Assignment1\_WenyueShi

Wenyue Shi

August 5, 2015

## Set a global seed

set.seed(1984)

## Exploratory analysis

Load the mosaic library and import the data

library(mosaic)

## Loading required package: car  
## Loading required package: dplyr  
##   
## Attaching package: 'dplyr'  
##   
## The following objects are masked from 'package:stats':  
##   
## filter, lag  
##   
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union  
##   
## Loading required package: lattice  
## Loading required package: ggplot2  
## Loading required package: mosaicData  
##   
## Attaching package: 'mosaic'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally  
##   
## The following object is masked from 'package:car':  
##   
## logit  
##   
## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cov, D, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var  
##   
## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

vote = read.csv('../data/georgia2000.csv')

Take a look at the data frame by looking at the summary.

summary(vote)

## 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   
##

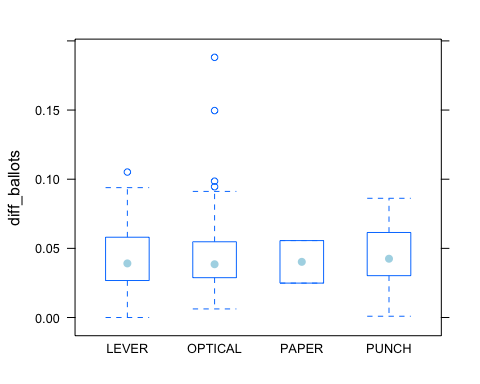
Calculate the undercount\_percentage

and add the percentage to the original data set.

diff = vote$ballots - vote$votes  
vote$diff <- diff  
diff\_ballots = vote$diff/vote$ballots  
vote$diff\_ballots = diff\_ballots

Boxplot the percentage:

bwplot(diff\_ballots~equip, data = vote, col = "Light Blue", outline = FALSE)



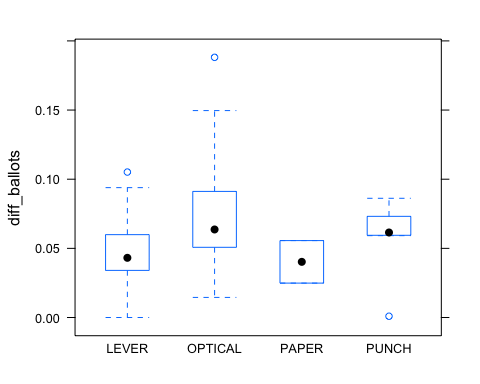
From the boxplot, in general, different equipment of voting does not lead to higher rate of undercount.

### Poor Areas

But then we want to see whether different equipment does affect undercount in poor area. Select the poor observation out, where

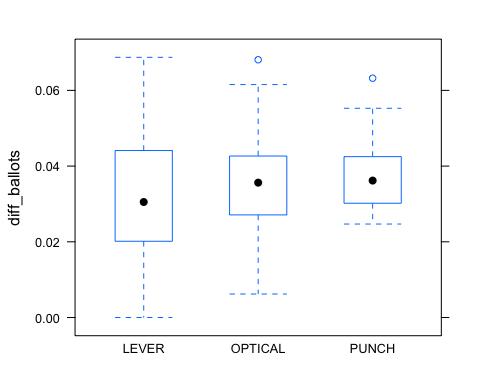
and plot the undercount percentage again with the boxplot.

poor <- vote[vote$poor == 1, ]  
bwplot(diff\_ballots~equip, data = poor)



Obviously, optical and punch lead to a higher level of undercount than lever and paper. To compare with non-poor areas, I did the similar thing to non-poor area.

notpoor <- vote[vote$poor == 0, ]  
bwplot(diff\_ballots~equip, data = notpoor)



Clearly, equipment doesn't obviously affect the undercount level, even though optical and punch still generate a little bit higher undercount.

Overall, poor areas have

* higher undercount\_percentage than non-poor areas
* remarkable different undercount value with different equipments

### Minority Areas

Now let's take a look at minority communities. Since perAA (the percentage of African American) is a quantatative feature, we'd like to run four linear regression between undercount percentaeg level and perAA under the situation of different equipments.

lever = vote[vote$equip == 'LEVER', ]  
optical = vote[vote$equip == 'OPTICAL', ]  
paper = vote[vote$equip == 'PAPER', ]  
punch = vote[vote$equip == 'PUNCH', ]  
  
lm.lever = lm(diff\_ballots~perAA, data = lever)  
summary(lm.lever)

##   
## Call:  
## lm(formula = diff\_ballots ~ perAA, data = lever)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.042857 -0.014572 -0.003374 0.015299 0.062319   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.043712 0.004783 9.139 1.15e-13 \*\*\*  
## perAA -0.006584 0.014897 -0.442 0.66   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02097 on 72 degrees of freedom  
## Multiple R-squared: 0.002706, Adjusted R-squared: -0.01115   
## F-statistic: 0.1954 on 1 and 72 DF, p-value: 0.6598

lm.optical = lm(diff\_ballots~perAA, data = optical)  
summary(lm.optical)

##   
## Call:  
## lm(formula = diff\_ballots ~ perAA, data = optical)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.037657 -0.012688 -0.002691 0.008405 0.132622   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.025166 0.005342 4.711 1.37e-05 \*\*\*  
## perAA 0.107561 0.022993 4.678 1.55e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02599 on 64 degrees of freedom  
## Multiple R-squared: 0.2548, Adjusted R-squared: 0.2432   
## F-statistic: 21.88 on 1 and 64 DF, p-value: 1.546e-05

lm.punch = lm(diff\_ballots~perAA, data = punch)  
summary(lm.punch)

##   
## Call:  
## lm(formula = diff\_ballots ~ perAA, data = punch)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.048304 -0.007817 0.004888 0.013611 0.026992   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.03063 0.01014 3.021 0.00859 \*\*  
## perAA 0.05517 0.02967 1.859 0.08269 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02035 on 15 degrees of freedom  
## Multiple R-squared: 0.1873, Adjusted R-squared: 0.1331   
## F-statistic: 3.458 on 1 and 15 DF, p-value: 0.08269

For Lever equipment, perAA seems don't have a correlation with undercounts. But for optical method and punch method, the coefficients of perAA are and 0.0826 perspectively, indicating that the more African American, the more likely to have undercount with Optical and Punch equipment.

## BootStraping:

Library related packages:

library(mosaic)  
library(fImport)

## Loading required package: timeDate  
## Loading required package: timeSeries

library(foreach)

Import the five year price from 2010-08-01 to 2015-07-31 of the following five asset class:

* US domestic equities (SPY: the S&P 500 stock index)
* US Treasury bonds (TLT)
* Investment-grade corporate bonds (LQD)
* Emerging-market equities (EEM)
* Real estate (VNQ)

Take a look at the first five rows.

mystocks = c("SPY", "TLT", "LQD", "EEM", "VNQ")  
myprices = yahooSeries(mystocks, from='2010-08-01', to='2015-07-30')  
head(myprices, 5)

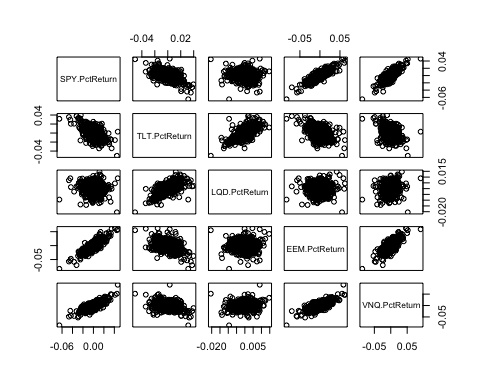
## GMT  
## SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adj.Close  
## 2010-08-02 111.99 112.94 111.54 112.76 188263200 101.8326  
## 2010-08-03 112.48 112.77 111.85 112.22 146657300 101.3450  
## 2010-08-04 112.53 113.11 112.16 112.97 158171700 102.0223  
## 2010-08-05 112.25 112.91 112.08 112.85 140473800 101.9139  
## 2010-08-06 111.74 112.57 110.92 112.39 239728300 101.4985  
## TLT.Open TLT.High TLT.Low TLT.Close TLT.Volume TLT.Adj.Close  
## 2010-08-02 99.24 99.33 98.75 98.75 5769200 84.60973  
## 2010-08-03 99.20 99.66 98.93 99.32 4363500 85.09811  
## 2010-08-04 99.50 99.51 98.56 98.56 3820400 84.44693  
## 2010-08-05 99.34 99.49 98.84 99.02 3704200 84.84106  
## 2010-08-06 99.79 100.21 99.49 100.10 6042400 85.76641  
## LQD.Open LQD.High LQD.Low LQD.Close LQD.Volume LQD.Adj.Close  
## 2010-08-02 109.91 109.95 109.56 109.63 764100 90.53107  
## 2010-08-03 109.90 110.04 109.70 109.90 1060700 90.75404  
## 2010-08-04 109.83 109.95 109.55 109.56 859900 90.47327  
## 2010-08-05 109.69 109.89 109.59 109.76 1093400 90.63843  
## 2010-08-06 110.19 110.48 110.06 110.39 685700 91.15867  
## EEM.Open EEM.High EEM.Low EEM.Close EEM.Volume EEM.Adj.Close  
## 2010-08-02 42.18 42.59 42.07 42.47 69623700 38.51627  
## 2010-08-03 42.14 42.43 41.93 42.27 60207900 38.33489  
## 2010-08-04 42.28 42.43 42.00 42.33 55875600 38.38930  
## 2010-08-05 42.02 42.20 41.87 42.14 43650600 38.21699  
## 2010-08-06 41.86 42.19 41.60 42.08 65731600 38.16258  
## VNQ.Open VNQ.High VNQ.Low VNQ.Close VNQ.Volume VNQ.Adj.Close  
## 2010-08-02 51.78 52.81 51.62 52.66 3018300 43.59576  
## 2010-08-03 52.53 52.57 51.78 52.15 1955500 43.17355  
## 2010-08-04 52.39 52.54 51.90 52.52 2041300 43.47986  
## 2010-08-05 52.24 52.50 51.75 51.86 1847300 42.93346  
## 2010-08-06 51.31 51.78 50.76 51.62 1836100 42.73477

Since we are going to simulate a short-term (4 weeks) return, we would like to know the day to day return of five stocks using their close price. The following function calculate the

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))  
}

Use the function to get the day to day returns of the five portfolio we selected. Plot the correlation between each pairs of the five assets and print out the standard deviation of their day-to-day returns of each assets.

myreturns = YahooPricesToReturns(myprices)  
pairs(myreturns)



apply(myreturns, 2, sd)

## SPY.PctReturn TLT.PctReturn LQD.PctReturn EEM.PctReturn VNQ.PctReturn   
## 0.009354486 0.009769863 0.003579966 0.013729404 0.011523419

From the plot of the returns, we can figure out the following fact:

* returns on SPY, EEM and VNQ are positively correlated
* returns on SPY and TLT are negatively correlated
* returns on TLT and LQD are positively correlated
* returns on LQD has no obviouse correlation with SPY, EEM and VNQ

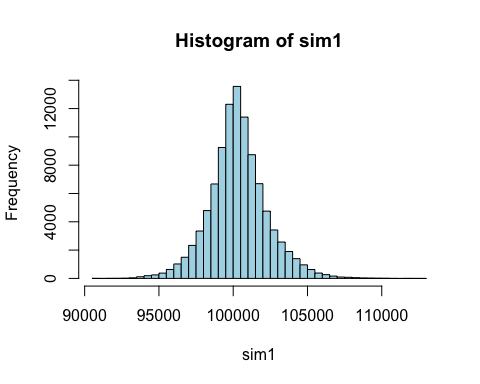
From the standard deviation of the returns, we can see that:

* EEM is the most risky assets, followed by VNQ
* LQD is the least risky assets

### Evenly Split Portfolio

Simulation on evenly split portfolio, with weights being 0.2, 0.2, 0.2, 0.2, 0.2

sim1 = foreach(i = 1:5000, .combine = 'cbind')%do% {  
 total\_wealth = 100000  
 weights = c(0.2, 0.2, 0.2, 0.2, 0.2)  
 holdings = total\_wealth \* weights  
 n\_days = 20  
 wealthtracker = rep(0, 20)  
 for (today in 1:n\_days) {  
 return.today = resample(myreturns, 1, orig.ids = FALSE)  
 holdings = holdings + holdings \* return.today  
 total\_wealth = sum(holdings)  
 wealthtracker[today] = total\_wealth  
 }  
 wealthtracker  
}  
  
hist(sim1, 70, col = "Light Blue")



risk1 = quantile(sim1, 0.05) - 100000  
risk1

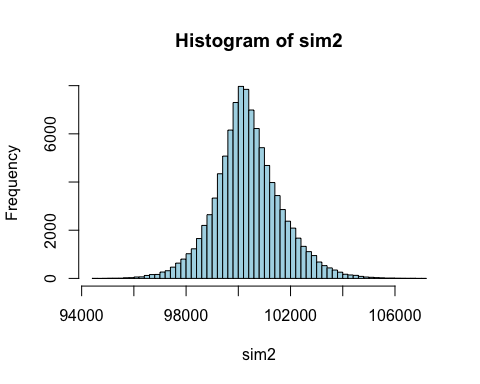
## 5%   
## -2792.256

### A Safer Portfolio

Based on the above analysis of standard deviation and correlation, a safer portfolio than evenly split will be the one with SPY, TLT and LQD, because SPY and TLT have a comparative level of risk but negatively correlated, and LQD is the safest assets among all of the assets.

The following is the simulation of a safer portfilio with weights being 0.3, 0.3, 0.4, 0, 0

sim2 = foreach(i = 1:5000, .combine = 'cbind')%do% {  
 total\_wealth = 100000  
 weights = c(0.3, 0.3, 0.4, 0, 0)  
 holdings = total\_wealth \* weights  
 n\_days = 20  
 wealthtracker = rep(0, 20)  
 for (today in 1:n\_days) {  
 return.today = resample(myreturns, 1, orig.ids = FALSE)  
 holdings = holdings + holdings \* return.today  
 total\_wealth = sum(holdings)  
 wealthtracker[today] = total\_wealth  
 }  
 wealthtracker  
}  
hist(sim2, 60, col = "Light Blue")



risk2 = quantile(sim2, 0.05) - 100000  
risk2

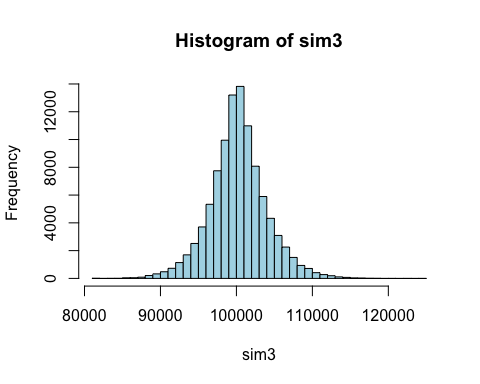
## 5%   
## -1662.459

### A More Aggressive Portfolio

Similarly, a more regressive portfolio than the evenly split one would include EEM, VNQ and SPY, since they are not only positively correlated to each other but also have relatively high risk.

The simulation is the following:

sim3 = foreach(i = 1:5000, .combine = 'cbind')%do% {  
 total\_wealth = 100000  
 weights = c(0.2, 0, 0, 0.6, 0.2)  
 holdings = total\_wealth \* weights  
 n\_days = 20  
 wealthtracker = rep(0, 20)  
 for (today in 1:n\_days) {  
 return.today = resample(myreturns, 1, orig.ids = FALSE)  
 holdings = holdings + holdings \* return.today  
 total\_wealth = sum(holdings)  
 wealthtracker[today] = total\_wealth  
 }  
 wealthtracker  
}  
hist(sim3, 60, col = "Light Blue")



risk3 = quantile(sim3, 0.05) - 100000  
risk3

## 5%   
## -5883.862

Now we have three options:

* The even split
  + 20% of your assets in each of the ETFs
  + The risk at 5% level is -2792.26.
* The safer option
  + 30% of your assets in SPY, 30% of you assets in TLT and the last 40% in LQD
  + The risk at 5% level is -1662.46.
* The more aggresive option
  + 20% of you assets in SPY, 20% of you assets in VNQ and the rest 60% in EEM
  + The risk at 5% level is -5883.86.

## Clustering and PCA

Load the library and the data.

library(caret)

##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:mosaic':  
##   
## dotPlot

library(ggplot2)  
library(cclust)  
wines = read.csv('../data/wine.csv')

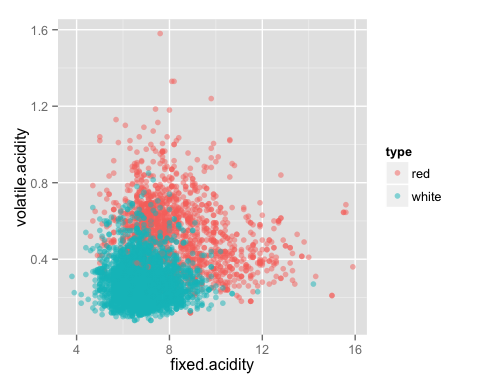
Summarize the data to see the columns. Remove the last two columns which are quality and color, since we are only clustering with the first 11 features. Scale the wine data.

wine = wines[, c(-12, -13)]  
wine\_scaled <- scale(wine, center = TRUE, scale = TRUE)

### k-means clustering

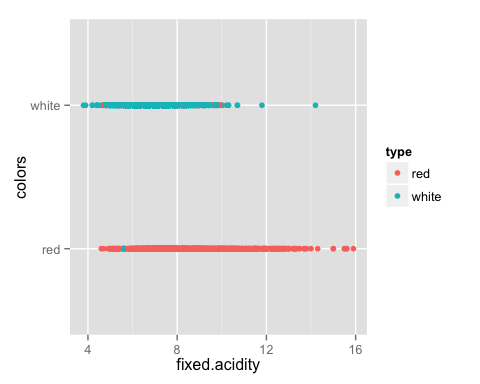
Since we know that we want two clusters seperating red and white wine, let's start with kmeans with 2 centers.

wine\_kmean <- kmeans(wine\_scaled, centers=2, nstart = 50)  
type = ifelse(wine\_kmean$cluster == 1, "white", "red")  
qplot(wines$fixed.acidity, wines$volatile.acidity, col = type, xlab = "fixed.acidity", ylab = "volatile.acidity", alpha = I(0.5))



The first plot shows the red wine tend to have lower level of both volatile acidity and fixed acidity. But it is unclear whether the red dots are truly red wine.

qplot(wines$fixed.acidity, wines$color, col = type, xlab = "fixed.acidity", ylab = "colors")



This plot indicates the true classification of red and white wine in the y-axis, with the dots color representing the clustering result. Clearly, kmeans did a good job in clustering the color of the wine. Most red dots fall into the true red wine category, and most green dots fall into the true white wine category.

A confusion matrix can show us the accuracy of the classification.

confusionMatrix(type, wines$color)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction red white  
## red 1575 68  
## white 24 4830  
##   
## Accuracy : 0.9858   
## 95% CI : (0.9827, 0.9886)  
## No Information Rate : 0.7539   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9622   
## Mcnemar's Test P-Value : 7.358e-06   
##   
## Sensitivity : 0.9850   
## Specificity : 0.9861   
## Pos Pred Value : 0.9586   
## Neg Pred Value : 0.9951   
## Prevalence : 0.2461   
## Detection Rate : 0.2424   
## Detection Prevalence : 0.2529   
## Balanced Accuracy : 0.9856   
##   
## 'Positive' Class : red   
##

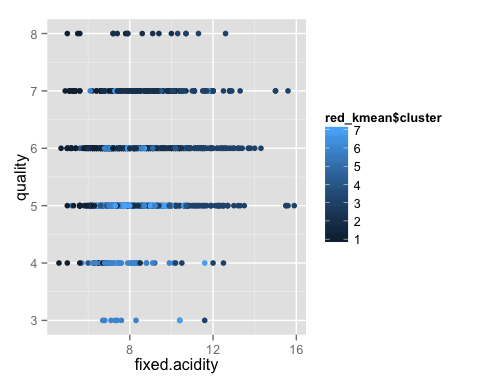
Of all the 1599 kinds of red wine, 24 of them were clustered to white wine. The accuracy of classifying red wine is around 98.5%. Of 4898 kinds of white wine, 68 of them are misclassified by kmeans, and the accuracy is 98.61%.

However, can they tell the difference between different wine quality? Let's split the data frame into two subset by their colors.

red = wines[wines$color == "red", ]  
white = wines[wines$color == "white", ]  
red\_scaled = scale(red[, c(-12, -13)], center = TRUE, scale = TRUE)  
white\_scaled = scale(white[, c(-12, -13)], center = TRUE, scale = TRUE)

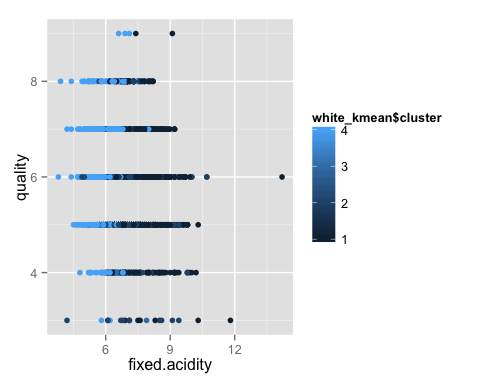
Try kmeans on each subset to see whether kmeans can rank them.

red\_kmean <- kmeans(red\_scaled, centers = 7, nstart = 100)  
qplot(red$fixed.acidity, red$quality, col = red\_kmean$cluster, xlab = "fixed.acidity", ylab = "quality")



From the plot, we cannot see a clearly clustering by ranks. Try the same with white wine.

white\_kmean <- kmeans(white\_scaled, centers = 4, nstart = 100)  
qplot(white$fixed.acidity, white$quality, col = white\_kmean$cluster, xlab = "fixed.acidity", ylab = "quality")



We still cannot tell the difference between each cluster in terms of their quality. It seems their quality are not determined by these 11 features.

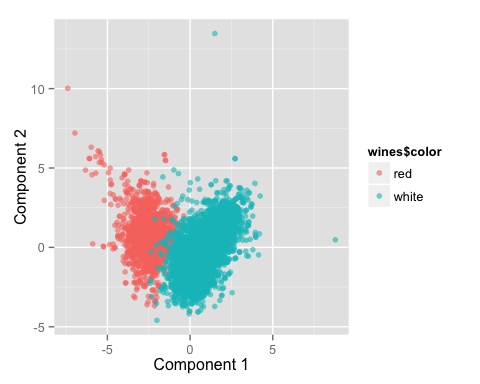
## Principal Component Analysis

Apply PCA to the scaled wine data. Get the component vectors and the projection position on those vectors, named as scores.

wine\_pca <- prcomp(wine\_scaled)  
loadings = wine\_pca$rotation  
scores = wine\_pca$x

Plot the red and white wine with x being the first component and y being the second component.

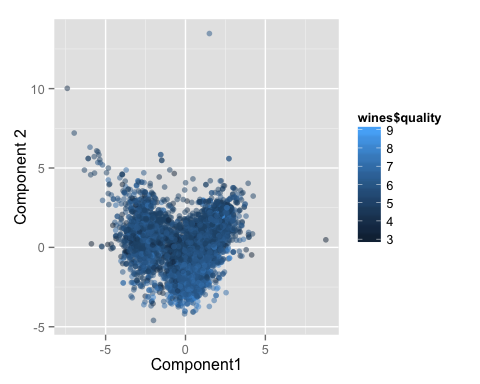
qplot(scores[,1], scores[,2], color=wines$color, xlab='Component 1', ylab='Component 2', alpha = I(0.6))



Clearly, most kinds of wine, including both red and white, fall into the range of [-5, 5] in temrs of component 2. On the contrary, component 1 can seperate the two kinds of wine into two category. That is to say, component 1 itself is interesting enough in terms of telling the difference between wine color, while component 2 is does not provide much extra information. Since component 2 is not interesting to us, there's no need to consider the rest components.

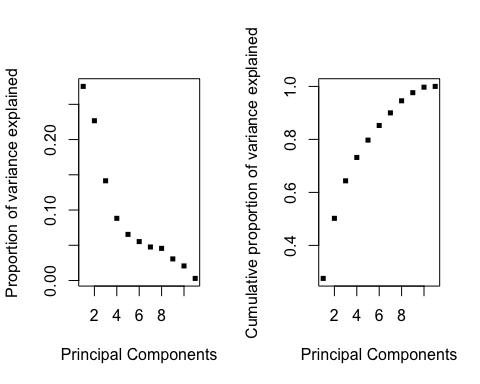
However, PCA doesn't seem work well when we want to know the scores of the wine. Let's take a look at the plot with color representing different scores.

qplot(scores[, 1], scores[, 2], color = wines$quality, xlab = "Component1", ylab = "Component 2", alpha = I(0.5))



Even component 1 and component 2 together are not enough to order the quality of these wine.

vars <- (wine\_pca$sdev)^2  
sum <- sum(vars)  
percent <- vars/sum  
par(mfrow = c(1, 2))  
plot(percent, xlab = "Principal Components", ylab = "Proportion of variance explained", cex = 0.7, pch = 15, col = 9)  
plot(cumsum(percent), xlab = "Principal Components", ylab = "Cumulative proportion of variance explained", cex = 0.7, pch = 15, col = 9)



par(mfrow = c(1, 1))

The above two graph showed that the first two components explaine nearly 38% of the variance, and we need more components to rank the wine.

In conclusion, PCA works well in distinguishing red and white wine, but not in ranking the quality.

## Market Segmentation

Load the library and the data. Take a look at the first row of the data set.

library(cluster)  
market <- read.csv("../data/social\_marketing.csv")  
head(market, 1)

## X chatter current\_events travel photo\_sharing uncategorized  
## 1 hmjoe4g3k 2 0 2 2 2  
## tv\_film sports\_fandom politics food family home\_and\_garden music news  
## 1 1 1 0 4 1 2 0 0  
## online\_gaming shopping health\_nutrition college\_uni sports\_playing  
## 1 0 1 17 0 2  
## cooking eco computers business outdoors crafts automotive art religion  
## 1 5 1 1 0 2 1 0 0 1  
## beauty parenting dating school personal\_fitness fashion small\_business  
## 1 0 1 1 0 11 0 0  
## spam adult  
## 1 0 0

We don't want the first column, "spam" column, "chatter" column and "uncategorized" column which does't provide interesting information about the user. So delete these columns.

Scale the data because some of the topic might be tweeted more often than others.

markets = market[, -1]  
markets = markets[, -which(names(markets)=="spam")]  
markets = markets[, -which(names(markets) == "chatter")]  
markets = markets[, -which(names(markets) == "uncategorized")]  
market\_scaled <- scale(markets, center = TRUE, scale = TRUE)

Apply kmeans to the scaled data frame.

center\_num = 8  
segments <- kmeans(market\_scaled, centers = center\_num, nstart = 100)

Take a look at the clustering centers.

segments$centers

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## 1 0.10234034 -0.11272269 -0.025459271 -0.10705254 2.0249394  
## 2 -0.09330702 -0.22207450 -0.147276970 -0.22458904 -0.3147726  
## 3 0.01025367 -0.15952768 0.002424409 -0.15545129 -0.2087017  
## 4 0.11780919 3.19186654 -0.057575455 -0.07686805 -0.2181462  
## 5 0.08895021 -0.18557373 -0.126954494 -0.03844082 0.6355539  
## 6 0.31876161 0.20430462 0.058709428 2.64827640 -0.1106166  
## 7 0.17632541 -0.05991035 1.279706183 -0.14787147 -0.2099234  
## 8 -0.05862658 -0.04421146 0.026826191 0.09302734 -0.1325942  
## politics food family home\_and\_garden music  
## 1 -0.2238752 1.80041211 1.45616807 0.15771029 0.032279123  
## 2 -0.2838071 -0.37481987 -0.26822345 -0.13468126 -0.180055714  
## 3 -0.1921082 0.41922856 -0.07854314 0.14777257 0.002940299  
## 4 3.0479280 0.14597546 -0.09151735 0.04035434 -0.045743648  
## 5 1.1907155 -0.17611672 0.23536706 0.13717972 -0.079069871  
## 6 -0.0952232 0.09674490 -0.12442881 0.29806205 1.088913110  
## 7 -0.1269455 -0.20220491 0.04039369 0.13899567 0.535429228  
## 8 -0.1622641 -0.09285402 0.20046769 0.06992411 -0.033867664  
## news online\_gaming shopping health\_nutrition college\_uni  
## 1 -0.12730023 -0.08179991 0.06534396 -0.15583728 -0.13316289  
## 2 -0.31671659 -0.22630957 -0.06882735 -0.32254820 -0.24471756  
## 3 -0.09113928 -0.11565666 0.04981961 2.12456190 -0.20787115  
## 4 1.11359940 -0.15476099 -0.01792300 -0.16009949 -0.03918070  
## 5 2.56887531 -0.13999124 -0.07224083 -0.26347758 -0.19941660  
## 6 0.01240974 -0.17354869 0.20355939 -0.18248717 0.40952284  
## 7 -0.08776632 -0.03405535 0.32163363 -0.06799234 -0.02106146  
## 8 -0.20220246 3.50100087 -0.07483086 -0.18919863 3.23523662  
## sports\_playing cooking eco computers business  
## 1 0.10056064 -0.1045525 0.20113505 0.08527873 0.10986672  
## 2 -0.23189096 -0.3242998 -0.17290368 -0.22056362 -0.14703738  
## 3 -0.02790323 0.3778584 0.54068822 -0.06625756 0.07953593  
## 4 0.02910975 -0.1837864 0.18755952 2.88959887 0.55359376  
## 5 -0.08695907 -0.2504689 -0.05835230 -0.19827530 -0.08066612  
## 6 0.11915055 -0.1629270 0.11899003 -0.15024959 0.41722581  
## 7 0.20034251 2.6867046 0.04897007 0.07089596 0.26084461  
## 8 2.07471923 -0.1327845 -0.04725155 -0.09012065 -0.09767463  
## outdoors crafts automotive art religion  
## 1 -0.08188459 0.68362688 0.11849451 -0.021052780 2.21225549  
## 2 -0.32975938 -0.22612614 -0.25185210 -0.235468568 -0.30720507  
## 3 1.62876906 0.05306240 -0.15349796 -0.086522057 -0.17410994  
## 4 -0.03883854 0.21241954 -0.12677206 -0.162205164 0.10385816  
## 5 0.28214013 -0.15514338 2.54644636 -0.175312889 -0.20195616  
## 6 -0.09591099 0.68543839 -0.18441845 2.381751806 -0.00347951  
## 7 0.02545895 0.10536135 0.03726068 -0.002247328 -0.12916572  
## 8 -0.14970455 0.02968489 0.06190603 0.267316090 -0.18946174  
## beauty parenting dating school personal\_fitness  
## 1 0.305945440 2.097901478 0.047234325 1.65058671 -0.11147215  
## 2 -0.270679450 -0.307765427 -0.087044097 -0.26053041 -0.32948234  
## 3 -0.215850225 -0.103591777 0.186560834 -0.15178486 2.07725967  
## 4 -0.182193847 0.008753798 0.368607935 -0.08655386 -0.14112898  
## 5 -0.182323209 0.020347760 -0.018290108 0.01254789 -0.24904831  
## 6 -0.006161053 -0.198057182 -0.047868942 -0.01216216 -0.15881086  
## 7 2.533378051 -0.067382729 0.119065896 0.20358928 -0.04259446  
## 8 -0.233020861 -0.147475858 -0.004293009 -0.19153363 -0.19210751  
## fashion small\_business adult  
## 1 0.02218561 0.09453593 1.348903e-02  
## 2 -0.26196120 -0.14389108 1.964096e-02  
## 3 -0.11033219 -0.10790216 -2.288329e-05  
## 4 -0.15609324 0.40802588 -1.157275e-01  
## 5 -0.22046241 -0.13263537 -8.934443e-02  
## 6 -0.02180280 0.82502146 -2.307904e-02  
## 7 2.59660035 0.20254493 1.301350e-02  
## 8 -0.06539505 0.10904749 -8.615448e-03

Unscale the data and get mu and sigma.

mu = attr(market\_scaled,"scaled:center")  
sigma = attr(market\_scaled,"scaled:scale")  
segments\_unscaled = segments$centers \* sigma + mu

### First Cluster

rbind(segments$centers[1, ], segments\_unscaled[1, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.1023403 -0.1127227 -0.02545927 -0.1070525 2.024939  
## [2,] 1.6561210 0.4376382 1.91090001 0.8904199 4.075406  
## politics food family home\_and\_garden music news  
## [1,] -0.2238752 1.800412 1.456168 0.1577103 0.03227912 -0.1273002  
## [2,] 0.6103146 2.395510 3.097752 0.4338241 1.45480138 1.1806714  
## online\_gaming shopping health\_nutrition college\_uni sports\_playing  
## [1,] -0.08179991 0.06534396 -0.1558373 -0.1331629 0.1005606  
## [2,] 0.71811450 1.11605018 1.3162718 1.9685223 0.5980016  
## cooking eco computers business outdoors crafts  
## [1,] -0.1045525 0.2011351 0.08527873 0.1098667 -0.08188459 0.6836269  
## [2,] 1.2105911 2.0286479 1.54362427 0.9155513 0.67039844 2.2042756  
## automotive art religion beauty parenting dating  
## [1,] 0.1184945 -0.02105278 2.212255 0.3059454 2.097901 0.04723433  
## [2,] 1.5272801 0.40866633 4.653867 3.5324705 5.612761 0.70478452  
## school personal\_fitness fashion small\_business adult  
## [1,] 1.650587 -0.1114721 0.02218561 0.09453593 0.01348903  
## [2,] 3.422570 1.3302309 0.70212073 0.58507318 0.72306310

Clearly, people who fall into the first cluster loves to tweet about online games, college unions, and sports playing. They are more likely to be male students in college who love games and sports.

### Second Cluster

rbind(segments$centers[2, ], segments\_unscaled[2, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] -0.09330702 -0.2220745 -0.1472770 -0.2245890 -0.3147726  
## [2,] 1.37174786 0.4505292 0.3989438 0.4069186 1.1268504  
## politics food family home\_and\_garden music  
## [1,] -0.2838071 -0.3748199 -0.2682235 -0.1346813 -0.1800557  
## [2,] 0.3116018 0.7126138 0.5818051 0.1590893 0.6599428  
## news online\_gaming shopping health\_nutrition college\_uni  
## [1,] -0.3167166 -0.2263096 -0.06882735 -0.3225482 -0.2447176  
## [2,] 0.3302155 0.3560275 0.29379055 0.8247851 0.8404953  
## sports\_playing cooking eco computers business outdoors  
## [1,] -0.2318910 -0.3242998 -0.1729037 -0.2205636 -0.1470374 -0.3297594  
## [2,] 0.5130675 0.4036207 1.2645416 1.5755560 0.3957518 0.6689136  
## crafts automotive art religion beauty parenting  
## [1,] -0.2261261 -0.2518521 -0.2354686 -0.3072051 -0.2706794 -0.3077654  
## [2,] 1.1053717 0.9338065 0.4978742 0.4026593 0.6212882 0.3817215  
## dating school personal\_fitness fashion small\_business  
## [1,] -0.0870441 -0.2605304 -0.3294823 -0.2619612 -0.1438911  
## [2,] 0.3629753 0.2465046 1.7967932 0.6552087 0.4793610  
## adult  
## [1,] 0.01964096  
## [2,] 0.95110285

People in the second cluster loves to talk about food, sports\_fandon, family , crafts, religious, parenting and schooling. They seems to be father with one or two children.

### Third Cluster

rbind(segments$centers[3, ], segments\_unscaled[3, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.01025367 -0.1595277 0.002424409 -0.1554513 -0.2087017  
## [2,] 2.72478547 0.8703990 0.651933316 0.6857753 1.1080100  
## politics food family home\_and\_garden music  
## [1,] -0.1921082 0.4192286 -0.07854314 0.1477726 0.002940299  
## [2,] 0.4813949 0.8349972 0.60085332 1.7137696 0.522846123  
## news online\_gaming shopping health\_nutrition college\_uni  
## [1,] -0.09113928 -0.1156567 0.04981961 2.124562 -0.2078711  
## [2,] 1.68562146 0.8739445 0.49366830 3.270065 0.4363961  
## sports\_playing cooking eco computers business outdoors  
## [1,] -0.02790323 0.3778584 0.5406882 -0.06625756 0.07953593 1.628769  
## [2,] 0.67934561 0.5699079 2.3575109 1.35752145 0.93852361 3.974636  
## crafts automotive art religion beauty parenting  
## [1,] 0.0530624 -0.153498 -0.08652206 -0.1741099 -0.2158502 -0.1035918  
## [2,] 1.9494705 1.877093 0.44518359 1.0432886 1.1275771 1.2019866  
## dating school personal\_fitness fashion small\_business  
## [1,] 0.1865608 -0.1517849 2.07726 -0.1103322 -0.1079022  
## [2,] 1.0083115 0.5873388 4.51601 0.9123164 0.3485336  
## adult  
## [1,] -2.288329e-05  
## [2,] 7.108194e-01

People in the third cluster love to tweet things about travelling, poiltics, news, computers. They seems to be professionals who have to travel a lot.

### Forth Cluster

rbind(segments$centers[4, ], segments\_unscaled[4, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.1178092 3.191867 -0.05757546 -0.07686805 -0.2181462  
## [2,] 1.2657066 9.786859 0.38337880 0.57385466 2.1009090  
## politics food family home\_and\_garden music news  
## [1,] 3.047928 0.1459755 -0.09151735 0.04035434 -0.04574365 1.113599  
## [2,] 7.608558 0.8212451 0.78265817 1.67723484 0.63215259 1.369472  
## online\_gaming shopping health\_nutrition college\_uni sports\_playing  
## [1,] -0.1547610 -0.017923 -0.1600995 -0.0391807 0.02910975  
## [2,] 0.4996434 1.503520 0.4027361 1.8638363 1.15114747  
## cooking eco computers business outdoors crafts  
## [1,] -0.18378638 0.1875595 2.889599 0.5535938 -0.03883854 0.2124195  
## [2,] 0.07004068 1.0762898 3.458030 1.6269389 0.31232804 1.7746510  
## automotive art religion beauty parenting dating  
## [1,] -0.1267721 -0.1622052 0.1038582 -0.1821938 0.008753798 0.3686079  
## [2,] 1.1822016 0.6082688 1.1864700 1.2363821 2.606600155 0.8169554  
## school personal\_fitness fashion small\_business adult  
## [1,] -0.08655386 -0.141129 -0.1560932 0.4080259 -0.1157275  
## [2,] 1.25388230 1.289044 1.1070196 1.2761697 0.6301842

People in this cluster like photo sharing, cooking, shopping, beauty things and fashion. Clearly, they are more likely to be young ladies.

### Fifth Cluster

rbind(segments$centers[5, ], segments\_unscaled[5, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.08895021 -0.1855737 -0.1269545 -0.03844082 0.6355539  
## [2,] 1.78622573 1.0536939 0.6291201 0.72202090 2.1245330  
## politics food family home\_and\_garden music news  
## [1,] 1.190715 -0.1761167 0.2353671 0.1371797 -0.07906987 2.568875  
## [2,] 4.408836 0.3013033 1.1303659 3.0714852 1.03942317 3.678942  
## online\_gaming shopping health\_nutrition college\_uni sports\_playing  
## [1,] -0.1399912 -0.07224083 -0.2634776 -0.1994166 -0.08695907  
## [2,] 0.7092028 1.41989526 0.4078834 0.3588106 0.58967778  
## cooking eco computers business outdoors crafts  
## [1,] -0.2504689 -0.0583523 -0.1982753 -0.08066612 0.2821401 -0.1551434  
## [2,] 1.2084448 0.4776924 1.3181513 0.94094539 0.9149648 0.6881576  
## automotive art religion beauty parenting dating  
## [1,] 2.546446 -0.1753129 -0.2019562 -0.1823232 0.02034776 -0.01829011  
## [2,] 3.123279 0.4391304 0.2114974 1.0737626 1.60843043 0.80487847  
## school personal\_fitness fashion small\_business adult  
## [1,] 0.01254789 -0.2490483 -0.2204624 -0.1326354 -0.08934443  
## [2,] 1.01951719 1.0337386 1.5760111 0.4075160 1.24717030

This group like to talk about home and gardening, health and nutrition, eco-friendly, outdoor activities and personal fitness. They care about health and environment!

### Sixth Cluster

rbind(segments$centers[6, ], segments\_unscaled[6, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.3187616 0.2043046 0.05870943 2.648276 -0.1106166  
## [2,] 2.7548349 3.4858248 0.56381557 7.831817 1.3549784  
## politics food family home\_and\_garden music news  
## [1,] -0.0952232 0.0967449 -0.1244288 0.2980621 1.088913 0.01240974  
## [2,] 1.2171241 0.8996807 0.6198449 1.5647071 4.135246 0.43183508  
## online\_gaming shopping health\_nutrition college\_uni sports\_playing  
## [1,] -0.1735487 0.2035594 -0.1824872 0.4095228 0.1191506  
## [2,] 0.4015362 3.2528020 0.8221662 1.1320869 1.1018956  
## cooking eco computers business outdoors crafts  
## [1,] -0.162927 0.1189900 -0.1502496 0.4172258 -0.09591099 0.6854384  
## [2,] 1.212629 0.8018307 0.3966557 1.2591863 1.40456183 1.0256365  
## automotive art religion beauty parenting dating  
## [1,] -0.1844184 2.381752 -0.00347951 -0.006161053 -0.1980572 -0.04786894  
## [2,] 1.3656794 5.656056 0.39701419 0.856889265 0.4459697 0.64680998  
## school personal\_fitness fashion small\_business adult  
## [1,] -0.01216216 -0.1588109 -0.0218028 0.8250215 -0.02307904  
## [2,] 0.32881795 1.1155102 1.4863137 1.9569694 0.95437648

People from this group love to talk about current events, tv and film, about music, about art and small business. These people are those fantastic artists, musicians, movie makers!

### Seventh Cluster

rbind(segments$centers[7, ], segments\_unscaled[7, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] 0.1763254 -0.05991035 1.279706 -0.1478715 -0.2099234  
## [2,] 1.7105638 1.37591028 2.578137 0.7262044 1.1523308  
## politics food family home\_and\_garden music  
## [1,] -0.1269455 -0.2022049 0.04039369 0.1389957 0.5354292  
## [2,] 1.9964766 0.3506883 1.55922214 1.8943698 2.3578771  
## news online\_gaming shopping health\_nutrition college\_uni  
## [1,] -0.08776632 -0.03405535 0.3216336 -0.06799234 -0.02106146  
## [2,] 0.67651751 0.72723197 1.6038072 1.02610322 0.40866032  
## sports\_playing cooking eco computers business outdoors  
## [1,] 0.2003425 2.686705 0.04897007 0.07089596 0.2608446 0.02545895  
## [2,] 1.0679400 10.035538 1.30840694 0.73269231 1.3166129 1.64319099  
## crafts automotive art religion beauty parenting  
## [1,] 0.1053613 0.03726068 -0.002247328 -0.1291657 2.533378 -0.06738273  
## [2,] 0.7877930 0.54098699 0.702166744 1.3623652 2.386998 1.76710503  
## dating school personal\_fitness fashion small\_business  
## [1,] 0.1190659 0.2035893 -0.04259446 2.596600 0.2025449  
## [2,] 1.3233981 0.7725185 0.81562618 3.172205 1.0548783  
## adult  
## [1,] 0.0130135  
## [2,] 0.3443802

People from this group usually talk about news and automotives. These people probably care more about cars news.

### Eighth Cluster

rbind(segments$centers[8, ], segments\_unscaled[8, ])

## current\_events travel photo\_sharing tv\_film sports\_fandom  
## [1,] -0.05862658 -0.04421146 0.02682619 0.09302734 -0.1325942  
## [2,] 0.79746881 0.59604885 0.76853134 0.39384051 1.1620594  
## politics food family home\_and\_garden music  
## [1,] -0.1622641 -0.09285402 0.2004677 0.06992411 -0.03386766  
## [2,] 1.0793757 0.70301291 1.3631121 2.00058025 2.41496792  
## news online\_gaming shopping health\_nutrition college\_uni  
## [1,] -0.2022025 3.501001 -0.07483086 -0.1891986 3.235237  
## [2,] 0.3506903 9.882827 1.43230837 1.0471370 4.695633  
## sports\_playing cooking eco computers business  
## [1,] 2.074719 -0.1327845 -0.04725155 -0.09012065 -0.09767463  
## [2,] 3.233003 0.8500260 1.08184337 0.36084519 0.53677011  
## outdoors crafts automotive art religion beauty  
## [1,] -0.1497045 0.02968489 0.06190603 0.2673161 -0.1894617 -0.2330209  
## [2,] 2.2878580 1.26789303 0.72208911 1.3264195 1.1519834 0.4392543  
## parenting dating school personal\_fitness fashion  
## [1,] -0.1474759 -0.004293009 -0.1915336 -0.1921075 -0.06539505  
## [2,] 0.3987907 0.699450278 1.2832272 0.3791561 1.77392264  
## small\_business adult  
## [1,] 0.1090475 -0.008615448  
## [2,] 1.3042146 0.387700535

Well, these people don't tweet a lot. But when they tweet, they tweet about adult things. And they seems to have little interest in other topics. So, they are probaly single males.