Solutions for Exercises 1

Vijayakumar Ganapathy

Aug 7, 2015

Question 1: Exploratory Analysis

Objective 1: To find whether voting certain kinds of voting equipment lead to higher rates of undercount.

Process outline:

- Look at the kinds of equipment and their frequency.
- If the equipments are not equally represented or do not have similar number of ballots, then take undercount%
- Look at the **undercount** (Ballots - votes) and undercount_% ((Ballots-votes)/Ballots*100) for all counties, and then look at it grouping by equipment as well

```
georgia = read.csv("georgia2000.csv")
names(georgia)
```

```
##  [1] "county"  "ballots" "votes"   "equip"   "poor"    "urban"   "atlanta"
##  [8] "perAA"   "gore"    "bush"
```

```
table(georgia$equip)
```

```
##
##   LEVER OPTICAL   PAPER   PUNCH
##      74      66       2      17
```

The equipments are definitely not equally represented. Now let us look at the ballots and votes in each equip.

```
agg = aggregate(subset(georgia, select=c(ballots,votes)),
by=list(equip=georgia$equip), FUN=sum)
print(agg)
```

```
##     equip ballots   votes
## 1   LEVER  427780  410764
## 2 OPTICAL 1436159 1397069
## 3   PAPER    3454    3341
## 4   PUNCH  823921  785459
```

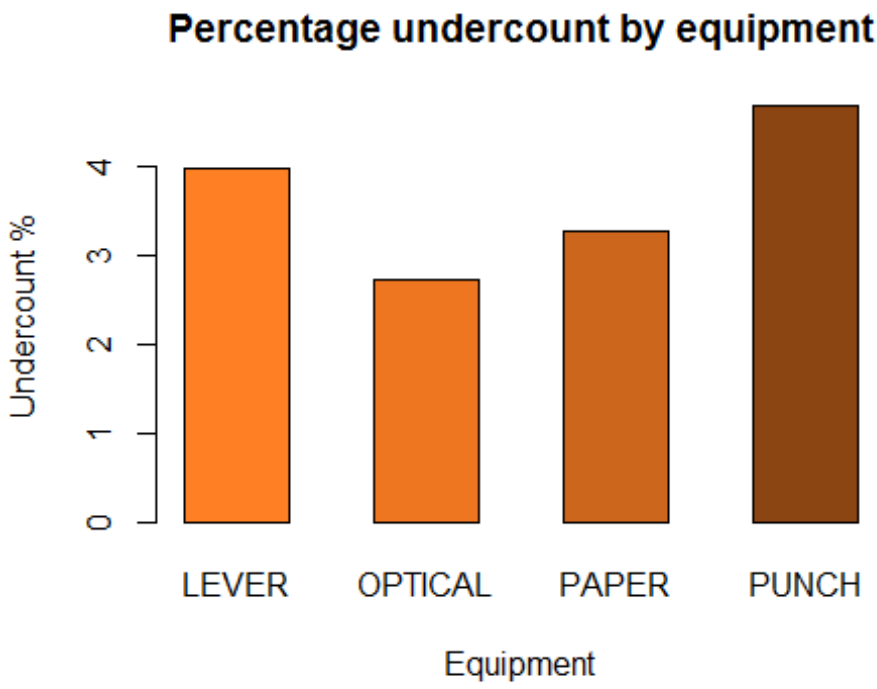
Number of ballots and votes cast using each equipment is very different from each other.

I am creating additional columns on undercount and undercount percentage

```
agg$undercount <- agg$ballots - agg$votes
agg$undercount_percent <- agg$undercount/agg$ballots*100
agg$numer_of_counties <- aggregate(georgia$county, by=list(georgia$equip),
FUN=length)$x
```

Now let us look at the Percentage undercount plots for each equipment

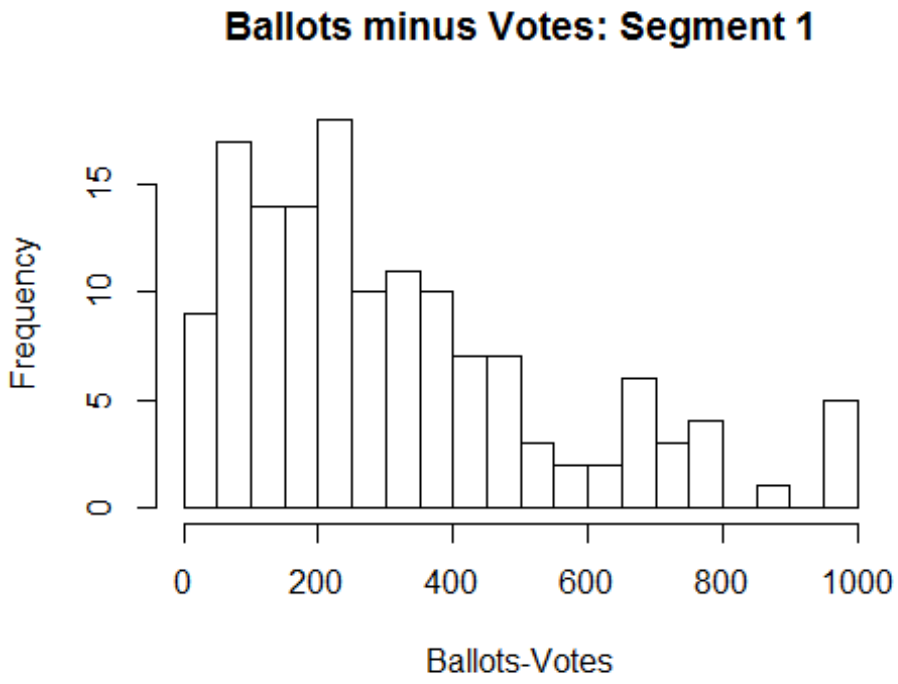
```
##     equip ballots   votes undercount undercount_percent numer_of_counties
## 1   LEVER  427780  410764      17016           3.977746                74
## 2 OPTICAL 1436159 1397069      39090           2.721843                66
## 3   PAPER    3454    3341        113           3.271569                 2
## 4   PUNCH  823921  785459      38462           4.668166                17
```



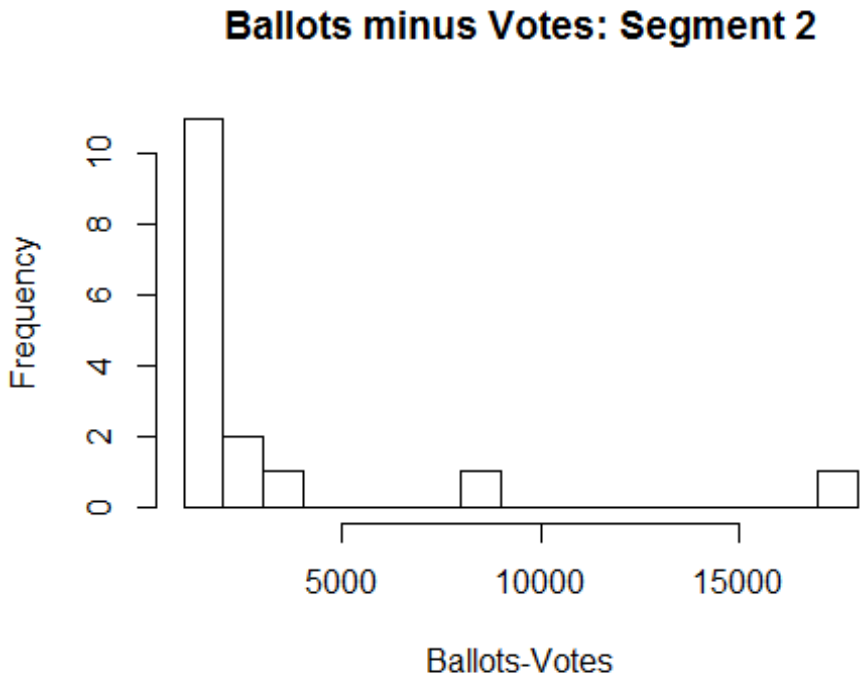
From the plot above, we can see that *Punch* leads the pack with the highest percentage of undercount, followed by Lever, Paper and Optical. However the paper percentage is not significant.

However there are a few areas with very high population which are causing a skew in our analysis. So let us and go ahead and explore the outliers

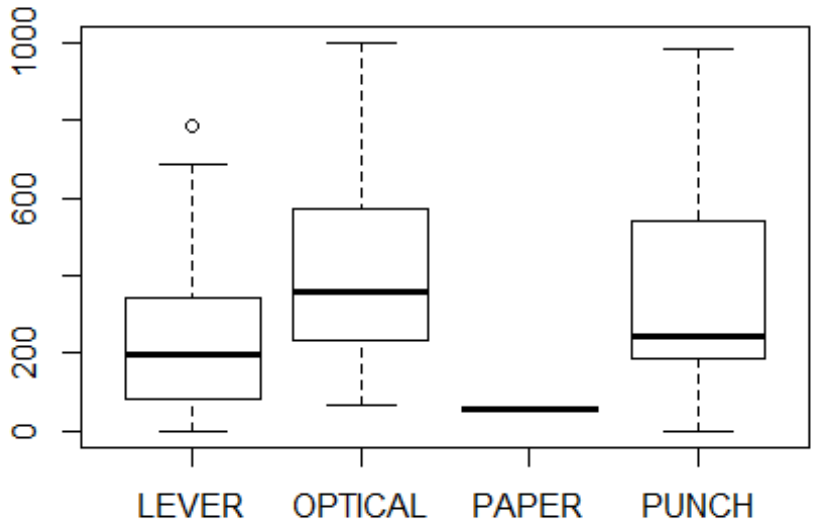
```
georgia$ballots_votes_diff <- georgia$ballots - georgia$votes

hist(georgia$ballots_votes_diff[georgia$ballots_votes_diff<=1000], breaks=20,
main="Ballots minus Votes: Segment 1", xlab="Ballots-Votes")
```




```
hist(georgia$ballots_votes_diff[georgia$ballots_votes_diff>1000], breaks=20,
main="Ballots minus Votes: Segment 2", xlab="Ballots-Votes")
```

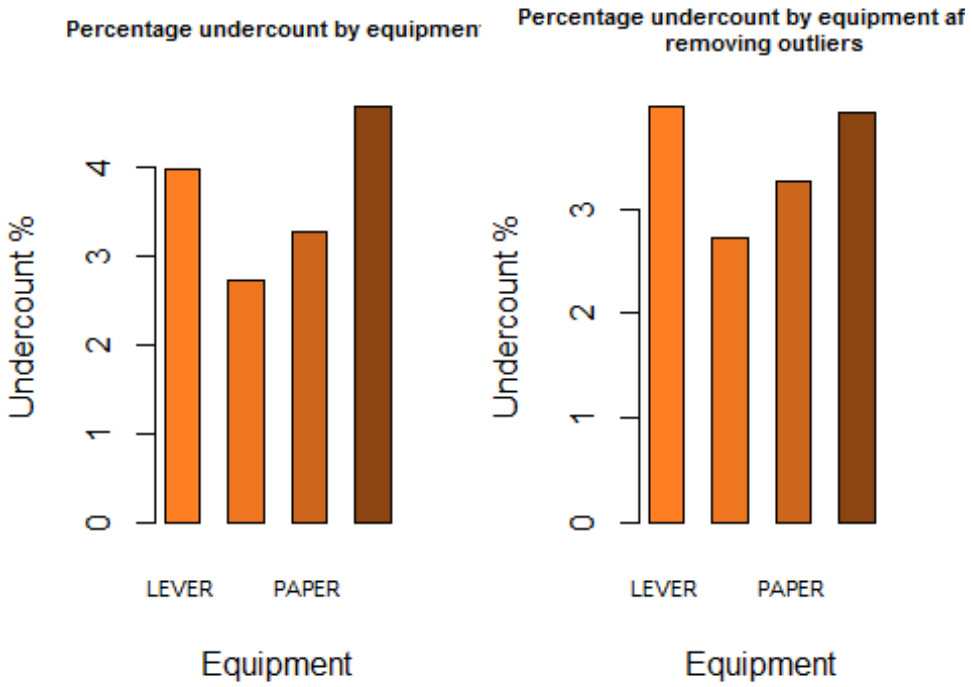


```
plot(georgia$equip[georgia$ballots_votes_diff<=1000],
georgia$ballots_votes_diff[georgia$ballots_votes_diff<=1000])
```



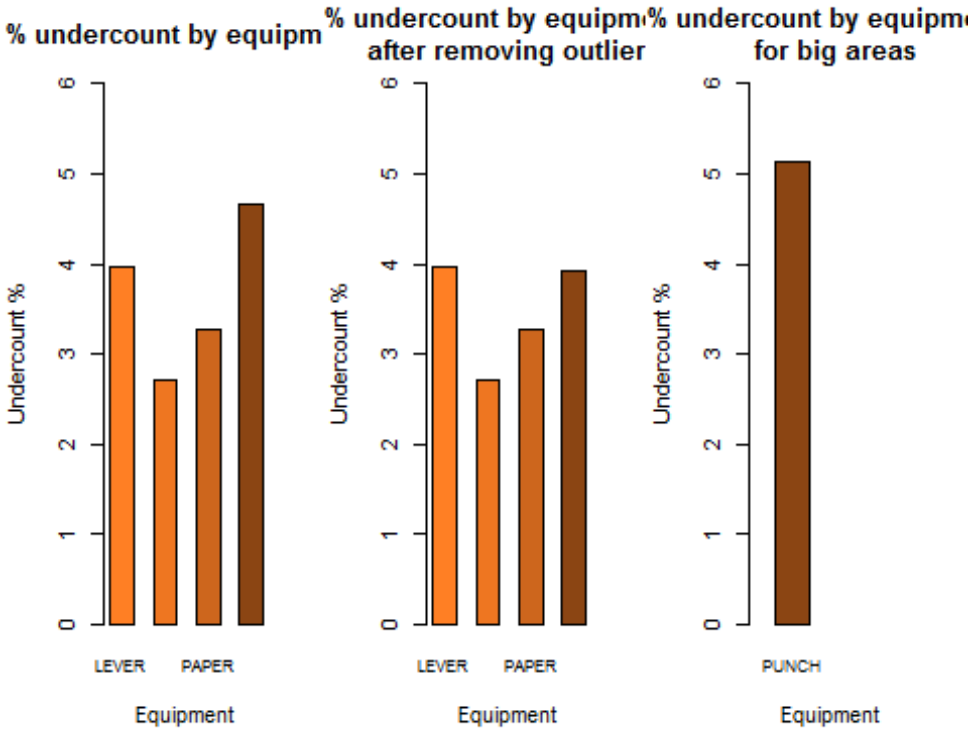
So now, let us remove areas with difference more than 5000, and repeat our initial analyses.

```
agg1 = aggregate(subset(georgia, ballots_votes_diff<=5000,
select=c(ballots,votes)),
by=list(equip=georgia$equip[georgia$ballots_votes_diff<=5000]), FUN=sum)

##     equip ballots   votes undercount undercount_percent numer_of_counties
## 1   LEVER  427780  410764      17016           3.977746                74
## 2 OPTICAL 1436159 1397069      39090           2.721843                66
## 3   PAPER    3454    3341        113           3.271569                 2
## 4   PUNCH  314594  302268      12326           3.918066                15
```



We can see all the outliers (with undercount of more than 5000) are coming from *Punch*. So the bigger areas use Punch. After removing these areas however, we see the undercount % of Punch has decreased. This means that Punch has higher undercount % in areas with more population as can be seen below.

```
##   equip ballots  votes undercount undercount_percent numer_of_counties
## 1 PUNCH  509327 483191      26136           5.131477                 2
```



- Punch and Lever have very similar undercount % after removing outliers.
- Punch has about 5.1% undercount in big areas.
- All equipments had similar undercount % in the range (2.7% to 4.7%) when I don't remove the outliers.
- But if this should be sorted in the decreasing order of undercount % then **Punch > Lever > Optical > Paper**.
- **Conclusion:** Punch has the most undercount %, but other equipments are not very far from Punch

Objective 2: To find whether voting certain kinds of voting equipment effect has a disparate impact of undercount on poor and minority communities.

Process outline:

- Look at poor areas and rich areas and see if there is a bias on undercount
- Look if there is any specific equipment causing the same
- Repeat the same process for minorities

First let us look at Undercount impact on poor and minorities. Poor is just a flag - 0 or 1. So we can directly group numbers based on it. However, perAA is a continuous variable. So I will split them into 10 bins and then look if there are any trends in undercount %

```
# Computations for poor

library(sqldf)
```

```
## Loading required package: gsubfn
## Loading required package: proto
## Loading required package: RSQLite
## Loading required package: DBI
```

```
poor_undercount=sqldf('select poor, sum(ballots) as Ballots, sum(votes) as
Votes, count(*) number_of_counties, sum(ballots)-sum(votes) undercount from
georgia group by 1')
```

```
## Loading required package: tcltk
```

```
poor_undercount$undercount_percent <-
poor_undercount$undercount/poor_undercount$Ballots*100
poor_undercount
```

```
##   poor Ballots   Votes number_of_counties undercount undercount_percent
## 1    0 2330529 2255047                 87      75482           3.238835
## 2    1  360785  341586                 72      19199           5.321452
```

```
# Computations for minorities
```

```
library(dplyr)
```

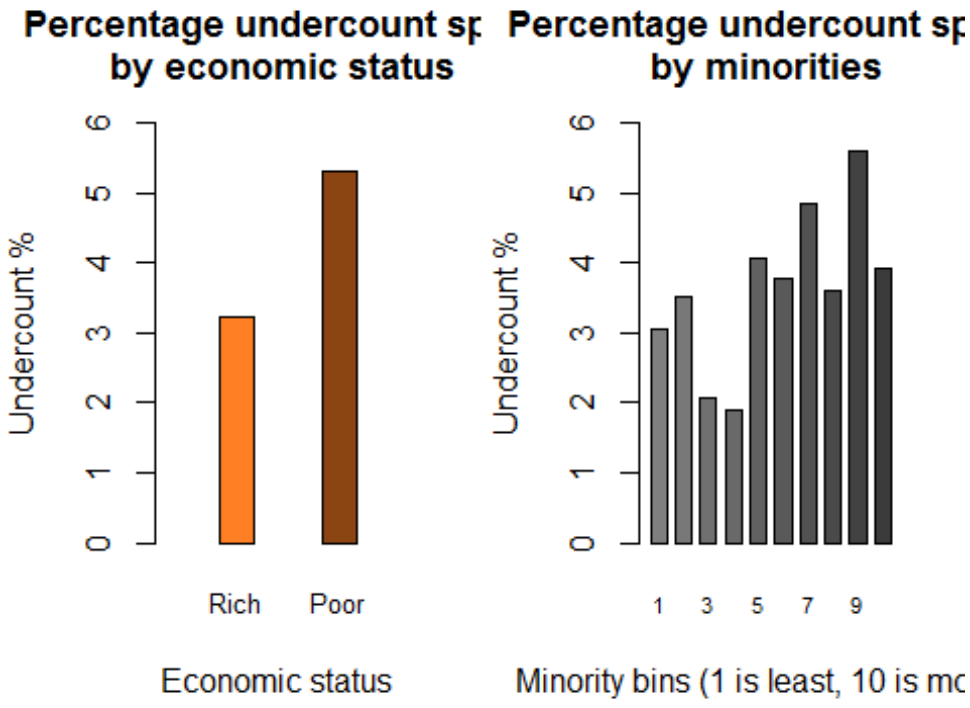
```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
georgia$decile_minority <- ntile(georgia$perAA, 10)
minority = sqldf('select decile_minority, sum(ballots) as Ballots, sum(votes)
as Votes, count(*) number_of_counties, sum(ballots)-sum(votes) undercount
from georgia group by 1')
minority$undercount_percent <- minority$undercount/minority$Ballots*100
minority
```

```
##    decile_minority Ballots  Votes number_of_counties undercount
## 1                1  200047 193915                 16       6132
## 2                2  217032 209380                 16       7652
## 3                3  448810 439452                 16       9358
## 4                4  453182 444599                 16       8583
## 5                5  130224 124924                 16       5300
## 6                6  114259 109934                 16       4325
## 7                7  104147  99095                 16       5052
## 8                8  155039 149471                 16       5568
## 9                9  513402 484587                 16      28815
## 10              10  355172 341276                 15      13896
```

```
##    undercount_percent
## 1            3.065280
## 2            3.525747
## 3            2.085069
## 4            1.893941
## 5            4.069910
## 6            3.785260
## 7            4.850836
## 8            3.591354
## 9            5.612561
## 10           3.912471
```

```r
# plots for both Poor and minorities

par(mfrow=c(1,2))

barplot(poor_undercount$undercount_percent, cex.names = 0.8, cex.main= 1.1,
space=2, main="Percentage undercount split \n by economic status",
xlim=c(0.4,7), ylim=c(0,6), xlab="Economic status", ylab="Undercount %",
col=c("chocolate1","chocolate4"), names.arg=c("Rich","Poor"))

barplot(minority$undercount_percent, cex.names = 0.7, cex.main= 1.1,
space=0.5, main="Percentage undercount split \n by minorities",
xlim=c(0.4,14), ylim=c(0,6), xlab="Minority bins (1 is least, 10 is most)",
ylab="Undercount %",
col=c("grey50","grey47","grey44","grey41","grey38","grey35","grey32","grey29"
,"grey26","grey23"), names.arg=minority$decile_minority)
```



Inferences from the above plots:

- **Poor:** There is a clear demarcation between the undercount % of poor (~3.2%) and rich people (~5.2%). So undercount has a higher impact on Poor. In the next set of analysis, let us see if any equipment is responsible.
- **Minorities:** The decile bins of minorities do not rank order on undercount %. However we do see that the top 6 bins (that is from 5 to 10) have higher undercount %. In the next set of analysis, let us see if any equipment is causing this.

```
# Computations for poor

equip_poor_undercount=sqldf('select equip, poor, sum(ballots) as Ballots,
sum(votes) as Votes, count(*) number_of_counties, sum(ballots)-sum(votes)
undercount from georgia group by 1,2')
equip_poor_undercount$undercount_percent <-
equip_poor_undercount$undercount/equip_poor_undercount$Ballots*100
equip_poor_undercount
```

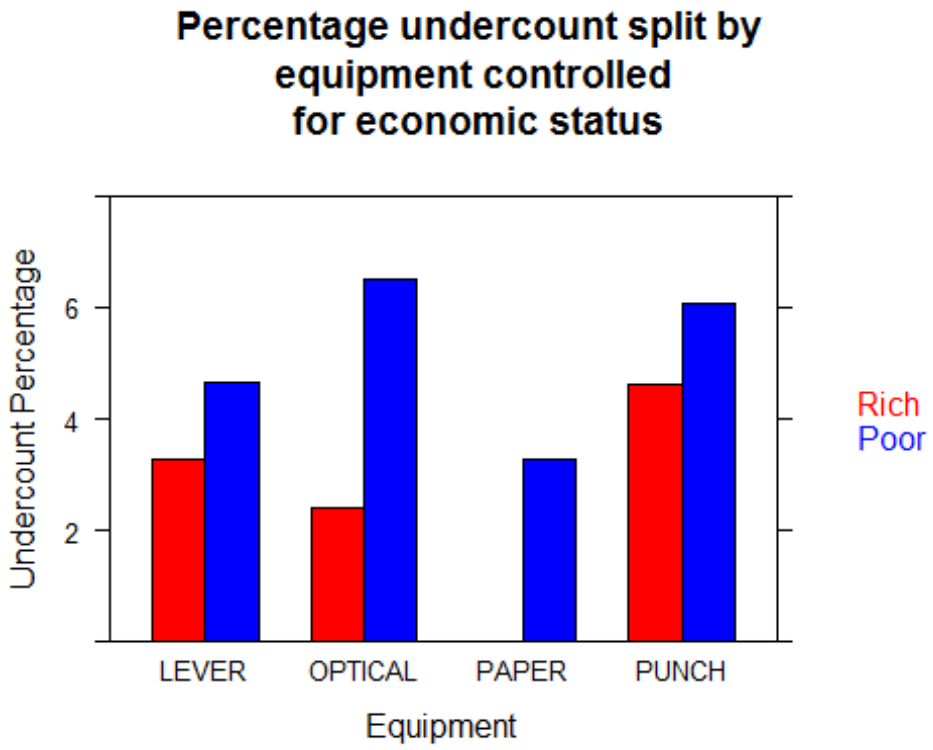
```
##     equip poor Ballots   Votes number_of_counties undercount
## 1   LEVER    0  208526  201710                 29       6816
## 2   LEVER    1  219254  209054                 45      10200
## 3 OPTICAL    0 1321694 1290061                 48      31633
## 4 OPTICAL    1  114465  107008                 18       7457
## 5   PAPER    1    3454    3341                  2        113
## 6   PUNCH    0  800309  763276                 10      37033
## 7   PUNCH    1   23612   22183                  7       1429
##   undercount_percent
## 1           3.268657
## 2           4.652139
## 3           2.393368
## 4           6.514655
## 5           3.271569
## 6           4.627338
## 7           6.052007
```

```
# Computations for minorities

equip_minority = sqldf('select equip, case when decile_minority in
(1,2,3,4,5) then "Bottom 5 bins" when decile_minority in (6,7,8,9,10) then
"Top 5 bins" end as Minority_Bin, sum(ballots) as Ballots, sum(votes) as
Votes, count(*) number_of_counties, sum(ballots)-sum(votes) undercount from
georgia group by 1,2')
equip_minority$undercount_percent <-
equip_minority$undercount/equip_minority$Ballots*100
equip_minority
```

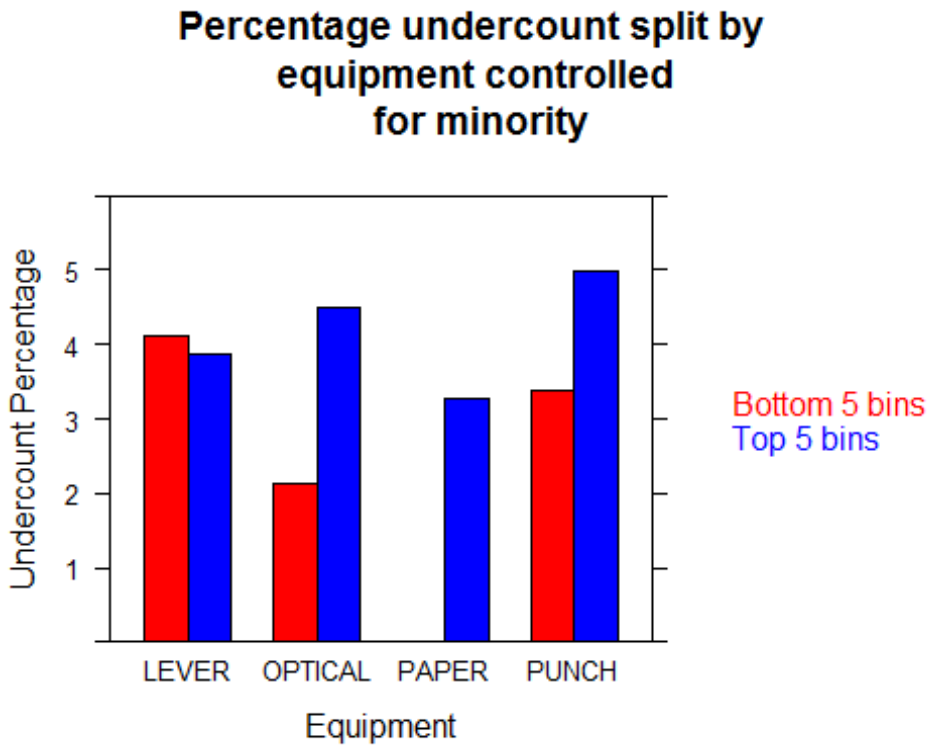
```
##     equip  Minority_Bin Ballots   Votes number_of_counties undercount
## 1   LEVER Bottom 5 bins  203319  194973                 32       8346
## 2   LEVER    Top 5 bins  224461  215791                 42       8670
## 3 OPTICAL Bottom 5 bins 1079207 1056148                 42      23059
```

```
## 4 OPTICAL    Top 5 bins  356952  340921                  24       16031
## 5   PAPER    Top 5 bins    3454    3341                   2         113
## 6   PUNCH Bottom 5 bins  166769  161149                   6        5620
## 7   PUNCH    Top 5 bins  657152  624310                  11       32842
##   undercount_percent
## 1           4.104880
## 2           3.862586
## 3           2.136661
## 4           4.491080
## 5           3.271569
## 6           3.369931
## 7           4.997626
```

```r
# plots for both Poor and minorities


library(lattice)

par(mfrow=c(1,1))

barchart(undercount_percent~equip, data=equip_poor_undercount, groups=poor,
xlab="Equipment", ylim=c(0,8), ylab="Undercount Percentage", main="Percentage
undercount split by \n equipment controlled \n for economic status",
col=c("red","blue"), key=list(space="right", text=list(c("Rich","Poor")),
col=c("red","blue")))
```



```
barchart(undercount_percent~equip, data=equip_minority, groups=Minority_Bin,
xlab="Equipment", ylim=c(0,6), ylab="Undercount Percentage", main="Percentage
undercount split by \n equipment controlled \n for minority",
col=c("red","blue"), key=list(space="right", text=list(c("Bottom 5 bins","Top
5 bins")), col=c("red","blue")))
```



Inferences from the above plots:

- **Poor:** From the economic status plot, we can see all equipments have more undercount impact on poor people. But the one with clearly the most impact is optical. This can be because **poor people might have trouble recording their votes through optical method.**
- **Minorities:** The top 5 bins (more minorities) and bottom 5 bins (less minorities) were grouped and then they were plotted for undercount % grouped by equipments. From the plots we can see that, top 5 bins (top 50 percentile) of counties suffer more impact of undercounting in all methods - Optical, Paper and Punch. Punch has the most impact in top 5 counties, but optical has most relative impact compared to bottom 5 bins counties. Again, on similar lines, **some African American people might not know to vote via optical method**

Conclusion:

- Under count seems to be an issue in almost all the counties.
- Most of the counties seem to use Optical and Lever.

- Optical seems to have the most undercount in absolute terms, however Punch has the most undercount % among all the equipments.
- There are only a handful of counties more than 1000 undercount. Majority of these counties use Punch. Punch has higher undercount rate in these counties as well.
- For both poor people and minority communities, Optical seems to cause the most impact in terms of undercount.

Question 2: Bootstrapping

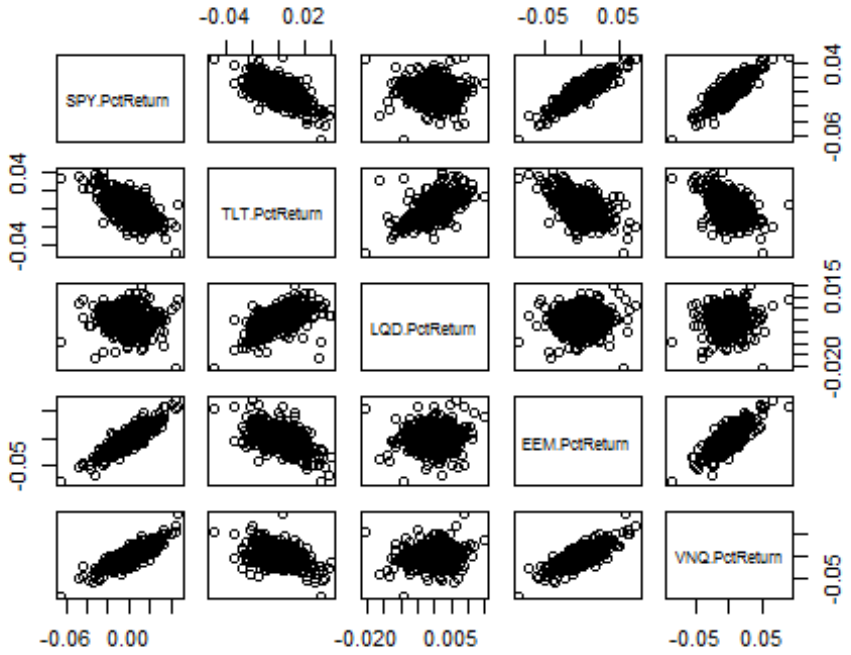
Objective: To explore the financial data from Yahoo and come to an understanding of the risk/return properties of the five ETFs - SPY, TLT, LQD, EEM, VNQ - and to build safe and risky portfolios from these ETFs.

```
library(mosaic)
library(fImport)
library(foreach)

# Import a few stocks
mystocks = c("SPY","TLT","LQD","EEM","VNQ")
myprices = yahooSeries(mystocks, from='2010-01-01', to='2014-12-31')

# The first few rows
head(myprices)
```

```
## GMT
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adj.Close
## 2010-01-04   112.37   113.39  111.51    113.33  118944600      101.4450
## 2010-01-05   113.26   113.68  112.85    113.63  111579900      101.7135
## 2010-01-06   113.52   113.99  113.43    113.71  116074400      101.7852
## 2010-01-07   113.50   114.33  113.18    114.19  131091100      102.2148
## 2010-01-08   113.89   114.62  113.66    114.57  126402800      102.5550
## 2010-01-11   115.08   115.13  114.24    114.73  106375700      102.6982
##            TLT.Open TLT.High TLT.Low TLT.Close TLT.Volume TLT.Adj.Close
## 2010-01-04    89.84    90.10   89.58     89.81    2829100      75.15839
## 2010-01-05    90.05    90.63   90.00     90.39    2841600      75.64378
## 2010-01-06    90.17    90.26   89.12     89.18    4099600      74.63117
## 2010-01-07    89.22    89.64   89.12     89.33    2793200      74.75670
## 2010-01-08    89.51    89.56   88.76     89.29    2910700      74.72323
## 2010-01-11    88.99    89.36   88.77     88.80    2181300      74.31317
##            LQD.Open LQD.High LQD.Low LQD.Close LQD.Volume LQD.Adj.Close
## 2010-01-04   104.77   104.77  104.34    104.70    2017600      83.95245
## 2010-01-05   104.98   105.45  104.86    105.20    1143800      84.35337
## 2010-01-06   105.39   105.45  104.82    104.89    1005500      84.10480
## 2010-01-07   104.97   105.22  104.87    105.02    1264100      84.20904
## 2010-01-08   105.14   105.25  104.97    105.25     704600      84.39346
## 2010-01-11   105.00   105.38  105.00    105.36     817500      84.48166
##            EEM.Open EEM.High EEM.Low EEM.Close EEM.Volume EEM.Adj.Close
```

```
## 2010-01-04    42.18    42.74   42.16     42.71   70761600      38.47913
## 2010-01-05    42.91    43.17   42.76     43.02   50196300      38.75843
## 2010-01-06    43.08    43.29   43.01     43.11   50670000      38.83951
## 2010-01-07    42.85    42.98   42.62     42.86   41803700      38.61428
## 2010-01-08    42.88    43.22   42.74     43.20   41118100      38.92060
## 2010-01-11    43.45    43.47   42.90     43.11   42546400      38.83951
##            VNQ.Open VNQ.High VNQ.Low VNQ.Close VNQ.Volume VNQ.Adj.Close
## 2010-01-04    45.22    45.42   44.20     44.55    2408400      36.26420
## 2010-01-05    44.50    44.58   43.93     44.50    2054200      36.22350
## 2010-01-06    44.52    44.84   44.30     44.42    2471200      36.15838
## 2010-01-07    44.46    45.10   43.95     44.90    2091700      36.54911
## 2010-01-08    44.81    44.85   44.18     44.57    2682000      36.28048
## 2010-01-11    44.87    45.06   44.58     44.83    1924800      36.49213
```

```
# Helper function from portfolio.R

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))
}

# Compute the returns from the closing prices
myreturns = YahooPricesToReturns(myprices)

# These returns can be viewed as draws from the joint distribution
pairs(myreturns)
```

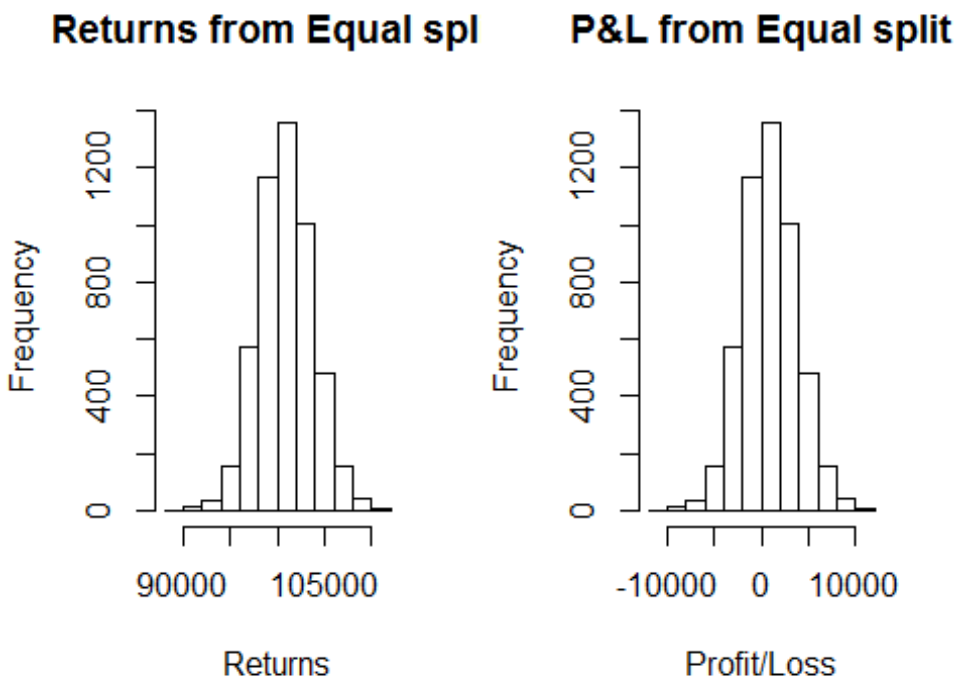



```
# Now simulate many different possible trading years!
set.seed(23432)
sim_all = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_all = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth        # to ensure portfolio is
rebalanced                                  # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_all[today] = totalwealth
  }
  wealthtracker_all
}

head(sim_all)
```

```
##                [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1  97173.32  96990.73  97007.15  96648.56  97718.09  97308.83
## result.2  99893.94 100230.53 100090.64  99953.28 100342.77 100945.31
## result.3 100164.24  99602.58  98933.71  99066.52  99438.85  98028.40
```

```
## result.4  99238.59  98180.29  98680.24 100154.30 101019.81 100939.19
## result.5 100691.59 100750.87 101818.11 102291.89 101467.16 102512.12
## result.6 100287.16 100718.20 101213.79 101391.34 100709.95 100849.20
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1  96700.80  96634.72  96333.92  96463.24  96964.00  96411.18
## result.2 101650.42 100804.92 100452.15 100670.86 101759.74 101639.18
## result.3  98277.13  97362.02  97431.26  97347.38  96719.43  96687.42
## result.4 100216.54  99554.19  99635.53 100196.09 100439.05 100346.93
## result.5 101917.44 101760.46 101793.11 101141.70 101380.05 101356.03
## result.6 101100.89 101471.21 101174.67 100966.47 101286.80 100625.95
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1  97282.45  98370.20  95949.95  95980.91  96472.06  96082.92
## result.2 102139.78 101191.15 101371.80 101386.10 102555.03 103780.95
## result.3  95743.78  95448.45  96161.53  96363.27  96612.10  96757.24
## result.4 100064.33 100326.54 100367.67 100290.68 100346.02 100438.79
## result.5 102567.62 102549.23 103594.13 103035.77 103156.43 103346.65
## result.6 100302.88  99835.36 100030.95 100700.47 101080.41 101292.34
##              [,19]     [,20]
## result.1  96192.07  95886.54
## result.2 104532.56 104802.48
## result.3  96250.58  95892.70
## result.4 100441.53 100516.35
## result.5 103428.63 103293.46
## result.6 101781.30 101376.76
```

```r
par(mfrow=c(1,2))

hist(sim_all[,n_days], xlab='Returns', main='Returns from Equal split')

# Profit/loss
hist(sim_all[,n_days]- 100000, xlab='Profit/Loss', main='P&L from Equal
split')
```




```
# Calculate 5% value at risk
Risk_all_equal = quantile(sim_all[,n_days], 0.05) - 100000
print("Overall Risk when we invest 20% in each is:")

## [1] "Overall Risk when we invest 20% in each is:"

print(Risk_all_equal)

##        5%
## -3709.551
```

The equal split seems to have a risk of about -$3710. The way this risk can be read is like this - if 100 customers invest $100000 in this stock, 5 customers will encounter a loss of $3710 or more. In other words, -$3710 is the 5 percentile point for profit.

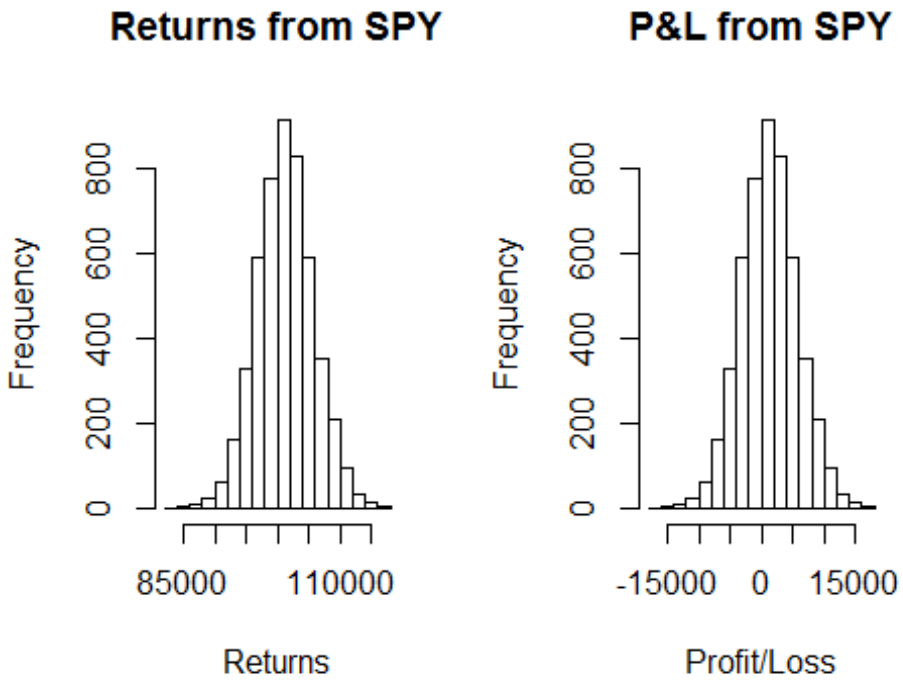
As far as returns are concerned, from the histograms we can see that we get profits in the range of (-$10000, $10000) which seems like a pretty good variance. We will get a better idea of how good this is when we look at individual risks and returns too. (please note profits = Total Returns - Investment, Investment is $100000 in all cases here. So I will be using ranges of profits to characterize the returns)

Now using a very similar procedure, let us calculate the individual risk for each ticker

- **For SPY:**

```
set.seed(23432)
sim_SPY = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
```

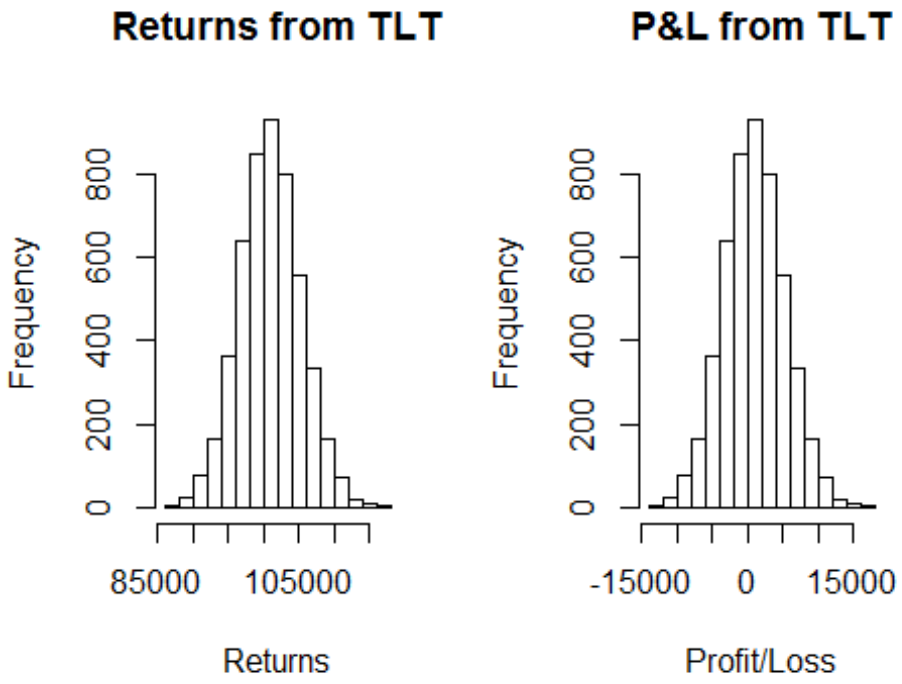
```
  weights = c(1, 0, 0, 0, 0)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_SPY = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_SPY[today] = totalwealth
  }
  wealthtracker_SPY
}

# Calculate 5% value at risk
Risk_SPY = quantile(sim_SPY[,n_days], 0.05) - 100000

par(mfrow=c(1,2))

hist(sim_SPY[,n_days], xlab='Returns', main='Returns from SPY')

# Profit/loss
hist(sim_SPY[,n_days]- 100000, xlab='Profit/Loss', main='P&L from SPY')
```



Now when we look at SPY, both the risk (-$6124) and the returns (-$15000, $15000) seem to be much higher than what we saw for the equal split

- **For TLT:**

```
set.seed(23432)
sim_TLT = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0, 1, 0, 0, 0)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_TLT = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_TLT[today] = totalwealth
  }
  wealthtracker_TLT
}

# Calculate 5% value at risk
Risk_TLT = quantile(sim_TLT[,n_days], 0.05) - 100000

par(mfrow=c(1,2))

hist(sim_TLT[,n_days], xlab='Returns', main='Returns from TLT')

# Profit/Loss
hist(sim_TLT[,n_days]- 100000, xlab='Profit/Loss', main='P&L from TLT')
```

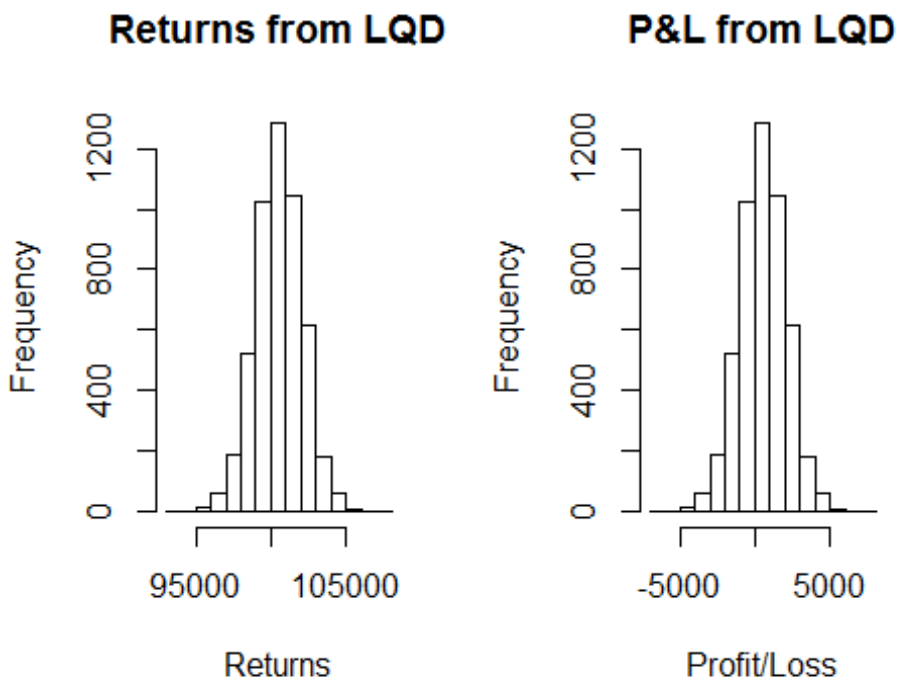


TLT has similar slightly higher risk (-$6124) than SPY but the returns (-$15000, $15000) are about the same. The tails on both the sides have similar lengths for SPY and TLT.

- **For LQD:**

```
set.seed(23432)
sim_LQD = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0, 0, 1, 0, 0)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_LQD = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_LQD[today] = totalwealth
  }
  wealthtracker_LQD
}

# Calculate 5% value at risk
Risk_LQD = quantile(sim_LQD[,n_days], 0.05) - 100000
```

```r
par(mfrow=c(1,2))

hist(sim_LQD[,n_days], xlab='Returns', main='Returns from LQD')

# Profit/loss
hist(sim_LQD[,n_days]- 100000, xlab='Profit/Loss', main='P&L from LQD')
```

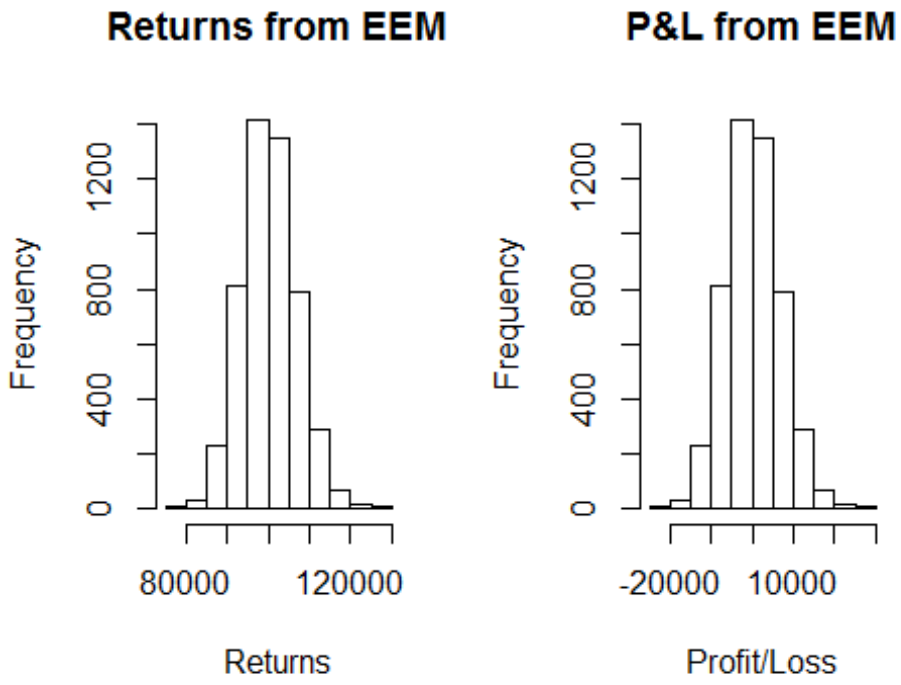


LQD seems to be the safest ETF so far with a risk of -$2032. Their returns are also pretty less ranging from -$5000 to $5000.

- **For EEM:**

```r
set.seed(23432)
sim_EEM = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0, 0, 0, 1, 0)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_EEM = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
```

```
    totalwealth = sum(holdings)
    wealthtracker_EEM[today] = totalwealth
  }
  wealthtracker_EEM
}

# Calculate 5% value at risk
Risk_EEM = quantile(sim_EEM[,n_days], 0.05) - 100000

par(mfrow=c(1,2))

hist(sim_EEM[,n_days], xlab='Returns', main='Returns from EEM')

# Profit/Loss
hist(sim_EEM[,n_days]- 100000, xlab='Profit/Loss', main='P&L from EEM')
```

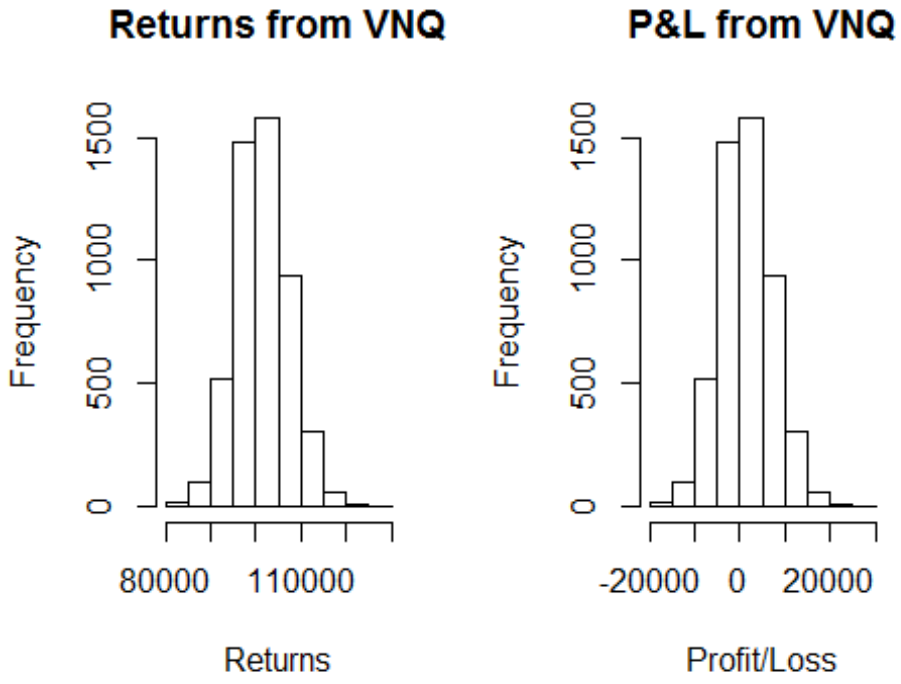


EEM has the highest risk among the ones we have seen (-$10059) and the returns are also well spread (-$25000, $30000). This hence will be one of those aggressive options.

- **For VNQ:**

```
set.seed(23432)
sim_VNQ = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0, 0, 0, 0, 1)
  n_days = 20
  holdings = weights * totalwealth
```

```
  wealthtracker_VNQ = rep(0, n_days) # Set up a placeholder to track total
wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                          # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_VNQ[today] = totalwealth
  }
  wealthtracker_VNQ
}

# Calculate 5% value at risk
Risk_VNQ = quantile(sim_VNQ[,n_days], 0.05) - 100000

par(mfrow=c(1,2))

hist(sim_VNQ[,n_days], xlab='Returns', main='Returns from VNQ')

# Profit/Loss
hist(sim_VNQ[,n_days]- 100000, xlab='Profit/Loss', main='P&L from VNQ')
```



VNQ has high risk (-$7606) which is higher than all others except EEM. The returns (-$20000, $25000) also has high variance.

Based on our findings from the above individual investments, let us now build a conservative and an aggressive portfolio. Before doing that however, we can look at all the individual risks at once.

```
## [1] "Overall Risk when we invest completely in SPY is:"

##        5%
## -6124.255

## [1] "Overall Risk when we invest completely in TLT is:"

##        5%
## -6127.355

## [1] "Overall Risk when we invest completely in LQD is:"

##        5%
## -2032.162

## [1] "Overall Risk when we invest completely in EEM is:"

##        5%
## -10058.54

## [1] "Overall Risk when we invest completely in VNQ is:"

##        5%
## -7606.152
```

As we can see from above, the tickers in *descending* order of risk based on my simulations are - **EEM > VNQ > TLT > SPY > LQD**.

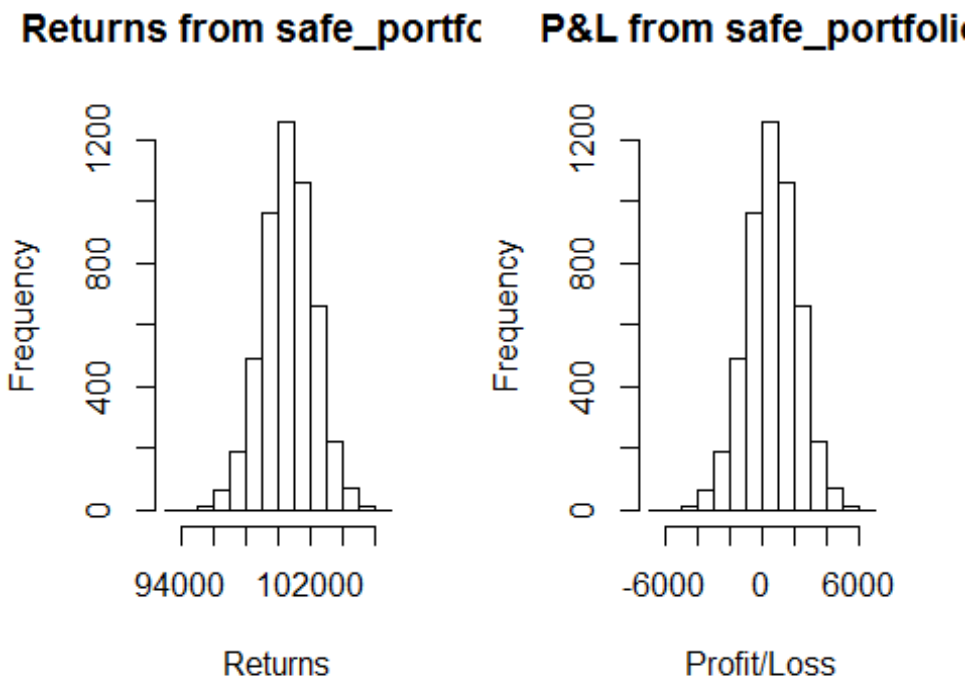
So I am going to try the following combinations:

- **for a safe/conservative portfolio:** LQD = 85% ; SPY = 10% ; TLT = 5% ; Please note that LQD which has the minimum Loss/Risk has been given 85% making this combination a safe bet.
- **for a risky/aggressive portfolio:** EEM = 65% ; VNQ = 25% ; TLT = 10% ; Please note that EEM which has the maximum Loss/Risk has been given 65%. However, since VNQ's risk is not far behind, it has been given 25% as well. This hence will constitute a risky bet!

Let us look at how the **safe portfolio** fares first

```
set.seed(23432)
sim_safe_portfolio = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0.05, 0.10, 0.85, 0, 0)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_safe_portfolio = rep(0, n_days) # Set up a placeholder to
track total wealth
  for(today in 1:n_days) {
```

```
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_safe_portfolio[today] = totalwealth
  }
  wealthtracker_safe_portfolio
}

par(mfrow=c(1,2))

hist(sim_safe_portfolio[,n_days], xlab='Returns', main='Returns from
safe_portfolio')

# Profit/loss
hist(sim_safe_portfolio[,n_days]- 100000, xlab='Profit/Loss', main='P&L from
safe_portfolio')
```



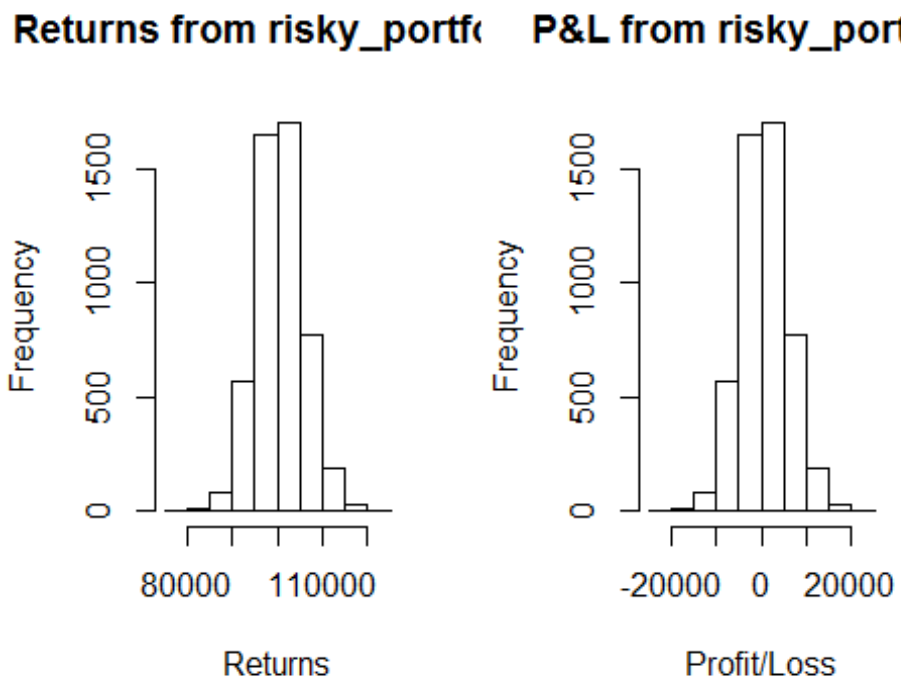

```
# Calculate 5% value at risk
Risk_safe_portfolio = quantile(sim_safe_portfolio[,n_days], 0.05) - 100000
print(Risk_safe_portfolio)
```

```
##        5%
## -2081.835
```

We can see that the risk of -$2350 is only slightly more risky than the least risky contributor LQD and still less risky than other tickers. Their returns are again in the lower range (-$6000, $6000).

Let us look at how the **risky portfolio** fares now!

```
set.seed(23432)
sim_risky_portfolio = foreach(i=1:5000, .combine='rbind') %do% {
  totalwealth = 100000
  weights = c(0, 0.10, 0, 0.65, 0.25)
  n_days = 20
  holdings = weights * totalwealth
  wealthtracker_risky_portfolio = rep(0, n_days) # Set up a placeholder to
track total wealth
  for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = weights * totalwealth         # to ensure portfolio is
rebalanced                                              # at zero transaction
cost
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker_risky_portfolio[today] = totalwealth
  }
  wealthtracker_risky_portfolio
}

par(mfrow=c(1,2))

hist(sim_risky_portfolio[,n_days], xlab='Returns', main='Returns from
risky_portfolio')

# Profit/loss
hist(sim_risky_portfolio[,n_days]- 100000, xlab='Profit/Loss', main='P&L from
risky_portfolio')
```




```
# Calculate 5% value at risk
Risk_risky_portfolio = quantile(sim_risky_portfolio[,n_days], 0.05) - 100000
print(Risk_risky_portfolio)
```

```
##        5%
## -7631.339
```

We can see that the risk of -$7133 is pretty high even though it is pretty lower than the most risky contributor EEM. It is only slightly more risky than VNQ, simply because we gave VNQ 25%, TLT 10% and EEM only 65%. We can very clearly see that the returns are also high (-20000, 20000) but not as high as the most aggressive individual stock - EEM.

Conclusion:

Comparing and summarizing the three portfolios we have built:

1. **Equal split:**
   - 5 percentile risk: -$3710
   - Profit window: -$10000 to $10000
2. **Safe/Conservative split:**
   - 5 percentile risk: -$2350
   - Profit window: -$6000 to $6000
3. **Risky/Aggressive split:**
   - 5 percentile risk: -$7133
   - Profit window: -$20000 to $20000

Question 3: Clustering and PCA

Objective: To apply unsupervised learning techniques to "unsupervised" information available from the wine data (i.e. their chemical properties) and use the color and quality data available to validate these techniques.

```
wine = read.csv("wine.csv")
dim(wine)
```

```
## [1] 6497   13
```

```
head(wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality color
## 1       5   red
## 2       5   red
## 3       5   red
## 4       6   red
## 5       5   red
## 6       5   red
```

```
par(mfrow=c(1,1))
```

Let us extract just the 11 chemical properties to run PCA and Clustering (I have chosen k-means clustering for this problem)

```
wine_chem <- wine[,c(1:11)]
names(wine_chem)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"
```

```
# lets scale this data

wine_chem <- scale(wine_chem,center = TRUE, scale=TRUE)
```

The chemical properties have been extracted and have been scaled. Now let us run K-Means clustering first on this.

To check if this method is capable of distinguishing reds from whites, we need to choose k=2, since we want to cluster this data into 2 groups.

```
set.seed(23432)
cluster_wine_chem <- kmeans(wine_chem, centers=2, nstart=50)
cluster_wine_chem$centers
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1    -0.2804833       -0.3953082   0.1143429      0.1998380 -0.3119753
## 2     0.8286464        1.1678795  -0.3378091     -0.5903919  0.9216848
##   free.sulfur.dioxide total.sulfur.dioxide    density         pH
## 1           0.2814861            0.4018607 -0.2306934 -0.1920315
## 2          -0.8316090           -1.1872380  0.6815493  0.5673286
##    sulphates     alcohol
## 1 -0.2853595  0.02562065
## 2  0.8430523 -0.07569241
```

Now let us extract color from the original data and see how the clustering and the color are related

```
wine_color <- wine$color
table(wine_color, cluster_wine_chem$cluster)
```

```
##
## wine_color    1    2
##      red     24 1575
##      white 4830   68
```

The above table can be read as this:

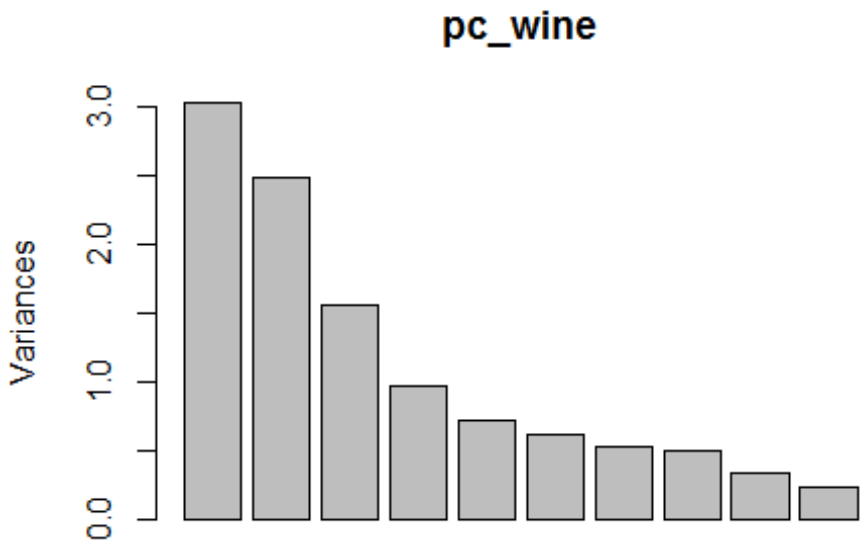
- Using the chemical properties of the wine data, the K-Means clustering algorithm with K=2, groups my data set into two clusters - 1 and 2.
- Cluster 1 has 4830 whites and 24 reds. Cluster 2 has 1575 reds and 68 whites.
- This means that the Cluster 1 is majorly white and Cluster 2 is majorly reds, implying the reds are distinguished from whites to a fairly good extent.
- However, let us try out PCA as well before we conclude on how good this method is!

```
#Let us run PCA on the chemical properties

pc_wine <- prcomp(wine_chem, scale=TRUE)
summary(pc_wine)
```

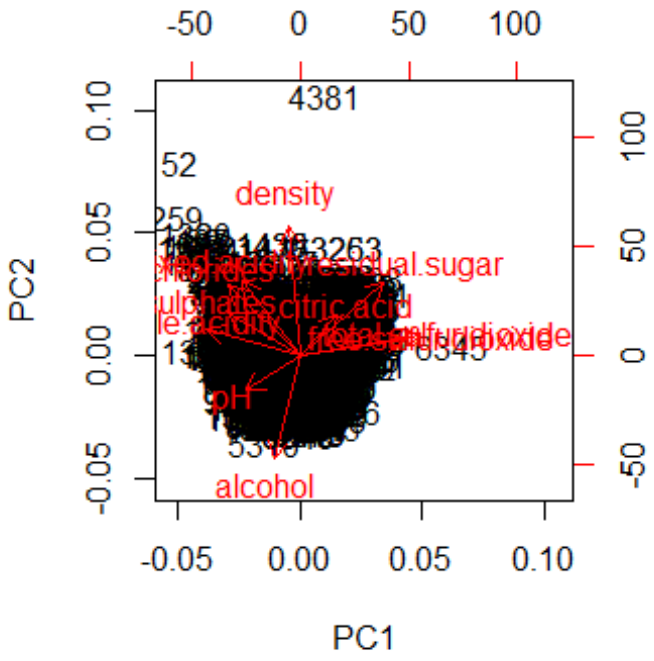
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
```

```
## Standard deviation     1.7407 1.5792 1.2475 0.98517 0.84845 0.77930
## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521
## Cumulative Proportion  0.2754 0.5021 0.6436 0.73187 0.79732 0.85253
##                            PC7     PC8     PC9   PC10    PC11
## Standard deviation     0.72330 0.70817 0.58054 0.4772 0.18119
## Proportion of Variance 0.04756 0.04559 0.03064 0.0207 0.00298
## Cumulative Proportion  0.90009 0.94568 0.97632 0.9970 1.00000
```

```
loadings=pc_wine$rotation
score=pc_wine$x
plot(pc_wine)
```

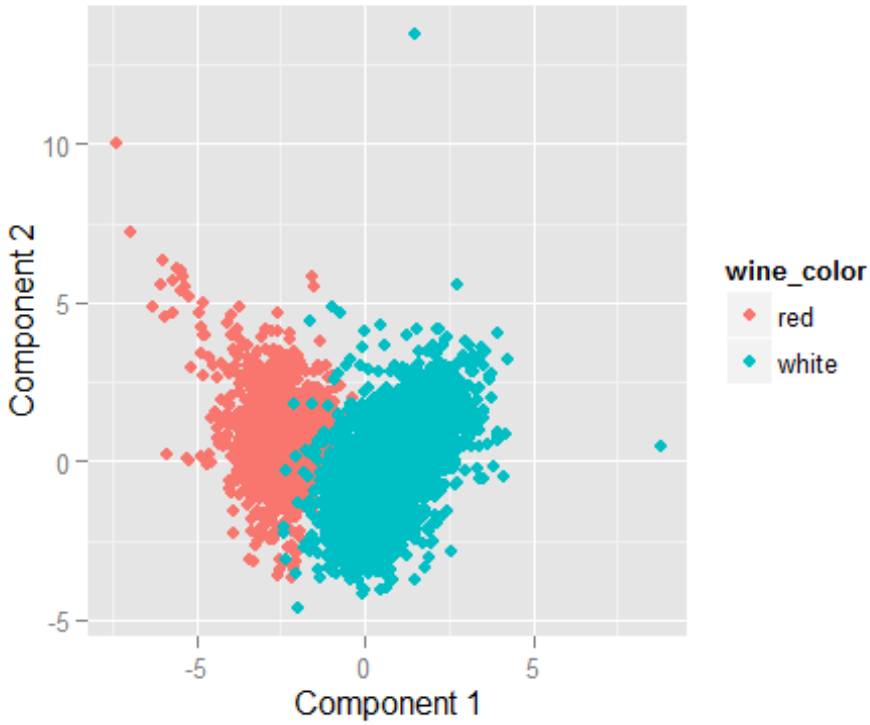



```
biplot(pc_wine)
```



Let us look at the plots of the Colors along the top 2 principal components and see if they show some grouping.

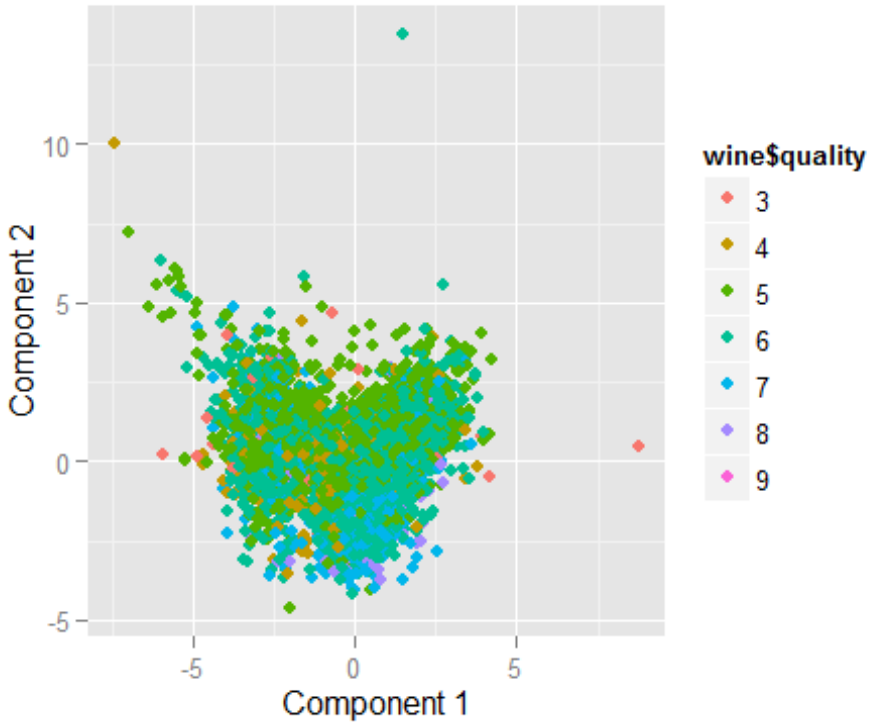
```
library(ggplot2)
qplot(score[,1], score[,2], color=wine_color, xlab='Component 1',
ylab='Component 2')
```



Oh yes they do!

Now let us look at how the quality is varying along these principal components.

```
wine$quality<-as.factor(wine$quality)
qplot(score[,1], score[,2], color=wine$quality, xlab='Component 1',
ylab='Component 2')
```



We can see that the different quality factors are present across both clusters and are not really distinguishable from each other. However, this is using the visualization only along the 2 top principal components. Probably they may cluster well when we take more principal components into account. But visualizing it on a 2-D plane is complicated.

Now let us do a k-means clustering with k=7 (since there are 7 categories of clustering from 3 to 9) and see how it distinguishes the quality.

```
set.seed(12321)
cluster_wine_chem_7 <- kmeans(wine_chem, centers=7, nstart=50)
table(wine$quality)

##
##    3    4    5    6    7    8    9
##   30  216 2138 2836 1079  193    5

table(wine$quality, cluster_wine_chem_7$cluster)

##
##       1   2   3   4   5   6   7
##   3   5   4   6   2   4   7   2
##   4  63  15  64   2  20  24  28
##   5 413 198 472  27  77 652 299
##   6 538 263 343  16 528 645 503
##   7 144 140  43   2 451 122 177
##   8  30  14   2   0  96  21  30
##   9   0   0   0   0   4   1   0
```

Inferences from the above 2 tables:

- From table 1, we can see that there are only few data points in 3 and 9 and only slightly more in 4 and 8. So even if the clusters 1 to 7 are distinguishing 5, 6 and 7 it would be good.
- From table 2, however, we can see 5, 6 and 7 are still not distinguishable using clusters 1 to 7. For example, cluster 1 has 200 data points with quality 5, 265 data points with quality 6 and 141 data points with quality 7 and so on.
- This means that clustering with k=7 is not distinguishing the quality well.

My choice of technique:

I would choose K-means clustering over PCA for this data due to the following reasons:

- K-Means clustering is a grouping algorithm and PCA even though helps in grouping, I would primarily use it as a feature reduction technique.
- In the plot that we looked at for PCA, we are able to look at only the 2 dimensions (along PC1 and PC2) and even though it shows some grouping, we can see a lot of overlap. Probably with more dimensions we will be able to segregate them better. But as I mentioned earlier, it is tough to visualize.
- Also, with PCA, there is no way (that I know of) of validating the output or calculating the correctness of your grouping.In K-Means however, we can predict the correctness of our groupings. When we were analyzing the table earlier, we did see that K-Means has given a very good grouping based on colors.
- Something that strikes me as a better method of grouping the "unsupervised" information is:
  - Eliminate the weakest 4 PCs after applying PCA on all the 11 properties.
  - And then apply K-Means clustering on the data represented using the remaining 7 PCs. This is something that I have not tried though, but intuitively it seems like a good idea.

Conclusion:

- Both k-means clustering and PCA are able to distinguish the dataset into 2 clusters. However, for this kind of data where all we need to do is group these into 2 segments and not necessarily reduce dimensions and also because of the inherent complexity of calculating correctness of grouping in PCA, I would choose k-means over PCA.
- Both k-means and PCA are not able to distinguish the quality of the wines.

Question 4: Market segmentation

Objective: To prepare a report for NutrientH20 that identifies any interesting market segments that appear to stand out in their social-media audience using the the twitter data their digital marketing agency has gathered.

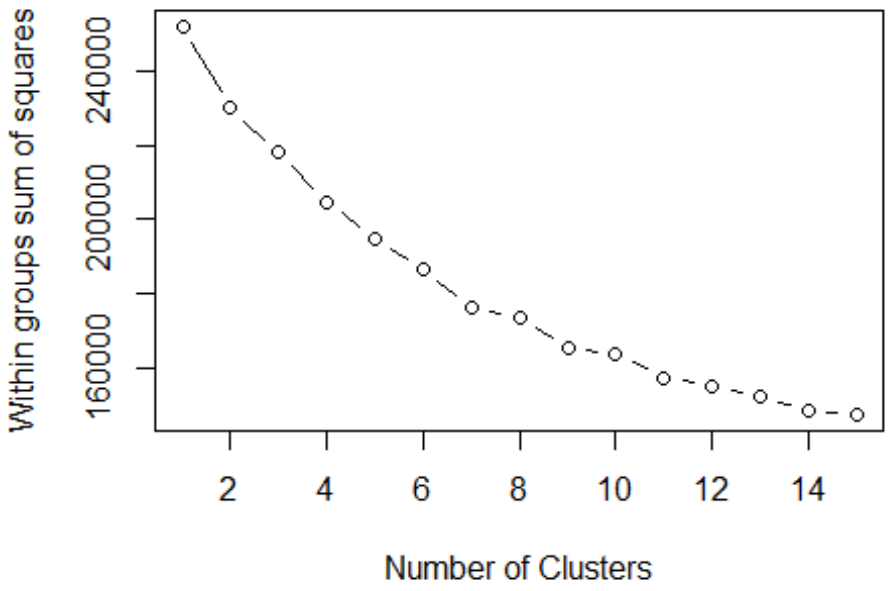
```r
# Reading the data
social = read.csv("social_marketing.csv")
par(mfrow=c(1,1))

# List of categories in the data frame
names(social)
```

```
##  [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"
```

```r
# Getting the categories
social_cat <- social[,c(2:37)]
```

To identify market segments I would like to run a K-Means clustering algorithm on all the categories. However, before running the K-Means, I would like to remove the following features - chatter, uncategorized, spam and adult - since they only add noise to the data and more importantly they are not useful in categorizing customers anyway.

```r
# Cleaning the data (Removing Chatter, Uncategorized)
social_cat <- social_cat[,-c(1,5,35,36)]

# scale the data
social_cat <- scale(social_cat, center=TRUE, scale=TRUE)

# before using k-means, lets find the best k using elbow function
wss <- (nrow(social_cat)-1)*sum(apply(social_cat,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(social_cat, centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum
of squares")
```



As we can see from the plot, the sum of squares is less only after 10. However, there is always a tradeoff between sum of squares and complexity. The problem I am looking at is to identify market segments using 32 categories (after cleaning). So grouping into 10 segments using just 32 variables is a stretch. Looking at the plot, 6 seems like a good K for K-Means.

```
# now let us cluster the data using K-Means with k = 6
cluster_social <- kmeans(social_cat, centers=6, nstart=50)
```

Now I have 6 clusters with me. They contain scaled data. What I am planning to do is just take the cluster values from the cluster_social we created and update it in our original data which is still unscaled. This will allow me to split the dataset into 6 groups and then analyze their market segments.

```
# adding cluster to original data
social$cluster <- as.numeric(cluster_social$cluster)

# Frequency of each cluster
table(social$cluster)

##
##    1    2    3    4    5    6
##  886  687  763  453  573 4520
```

We can see from the table above that Group 2 has the most representation. Now that we have the original data along with clusters, let us split them into their individual clusters now.

```
# creating individual datasets for analysis
# have removed the alphanumeric masked twitter name since it is not useful

social_group1 = social[social$cluster==1,c(2:37)]
social_group2 = social[social$cluster==2,c(2:37)]
social_group3 = social[social$cluster==3,c(2:37)]
social_group4 = social[social$cluster==4,c(2:37)]
social_group5 = social[social$cluster==5,c(2:37)]
social_group6 = social[social$cluster==6,c(2:37)]

# removing noise since it is not useful to characterize the market segments

social_group1 = social_group1[,-c(1,5,35,36)]
social_group2 = social_group2[,-c(1,5,35,36)]
social_group3 = social_group3[,-c(1,5,35,36)]
social_group4 = social_group4[,-c(1,5,35,36)]
social_group5 = social_group5[,-c(1,5,35,36)]
social_group6 = social_group6[,-c(1,5,35,36)]
```

Let me now analyze each market segment one by one.

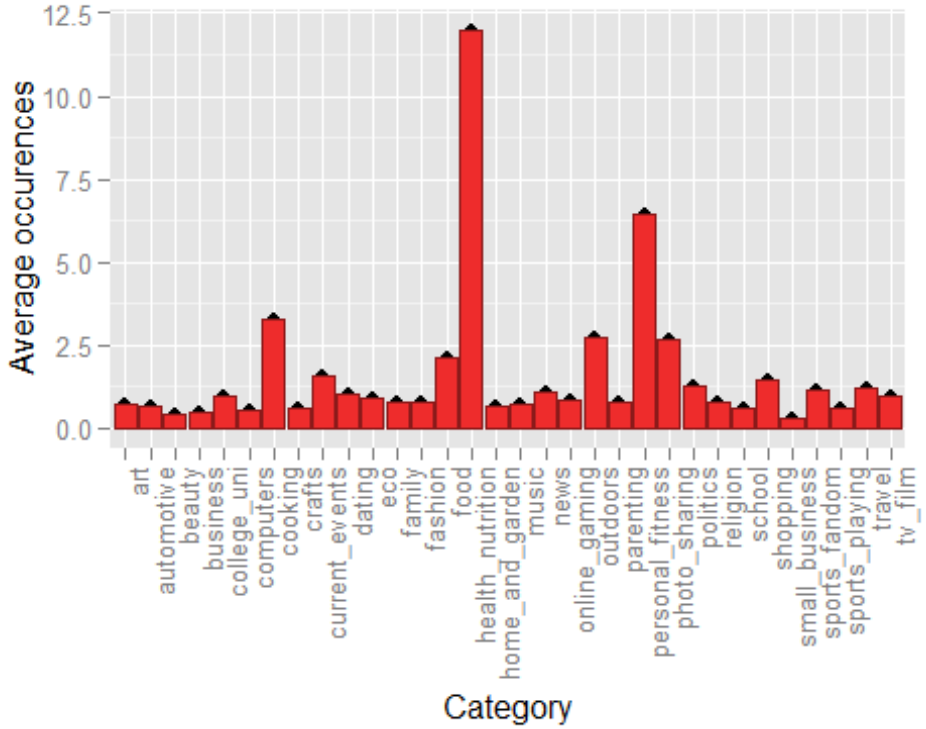
The plan here is:

- to take all clusters/groups one by one,
- look at the average occurences of each category in the group (i.e. if there are 500 customers in a group and the sum of category *travel* for all of those 500 customers is 2400, then average occurences of *travel* in *that group*) using bar plots,
- identify the stand-out categories in terms of average occurences and
- then try to label them based on the stand-out categories.

Market Segment 1:

Aside: Naming of this market segment is arbitrary and it gets its name only because it is derived from cluster 1

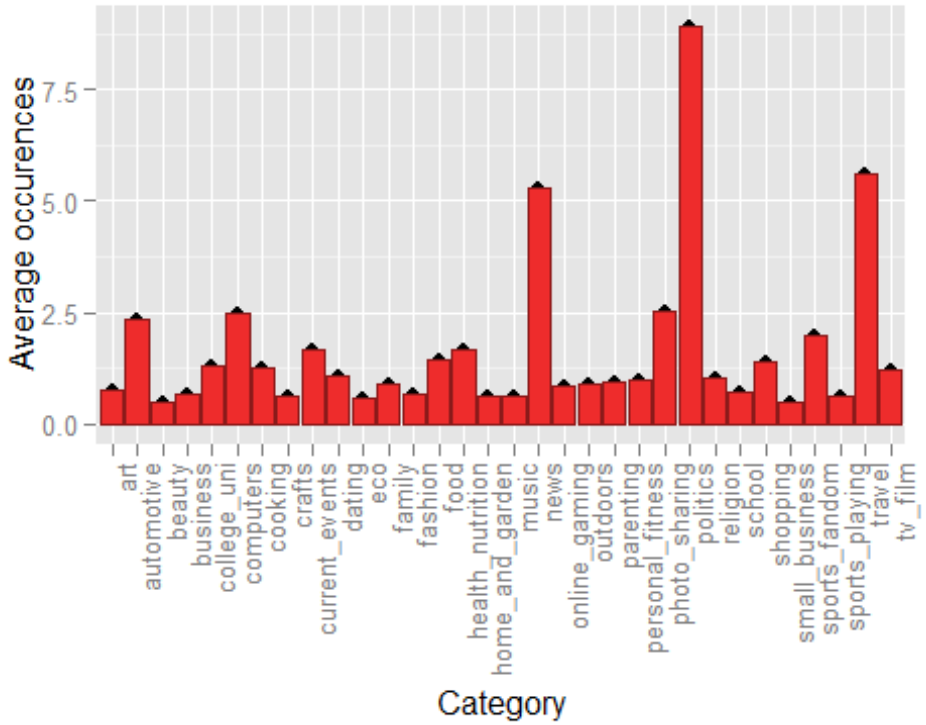
```
#identifying most frequent variables in cluster 1

library(ggplot2)
qplot(x=names(social_group1), y=apply(social_group1, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see the following standout categories - **politics, travel and news**. This indicates that these users are people who follow politics and news very much. They also seem to be active on the traveling front. Looking at the very poor averages for art, beauty and fashion, we can extrapolate that majority of this category should be men (though not necessarily always). I would like to call this market segment - ***Travelers who are into Current affairs***

Market Segment 2:

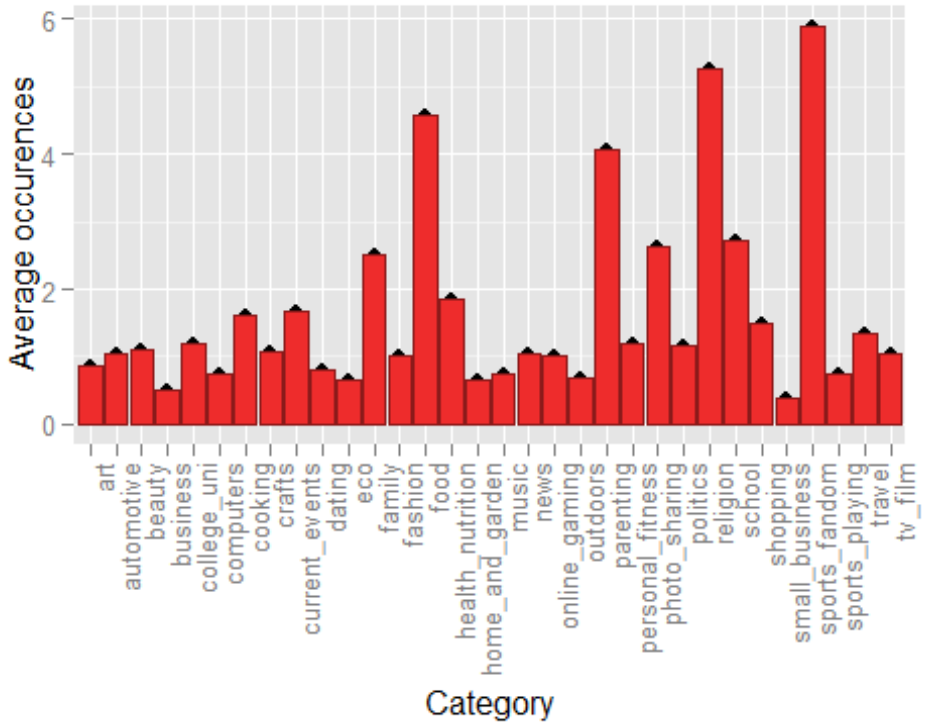
```
#identifying most frequent variables in cluster 2

qplot(x=names(social_group2), y=apply(social_group2, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see the following standout categories - **college/university, online gaming**. There is a decent amount of **photo sharing** as well. More gaming, less photo sharing, again we can say majority of this group should be guys. I would like to call this segment - ***Student gamers***

Market Segment 3:

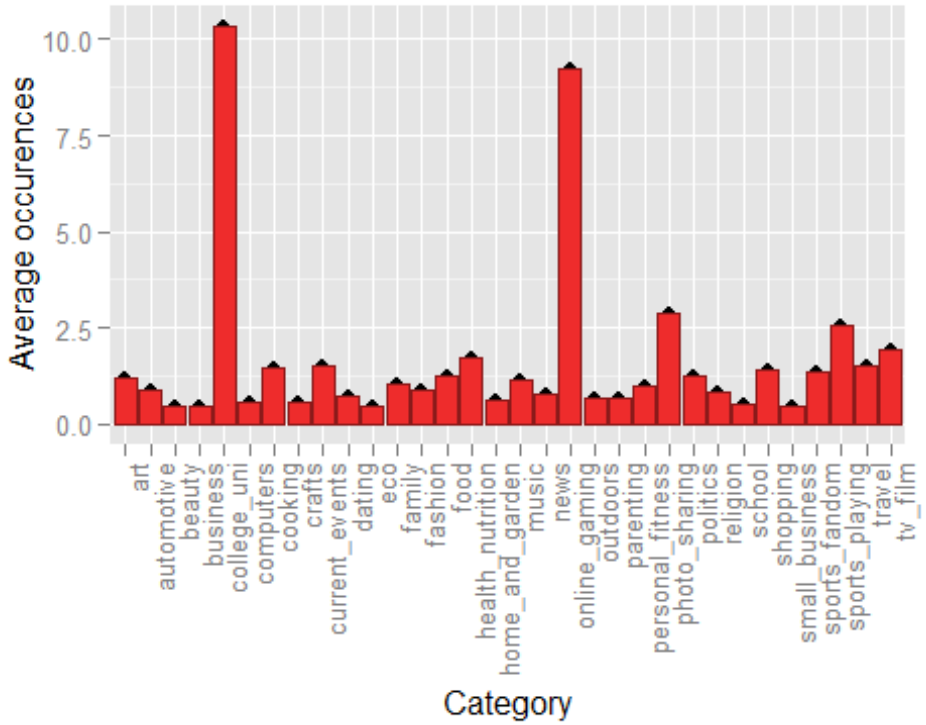
```
#identifying most frequent variables in cluster 3

qplot(x=names(social_group3), y=apply(social_group3, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see the following standout categories - **Cooking, beauty, fashion and photo sharing**. This group should mostly contain women. I would like to call this market segment - ***Women***

Market Segment 4:

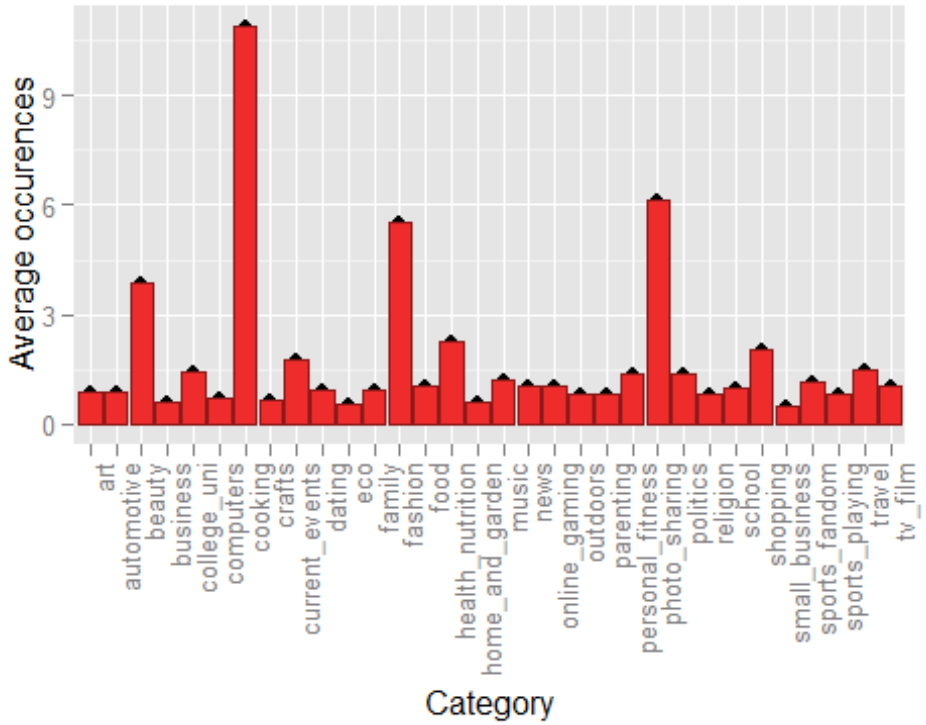
```
#identifying most frequent variables in cluster 4

qplot(x=names(social_group4), y=apply(social_group4, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see one standout category - **photo sharing**. But there are other categories like **current events, shopping, travel, tv/film, sports, politics, health/nutrition** as well. These kind of tweets come from people who follow pop culture closely or celebrities themselves. This also consists of people who just like using social media for fun - for example a tweet of an instagram upload would fall in this category. I would call this market segment - ***Pop culture/Social Media enthusiasts***

Market Segment 5:

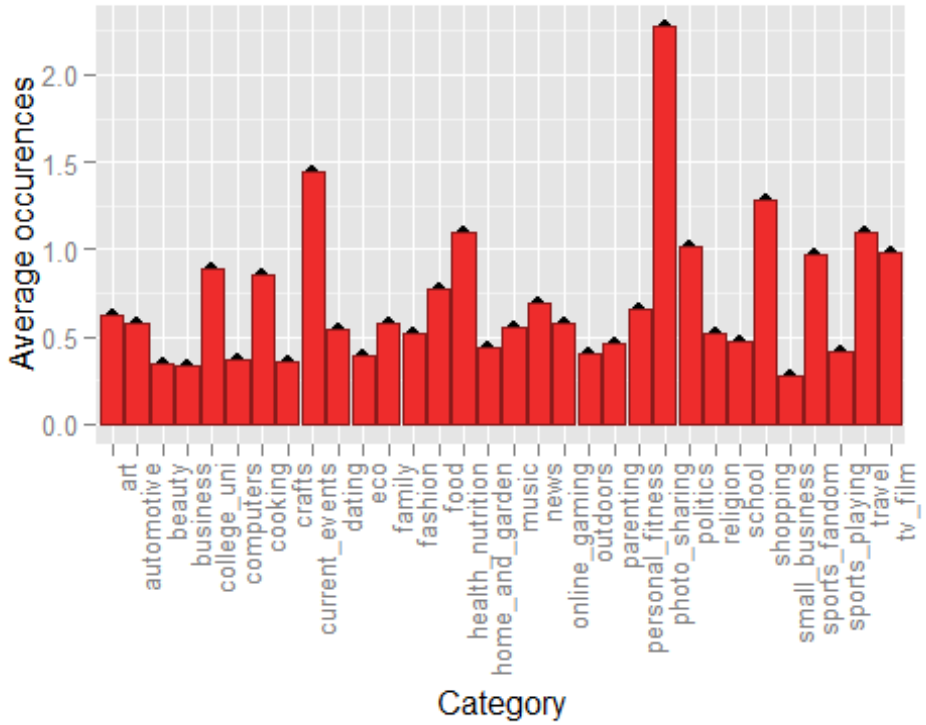
```
#identifying most frequent variables in cluster 5

qplot(x=names(social_group5), y=apply(social_group5, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see the following standout categories - **sports random, religion, parenting, food**. There are other relatively smaller categories too which have good representation - **family, photo-sharing, school, health/nutrition**. These users should then mostly be elders who are parents and are into sports as well. I would like to call this market segment - ***Sports-loving parents***

Market Segment 6:

```
#identifying most frequent variables in cluster 6

qplot(x=names(social_group6), y=apply(social_group6, MARGIN=2, FUN=mean),
xlab='Category',ylab='Average occurences') + geom_bar(stat = 'identity', fill
= "firebrick2", color = "firebrick4")+ theme(axis.text.x =
element_text(angle=90, hjust=1))
```



From the above plot, we can clearly see the following standout categories - **health/nutrition, personal fitness, cooking**. There people are into gymming and physical fitness a lot. Even the cooking bit should be related to fitness. So I would like to call this segment - ***Fitness Freaks***

Conclusion:

If this was a project given to me by NutrientH20, I would submit the following:

- 6 lists of userids belonging to these 6 market segments
- A report on different market segments identified, type of users in each segment and recommendations on targeting the user segment.

(However I am not aware of the exact business of NutrientH20 or the brand it represents. Hence, I have given recommendations ranging from beauty products campaigns to antiviruses campaigns!)

Report: I have grouped the users from the twitter data into 6 segments. Please find below the different segments and potential ad campaigns that can be targeted at different users.

1. **Market segment 1 - Travelers who are into Current affairs**
   - These people seem to be great followers of news and politics
   - Potential targets for ad campaigns on news papers, news and economy magazines, travel magazines
2. **Market segment 2 - Student gamers**

- This segment is full of college students who are into online gaming. They tend to be nerds as well
- Potential targets for ad campaigns on latest video games, latest software packages, antiviruses, graphics cards/GPU/processors, stationary stores and all other student-centric campaigns

3. **Market segment 3 - Women**
   - This segment is full of women who are into fashion, cooking and beauty
   - Potential targets for ad campaigns on toiletries, boutiques, websites containing info on cooking or beauty tips
4. **Market segment 4 - Pop culture/Social Media enthusiasts**
   - This segment consists of people who are into pop culture and using social media
   - Potential targets for ad campaigns on new social media apps, new technology, movies, tv series etc.
5. **Market segment 5 - Sports-loving parents**
   - We have family men and women who often tweet about parenting, family and schools in this segment. They also are very much into sports and religion
   - Potential targets for ad campaigns on sports magazines, school, education websites, parenting books, sports stores
6. **Market segment 6 - Fitness Freaks**
   - This group consists of health conscious fitness freaks
   - Potential targets for ad campaigns on gyms, fitness apps, fitness products, health books and websites

---