- function(y,class){
 return(glm(y~class)$aic)
}
df_temp <- data.frame(names = rownames(t(df_appnum.raw)))
for (i in 1:length(ls_nm)){
 df_appnum.raw$cluster <- ls_df[[i]]$cluster
 df_aic <- data.frame(apply(df_appnum.raw, 2,
                    apply.glm.f, class=df_appnum.raw$cluster))
 names(df_aic)[1] <- ls_nm[[i]]
 df_aic$names <- rownames(df_aic)
 df_temp <- merge(x=df_temp, y=df_aic, by.x='names', all=TRUE)
}
rownames(df_temp) <- df_temp[, 1]
df_temp <- df_temp[-c(1:2),-1]
df_aic <- df_temp[is.finite(rowSums(df_temp)), ]
colnames(df_aic) <- c('k-means + Euclidean', 'Medoid + Gower',
                'Ward + Eucildean', 'Ward + Gower')
write.table(round(df_aic, 3), 'aic.csv')
apply(df_aic, 2, mean)
sim.chisq.pval.f <- function(x,class){
 return(chisq.test(x,class,
              rescale.p=TRUE,
              simulate.p.value=TRUE)$p.value)
}

df_temp <- data.frame(names = rownames(t(df_appnum.raw)))
for (i in 1:length(ls_nm)){
 df_appnum.raw$cluster <- ls_df[[i]]$cluster
 df_chi <- data.frame(apply(df_appnum.att, 2,
                    sim.chisq.pval.f, class=df_appnum.raw$cluster))
 names(df_chi)[1] <- ls_nm[[i]]
```

```
  df_chi$names <- rownames(df_chi)
  df_temp <- merge(x=df_temp, y=df_chi, by.x='names', all=TRUE)
}
rownames(df_temp) <- df_temp[, 1]
df_temp <- df_temp[-c(1:2),-1]
df_chi <- df_temp[is.finite(rowSums(df_temp)), ]
colnames(df_chi) <- c('k-means + Euclidean', 'Medoid + Gower',
                      'Ward + Eucildean', 'Ward + Gower')
apply(df_chi, 2, mean)
rm(ls_nm, ls_df, df_temp)
```