STA 380 Homework 1

Francisco Sananez

August 6, 2015

Exploratory Analysis

To begin the "georgia2000.csv" data was loaded and saved as Georgia_Counties. Thanks to functions like *head()* and *summary()* one can get a feel of how the data is portrayed and especially see some useful information about each column.

```
Georgia_Counties <- read.csv("georgia2000.csv")
head(Georgia_Counties)
```

```
##     county ballots votes   equip poor urban atlanta perAA gore bush
## 1  APPLING    6617  6099   LEVER    1     0       0 0.182 2093 3940
## 2 ATKINSON    2149  2071   LEVER    1     0       0 0.230  821 1228
## 3    BACON    3347  2995   LEVER    1     0       0 0.131  956 2010
## 4    BAKER    1607  1519 OPTICAL    1     0       0 0.476  893  615
## 5  BALDWIN   12785 12126   LEVER    0     0       0 0.359 5893 6041
## 6    BANKS    4773  4533   LEVER    0     0       0 0.024 1220 3202
```

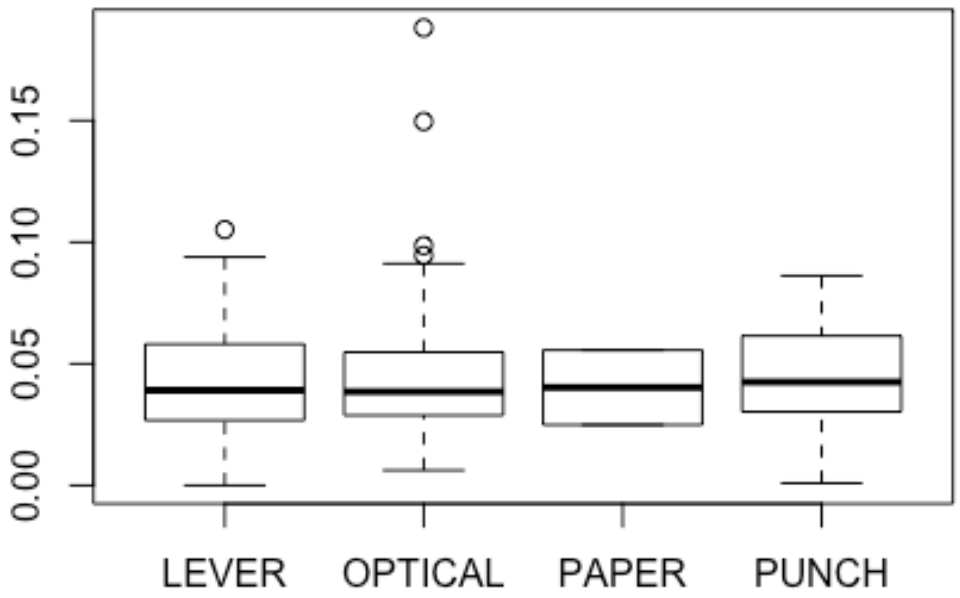
```
summary(Georgia_Counties)
```

```
##       county       ballots           votes            equip
##  APPLING :  1   Min.   :   881   Min.   :   832   LEVER  :74
##  ATKINSON:  1   1st Qu.:  3694   1st Qu.:  3506   OPTICAL:66
##  BACON   :  1   Median :  6712   Median :  6299   PAPER  : 2
##  BAKER   :  1   Mean   : 16926   Mean   : 16331   PUNCH  :17
##  BALDWIN :  1   3rd Qu.: 12251   3rd Qu.: 11846
##  BANKS   :  1   Max.   :280975   Max.   :263211
##  (Other) :153
##       poor            urban           atlanta            perAA
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.1115
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.2330
##  Mean   :0.4528   Mean   :0.2642   Mean   :0.09434   Mean   :0.2430
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.3480
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :0.7650
##
##       gore             bush
##  Min.   :  249   Min.   :  271
##  1st Qu.: 1386   1st Qu.: 1804
##  Median : 2326   Median : 3597
##  Mean   : 7020   Mean   : 8929
##  3rd Qu.: 4430   3rd Qu.: 7468
```

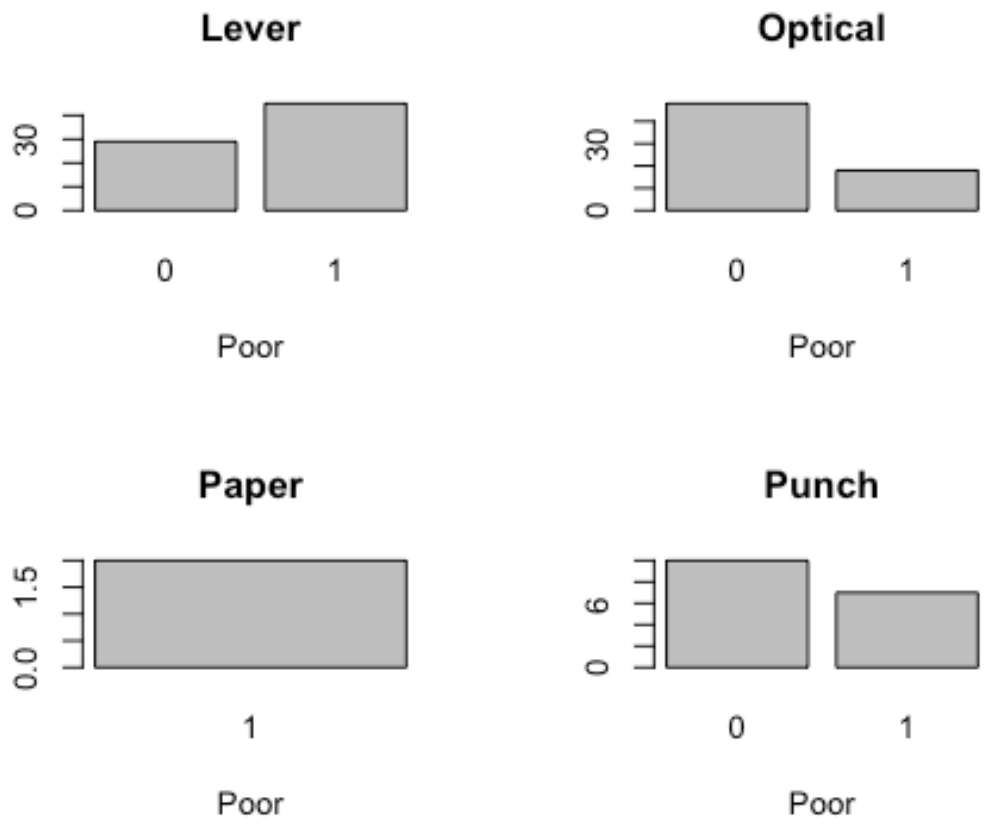
```
##  Max.   :154509   Max.   :140494
##
```

For this exercise we are interested in specific columns of this data set, it's the case of "ballots", "votes", "equip", "poor" and "perAA". Essentially we would like to know if there is any relation between "undercounts ratio" ((Ballots-Votes)/ballots) and the equipment used for the voting procedure. First we must calculate such undercounts and plot them against different equipment to see if there is any discrepancies between them.

```
Georgia_Counties$Undercount = (Georgia_Counties$ballots -
Georgia_Counties$votes) / Georgia_Counties$ballots
```



We can see with the plot given that the medians throughout all equipments are very close to each other, indicating no huge discrepancy between them. Although it's essential to point out some outliers counties using lever and optical equipment. Now, even though, there are no discrepancies between equipments one might think that counties with higher poor index or minorities are being affected over other counties. In order to display if this is the case, four bar plots were made (one for each equipment) showing the usage of such equipment by number of counties splitted by their poor index.



If we compare this plots to the box plot before we see that minorities are not the only ones being affected by undercount. We might say that undercounts are "higher" (outliers) with optical and lever because of the "higher" complexity it involves against paper or punch, equipment voters have used for longer time.

Bootstrapping

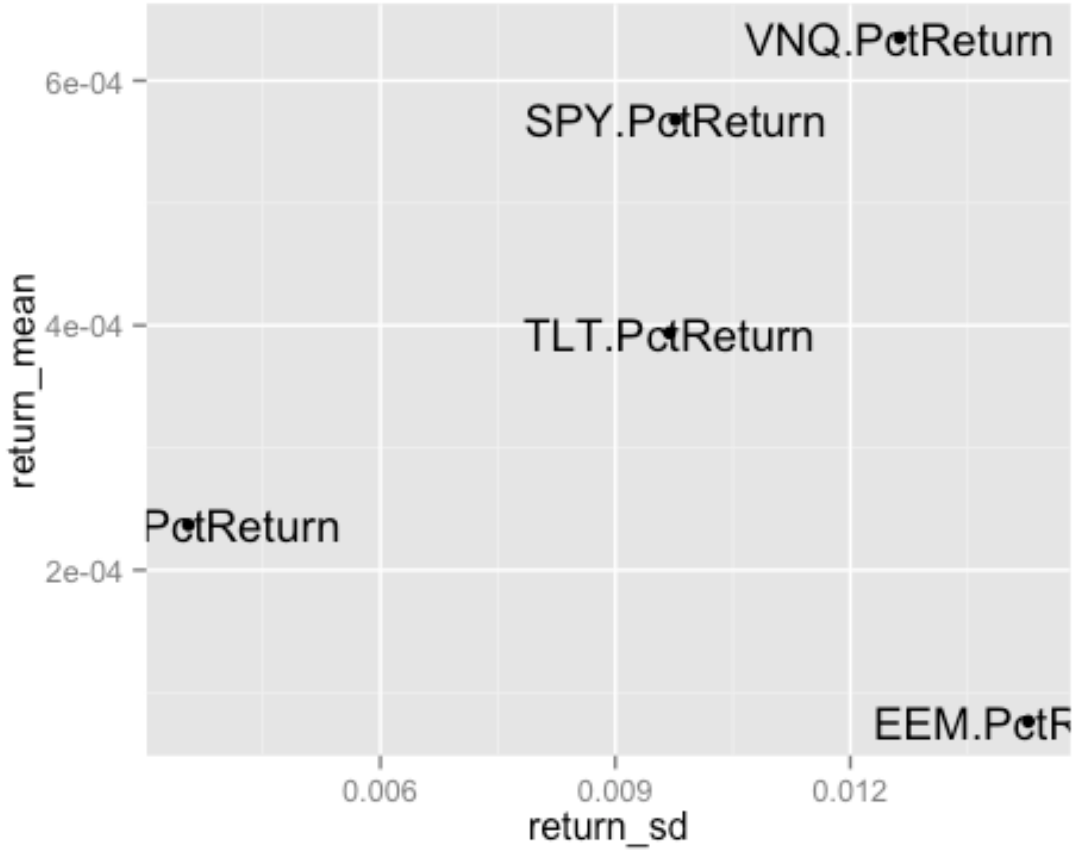
Now for this second exercise, we download the information of five ETFs in order to predict future returns by bootstrapping. After we load each ETFs' data we use a helper function, provided by prof. Scott, that provides the returns of each ETFs for each each day in our time interval.

```
stocks = c("SPY","TLT","LQD","EEM","VNQ")
prices = yahooSeries(stocks, from='2010-01-01', to='2015-08-02')
investment = 100000

YahooPricesToReturns = function(series)
{
  mycols = grep('Adj.Close', colnames(series))
  closingprice = series[,mycols]
  N = nrow(closingprice)
```

```
  percentreturn = as.data.frame(closingprice[2:N,]) /
as.data.frame(closingprice[1:(N-1),]) - 1
  mynames = strsplit(colnames(percentreturn), '.', fixed=TRUE)
  mynames = lapply(mynames, function(x) return(paste0(x[1],
".PctReturn")))
  colnames(percentreturn) = mynames
  as.matrix(na.omit(percentreturn))
}

returns = YahooPricesToReturns(prices)
```

With the returns we can now calculate to a degree of accuracy the risk-return trade of for each ETFs, they will come in handy when defining our weighted portfolios.

```
return_mean = apply(returns, 2, mean)
return_sd = apply(returns, 2, sd)
risk_return = round((return_mean/ return_sd)*100,digits=2)
```



We have plotted the risk-return relation so it's easier to appreciate and study the differences between all five ETFs. With that in mind we set out to create our three portfolios to study. Our first is a simple even weighted portfolio with all five ETFs in 20% of the wealth. For our second portfolio, the aggressive one, we have decided, based on how volatile equities are compared to fund and by looking at our previous

plot, that SPY should get around 45% of our distributed wealth, TLT 10%, LQD 5%, EEM 25% and VNQ 15%. On the contrary, for our safe portfolio we have decided the following weights, SPY 20%, TLT 25%, LQD 50%, EEM 0% and VNQ 5%. We proceed by running bootstrap for each portfolio with their respected weights.

```
even = c(0.2,0.2,0.2,0.2,0.2)
aggre = c(0.45,0.10,0.05,0.25,0.15)
safe = c(0.25,0.25,0.50,0.0,0.05)

# Even Portfolio Bootstrap Simulation
set.seed(101)
even_bootstrap = foreach(i = 1:5000, .combine = 'rbind') %do%
{
  holdings = even*investment
  days = 20
  wealth_tracker = 1:days
  for (today in 1:days)
  {
    todays_returns = resample(returns, 1, orig.ids = FALSE)
    holdings = holdings + holdings*todays_returns
    wealth = sum(holdings)
    wealth_tracker[today] = wealth
  }
  wealth_tracker
}

# Aggresive Portfolio Bootstrap Simulation
set.seed(202)
aggre_bootstrap = foreach(i = 1:5000, .combine = 'rbind') %do%
{
  holdings = aggre*investment
  days = 20
  wealth_tracker = 1:days
  for (today in 1:days)
  {
    todays_returns = resample(returns, 1, orig.ids = FALSE)
    holdings = holdings + holdings*todays_returns
    wealth = sum(holdings)
    wealth_tracker[today] = wealth
  }
  wealth_tracker
}

# Safe Portfolio Bootstrap Simulation
set.seed(303)
safe_bootstrap = foreach(i = 1:5000, .combine = 'rbind') %do%
{
  holdings = safe*investment
  days = 20
```

```
  wealth_tracker = 1:days
  for (today in 1:days)
  {
    todays_returns = resample(returns, 1, orig.ids = FALSE)
    holdings = holdings + holdings*todays_returns
    wealth = sum(holdings)
    wealth_tracker[today] = wealth
  }
  wealth_tracker
}

#Even Portfolio 5% Value at Risk
quantile(even_bootstrap[,days],0.05)
```

```
##       5%
## 96301.57
```

```
#Aggresive Portfolio 5% Value at Risk
quantile(aggre_bootstrap[,days],0.05)
```

```
##       5%
## 94623.93
```

```
#Safe Portfolio 5% Value at Risk
quantile(safe_bootstrap[,days], 0.05)
```

```
##       5%
## 102792.1
```

We see that with the corresponding weights the even portfolio has a 95% chance of ending up losing money and have a final wealth of around 96K, for the aggressive portfolio we lost even more money at a 95% chance with a final wealth of around 94, finally for the safe portfolio we are almost even with the starting wealth with just 2k over it.

Clustering and PCA

For our third exercise we would like to prove which method is better at separating wine bottles between the red and the whites. Also is the same method able to recreate its accuracy with quality instead. First we read and print out a summary of the wine data set to get a feel of how the data is portrayed. Then we start with clustering to see how well it does dividing the different wines.

```
wine <- read.csv("wine.csv")
summary(wine)
```

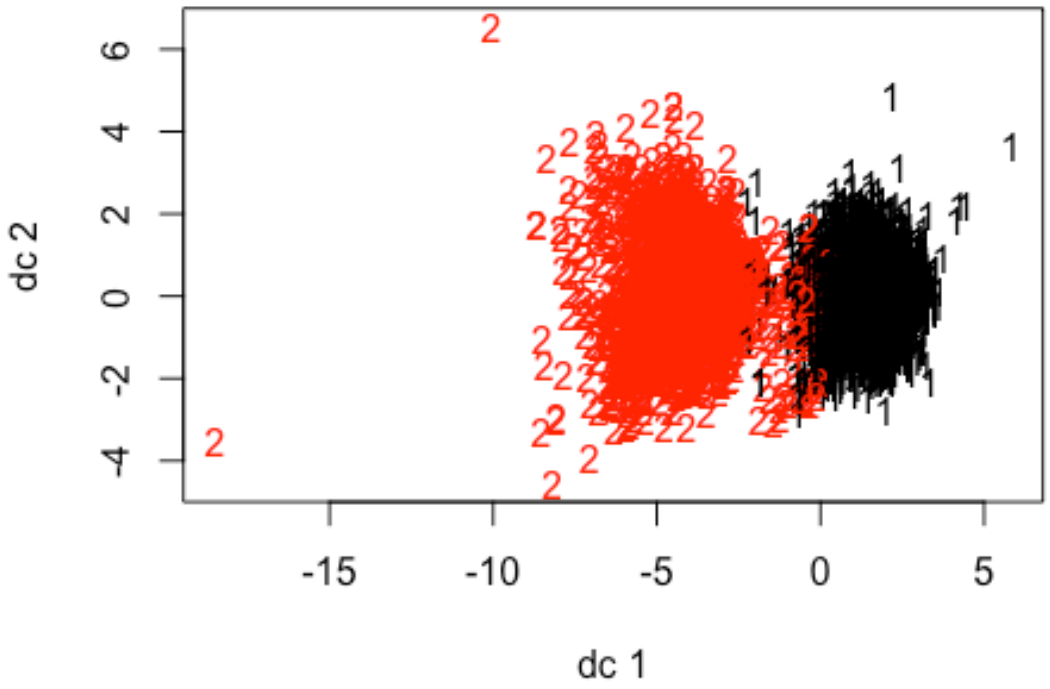
```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800
##  Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000
```

```
##  Mean   : 7.215   Mean   :0.3397   Mean   :0.3186   Mean   : 5.443
##  3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100
##  Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.00900   Min.   :  1.00      Min.   :  6.0
##  1st Qu.:0.03800   1st Qu.: 17.00      1st Qu.: 77.0
##  Median :0.04700   Median : 29.00      Median :118.0
##  Mean   :0.05603   Mean   : 30.53      Mean   :115.7
##  3rd Qu.:0.06500   3rd Qu.: 41.00      3rd Qu.:156.0
##  Max.   :0.61100   Max.   :289.00      Max.   :440.0
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50
##  Median :0.9949   Median :3.210   Median :0.5100   Median :10.30
##  Mean   :0.9947   Mean   :3.219   Mean   :0.5313   Mean   :10.49
##  3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30
##  Max.   :1.0390   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality        color
##  Min.   :3.000   red  :1599
##  1st Qu.:5.000   white:4898
##  Median :6.000
##  Mean   :5.818
##  3rd Qu.:6.000
##  Max.   :9.000
```

```
# Clustering Color
X = wine[,-13]
wine_scaled <- scale(X, center=TRUE, scale=TRUE)

set.seed(101)
cluster_all_prop <- kmeans(wine_scaled, centers=2, nstart=5)

crosstabulate = table(wine$color, cluster_all_prop$cluster)
randIndex(crosstabulate)
```

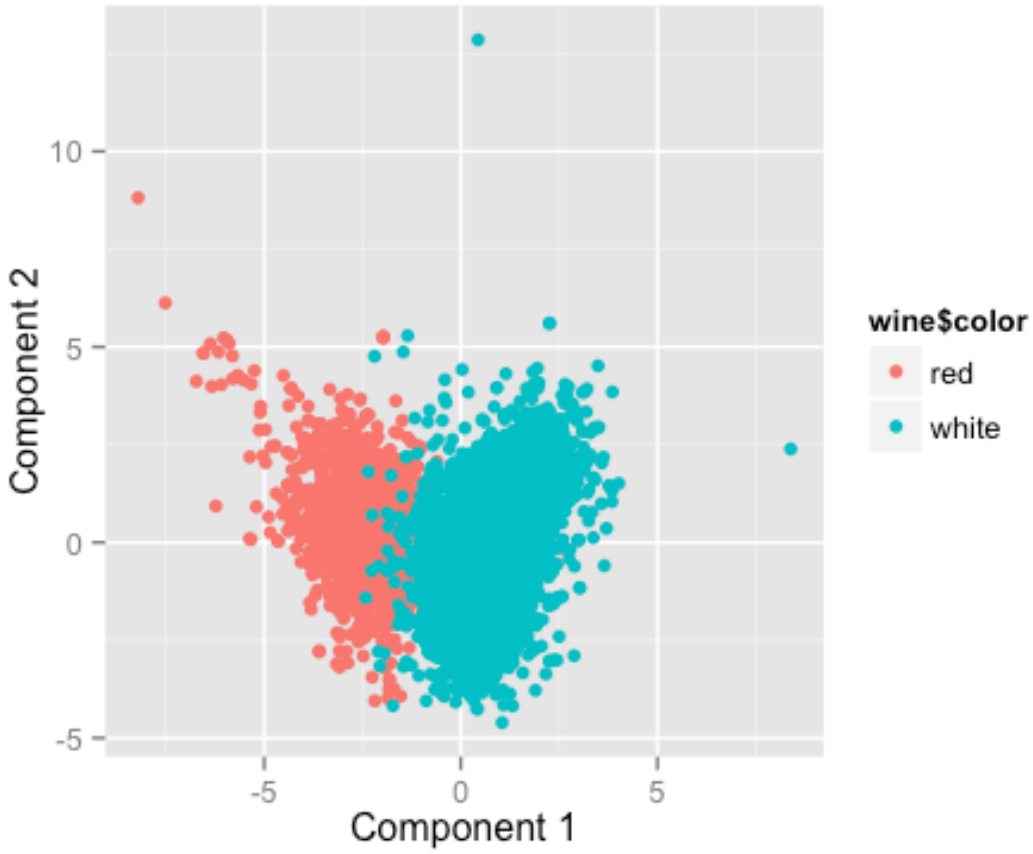
```
##       ARI
## 0.9233953
```

This result are very appealing, by cross tabulation we see that the clustering method actually got a 92% accuracy at defining what color a wine is based on its 11 properties. We can even see it very clearly in the next plot.



Now lets try PCA to see if it has an accuracy as high as the clustering method.

```
# PCA on Color
pc2 = prcomp(X, scale=TRUE)
loadings = pc2$rotation
scores = pc2$x
```



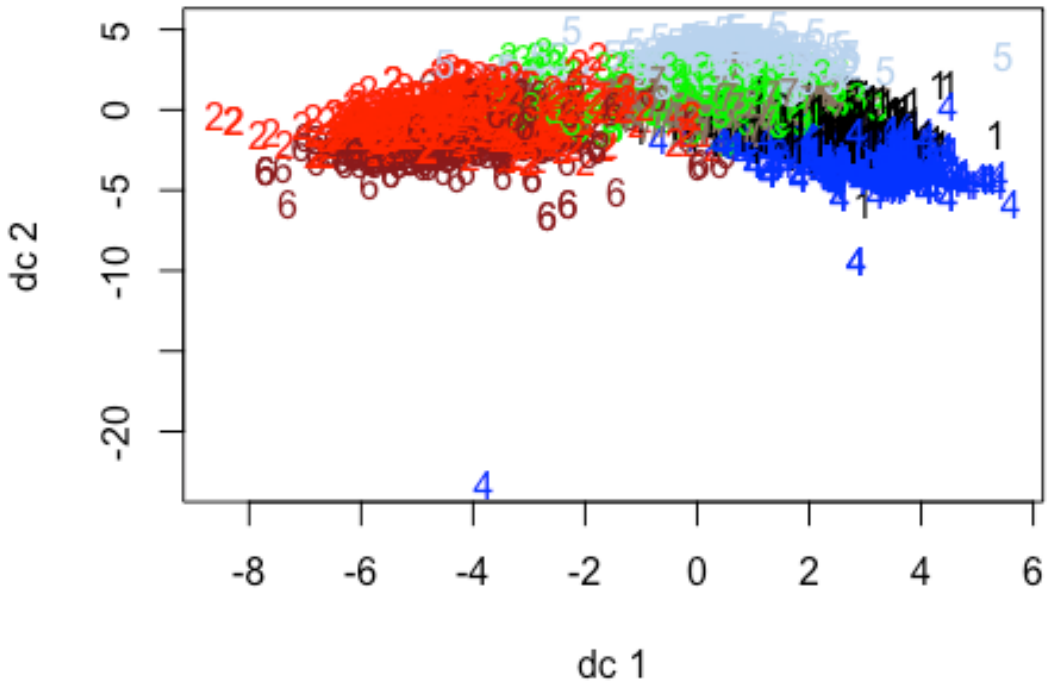
Even though we don't have a precise percentage of the accuracy we can see two very distinct groups in the plot, meaning that PCA also did a very good job at predicting it. Now can we say the same about both methods towards figuring out the quality of the wine?

```
# Clustering Quality
X = wine[,-(12:13)]
wine_scaled <- scale(X, center=TRUE, scale=TRUE)

set.seed(101)
cluster_all_prop <- kmeans(wine_scaled, centers=7, iter.max=30,
nstart=5)
cluster_all_prop$size
```

```
## [1]  926  937  926  871 1093  635 1109
```

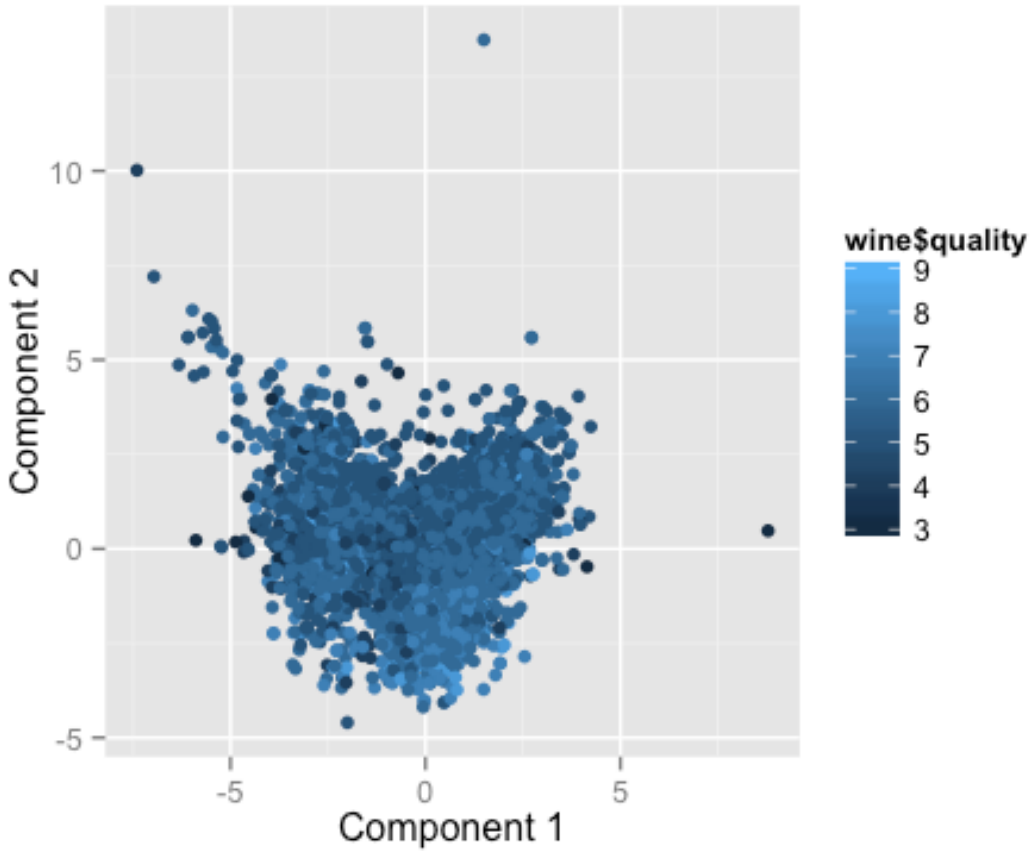
```
crosstabulate = table(wine$quality, cluster_all_prop$cluster)
randIndex(crosstabulate)
```

```
##         ARI
## 0.02945214
```



Just from the cross tabulation result we can see that k mean clustering wont do us any good when trying to identify the wine quality, we can even see it better in the plot they are all pretty much on top of each other not defining a clear group between them. Can PCA do any better?

```
# PCA on Quality
pc2 = prcomp(X, scale=TRUE)
loadings = pc2$rotation
scores = pc2$x
```



And no it doesn't, it appears that for both methods determining the quality of the wine with the 11 properties it's very difficult.

Market segmentation

Now for the last exercise, social media has become a major source of information about costumers for big companies. We got a twitter dataset of a "nameless" company that needs to be analyzed in order to ectract diferent market segments of such company. First as always we load and print a summary of data. Hierachical clustering seems like a good method to approach this problem, and after tweeking it, by trial and error, we ha come to the conclusion that trimming it down to 6 clusters throws the best results.

```
twitter <- read.csv("social_marketing.csv")
summary(twitter)
```

```
##          X           chatter       current_events      travel       
##  123pxkyqj:   1   Min.   : 0.000   Min.   :0.000   Min.   : 0.000  
##  12grikctu:   1   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 0.000  
##  12klxic7j:   1   Median : 3.000   Median :1.000   Median : 1.000  
##  12t4msroj:   1   Mean   : 4.399   Mean   :1.526   Mean   : 1.585  
```

```
##  12yam59l3:   1   3rd Qu.: 6.000   3rd Qu.:2.000   3rd Qu.: 2.000
##  132y8f6aj:   1   Max.   :26.000   Max.   :8.000   Max.   :26.000
##  (Other)  :7876
##  photo_sharing    uncategorized      tv_film      sports_fandom
##  Min.   : 0.000   Min.   :0.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 1.000   1st Qu.:0.000   1st Qu.: 0.00   1st Qu.: 0.000
##  Median : 2.000   Median :1.000   Median : 1.00   Median : 1.000
##  Mean   : 2.697   Mean   :0.813   Mean   : 1.07   Mean   : 1.594
##  3rd Qu.: 4.000   3rd Qu.:1.000   3rd Qu.: 1.00   3rd Qu.: 2.000
##  Max.   :21.000   Max.   :9.000   Max.   :17.00   Max.   :20.000
##
##     politics           food            family        home_and_garden
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 1.000   Median : 1.000   Median : 1.0000   Median :0.0000
##  Mean   : 1.789   Mean   : 1.397   Mean   : 0.8639   Mean   :0.5207
##  3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000
##  Max.   :37.000   Max.   :16.000   Max.   :10.0000   Max.   :5.0000
##
##      music              news        online_gaming       shopping
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.000   Median : 0.000   Median : 1.000
##  Mean   : 0.6793   Mean   : 1.206   Mean   : 1.209   Mean   : 1.389
##  3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.000   3rd Qu.: 2.000
##  Max.   :13.0000   Max.   :20.000   Max.   :27.000   Max.   :12.000
##
##  health_nutrition  college_uni     sports_playing      cooking
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.000
##  Median : 1.000   Median : 1.000   Median :0.0000   Median : 1.000
##  Mean   : 2.567   Mean   : 1.549   Mean   :0.6392   Mean   : 1.998
##  3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.: 2.000
##  Max.   :41.000   Max.   :30.000   Max.   :8.0000   Max.   :33.000
##
##       eco           computers          business         outdoors
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.0000   Median : 0.0000   Median :0.0000   Median : 0.0000
##  Mean   :0.5123   Mean   : 0.6491   Mean   :0.4232   Mean   : 0.7827
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000
##  Max.   :6.0000   Max.   :16.0000   Max.   :6.0000   Max.   :12.0000
##
##      crafts         automotive           art              religion
##  Min.   :0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   :0.5159   Mean   : 0.8299   Mean   : 0.7248   Mean   : 1.095
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.000
##  Max.   :7.0000   Max.   :13.0000   Max.   :18.0000   Max.   :20.000
```

```
##
##      beauty          parenting           dating            school
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :
0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:
0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median :
0.0000
##  Mean   : 0.7052   Mean   : 0.9213   Mean   : 0.7109   Mean   :
0.7677
##  3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.:
1.0000
##  Max.   :14.0000   Max.   :14.0000   Max.   :24.0000   Max.
:11.0000
##
##  personal_fitness    fashion        small_business        spam
##  Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.000   Median : 0.0000   Median :0.0000   Median :0.00000
##  Mean   : 1.462   Mean   : 0.9966   Mean   :0.3363   Mean   :0.00647
##  3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :19.000   Max.   :18.0000   Max.   :6.0000   Max.   :2.00000
##
##      adult
##  Min.   : 0.0000
##  1st Qu.: 0.0000
##  Median : 0.0000
##  Mean   : 0.4033
##  3rd Qu.: 0.0000
##  Max.   :26.0000
##
```

```
head(twitter)
```

```
##           X chatter current_events travel photo_sharing
uncategorized
## 1 hmjoe4g3k       2              0      2             2
2
## 2 clk1m5w8s       3              3      2             1
1
## 3 jcsovtak3       6              3      4             3
1
## 4 3oeb4hiln       1              5      2             2
0
## 5 fd75x1vgk       5              2      0             6
1
## 6 h6nvj91yp       6              4      2             7
0
##   tv_film sports_fandom politics food family home_and_garden music
news
```

```
## 1       1             1        0    4      1               2     0
0
## 2       1             4        1    2      2               1     0
0
## 3       5             0        2    1      1               1     1
1
## 4       1             0        1    0      1               0     0
0
## 5       0             0        2    0      1               0     0
0
## 6       1             1        0    2      1               1     1
0
##   online_gaming shopping health_nutrition college_uni sports_playing
## 1             0        1               17           0              2
## 2             0        0                0           0              1
## 3             0        2                0           0              0
## 4             0        0                0           1              0
## 5             3        2                0           4              0
## 6             0        5                0           0              0
##   cooking eco computers business outdoors crafts automotive art
religion
## 1       5   1         1        0        2      1          0   0
1
## 2       0   0         0        1        0      2          0   0
0
## 3       2   1         0        0        0      2          0   8
0
## 4       0   0         0        1        0      3          0   2
0
## 5       1   0         1        0        1      0          0   0
0
## 6       0   0         1        1        0      0          1   0
0
##   beauty parenting dating school personal_fitness fashion
small_business
## 1      0         1      1      0               11       0
0
## 2      0         0      1      4                0       0
0
## 3      1         0      1      0                0       1
0
## 4      1         0      0      0                0       0
0
## 5      0         0      0      0                0       0
1
## 6      0         0      0      0                0       0
0
##   spam adult
## 1    0     0
## 2    0     0
```

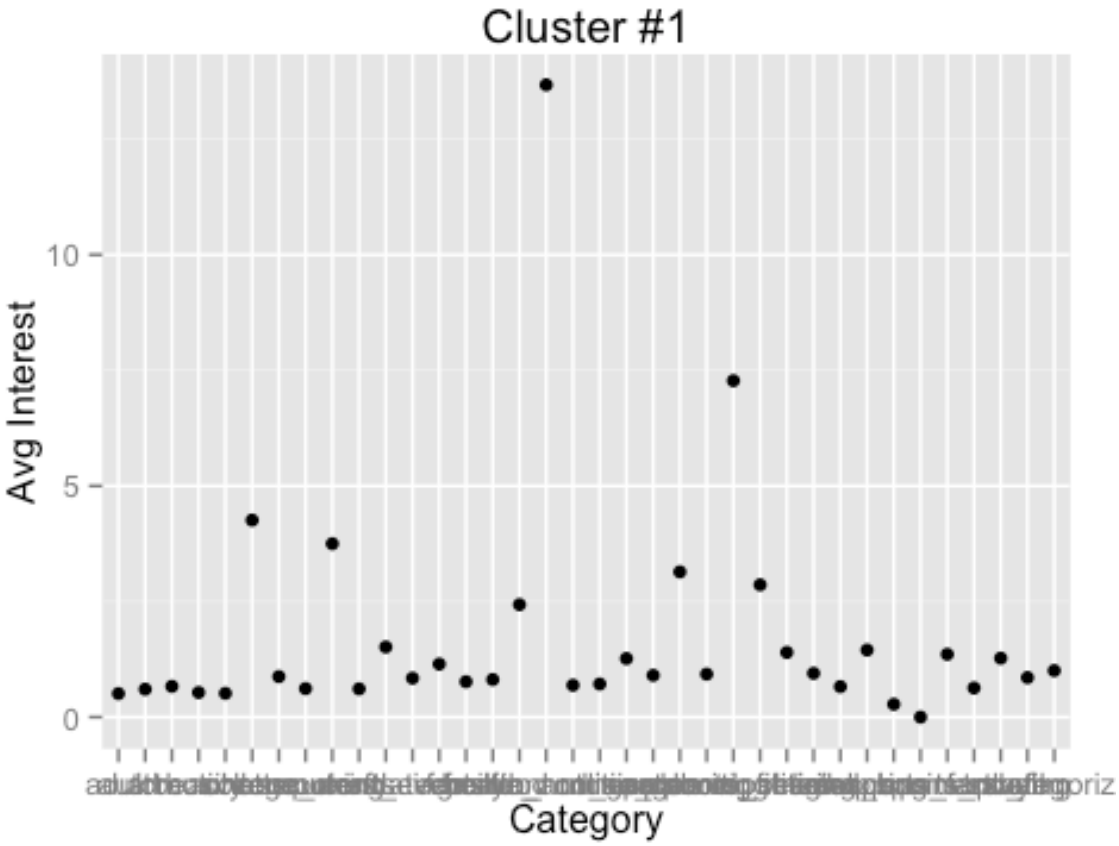
```
## 3    0     0
## 4    0     0
## 5    0     0
## 6    0     0
```

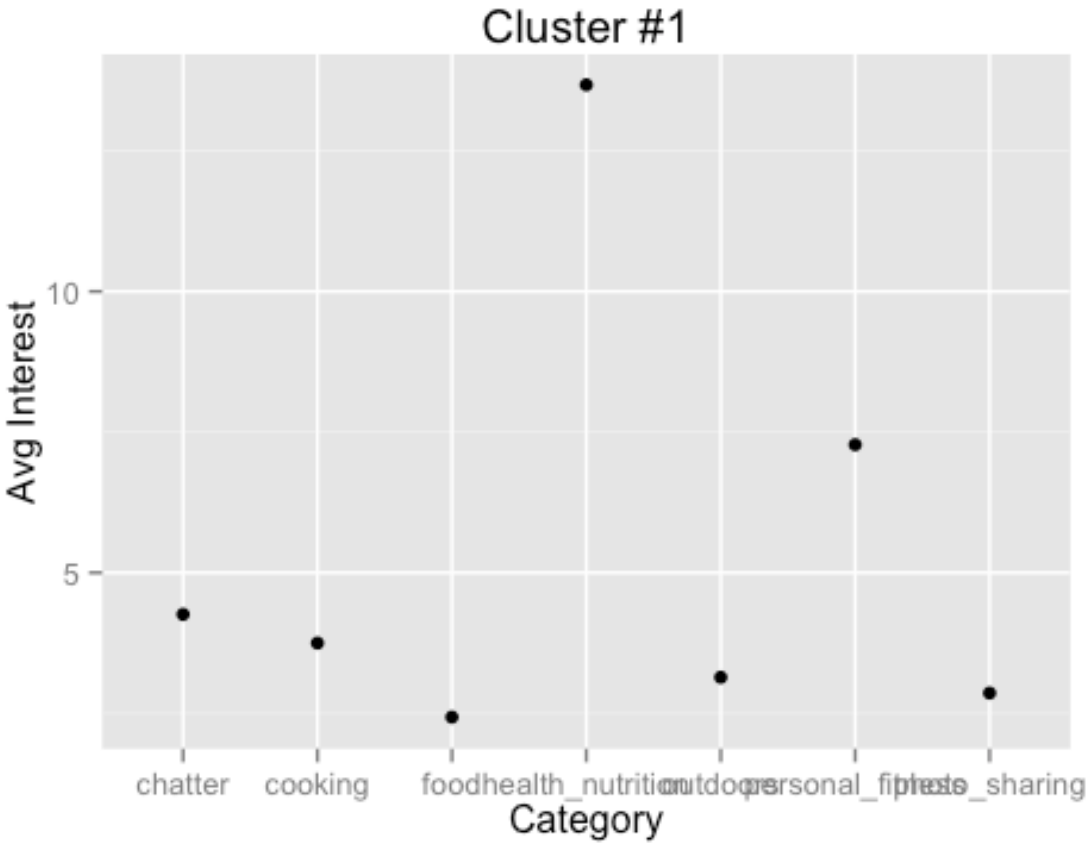
```
names(twitter)
```

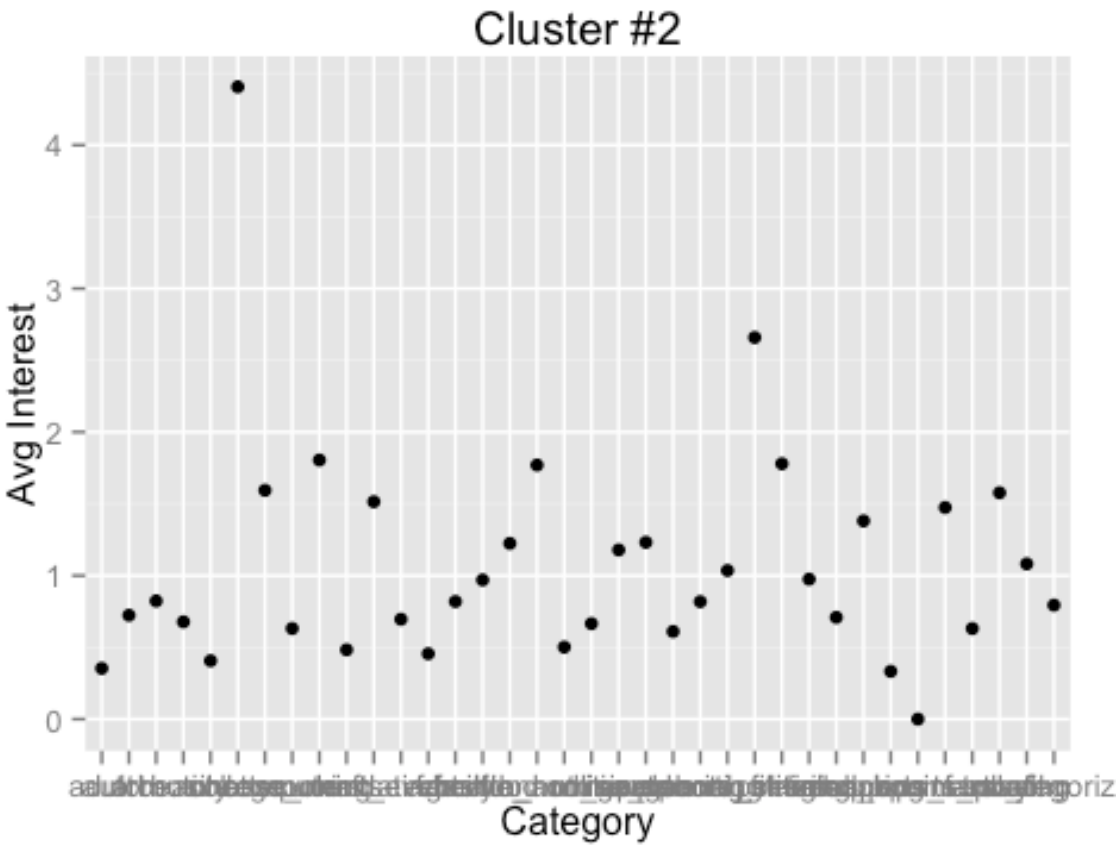
```
##  [1] "X"                "chatter"          "current_events"
##  [4] "travel"           "photo_sharing"    "uncategorized"
##  [7] "tv_film"          "sports_fandom"    "politics"
## [10] "food"             "family"           "home_and_garden"
## [13] "music"            "news"             "online_gaming"
## [16] "shopping"         "health_nutrition" "college_uni"
## [19] "sports_playing"   "cooking"          "eco"
## [22] "computers"        "business"         "outdoors"
## [25] "crafts"           "automotive"       "art"
## [28] "religion"         "beauty"           "parenting"
## [31] "dating"           "school"           "personal_fitness"
## [34] "fashion"          "small_business"   "spam"
## [37] "adult"
```

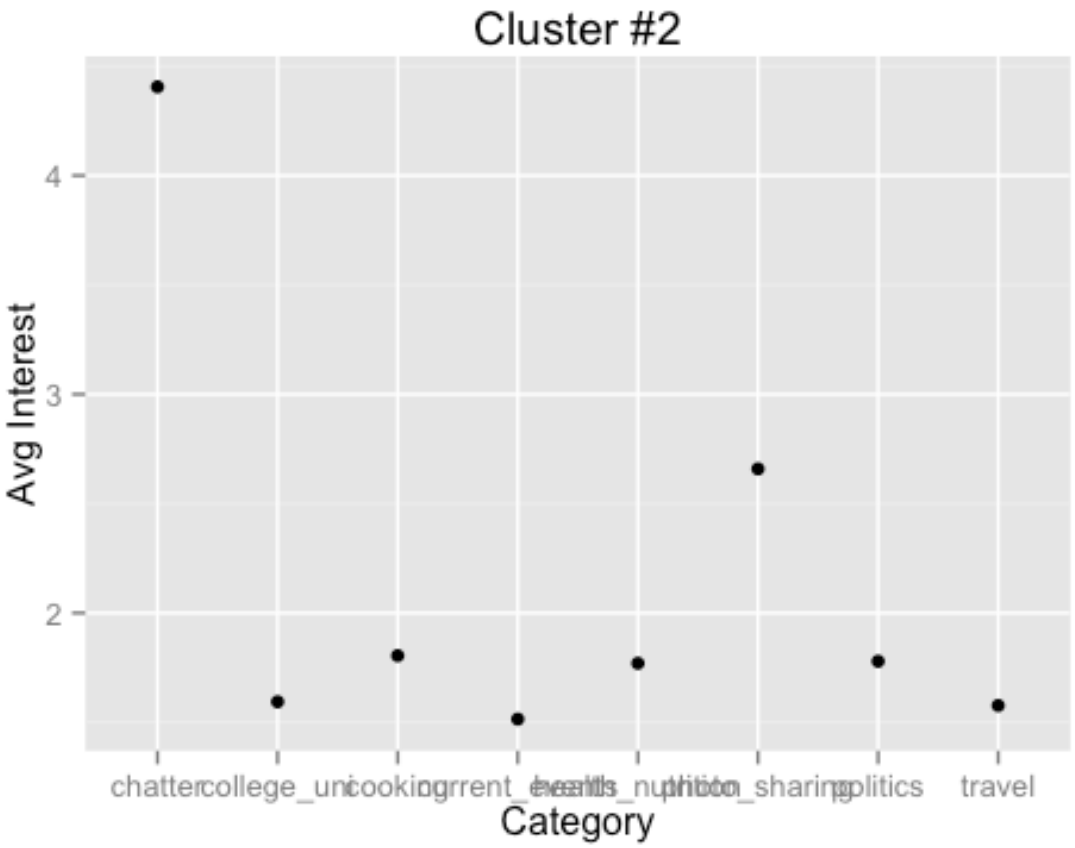
```
twitter_scaled <- scale(twitter[,-1], center=TRUE, scale=TRUE)

twitter_distance_matrix = dist(twitter_scaled, method='euclidean')

hier_twitter = hclust(twitter_distance_matrix, method='complete')

interest_clusters = cutree(hier_twitter, k=6)
```

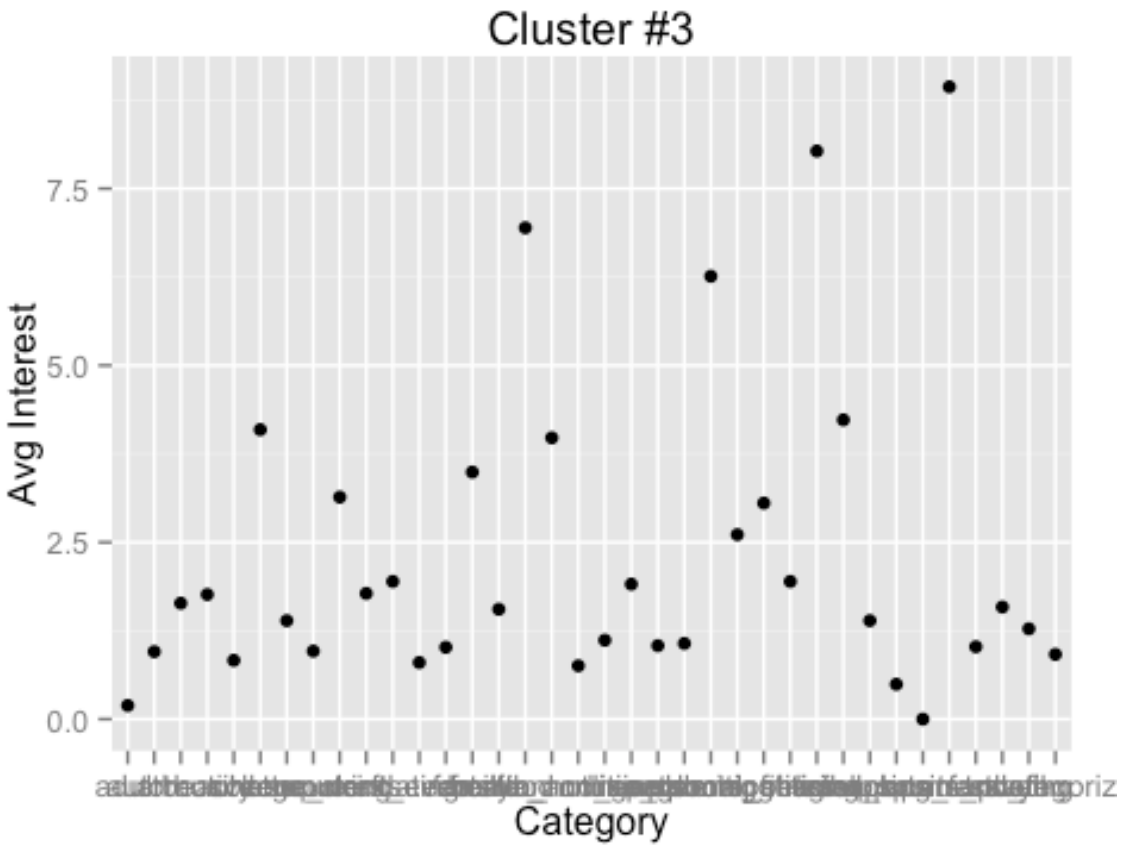
For each individual cluster we have run the same code, that follows: mask the results by clustered against the whole data set to get a subset by cluster. Find the mean for each category and plot those means. Trim the plot base on the most significant categories, name the category

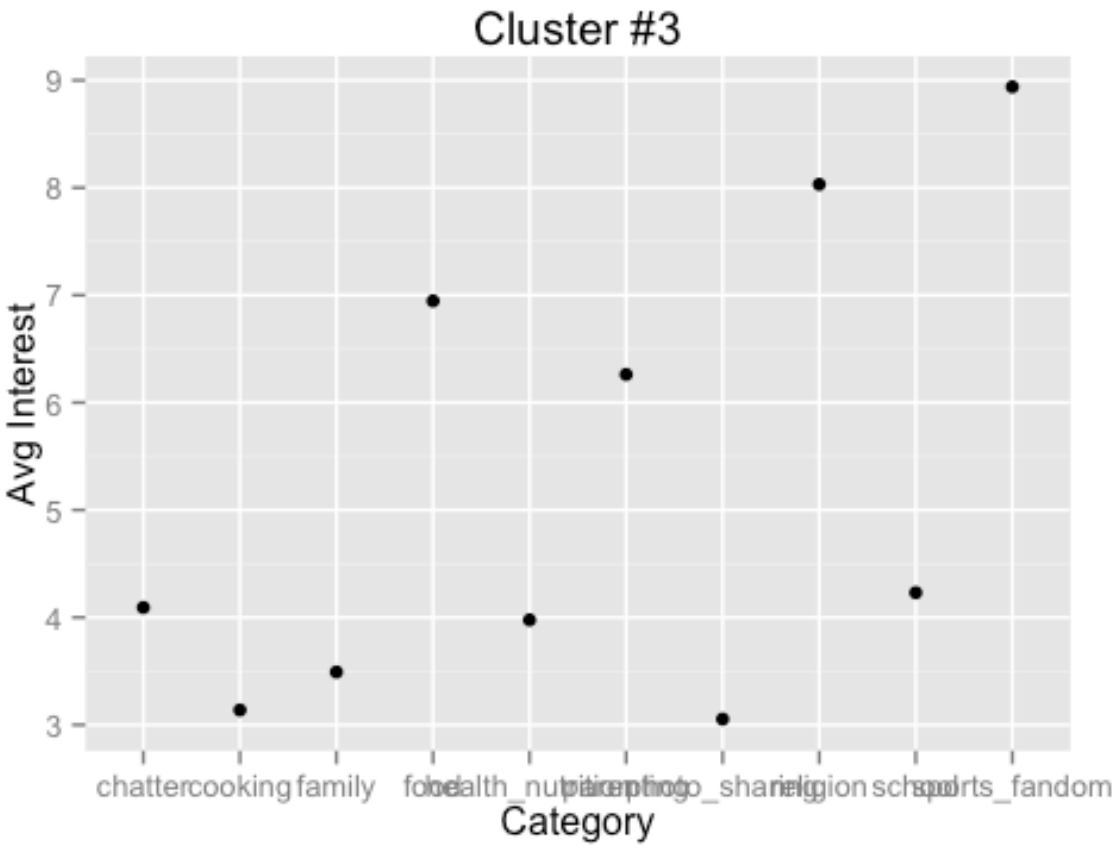

Cluster #1
Avg Interest
10
5
0
Category

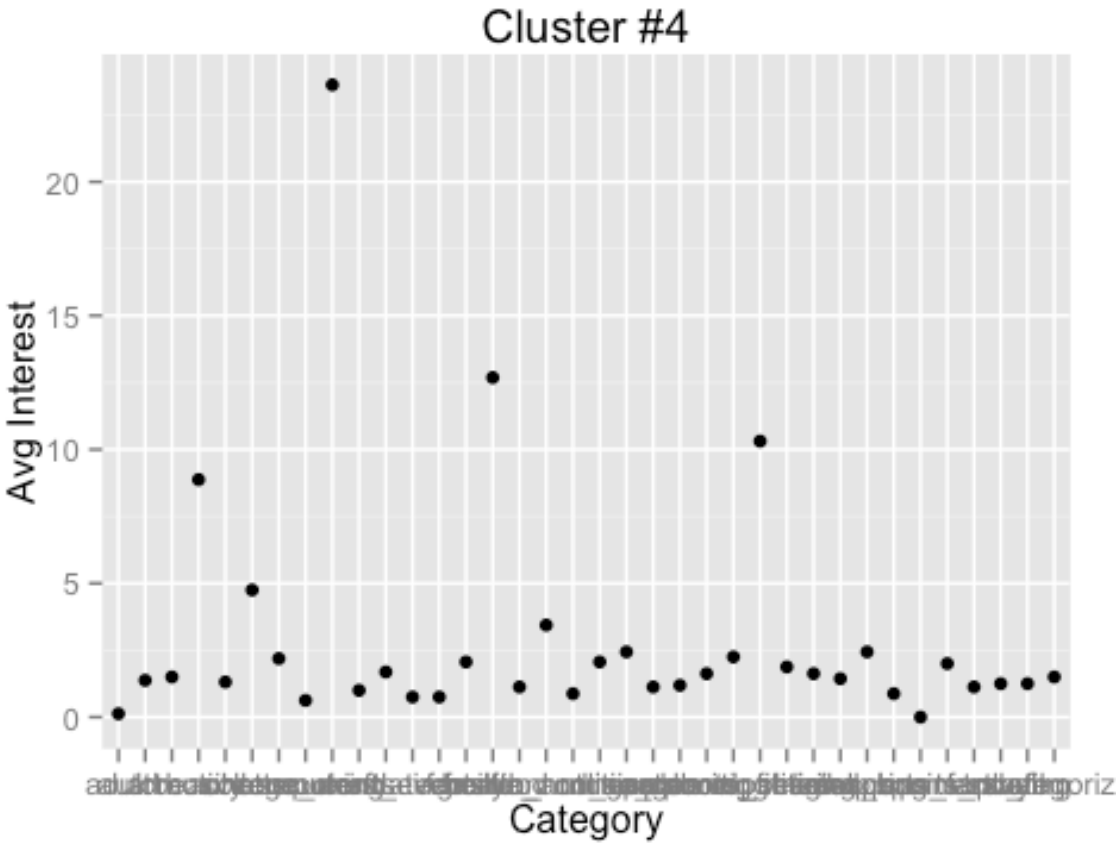

Cluster #1
Avg Interest
10
5
chatter
cooking
food
health_nutrition
outdoors
personal_fitness
photo_sharing
Category


Cluster #2
Avg Interest
4
3
2
1
0
Category

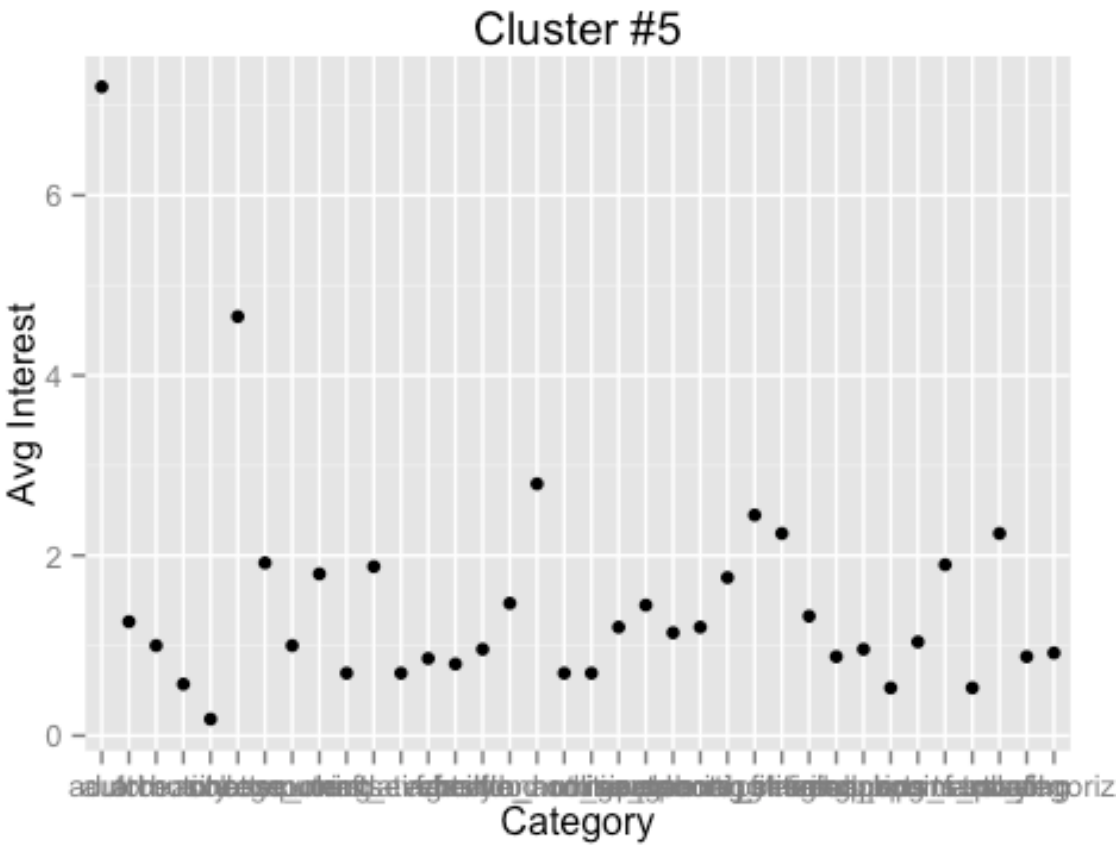

Cluster #2
Avg Interest
4
3
2
chatter
college_un
cooking
politics
travel
Category


Cluster #3
Avg Interest
7.5
5.0
2.5
0.0
Category

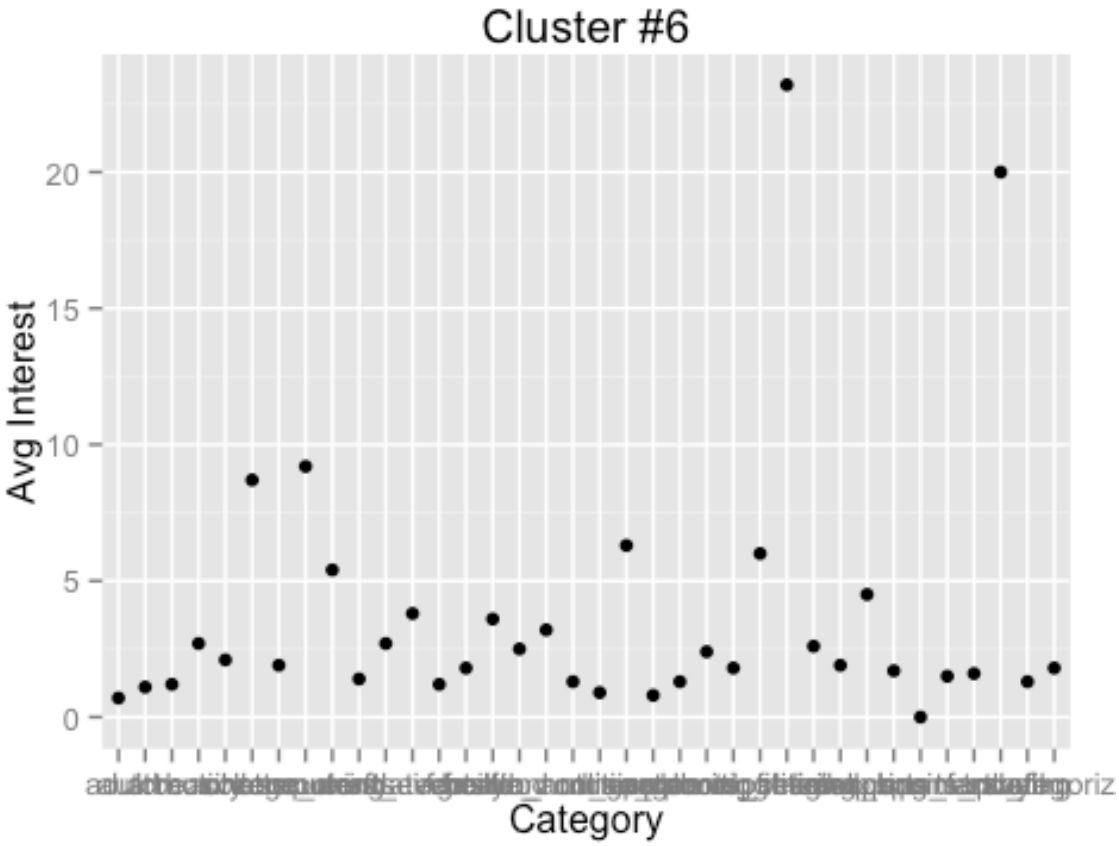

Cluster #3
Avg Interest
9
8
7
6
5
4
3
chattercooking family
sports_fandom
Category


Cluster #4
Avg Interest
20
15
10
5
0
Category


Cluster #4
Avg Interest
20
15
10
5
beauty
chatter
cooking
fashion
health_nutrition
photo_sharing
Category


Cluster #5
Avg Interest
6
4
2
0
Category


Cluster #5
Avg Interest
7
6
5
4
3
2
adult
chatter
health_nutrition
photo_sharing
politics
travel
Category


Cluster #6
Avg Interest
20
15
10
5
0
Category

Now by analyzing each trim we can figure out the differente market segments as follows: CLuster #1 --> Fitness/Nutrition, CLuster #2 --> Cooking Photo Enthusiast, CLuster #3 --> Sports, Religious Parents, CLuster #4 --> Cooking Divas, CLuster #5 --> Gossip/Adult , CLuster #6 --> Traveler/Cultured.