Exercise1

Bonfante

August 6, 2015

(1)Explanatory Analysis

Does using certain kinds of voting equipment lead to higher rates of undercount?

Read in data and create new column for the amount of undercounted votes

```
votes = read.csv('georgia2000.csv', header = TRUE)
summary(votes)
```

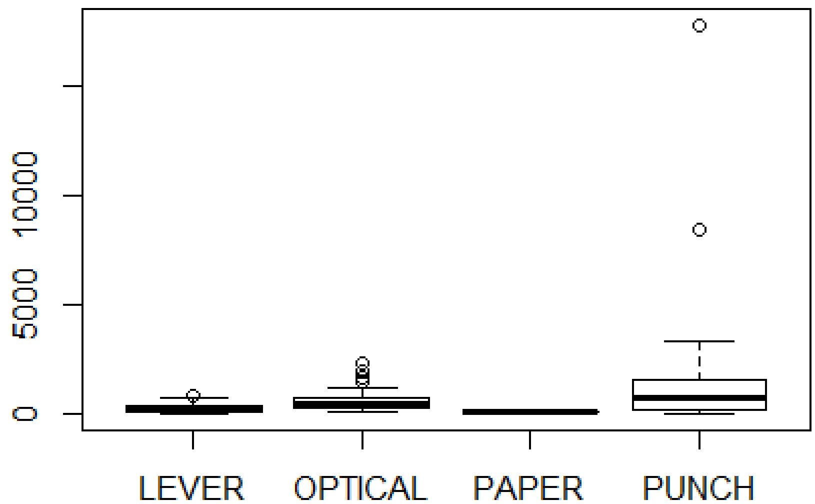
```
##       county       ballots           votes             equip   
##  APPLING :  1   Min.   :   881   Min.   :   832   LEVER  :74  
##  ATKINSON:  1   1st Qu.:  3694   1st Qu.:  3506   OPTICAL:66  
##  BACON   :  1   Median :  6712   Median :  6299   PAPER  : 2  
##  BAKER   :  1   Mean   : 16927   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  
## 
```

```
votes$undercount = votes$ballots - votes$votes
summary(votes$undercount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   152.5   296.0   595.5   523.5 17760.0
```

Boxplot Undercount against equipment. Lever and Paper are extremely accurate, while optical and punch are less reliable

```
boxplot(undercount~equip, data = votes)
```




```
lm.fit = glm(undercount~equip, data = votes)
summary(lm.fit)

## 
## Call:
## glm(formula = undercount ~ equip, data = votes)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2260.5   -246.8   -110.9    116.1  15501.5  
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      229.9      171.9   1.338    0.183    
## equipOPTICAL     362.3      250.3   1.447    0.150    
## equipPAPER      -173.4     1059.5  -0.164    0.870    
## equipPUNCH      2032.5      397.7   5.111 9.32e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 2186205)
## 
```

```
##     Null deviance: 396571504  on 158  degrees of freedom
## Residual deviance: 338861726  on 155  degrees of freedom
## AIC: 2778.2
##
## Number of Fisher Scoring iterations: 2
```

If so, should we worry that this effect has a disparate impact on poor and minority communities?

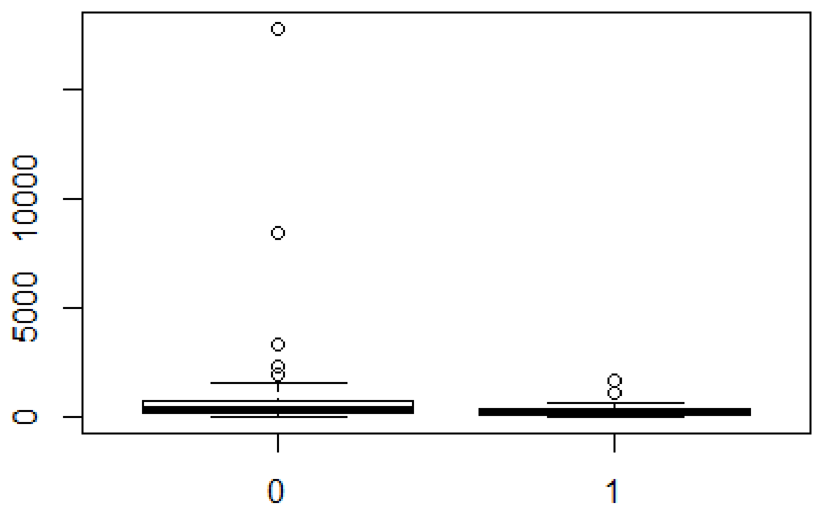
Xtab and Boxplot show that the poor have on average less undercounting and the large outliers pertain to the rich.

```
x1 = xtabs(~undercount + poor, data = votes)
x1

##           poor
## undercount 0 1
##       0    1 1
##       2    0 1
##       20   1 0
##       23   1 0
##       27   0 1
##       35   1 0
##       41   0 1
##       49   0 1
##       52   0 1
##       56   0 1
##       57   0 1
##       64   0 1
##       65   2 2
##       67   1 0
##       70   0 1
##       78   0 2
##       80   0 1
##       81   0 1
##       85   1 0
##       88   0 2
##       102  1 0
##       105  1 0
##       107  0 1
##       110  1 0
##       111  0 1
##       113  0 1
##       118  1 0
##       119  0 1
##       123  1 1
##       124  1 1
##       136  1 0
##       147  0 1
##       158  0 1
```

```
##       159   0 1
##       166   0 1
##       167   0 1
##       169   0 1
##       170   1 0
##       171   0 1
##       176   1 0
##       178   0 1
##       181   0 1
##       193   1 0
##       195   2 0
##       197   0 1
##       201   1 0
##       203   0 1
##       205   0 2
##       213   0 1
##       216   1 0
##       217   1 0
##       222   1 0
##       235   0 1
##       236   0 1
##       240   1 0
##       241   1 0
##       243   1 0
##       244   0 1
##       246   2 0
##       248   0 1
##       250   0 1
##       257   1 0
##       261   1 0
##       265   1 0
##       269   0 1
##       272   1 0
##       282   0 1
##       295   1 0
##       296   1 1
##       297   1 0
##       301   1 0
##       304   0 1
##       307   0 1
##       317   1 0
##       330   1 0
##       336   0 1
##       338   0 1
##       344   0 1
##       345   1 0
##       346   1 0
##       348   0 1
##       352   0 1
##       353   0 2
```

```
##       362   1 1
##       364   0 1
##       370   1 0
##       383   0 1
##       393   1 0
##       397   0 1
##       404   1 0
##       407   1 0
##       413   1 0
##       415   0 1
##       420   1 0
##       421   1 0
##       442   1 0
##       452   0 1
##       457   0 1
##       469   0 1
##       475   0 1
##       479   1 0
##       492   0 1
##       493   0 1
##       509   1 0
##       518   0 1
##       529   0 1
##       552   1 0
##       589   1 0
##       615   1 0
##       618   0 1
##       655   0 1
##       659   1 0
##       675   1 0
##       686   1 0
##       687   1 0
##       694   1 0
##       702   1 0
##       719   1 0
##       739   1 0
##       759   1 0
##       767   1 0
##       782   1 0
##       786   1 0
##       879   1 0
##       955   1 0
##       980   1 0
##       984   1 0
##       995   1 0
##       999   1 0
##       1080  0 1
##       1097  1 0
##       1110  1 0
##       1151  1 0
```

```
##       1193  1 0
##       1323  1 0
##       1499  1 0
##       1598  1 0
##       1688  0 1
##       1907  1 0
##       1911  1 0
##       2299  1 0
##       2308  1 0
##       3366  1 0
##       8372  1 0
##       17764 1 0

boxplot(undercount~poor, data = votes)
```

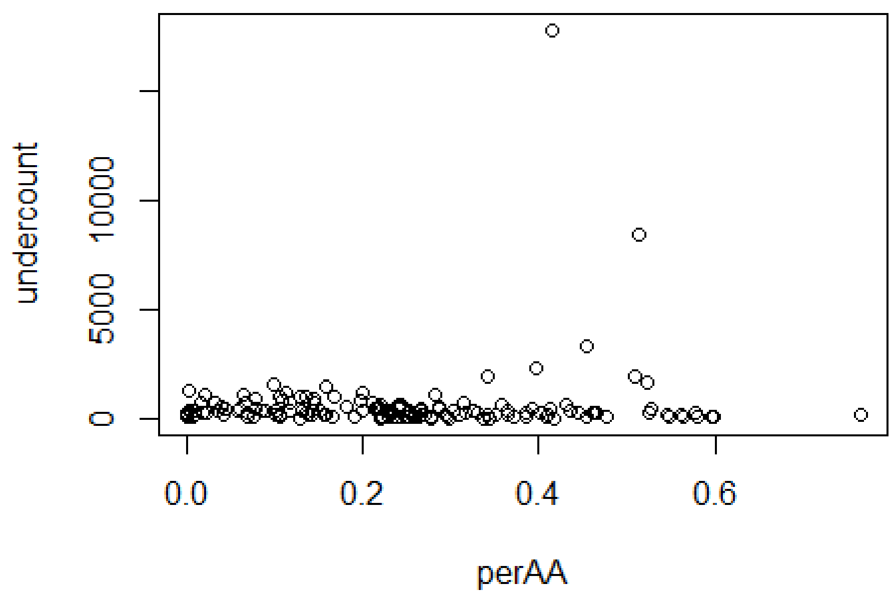


We can also see that the rich use the undercounted equipment much more frequently than the poor people.

```
x2 = xtabs(~equip + poor, data = votes)
p1 = prop.table(x2, margin =1)
p1

##          poor
## equip             0         1
##   LEVER   0.3918919 0.6081081
##   OPTICAL 0.7272727 0.2727273
##   PAPER   0.0000000 1.0000000
##   PUNCH   0.5882353 0.4117647
```

plotting Undercount against Percent AA shows little correlation, but a few large ouliers can be seen in higher AA populations.

```
plot(undercount~perAA, data = votes)
```




```
x = glm(undercount~perAA, data = votes)
summary(x)

##
## Call:
## glm(formula = undercount ~ perAA, data = votes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1046.2   -482.8   -221.3     21.9  16963.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    307.0      225.0   1.365    0.174
## perAA         1187.2      769.5   1.543    0.125
##
## (Dispersion parameter for gaussian family taken to be 2488210)
##
##     Null deviance: 396571504  on 158  degrees of freedom
## Residual deviance: 390648953  on 157  degrees of freedom
## AIC: 2796.8
```

```
##
## Number of Fisher Scoring iterations: 2
```

Conclusion

In conclusion it is not the poor population that is discriminated against but rather the rich population. When looking at the AA population there is very little correlation to undercounting other than a few outliers.

(2)Bootstrapping

Analyze the 5 ETFs

Import the ETFs and view the first 5 rows

```
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

library(fImport)

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

library(foreach)

#Import stocks
stocks = c("SPY", "TLT", "LQD", "EEM", "VNQ")
prices = yahooSeries(stocks, from='2011-01-01', to='2015-08-05')

# The first few rows
head(prices)

## GMT
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adj.Close
## 2011-01-03   126.71   127.60  125.70    127.05  138725200      115.9587
## 2011-01-04   127.33   127.37  126.19    126.98  137409700      115.8948
## 2011-01-05   126.58   127.72  126.46    127.64  133975300      116.4972
## 2011-01-06   127.69   127.83  127.01    127.39  122519000      116.2690
## 2011-01-07   127.56   127.77  126.15    127.14  156034600      116.0409
## 2011-01-10   126.58   127.16  126.20    126.98  122401700      115.8948
##            TLT.Open TLT.High TLT.Low TLT.Close TLT.Volume TLT.Adj.Close
## 2011-01-03    93.20    94.31   92.95     93.41   13799400      81.38658
## 2011-01-04    93.41    93.77   92.91     93.52   10466900      81.48242
## 2011-01-05    92.49    92.69   91.19     91.46   17568000      79.68757
## 2011-01-06    91.54    92.12   91.16     91.86    9317500      80.03609
## 2011-01-07    91.51    92.63   91.03     92.35   12694600      80.46302
## 2011-01-10    92.58    92.99   92.18     92.85    8295200      80.89866
##            LQD.Open LQD.High LQD.Low LQD.Close LQD.Volume LQD.Adj.Close
## 2011-01-03   108.22   108.89  108.06    108.86    2447900      91.64534
## 2011-01-04   108.98   109.19  108.75    109.00     962500      91.76320
## 2011-01-05   108.51   108.58  108.00    108.20    1209200      91.08971
## 2011-01-06   108.32   108.64  108.20    108.36    1089700      91.22441
## 2011-01-07   108.26   109.06  108.20    108.91    1118600      91.68744
## 2011-01-10   108.78   109.20  108.78    109.10     909200      91.84739
##            EEM.Open EEM.High EEM.Low EEM.Close EEM.Volume EEM.Adj.Close
## 2011-01-03    48.03    48.31   48.03     48.10   40116200      43.98749
## 2011-01-04    48.26    48.32   47.75     48.32   45176300      44.18868
## 2011-01-05    47.90    48.30   47.88     48.20   47527100      44.07894
## 2011-01-06    48.01    48.04   47.58     47.69   43977400      43.61254
## 2011-01-07    47.51    47.62   46.93     47.25   57124900      43.21016
## 2011-01-10    46.80    46.85   46.51     46.76   62556500      42.76206
##            VNQ.Open VNQ.High VNQ.Low VNQ.Close VNQ.Volume VNQ.Adj.Close
## 2011-01-03    55.76    56.44   55.73     56.40    1957400      47.62828
## 2011-01-04    56.48    56.63   55.00     55.32    2315700      46.71624
## 2011-01-05    55.15    55.63   55.14     55.52    1512700      46.88514
```

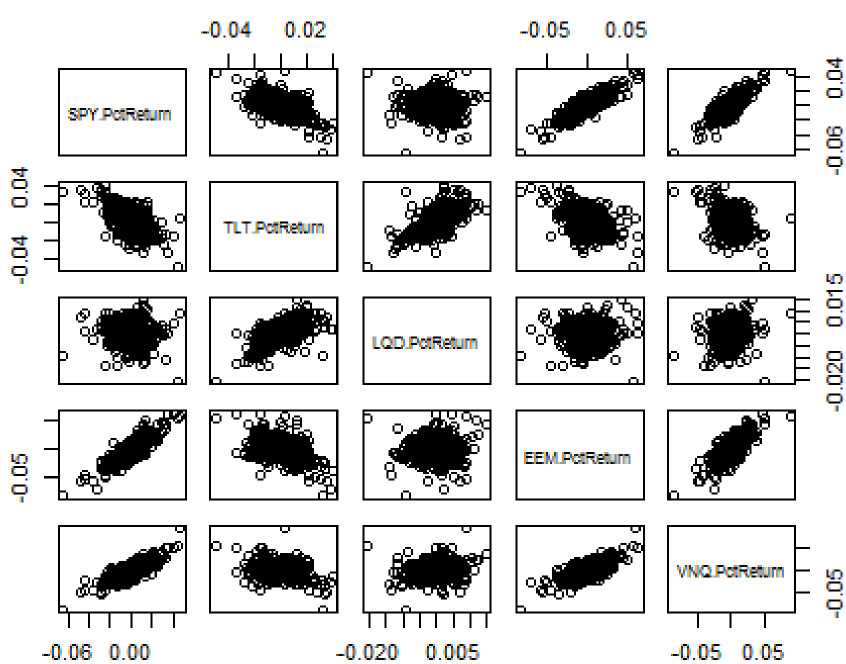
```
## 2011-01-06     55.67     55.68    54.97      55.00     1819300        46.44601
## 2011-01-07     55.29     55.45    54.50      55.02     2001200        46.46290
## 2011-01-10     54.80     55.14    54.42      55.01     1668400        46.45446
```

Creat a function to calculate daily returns of each ETF and then compute the returns

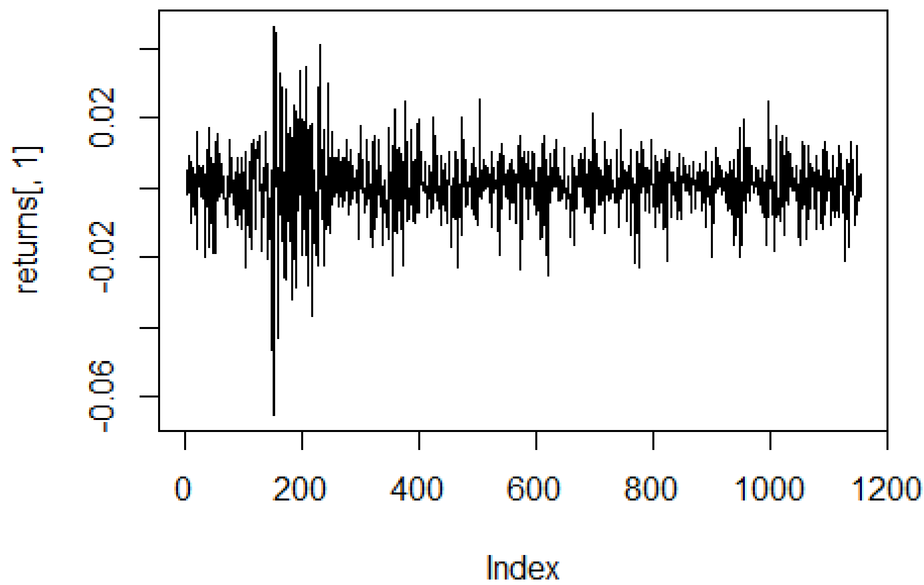
```
YahooPricesToReturns = function(series) {
    mycols = grep('Adj.Close', colnames(series))
    closingprice = series[,mycols]
    N = nrow(closingprice)
    percentreturn = as.data.frame(closingprice[2:N,]) /
as.data.frame(closingprice[1:(N-1),]) - 1
    mynames = strsplit(colnames(percentreturn),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


returns = YahooPricesToReturns(prices)
```

Plot the Returns

```
pairs(returns)
```




```
plot(returns[,1], type='l')
```



Calculate the betas of each stock determined against the market to see which investments are riskier

```
lm_TLT = lm(returns[,2]~returns[,1])
lm_LQD = lm(returns[,3]~returns[,1])
lm_EEM = lm(returns[,4]~returns[,1])
lm_VNQ = lm(returns[,5]~returns[,1])

coef(lm_TLT); coef(lm_LQD); coef(lm_EEM); coef(lm_VNQ)
```

```
##   (Intercept)  returns[, 1]
##  0.0007091476 -0.5626797102

##   (Intercept)  returns[, 1]
##  0.0002286759 -0.0428450593

##   (Intercept)  returns[, 1]
## -0.0007506257  1.2301980660

##   (Intercept)  returns[, 1]
## -0.0000285025  0.9399541016
```

Look at the residuals and their correlations

```
residuals = cbind(resid(lm_TLT), resid(lm_LQD), resid(lm_EEM), resid(lm_VNQ))

cor(residuals)
```

```
##             [,1]      [,2]       [,3]      [,4]
## [1,] 1.00000000 0.7888184 0.03087361 0.2707303
## [2,] 0.78881844 1.0000000 0.17097169 0.3493197
## [3,] 0.03087361 0.1709717 1.00000000 0.1280538
## [4,] 0.27073031 0.3493197 0.12805378 1.0000000
```

Evenly weighted Portfolio:

Set the seed and simulate performance for the safe portfolio. Here the bonds are evenly weighted

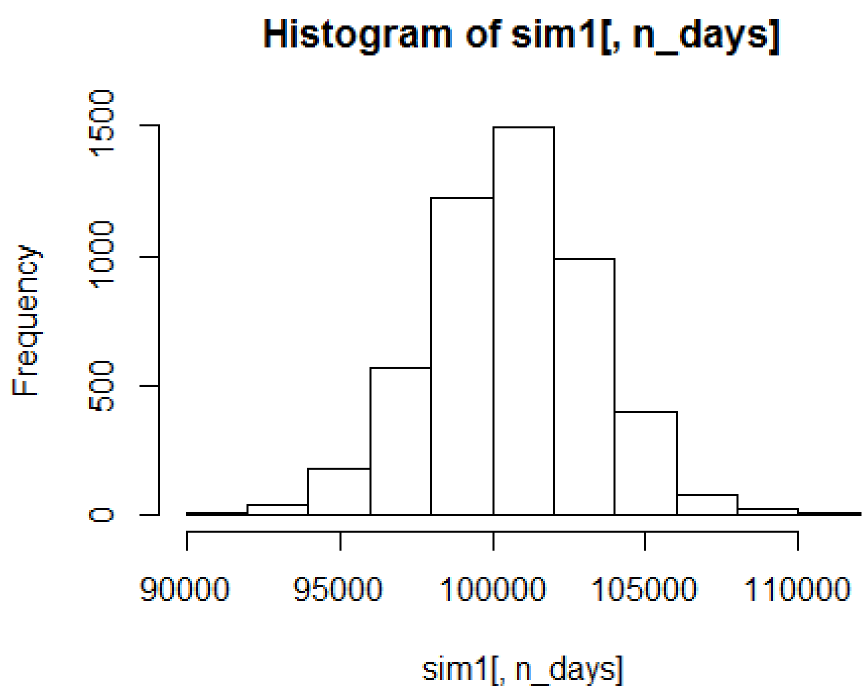
```
n_days=20
set.seed(3000)

sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days) # Set up a placeholder to track total
wealth
    for(today in 1:n_days) {
        return.today = resample(returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}
```

Show a selection of Sim1 and plot a histogram of the wealth over 20 days

```
head(sim1)

##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1 100842.83 100807.89 100843.17 100762.98 101161.61 100701.92
## result.2 100694.09  99254.37  99456.25  98838.68  99004.05  98161.16
## result.3 100035.54 100094.03 100571.32 100188.06 100506.17 100278.45
## result.4 100507.56 100809.95 100417.39 100931.90 101408.68 101130.46
## result.5 100123.68 100640.27 100986.27 101252.22 100583.03  99696.41
## result.6  99623.42  99608.27 100005.36 100844.86 100615.73  99591.93
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1 100961.95 100710.30 100564.89 100618.91 100304.57 100498.93
## result.2  98235.78  98357.28  98027.71  97528.78  97982.59  97877.37
## result.3  99826.58  99647.02  99619.48  98939.66  99314.46  99202.39
## result.4 101489.06 101349.91 101979.57 101991.98 102081.31 102119.02
## result.5 100165.18 100250.51  99830.64  99937.80  99201.50  99631.72
## result.6  99992.18 100027.65  99887.70  99536.16  99335.44  99109.74
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1 100695.81 100127.85  99789.70 100093.82 100470.40 100429.44
## result.2  98457.40  97822.33  97784.23  96980.26  97751.36  98001.20
```

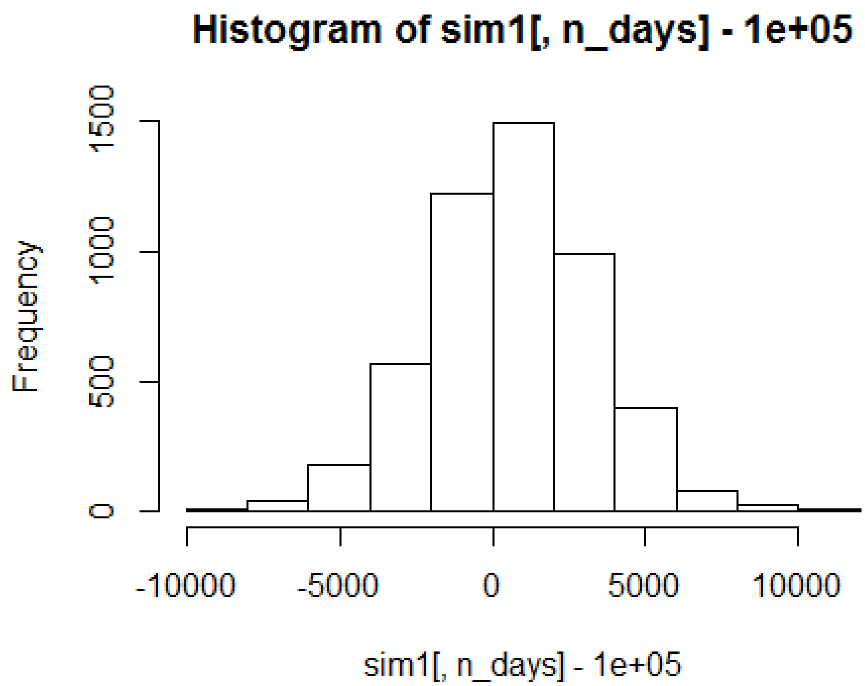
```
## result.3  98964.03  98092.31  98548.75  98521.51  98676.78  98536.17
## result.4 102905.05 102857.38 102746.25 101569.24 101989.80 102049.98
## result.5  99996.01 100433.81 100623.57 100283.38  99274.82  99311.50
## result.6  99487.26  99459.95  99126.93  98768.10  98094.09  97428.78
##              [,19]     [,20]
## result.1  99349.71  99498.96
## result.2  97978.77  97370.37
## result.3  98176.61  98209.07
## result.4 102263.94 102222.25
## result.5 100053.44 100646.38
## result.6  96710.97  97117.65
```

```r
hist(sim1[,n_days])
```



plot a histogram showing profit/loss over the 20 days

```r
hist(sim1[,n_days]- 100000)
```



Calculate 5% value at risk

```
quantile(sim1[,n_days], 0.05) - 100000

##       5% 
## -3869.17
```

Show the average profit or loss

```
mean(sim1[,n_days]- 100000)

## [1] 618.9943
```

Safe Portfolio

Set the seed and simulate performance for the safe portfolio. Here the bonds have been heavily weigted

```
set.seed(4000)

sim2 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.15, 0.3, 0.4, 0.0, 0.15)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days) # Set up a placeholder to track total 
wealth
    for(today in 1:n_days) {
        return.today = resample(returns, 1, orig.ids=FALSE)
```

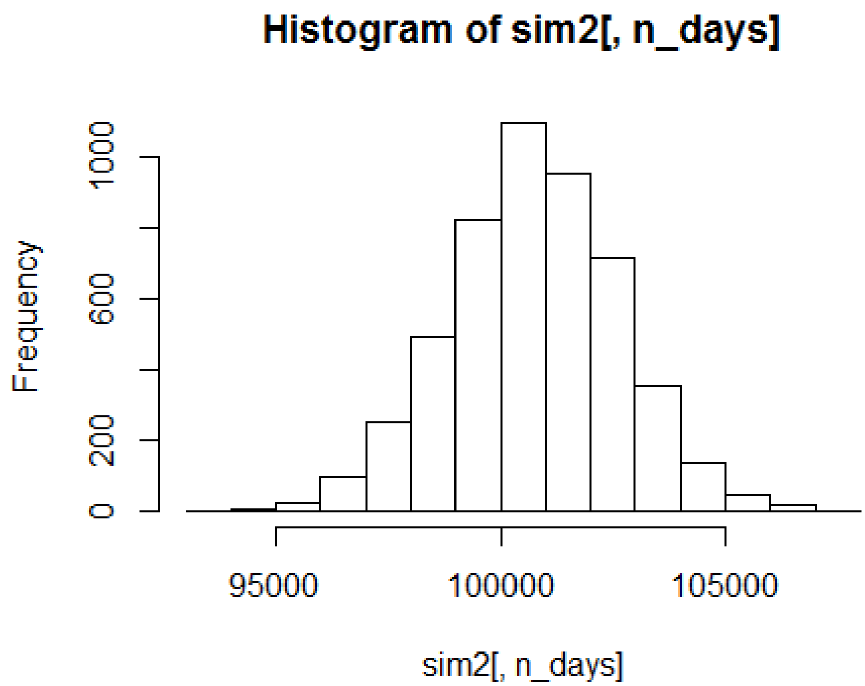
```
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}
```

Show a selection of Sim2 and plot a histogram of the wealth over 20 days

```
head(sim2)
```

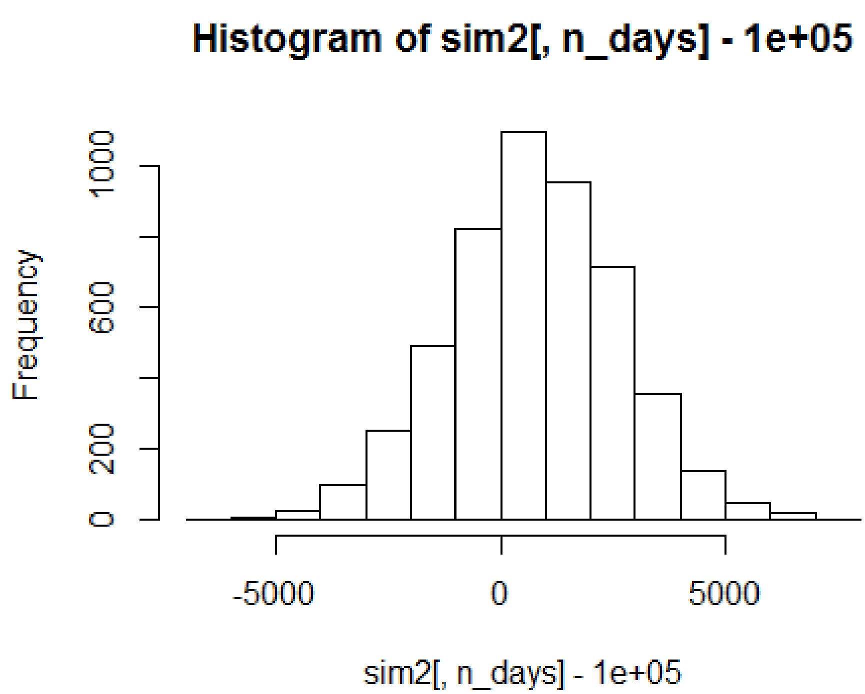
```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## result.1 100442.76 100730.17 100725.81 101085.10 100708.31 100661.34
## result.2 100737.43 100878.77 101143.97 100827.06 101024.26 100842.55
## result.3  99303.76  99040.03  98752.78  98297.78  98786.38  98816.78
## result.4 100384.21 100565.02 100190.17 100784.47 101342.15 101563.36
## result.5 100162.49 100117.57  99991.39  99684.05  99777.25  99811.77
## result.6 100151.03  99963.78 100400.43 100980.61 100552.75 100840.47
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1 101005.36 100936.26 100638.48 100826.11 100525.77 100435.80
## result.2 100817.24 100824.81 101162.39 101946.97 101304.65 101364.11
## result.3  98363.90  98512.89  98543.20  98347.81  98216.66  99054.20
## result.4 100923.46 101393.27 101839.20 102330.46 102775.77 103111.03
## result.5  99506.18  99807.34  99816.33  99373.98  99507.36  99782.32
## result.6 101073.80 101209.46 101326.76 101540.99 101206.85 101230.32
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1 100632.23 100105.05 100382.90 100388.20 100582.88 101301.50
## result.2 102153.95 102478.92 103193.41 102976.79 102800.60 103031.71
## result.3  98155.65  98306.50  98279.83  97618.74  97730.57  97730.30
## result.4 103216.85 103022.95 103296.97 103433.82 103666.35 103692.58
## result.5  99396.68  99030.88  98884.85  97979.98  97706.00  97779.11
## result.6 101936.11 101457.89 101356.30 101675.97 101401.39 101550.43
##              [,19]     [,20]
## result.1 101048.14 100433.33
## result.2 103018.87 103373.32
## result.3  97874.27  98030.42
## result.4 103944.47 104100.60
## result.5  97910.35  97957.72
## result.6 102043.03 102749.30
```

```
hist(sim2[,n_days])
```



plot a histogram showing profit/loss over the 20 days

```
hist(sim2[,n_days]- 100000)
```



Calculate 5% value at risk

```
quantile(sim2[,n_days], 0.05) - 100000

##        5%
## -2417.244
```

Show the average profit or loss

```
#Average Profit/Loss
mean(sim2[,n_days]- 100000)

## [1] 734.0009
```

Risky Portfolio

Set the seed and simulate performance for the risky portfolio. Here the bonds have been excluded

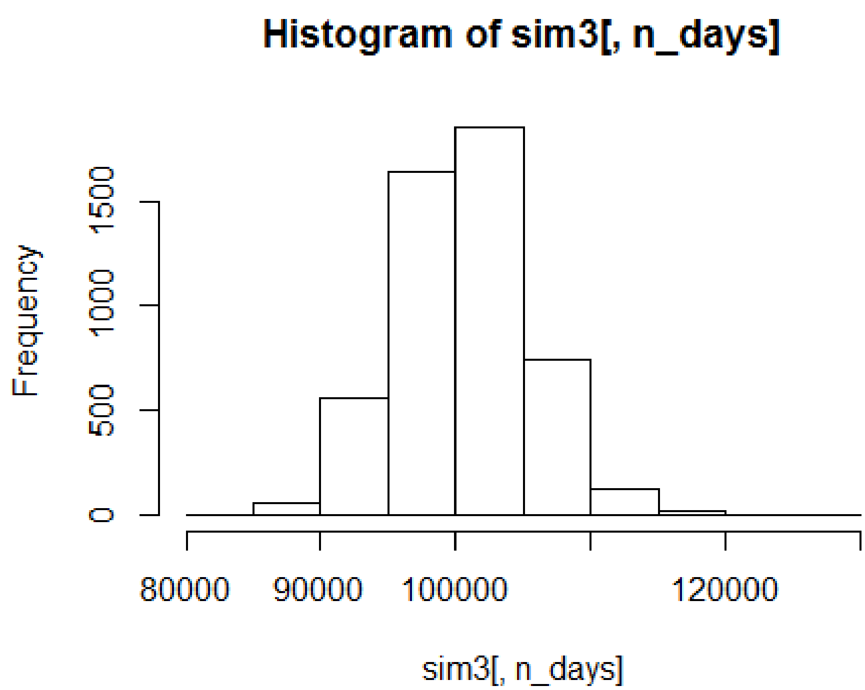
```
set.seed(2000)

sim3 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 100000
    weights = c(0.4, 0.0, 0.0, 0.4, 0.2)
    holdings = weights * totalwealth
    wealthtracker = rep(0, n_days) # Set up a placeholder to track total
wealth
    for(today in 1:n_days) {
        return.today = resample(returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        holdings = weights * totalwealth
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}
```

Show a selection of Sim3 and plot a histogram of the wealth over 20 days

```
head(sim3)

##               [,1]     [,2]     [,3]      [,4]     [,5]      [,6]
## result.1  99880.23 100184.5 100017.2 100458.52 101228.9 100881.73
## result.2 101058.13 101232.2 101793.4 101916.34 102147.7 102571.41
## result.3  99817.85 100037.2 100036.0 101475.47 101232.1 100535.13
## result.4 103068.40 101579.9 102820.5 101010.85 100117.7  99469.64
## result.5 101240.73 101678.6  99453.0  99973.87  97625.2  97651.87
## result.6 100941.02 100487.9 100083.9  99786.64 101138.2 101015.01
##               [,7]      [,8]      [,9]     [,10]     [,11]     [,12]
## result.1 100010.42  99775.03 101503.20 101546.61 101648.38 104827.07
## result.2 102658.24 102779.24 101071.49 100270.67  99159.27 101093.51
## result.3 100445.79  99933.25  98876.02  99418.32  99118.70  98061.48
```

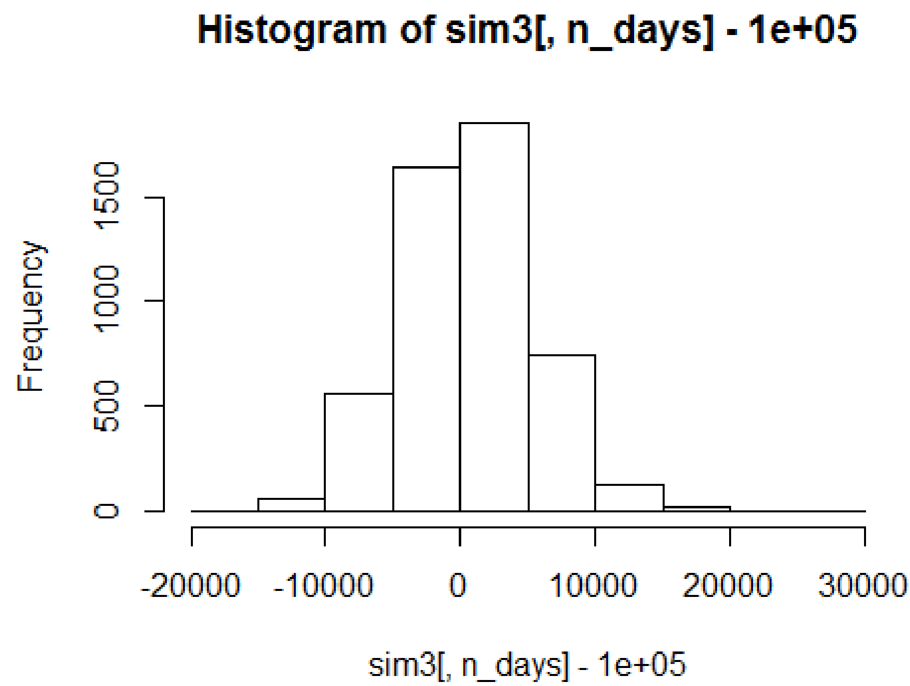
```
## result.4  99967.10 101444.45 102918.13 103670.91 102957.16 102449.68
## result.5  97830.44  97608.02  97571.06  97395.53  98389.70  97996.39
## result.6 105837.69 104193.51 104422.44 104478.70 103075.06 104287.14
##              [,13]     [,14]     [,15]     [,16]     [,17]     [,18]
## result.1 105488.89 104689.64 102435.51 102386.41 102468.40 103490.39
## result.2 101123.72 101465.81 100815.20 101227.69 101257.94 102329.38
## result.3  96881.90  97980.65  98390.16  99041.60  99709.57 100147.67
## result.4 102199.06 102093.22 102870.53 104506.71 104739.68 104145.47
## result.5  98593.56  98828.27  97126.38  98523.94  99362.55  99971.82
## result.6 104206.42 104989.69 103667.83 104575.87 105483.84 105498.81
##              [,19]     [,20]
## result.1 104561.39 105785.17
## result.2 102420.84 101334.10
## result.3  99854.57 101031.83
## result.4 104000.14 104853.38
## result.5  98516.27  99574.83
## result.6 106494.24 105646.79
```

```r
hist(sim3[,n_days])
```



plot a histogram showing profit/loss over the 20 days

```r
hist(sim3[,n_days]- 100000)
```



Calculate 5% value at risk

```
quantile(sim3[,n_days], 0.05) - 100000

##        5%
## -7258.986
```

Show the average profit or loss

```
mean(sim3[,n_days]- 100000)

## [1] 593.1402
```

Conclusion

Ultimately the safe portfolio had the best average return and the least amount at stake. The risky portfoli had the worst return and the most money at stake.

(3)Clustering And PCA

Red Vs White Clustering

Read in the data

```r
wine = read.csv('wine.csv')

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
```

```r
names(wine)
```

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

Remove the last 2 columns and scale the data.

```r
wine_num = wine [,(1:11)]
wine_scaled = scale(wine_num, center = TRUE, scale = TRUE)
```

cluster the data using 2 centers, in hopes of distinguishing red from white wine.

```r
cluster_2 = kmeans(wine_scaled, centers = 2, nstart = 50)
```

Capture the mean and SD of the scaled data.

```r
sigma = attr(wine_scaled,"scaled:scale")
mu = attr(wine_scaled,"scaled:center")
```

Unscale the data.

```r
cluster_2$center
```

```
##   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
```

```
cluster_2$center[1,]*sigma + mu
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.85167903           0.27458385           0.33524928
##       residual.sugar            chlorides  free.sulfur.dioxide
##           6.39402555           0.04510424          35.52152864
## total.sulfur.dioxide              density                   pH
##         138.45848785           0.99400486           3.18762464
##            sulphates              alcohol
##           0.48880511          10.52235888
```

```
cluster_2$center[2,]*sigma + mu
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            8.2895922            0.5319416            0.2695435
##       residual.sugar            chlorides  free.sulfur.dioxide
##            2.6342666            0.0883238           15.7647596
## total.sulfur.dioxide              density                   pH
##           48.6396835            0.9967404            3.3097200
##            sulphates              alcohol
##            0.6567194           10.4015216
```

See which wines are in each cluster

```
which(cluster_2$cluster == 1)
```

```
##    [1]   50  355  495  592  635  650  837  838 1018 1019 1080 1082 1091
##   [14] 1115 1132 1157 1236 1245 1287 1390 1457 1491 1567 1575 1600 1601
##   [27] 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614
##   [40] 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627
##   [53] 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640
##   [66] 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653
##   [79] 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666
##   [92] 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
##  [105] 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692
##  [118] 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705
##  [131] 1706 1707 1708 1709 1710 1711 1712 1713 1714 1716 1717 1718 1719
##  [144] 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732
##  [157] 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745
##  [170] 1746 1748 1749 1750 1751 1752 1753 1755 1756 1757 1758 1759 1760
##  [183] 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773
##  [196] 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786
```

```
##  [209] 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799
##  [222] 1800 1801 1802 1803 1804 1805 1806 1807 1809 1810 1811 1812 1813
##  [235] 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826
##  [248] 1827 1828 1829 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840
##  [261] 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853
##  [274] 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866
##  [287] 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879
##  [300] 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892
##  [313] 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905
##  [326] 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918
##  [339] 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
##  [352] 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944
##  [365] 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957
##  [378] 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970
##  [391] 1971 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
##  [404] 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997
##  [417] 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
##  [430] 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
##  [443] 2024 2025 2026 2027 2028 2029 2030 2031 2032 2034 2035 2036 2037
##  [456] 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050
##  [469] 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063
##  [482] 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076
##  [495] 2077 2078 2079 2080 2081 2082 2083 2085 2086 2087 2088 2089 2090
##  [508] 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103
##  [521] 2104 2105 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117
##  [534] 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130
##  [547] 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143
##  [560] 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156
##  [573] 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169
##  [586] 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182
##  [599] 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195
##  [612] 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208
##  [625] 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222
##  [638] 2223 2224 2225 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236
##  [651] 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249
##  [664] 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2263
##  [677] 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276
##  [690] 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2288 2289 2290
##  [703] 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303
##  [716] 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316
##  [729] 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329
##  [742] 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342
##  [755] 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355
##  [768] 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368
##  [781] 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381
##  [794] 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394
##  [807] 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407
##  [820] 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420
##  [833] 2421 2422 2423 2424 2425 2426 2427 2428 2429 2431 2432 2433 2435
##  [846] 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448
```

```
##  [859] 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461
##  [872] 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474
##  [885] 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487
##  [898] 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500
##  [911] 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513
##  [924] 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526
##  [937] 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539
##  [950] 2540 2541 2542 2543 2544 2545 2546 2547 2549 2550 2551 2552 2553
##  [963] 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566
##  [976] 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579
##  [989] 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592
## [1002] 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605
## [1015] 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618
## [1028] 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631
## [1041] 2632 2633 2635 2638 2639 2641 2642 2643 2644 2645 2646 2647 2648
## [1054] 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661
## [1067] 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674
## [1080] 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687
## [1093] 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700
## [1106] 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713
## [1119] 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727
## [1132] 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740
## [1145] 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2753 2754
## [1158] 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767
## [1171] 2768 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780
## [1184] 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793
## [1197] 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806
## [1210] 2807 2808 2809 2810 2811 2812 2813 2814 2815 2816 2818 2819 2820
## [1223] 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830 2831 2832 2833
## [1236] 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845 2846
## [1249] 2847 2848 2849 2850 2851 2852 2853 2854 2855 2856 2857 2858 2859
## [1262] 2860 2861 2862 2863 2864 2865 2866 2867 2868 2869 2870 2871 2873
## [1275] 2874 2875 2876 2877 2878 2879 2880 2881 2882 2883 2884 2885 2886
## [1288] 2887 2888 2889 2890 2891 2892 2893 2894 2895 2896 2897 2898 2899
## [1301] 2900 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910 2911 2912
## [1314] 2913 2914 2915 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925
## [1327] 2926 2927 2928 2929 2930 2931 2932 2933 2934 2935 2936 2937 2938
## [1340] 2939 2940 2941 2942 2943 2944 2945 2946 2947 2948 2949 2950 2951
## [1353] 2952 2953 2954 2955 2956 2957 2958 2959 2960 2961 2962 2963 2964
## [1366] 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975 2976 2977
## [1379] 2978 2979 2980 2981 2982 2983 2984 2985 2987 2988 2989 2990 2991
## [1392] 2992 2993 2995 2996 2997 2998 2999 3000 3001 3002 3003 3004 3005
## [1405] 3006 3007 3008 3009 3010 3011 3012 3013 3014 3015 3016 3017 3018
## [1418] 3019 3020 3021 3022 3023 3024 3025 3026 3027 3028 3029 3030 3031
## [1431] 3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042 3043 3044
## [1444] 3045 3046 3047 3048 3049 3050 3051 3052 3053 3054 3055 3056 3057
## [1457] 3058 3059 3060 3061 3062 3063 3064 3065 3066 3067 3068 3069 3070
## [1470] 3071 3072 3073 3074 3075 3076 3077 3078 3079 3080 3081 3082 3083
## [1483] 3084 3085 3086 3087 3088 3089 3090 3091 3092 3093 3094 3095 3097
## [1496] 3098 3099 3100 3101 3102 3103 3104 3105 3106 3107 3108 3109 3110
```

```
## [1509] 3111 3112 3113 3114 3115 3116 3117 3118 3119 3120 3121 3122 3123
## [1522] 3124 3125 3126 3127 3128 3129 3130 3131 3132 3133 3134 3135 3136
## [1535] 3137 3138 3139 3140 3141 3142 3143 3144 3145 3146 3147 3148 3149
## [1548] 3150 3151 3152 3153 3154 3155 3156 3157 3158 3159 3160 3161 3162
## [1561] 3163 3164 3165 3166 3167 3168 3169 3170 3171 3172 3173 3174 3175
## [1574] 3176 3178 3179 3180 3181 3182 3183 3184 3185 3186 3187 3188 3189
## [1587] 3190 3191 3192 3193 3194 3195 3196 3197 3198 3199 3200 3201 3202
## [1600] 3203 3204 3205 3206 3207 3208 3209 3210 3211 3212 3213 3214 3215
## [1613] 3216 3217 3218 3219 3220 3221 3222 3223 3224 3225 3226 3227 3228
## [1626] 3229 3230 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240 3241
## [1639] 3242 3243 3244 3245 3246 3247 3248 3249 3250 3251 3252 3253 3254
## [1652] 3255 3256 3257 3258 3259 3260 3261 3262 3263 3264 3265 3266 3267
## [1665] 3268 3269 3270 3271 3272 3273 3274 3275 3276 3277 3278 3279 3280
## [1678] 3281 3282 3283 3284 3285 3286 3287 3288 3289 3290 3291 3292 3293
## [1691] 3294 3295 3296 3297 3298 3299 3300 3301 3302 3303 3304 3305 3306
## [1704] 3307 3309 3310 3311 3312 3313 3314 3315 3316 3317 3318 3319 3320
## [1717] 3321 3322 3323 3324 3325 3326 3327 3328 3329 3330 3331 3332 3333
## [1730] 3334 3335 3336 3337 3338 3339 3340 3341 3342 3343 3344 3345 3346
## [1743] 3347 3348 3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359
## [1756] 3360 3361 3362 3363 3364 3365 3366 3367 3368 3369 3370 3371 3372
## [1769] 3373 3374 3375 3376 3377 3378 3379 3380 3381 3382 3384 3385 3386
## [1782] 3387 3388 3389 3390 3391 3392 3393 3394 3395 3396 3397 3398 3399
## [1795] 3400 3401 3402 3403 3404 3405 3406 3407 3408 3409 3410 3411 3412
## [1808] 3413 3414 3415 3416 3417 3418 3419 3420 3421 3422 3423 3424 3425
## [1821] 3426 3427 3428 3429 3430 3431 3432 3433 3434 3436 3437 3438 3439
## [1834] 3440 3441 3442 3443 3444 3445 3446 3447 3448 3449 3450 3451 3452
## [1847] 3453 3454 3455 3457 3458 3459 3460 3461 3462 3463 3464 3466 3467
## [1860] 3468 3469 3470 3471 3472 3473 3474 3475 3476 3477 3478 3479 3480
## [1873] 3481 3482 3483 3484 3485 3486 3487 3488 3489 3490 3491 3492 3493
## [1886] 3494 3495 3496 3497 3498 3499 3500 3501 3502 3503 3504 3505 3506
## [1899] 3507 3508 3509 3510 3511 3512 3513 3514 3515 3516 3517 3518 3519
## [1912] 3520 3521 3522 3523 3524 3525 3527 3528 3529 3530 3531 3533 3534
## [1925] 3535 3536 3537 3538 3539 3540 3541 3542 3543 3544 3545 3546 3547
## [1938] 3548 3549 3550 3552 3553 3554 3555 3556 3557 3558 3559 3560 3561
## [1951] 3562 3563 3564 3565 3566 3567 3568 3569 3570 3571 3572 3573 3574
## [1964] 3575 3576 3577 3578 3579 3580 3581 3582 3583 3584 3585 3586 3587
## [1977] 3588 3589 3590 3591 3592 3593 3594 3595 3596 3597 3598 3599 3600
## [1990] 3601 3602 3603 3604 3605 3606 3607 3608 3609 3610 3611 3612 3613
## [2003] 3614 3615 3616 3617 3618 3619 3620 3621 3622 3623 3625 3627 3628
## [2016] 3629 3630 3631 3632 3633 3634 3635 3636 3637 3638 3639 3640 3641
## [2029] 3642 3643 3644 3645 3646 3647 3648 3649 3650 3651 3652 3653 3654
## [2042] 3655 3656 3657 3658 3659 3660 3661 3662 3663 3664 3665 3666 3667
## [2055] 3668 3669 3670 3671 3672 3673 3674 3675 3676 3677 3678 3679 3680
## [2068] 3681 3682 3683 3684 3685 3686 3687 3688 3689 3690 3691 3693 3694
## [2081] 3695 3696 3697 3698 3699 3700 3701 3702 3703 3704 3705 3706 3707
## [2094] 3708 3709 3710 3711 3712 3713 3714 3715 3716 3717 3718 3719 3720
## [2107] 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733
## [2120] 3734 3735 3736 3737 3738 3739 3740 3741 3742 3743 3744 3745 3746
## [2133] 3747 3748 3749 3750 3751 3752 3753 3755 3756 3757 3758 3759 3760
## [2146] 3761 3763 3764 3765 3766 3767 3768 3769 3770 3771 3772 3773 3774
```

```
## [2159] 3775 3776 3777 3778 3779 3780 3781 3782 3783 3784 3785 3787 3788
## [2172] 3789 3790 3791 3792 3793 3794 3795 3796 3797 3798 3799 3800 3801
## [2185] 3802 3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814
## [2198] 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827
## [2211] 3828 3829 3830 3831 3832 3833 3834 3835 3836 3837 3838 3839 3840
## [2224] 3841 3842 3843 3844 3845 3846 3847 3848 3849 3850 3851 3852 3853
## [2237] 3854 3855 3856 3857 3858 3859 3860 3861 3862 3863 3864 3865 3866
## [2250] 3867 3868 3869 3870 3871 3872 3873 3874 3875 3876 3877 3878 3879
## [2263] 3880 3881 3882 3883 3884 3885 3886 3887 3888 3889 3890 3891 3892
## [2276] 3893 3894 3895 3896 3897 3898 3899 3900 3901 3902 3903 3904 3905
## [2289] 3906 3907 3908 3909 3910 3911 3912 3913 3914 3915 3916 3917 3918
## [2302] 3919 3920 3921 3922 3923 3924 3925 3926 3927 3928 3929 3930 3931
## [2315] 3932 3933 3934 3935 3936 3937 3938 3939 3940 3941 3942 3943 3944
## [2328] 3945 3946 3947 3948 3950 3951 3952 3953 3954 3955 3956 3957 3958
## [2341] 3959 3960 3961 3962 3963 3964 3965 3966 3967 3968 3969 3970 3971
## [2354] 3972 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983 3984 3985
## [2367] 3986 3987 3988 3989 3990 3991 3992 3993 3994 3995 3996 3997 3998
## [2380] 3999 4000 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011
## [2393] 4012 4013 4014 4015 4016 4017 4018 4019 4020 4021 4023 4025 4026
## [2406] 4027 4028 4029 4030 4031 4032 4033 4034 4035 4036 4037 4038 4039
## [2419] 4040 4041 4042 4043 4044 4045 4046 4047 4048 4049 4050 4051 4052
## [2432] 4053 4054 4055 4056 4057 4058 4059 4060 4061 4062 4063 4064 4065
## [2445] 4066 4067 4068 4069 4070 4071 4072 4073 4074 4076 4077 4078 4079
## [2458] 4080 4081 4082 4083 4084 4085 4086 4087 4088 4089 4091 4092 4093
## [2471] 4094 4095 4096 4097 4098 4099 4100 4101 4102 4103 4104 4105 4106
## [2484] 4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4117 4118 4119
## [2497] 4120 4121 4122 4123 4124 4125 4126 4127 4128 4129 4130 4131 4132
## [2510] 4133 4134 4135 4136 4137 4138 4139 4140 4141 4142 4143 4144 4145
## [2523] 4146 4147 4148 4149 4150 4151 4152 4153 4154 4155 4156 4157 4158
## [2536] 4159 4160 4161 4162 4163 4164 4165 4166 4167 4168 4169 4170 4171
## [2549] 4172 4173 4174 4175 4176 4177 4178 4179 4180 4181 4182 4183 4184
## [2562] 4185 4186 4187 4188 4190 4191 4192 4193 4195 4196 4197 4198 4199
## [2575] 4200 4201 4202 4203 4204 4205 4206 4207 4208 4209 4210 4211 4212
## [2588] 4213 4214 4215 4216 4217 4218 4219 4220 4221 4222 4223 4224 4225
## [2601] 4226 4227 4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238
## [2614] 4239 4240 4241 4243 4244 4245 4246 4247 4248 4249 4250 4251 4252
## [2627] 4253 4254 4255 4256 4257 4258 4259 4260 4261 4262 4263 4264 4265
## [2640] 4266 4267 4269 4270 4271 4272 4273 4274 4275 4276 4277 4278 4279
## [2653] 4280 4281 4282 4283 4284 4285 4286 4287 4288 4289 4290 4291 4292
## [2666] 4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304 4305
## [2679] 4306 4307 4308 4309 4310 4311 4312 4313 4314 4315 4316 4317 4318
## [2692] 4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331
## [2705] 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344
## [2718] 4345 4346 4347 4348 4349 4350 4351 4352 4353 4354 4355 4356 4357
## [2731] 4358 4359 4360 4361 4362 4363 4364 4365 4366 4367 4368 4369 4370
## [2744] 4371 4372 4373 4374 4375 4376 4377 4378 4379 4380 4382 4383 4384
## [2757] 4385 4386 4387 4388 4389 4390 4391 4392 4393 4394 4395 4396 4397
## [2770] 4398 4399 4400 4401 4402 4403 4404 4405 4406 4407 4408 4409 4410
## [2783] 4411 4412 4413 4414 4415 4416 4417 4418 4419 4420 4421 4422 4423
## [2796] 4424 4425 4426 4427 4428 4429 4430 4431 4432 4433 4434 4435 4436
```

```
## [2809] 4437 4438 4439 4440 4441 4442 4443 4444 4445 4446 4447 4448 4450
## [2822] 4451 4452 4453 4454 4455 4456 4457 4458 4459 4460 4461 4462 4463
## [2835] 4464 4465 4466 4467 4468 4469 4470 4471 4472 4473 4474 4475 4476
## [2848] 4477 4478 4479 4480 4481 4482 4483 4484 4485 4486 4487 4488 4489
## [2861] 4490 4491 4492 4493 4494 4495 4496 4497 4498 4499 4500 4501 4502
## [2874] 4503 4504 4505 4506 4507 4508 4509 4510 4511 4512 4513 4514 4515
## [2887] 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528
## [2900] 4529 4530 4531 4532 4533 4534 4535 4536 4537 4538 4539 4540 4541
## [2913] 4542 4543 4544 4545 4546 4547 4548 4549 4550 4551 4552 4553 4554
## [2926] 4555 4556 4557 4558 4559 4560 4561 4562 4563 4564 4565 4566 4567
## [2939] 4568 4569 4570 4571 4572 4573 4574 4575 4576 4577 4578 4579 4580
## [2952] 4581 4582 4583 4584 4585 4586 4587 4588 4589 4590 4591 4592 4593
## [2965] 4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606
## [2978] 4607 4608 4609 4610 4611 4612 4613 4614 4615 4616 4617 4618 4619
## [2991] 4620 4621 4622 4623 4624 4625 4626 4627 4628 4629 4630 4631 4632
## [3004] 4633 4634 4635 4636 4637 4638 4639 4640 4641 4642 4643 4644 4645
## [3017] 4646 4647 4648 4649 4650 4651 4652 4653 4654 4655 4656 4657 4658
## [3030] 4659 4660 4661 4662 4663 4664 4665 4666 4667 4668 4669 4670 4671
## [3043] 4672 4673 4674 4675 4676 4677 4678 4679 4680 4681 4682 4683 4684
## [3056] 4685 4686 4687 4688 4689 4690 4691 4692 4693 4694 4695 4696 4697
## [3069] 4698 4699 4700 4701 4702 4703 4704 4705 4706 4707 4708 4709 4710
## [3082] 4711 4712 4713 4714 4715 4716 4717 4718 4719 4720 4721 4722 4723
## [3095] 4724 4725 4726 4727 4728 4729 4730 4731 4732 4733 4734 4735 4736
## [3108] 4737 4738 4739 4740 4741 4742 4743 4744 4745 4746 4747 4748 4749
## [3121] 4750 4751 4752 4753 4754 4755 4756 4757 4758 4759 4760 4761 4762
## [3134] 4763 4764 4765 4766 4767 4768 4769 4770 4771 4772 4773 4774 4775
## [3147] 4776 4777 4778 4779 4780 4781 4782 4783 4784 4785 4786 4787 4788
## [3160] 4789 4790 4791 4792 4793 4794 4795 4796 4797 4798 4799 4800 4801
## [3173] 4802 4803 4804 4805 4806 4807 4808 4809 4810 4811 4812 4813 4814
## [3186] 4815 4816 4817 4818 4819 4820 4821 4822 4823 4824 4825 4826 4827
## [3199] 4828 4829 4830 4831 4832 4833 4834 4835 4836 4837 4838 4839 4840
## [3212] 4841 4842 4843 4844 4845 4846 4847 4848 4849 4850 4851 4852 4853
## [3225] 4854 4855 4856 4857 4858 4859 4860 4861 4862 4863 4864 4865 4866
## [3238] 4867 4868 4869 4870 4871 4872 4873 4874 4875 4876 4877 4878 4879
## [3251] 4880 4881 4882 4883 4884 4885 4886 4887 4888 4889 4890 4891 4892
## [3264] 4893 4894 4895 4896 4897 4898 4899 4900 4901 4902 4903 4904 4905
## [3277] 4906 4907 4908 4909 4910 4911 4912 4913 4914 4915 4916 4917 4918
## [3290] 4919 4920 4921 4922 4923 4924 4925 4926 4927 4928 4929 4930 4931
## [3303] 4932 4933 4934 4935 4936 4937 4938 4939 4940 4941 4942 4943 4944
## [3316] 4945 4946 4947 4948 4949 4950 4951 4952 4953 4954 4955 4956 4957
## [3329] 4958 4959 4960 4961 4962 4963 4964 4965 4966 4967 4968 4969 4970
## [3342] 4971 4972 4973 4974 4975 4976 4977 4978 4979 4980 4981 4982 4983
## [3355] 4984 4985 4986 4987 4988 4989 4990 4991 4992 4993 4994 4995 4996
## [3368] 4997 4998 4999 5000 5001 5002 5003 5004 5005 5006 5007 5008 5009
## [3381] 5010 5011 5012 5013 5014 5015 5016 5017 5018 5019 5020 5021 5022
## [3394] 5023 5024 5025 5026 5027 5028 5029 5030 5031 5032 5033 5034 5035
## [3407] 5036 5037 5038 5039 5040 5041 5042 5043 5044 5045 5046 5047 5048
## [3420] 5049 5050 5051 5052 5053 5054 5055 5056 5057 5058 5059 5060 5061
## [3433] 5062 5063 5064 5065 5066 5067 5068 5069 5070 5071 5072 5073 5074
## [3446] 5075 5076 5077 5078 5079 5080 5081 5082 5083 5084 5085 5086 5087
```

```
## [3459] 5088 5089 5090 5091 5092 5093 5094 5095 5096 5097 5098 5099 5100
## [3472] 5101 5102 5103 5104 5105 5106 5107 5108 5109 5110 5111 5112 5113
## [3485] 5114 5115 5116 5117 5118 5119 5120 5121 5122 5123 5124 5125 5126
## [3498] 5127 5128 5129 5130 5131 5132 5133 5134 5135 5136 5137 5138 5139
## [3511] 5140 5141 5142 5143 5144 5145 5146 5147 5148 5149 5150 5151 5152
## [3524] 5153 5154 5155 5156 5157 5158 5159 5160 5161 5162 5163 5164 5165
## [3537] 5166 5167 5168 5169 5170 5172 5173 5174 5175 5176 5177 5178 5179
## [3550] 5180 5181 5182 5183 5184 5185 5186 5187 5188 5189 5190 5191 5192
## [3563] 5193 5194 5195 5196 5197 5198 5199 5200 5201 5202 5203 5204 5205
## [3576] 5206 5207 5208 5209 5210 5211 5212 5213 5214 5215 5216 5217 5218
## [3589] 5219 5220 5221 5222 5223 5224 5225 5226 5227 5228 5229 5230 5231
## [3602] 5232 5233 5234 5235 5236 5237 5238 5239 5240 5241 5242 5243 5244
## [3615] 5245 5246 5247 5248 5249 5250 5251 5252 5253 5254 5255 5256 5257
## [3628] 5258 5259 5260 5261 5263 5264 5265 5266 5267 5268 5269 5270 5271
## [3641] 5272 5273 5274 5275 5276 5277 5278 5279 5280 5281 5282 5283 5284
## [3654] 5285 5286 5287 5288 5289 5290 5291 5292 5293 5294 5295 5296 5297
## [3667] 5298 5299 5300 5301 5302 5303 5304 5305 5306 5307 5308 5309 5310
## [3680] 5311 5312 5313 5314 5315 5316 5317 5318 5319 5320 5321 5322 5323
## [3693] 5324 5325 5326 5327 5328 5329 5330 5331 5332 5333 5334 5335 5336
## [3706] 5337 5338 5339 5340 5341 5342 5343 5344 5345 5346 5347 5348 5349
## [3719] 5350 5351 5352 5353 5354 5355 5356 5357 5358 5359 5360 5361 5362
## [3732] 5363 5364 5365 5366 5367 5368 5369 5370 5371 5372 5373 5374 5375
## [3745] 5376 5377 5378 5379 5380 5381 5382 5383 5384 5385 5386 5387 5388
## [3758] 5389 5390 5391 5392 5393 5394 5395 5396 5397 5398 5399 5400 5401
## [3771] 5402 5403 5404 5405 5406 5407 5408 5409 5410 5411 5412 5413 5414
## [3784] 5415 5416 5417 5418 5419 5420 5421 5422 5423 5424 5425 5426 5427
## [3797] 5428 5429 5430 5431 5432 5433 5434 5435 5436 5437 5438 5439 5440
## [3810] 5441 5442 5443 5444 5445 5446 5447 5449 5450 5451 5452 5453 5454
## [3823] 5455 5456 5457 5458 5459 5460 5461 5462 5463 5464 5465 5466 5467
## [3836] 5468 5469 5470 5471 5472 5473 5474 5475 5476 5477 5478 5479 5480
## [3849] 5481 5482 5483 5484 5485 5486 5487 5488 5489 5490 5491 5492 5493
## [3862] 5494 5495 5496 5497 5498 5499 5500 5502 5503 5504 5505 5506 5507
## [3875] 5508 5509 5510 5511 5512 5513 5514 5515 5516 5517 5518 5519 5520
## [3888] 5521 5522 5523 5524 5525 5526 5527 5528 5529 5530 5531 5532 5533
## [3901] 5534 5535 5536 5537 5538 5539 5540 5541 5542 5543 5544 5545 5546
## [3914] 5547 5548 5549 5550 5551 5552 5553 5554 5555 5556 5557 5558 5559
## [3927] 5560 5561 5562 5563 5564 5565 5566 5567 5568 5569 5570 5571 5573
## [3940] 5574 5575 5576 5577 5578 5579 5580 5581 5582 5583 5584 5585 5586
## [3953] 5587 5588 5589 5590 5591 5592 5593 5594 5595 5596 5597 5598 5599
## [3966] 5600 5601 5602 5603 5604 5605 5606 5607 5608 5609 5610 5611 5612
## [3979] 5613 5614 5615 5616 5617 5618 5619 5620 5621 5622 5623 5624 5625
## [3992] 5626 5627 5628 5629 5630 5631 5632 5633 5634 5635 5636 5637 5638
## [4005] 5640 5641 5642 5643 5644 5645 5646 5647 5648 5649 5650 5651 5652
## [4018] 5653 5654 5655 5656 5657 5658 5659 5660 5661 5662 5663 5664 5665
## [4031] 5666 5667 5668 5669 5670 5671 5672 5673 5674 5675 5676 5677 5678
## [4044] 5679 5680 5681 5682 5683 5684 5685 5686 5687 5688 5689 5690 5691
## [4057] 5692 5693 5694 5695 5696 5697 5698 5699 5700 5701 5702 5703 5704
## [4070] 5705 5706 5707 5708 5709 5710 5711 5712 5713 5714 5715 5716 5717
## [4083] 5718 5719 5720 5721 5722 5723 5724 5725 5726 5727 5728 5729 5730
## [4096] 5731 5732 5733 5734 5735 5736 5737 5738 5739 5740 5741 5742 5743
```

```
## [4109] 5744 5745 5746 5747 5748 5749 5750 5751 5752 5753 5754 5755 5756
## [4122] 5757 5758 5759 5760 5761 5762 5763 5764 5765 5766 5767 5768 5769
## [4135] 5770 5771 5772 5773 5774 5775 5776 5777 5778 5779 5780 5781 5782
## [4148] 5783 5784 5785 5786 5787 5788 5789 5790 5791 5792 5793 5794 5795
## [4161] 5796 5797 5798 5799 5800 5801 5802 5803 5804 5805 5806 5807 5808
## [4174] 5809 5810 5811 5812 5814 5815 5816 5817 5818 5819 5820 5821 5822
## [4187] 5823 5824 5825 5826 5827 5828 5829 5830 5831 5832 5833 5834 5835
## [4200] 5836 5837 5838 5839 5840 5841 5842 5843 5844 5845 5846 5847 5848
## [4213] 5849 5850 5851 5852 5853 5854 5855 5856 5857 5858 5859 5860 5861
## [4226] 5862 5863 5864 5865 5866 5867 5868 5869 5870 5871 5872 5873 5874
## [4239] 5875 5876 5877 5878 5879 5880 5881 5882 5883 5884 5885 5886 5887
## [4252] 5888 5889 5890 5891 5892 5893 5894 5895 5896 5897 5898 5899 5900
## [4265] 5901 5902 5903 5904 5905 5906 5907 5908 5909 5910 5911 5912 5913
## [4278] 5914 5915 5917 5918 5919 5920 5921 5922 5923 5924 5925 5926 5927
## [4291] 5928 5929 5930 5931 5932 5933 5934 5935 5936 5937 5938 5939 5940
## [4304] 5941 5942 5943 5944 5945 5946 5947 5948 5949 5950 5951 5952 5953
## [4317] 5954 5955 5956 5957 5958 5959 5960 5961 5962 5963 5964 5965 5966
## [4330] 5967 5968 5969 5970 5971 5972 5973 5974 5975 5976 5977 5978 5979
## [4343] 5980 5981 5982 5983 5984 5985 5986 5987 5988 5989 5990 5991 5992
## [4356] 5993 5994 5995 5996 5997 5998 5999 6000 6001 6002 6003 6004 6005
## [4369] 6006 6007 6008 6009 6010 6011 6012 6013 6014 6015 6016 6017 6018
## [4382] 6019 6020 6021 6022 6023 6024 6025 6026 6027 6028 6029 6030 6031
## [4395] 6032 6033 6034 6035 6036 6037 6038 6039 6040 6041 6042 6043 6044
## [4408] 6045 6046 6047 6048 6049 6050 6051 6052 6053 6054 6055 6056 6057
## [4421] 6058 6059 6060 6061 6062 6063 6064 6065 6066 6067 6068 6069 6070
## [4434] 6071 6072 6074 6075 6076 6077 6078 6079 6080 6081 6082 6083 6084
## [4447] 6085 6086 6087 6088 6089 6090 6091 6092 6093 6094 6095 6096 6097
## [4460] 6098 6099 6100 6101 6102 6103 6104 6105 6106 6107 6108 6109 6110
## [4473] 6111 6112 6113 6114 6115 6116 6117 6118 6119 6120 6121 6122 6123
## [4486] 6124 6125 6126 6127 6128 6129 6130 6131 6132 6133 6134 6135 6136
## [4499] 6137 6138 6139 6140 6141 6142 6143 6144 6145 6146 6147 6148 6149
## [4512] 6150 6151 6152 6153 6154 6155 6156 6157 6158 6159 6160 6161 6162
## [4525] 6163 6164 6165 6166 6167 6168 6169 6170 6171 6172 6173 6174 6175
## [4538] 6176 6177 6178 6179 6180 6181 6182 6183 6184 6185 6186 6187 6188
## [4551] 6189 6190 6191 6192 6193 6194 6195 6196 6197 6198 6199 6200 6201
## [4564] 6202 6203 6204 6205 6206 6207 6208 6209 6210 6211 6212 6213 6214
## [4577] 6215 6216 6217 6218 6219 6220 6221 6222 6223 6224 6225 6226 6227
## [4590] 6228 6229 6230 6231 6232 6233 6234 6235 6236 6237 6238 6239 6240
## [4603] 6241 6242 6243 6244 6245 6246 6247 6248 6251 6252 6253 6254 6255
## [4616] 6256 6257 6258 6259 6260 6261 6262 6263 6264 6265 6266 6267 6268
## [4629] 6269 6270 6271 6272 6273 6274 6275 6276 6277 6278 6279 6280 6281
## [4642] 6282 6283 6284 6285 6286 6287 6288 6289 6290 6291 6292 6293 6294
## [4655] 6295 6296 6297 6298 6299 6300 6301 6302 6303 6304 6305 6306 6307
## [4668] 6308 6309 6310 6311 6312 6313 6314 6315 6316 6317 6318 6319 6320
## [4681] 6321 6322 6323 6324 6325 6326 6327 6328 6329 6330 6331 6332 6333
## [4694] 6334 6335 6336 6337 6338 6339 6340 6341 6342 6343 6344 6345 6346
## [4707] 6347 6348 6349 6350 6351 6352 6353 6354 6355 6356 6357 6358 6359
## [4720] 6360 6361 6362 6363 6364 6365 6366 6367 6368 6369 6370 6371 6372
## [4733] 6373 6374 6375 6376 6377 6378 6379 6380 6381 6382 6383 6384 6385
## [4746] 6386 6387 6388 6389 6390 6391 6393 6394 6395 6396 6397 6398 6399
```

```
## [4759] 6400 6401 6402 6403 6404 6405 6406 6407 6408 6409 6410 6411 6412
## [4772] 6413 6414 6416 6417 6418 6419 6420 6421 6422 6423 6424 6425 6426
## [4785] 6427 6428 6429 6430 6431 6432 6433 6434 6435 6436 6437 6438 6439
## [4798] 6440 6441 6442 6443 6444 6446 6447 6448 6449 6450 6451 6452 6453
## [4811] 6454 6455 6456 6457 6458 6459 6460 6461 6462 6463 6464 6465 6466
## [4824] 6467 6468 6469 6470 6471 6472 6473 6474 6475 6476 6477 6478 6479
## [4837] 6480 6481 6482 6483 6484 6485 6486 6487 6488 6489 6490 6491 6492
## [4850] 6493 6494 6495 6496 6497
```

```
which(cluster_2$cluster == 2)
```

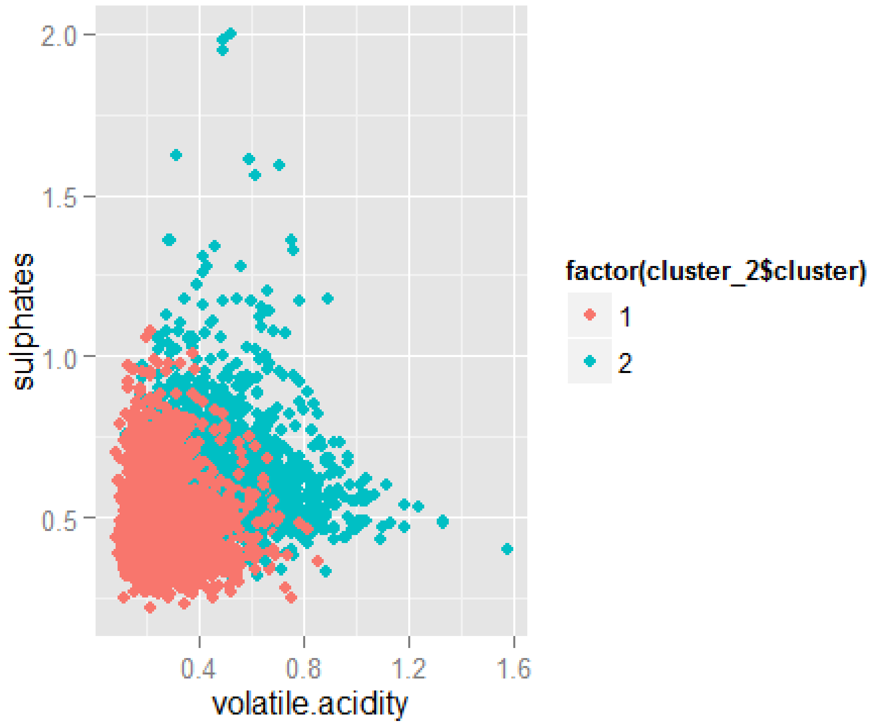
```
##    [1]    1    2    3    4    5    6    7    8    9   10   11   12   13
##   [14]   14   15   16   17   18   19   20   21   22   23   24   25   26
##   [27]   27   28   29   30   31   32   33   34   35   36   37   38   39
##   [40]   40   41   42   43   44   45   46   47   48   49   51   52   53
##   [53]   54   55   56   57   58   59   60   61   62   63   64   65   66
##   [66]   67   68   69   70   71   72   73   74   75   76   77   78   79
##   [79]   80   81   82   83   84   85   86   87   88   89   90   91   92
##   [92]   93   94   95   96   97   98   99  100  101  102  103  104  105
##  [105]  106  107  108  109  110  111  112  113  114  115  116  117  118
##  [118]  119  120  121  122  123  124  125  126  127  128  129  130  131
##  [131]  132  133  134  135  136  137  138  139  140  141  142  143  144
##  [144]  145  146  147  148  149  150  151  152  153  154  155  156  157
##  [157]  158  159  160  161  162  163  164  165  166  167  168  169  170
##  [170]  171  172  173  174  175  176  177  178  179  180  181  182  183
##  [183]  184  185  186  187  188  189  190  191  192  193  194  195  196
##  [196]  197  198  199  200  201  202  203  204  205  206  207  208  209
##  [209]  210  211  212  213  214  215  216  217  218  219  220  221  222
##  [222]  223  224  225  226  227  228  229  230  231  232  233  234  235
##  [235]  236  237  238  239  240  241  242  243  244  245  246  247  248
##  [248]  249  250  251  252  253  254  255  256  257  258  259  260  261
##  [261]  262  263  264  265  266  267  268  269  270  271  272  273  274
##  [274]  275  276  277  278  279  280  281  282  283  284  285  286  287
##  [287]  288  289  290  291  292  293  294  295  296  297  298  299  300
##  [300]  301  302  303  304  305  306  307  308  309  310  311  312  313
##  [313]  314  315  316  317  318  319  320  321  322  323  324  325  326
##  [326]  327  328  329  330  331  332  333  334  335  336  337  338  339
##  [339]  340  341  342  343  344  345  346  347  348  349  350  351  352
##  [352]  353  354  356  357  358  359  360  361  362  363  364  365  366
##  [365]  367  368  369  370  371  372  373  374  375  376  377  378  379
##  [378]  380  381  382  383  384  385  386  387  388  389  390  391  392
##  [391]  393  394  395  396  397  398  399  400  401  402  403  404  405
##  [404]  406  407  408  409  410  411  412  413  414  415  416  417  418
##  [417]  419  420  421  422  423  424  425  426  427  428  429  430  431
##  [430]  432  433  434  435  436  437  438  439  440  441  442  443  444
##  [443]  445  446  447  448  449  450  451  452  453  454  455  456  457
##  [456]  458  459  460  461  462  463  464  465  466  467  468  469  470
##  [469]  471  472  473  474  475  476  477  478  479  480  481  482  483
##  [482]  484  485  486  487  488  489  490  491  492  493  494  496  497
##  [495]  498  499  500  501  502  503  504  505  506  507  508  509  510
```

```
##  [508]  511  512  513  514  515  516  517  518  519  520  521  522  523
##  [521]  524  525  526  527  528  529  530  531  532  533  534  535  536
##  [534]  537  538  539  540  541  542  543  544  545  546  547  548  549
##  [547]  550  551  552  553  554  555  556  557  558  559  560  561  562
##  [560]  563  564  565  566  567  568  569  570  571  572  573  574  575
##  [573]  576  577  578  579  580  581  582  583  584  585  586  587  588
##  [586]  589  590  591  593  594  595  596  597  598  599  600  601  602
##  [599]  603  604  605  606  607  608  609  610  611  612  613  614  615
##  [612]  616  617  618  619  620  621  622  623  624  625  626  627  628
##  [625]  629  630  631  632  633  634  636  637  638  639  640  641  642
##  [638]  643  644  645  646  647  648  649  651  652  653  654  655  656
##  [651]  657  658  659  660  661  662  663  664  665  666  667  668  669
##  [664]  670  671  672  673  674  675  676  677  678  679  680  681  682
##  [677]  683  684  685  686  687  688  689  690  691  692  693  694  695
##  [690]  696  697  698  699  700  701  702  703  704  705  706  707  708
##  [703]  709  710  711  712  713  714  715  716  717  718  719  720  721
##  [716]  722  723  724  725  726  727  728  729  730  731  732  733  734
##  [729]  735  736  737  738  739  740  741  742  743  744  745  746  747
##  [742]  748  749  750  751  752  753  754  755  756  757  758  759  760
##  [755]  761  762  763  764  765  766  767  768  769  770  771  772  773
##  [768]  774  775  776  777  778  779  780  781  782  783  784  785  786
##  [781]  787  788  789  790  791  792  793  794  795  796  797  798  799
##  [794]  800  801  802  803  804  805  806  807  808  809  810  811  812
##  [807]  813  814  815  816  817  818  819  820  821  822  823  824  825
##  [820]  826  827  828  829  830  831  832  833  834  835  836  839  840
##  [833]  841  842  843  844  845  846  847  848  849  850  851  852  853
##  [846]  854  855  856  857  858  859  860  861  862  863  864  865  866
##  [859]  867  868  869  870  871  872  873  874  875  876  877  878  879
##  [872]  880  881  882  883  884  885  886  887  888  889  890  891  892
##  [885]  893  894  895  896  897  898  899  900  901  902  903  904  905
##  [898]  906  907  908  909  910  911  912  913  914  915  916  917  918
##  [911]  919  920  921  922  923  924  925  926  927  928  929  930  931
##  [924]  932  933  934  935  936  937  938  939  940  941  942  943  944
##  [937]  945  946  947  948  949  950  951  952  953  954  955  956  957
##  [950]  958  959  960  961  962  963  964  965  966  967  968  969  970
##  [963]  971  972  973  974  975  976  977  978  979  980  981  982  983
##  [976]  984  985  986  987  988  989  990  991  992  993  994  995  996
##  [989]  997  998  999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009
## [1002] 1010 1011 1012 1013 1014 1015 1016 1017 1020 1021 1022 1023 1024
## [1015] 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037
## [1028] 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
## [1041] 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063
## [1054] 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076
## [1067] 1077 1078 1079 1081 1083 1084 1085 1086 1087 1088 1089 1090 1092
## [1080] 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105
## [1093] 1106 1107 1108 1109 1110 1111 1112 1113 1114 1116 1117 1118 1119
## [1106] 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1133
## [1119] 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146
## [1132] 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1158 1159 1160
## [1145] 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173
```

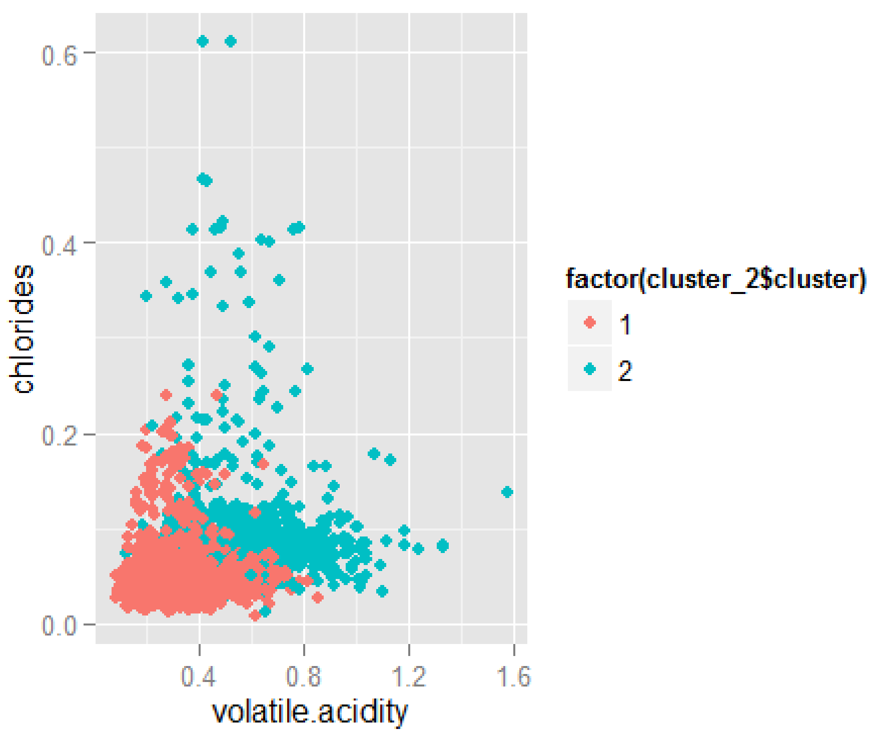
```
## [1158] 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186
## [1171] 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199
## [1184] 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212
## [1197] 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225
## [1210] 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1237 1238 1239
## [1223] 1240 1241 1242 1243 1244 1246 1247 1248 1249 1250 1251 1252 1253
## [1236] 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266
## [1249] 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
## [1262] 1280 1281 1282 1283 1284 1285 1286 1288 1289 1290 1291 1292 1293
## [1275] 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306
## [1288] 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319
## [1301] 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332
## [1314] 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345
## [1327] 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358
## [1340] 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371
## [1353] 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384
## [1366] 1385 1386 1387 1388 1389 1391 1392 1393 1394 1395 1396 1397 1398
## [1379] 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411
## [1392] 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424
## [1405] 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437
## [1418] 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450
## [1431] 1451 1452 1453 1454 1455 1456 1458 1459 1460 1461 1462 1463 1464
## [1444] 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477
## [1457] 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
## [1470] 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
## [1483] 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517
## [1496] 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530
## [1509] 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543
## [1522] 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556
## [1535] 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1568 1569 1570
## [1548] 1571 1572 1573 1574 1576 1577 1578 1579 1580 1581 1582 1583 1584
## [1561] 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597
## [1574] 1598 1599 1715 1747 1754 1808 1830 1972 2033 2084 2106 2209 2226
## [1587] 2262 2287 2430 2434 2548 2634 2636 2637 2640 2714 2752 2817 2872
## [1600] 2986 2994 3096 3177 3308 3383 3435 3456 3465 3526 3532 3551 3624
## [1613] 3626 3692 3754 3762 3786 3949 3973 4022 4024 4075 4090 4189 4194
## [1626] 4242 4268 4381 4449 5171 5262 5448 5501 5572 5639 5813 5916 6073
## [1639] 6249 6250 6392 6415 6445
```

Plots displaying cluster association

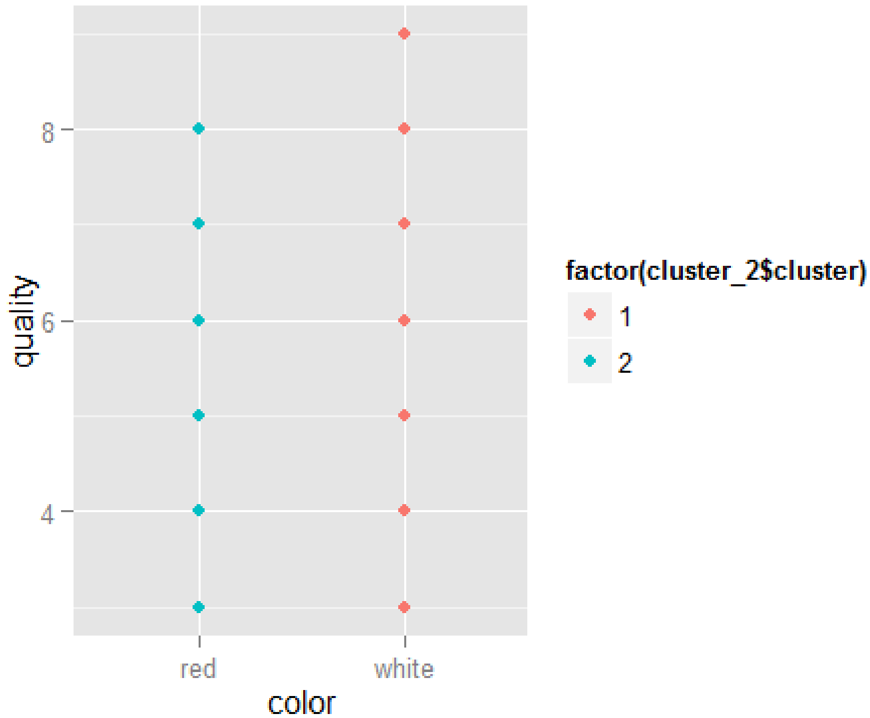
```
qplot(volatile.acidity,sulphates, data=wine, color=factor(cluster_2$cluster))
```




```
qplot(volatile.acidity, chlorides, data=wine,
color=factor(cluster_2$cluster))
```




```
qplot(color, quality, data=wine, color=factor(cluster_2$cluster))
```




```
qplot(wine$color, cluster_2$cluster, data=wine,
color=factor(cluster_2$cluster))
```

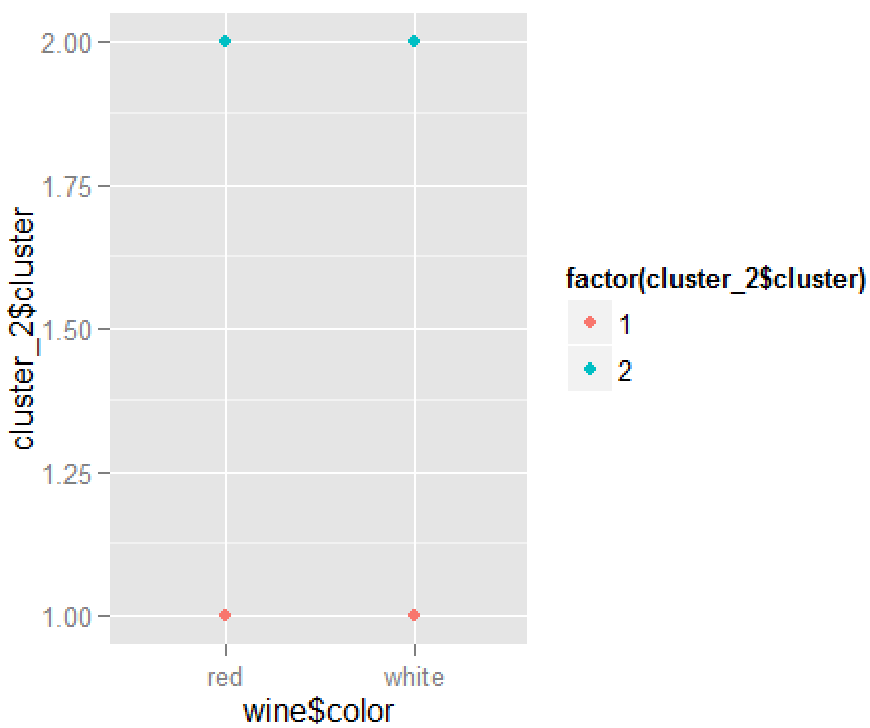


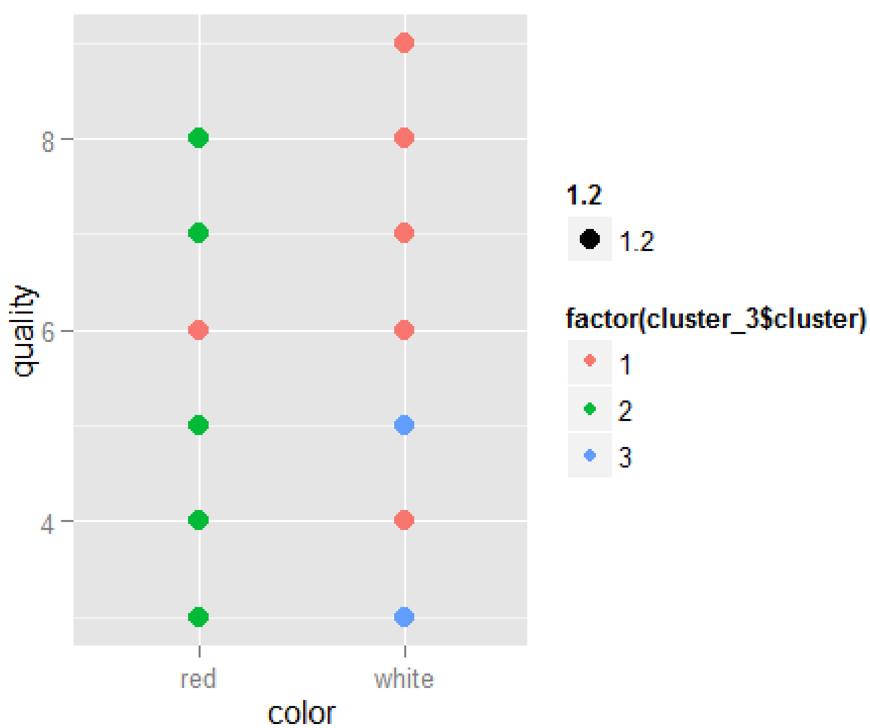
Table displaying that the clusters succesfully split red and white wine with minimal error

```
table(wine$color,cluster_2$cluster)

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

Quality Clustering

Cluster the data into 3 groups

```
cluster_3 = kmeans(wine_scaled, centers = 3, nstart = 50)
table(wine$quality,cluster_3$cluster)

##
##         1    2    3
##   3     8   10   12
##   4   100   68   48
##   5   638  696  804
##   6  1371  622  843
##   7   740  182  157
##   8   148   15   30
##   9     4    0    1

qplot(color, quality, data=wine, color=factor(cluster_3$cluster),cex= 1.2)
```



It is dificult to distinguish the wines by quality.

PCA

Run PCA on the data and look at the result

```
pca_wine = prcomp(wine_num, scale.=TRUE)
pca_wine
```

```
## Standard deviations:
##  [1] 1.7406518 1.5791852 1.2475364 0.9851660 0.8484544 0.7793021 0.7232971
##  [8] 0.7081739 0.5805377 0.4771748 0.1811927
## 
## Rotation:
##                              PC1         PC2         PC3         PC4
## fixed.acidity        -0.23879890  0.33635454 -0.43430130  0.16434621
## volatile.acidity     -0.38075750  0.11754972  0.30725942  0.21278489
## citric.acid           0.15238844  0.18329940 -0.59056967 -0.26430031
## residual.sugar        0.34591993  0.32991418  0.16468843  0.16744301
## chlorides            -0.29011259  0.31525799  0.01667910 -0.24474386
## free.sulfur.dioxide   0.43091401  0.07193260  0.13422395 -0.35727894
## total.sulfur.dioxide  0.48741806  0.08726628  0.10746230 -0.20842014
## density              -0.04493664  0.58403734  0.17560555  0.07272496
## pH                   -0.21868644 -0.15586900  0.45532412 -0.41455110
## sulphates            -0.29413517  0.19171577 -0.07004248 -0.64053571
## alcohol              -0.10643712 -0.46505769 -0.26110053 -0.10680270
##                             PC5         PC6         PC7          PC8
## fixed.acidity        -0.1474804 -0.20455371 -0.28307944  0.401235645
## volatile.acidity      0.1514560 -0.49214307 -0.38915976 -0.087435088
## citric.acid          -0.1553487  0.22763380 -0.38128504 -0.293412336
## residual.sugar       -0.3533619 -0.23347775  0.21797554 -0.524872935
## chlorides             0.6143911  0.16097639 -0.04606816 -0.471516850
## free.sulfur.dioxide   0.2235323 -0.34005140 -0.29936325  0.207807585
## total.sulfur.dioxide  0.1581336 -0.15127722 -0.13891032  0.128621319
## density              -0.3065613  0.01874307 -0.04675897  0.004831136
## pH                   -0.4533764  0.29657890 -0.41890702 -0.028643277
## sulphates            -0.1365769 -0.29692579  0.52534311  0.165818022
## alcohol              -0.1888920 -0.51837780 -0.10410343 -0.399233887
##                             PC9         PC10          PC11
## fixed.acidity         0.3440567 -0.281267685 -0.3346792663
## volatile.acidity     -0.4969327  0.152176731 -0.0847718098
## citric.acid          -0.4026887  0.234463340  0.0011089514
## residual.sugar        0.1080032 -0.001372773 -0.4497650778
## chlorides             0.2964437 -0.196630217 -0.0434375867
## free.sulfur.dioxide   0.3666563  0.480243340  0.0002125351
## total.sulfur.dioxide -0.3206955 -0.713663486  0.0626848131
## density               0.1128800 -0.003908289  0.7151620723
## pH                    0.1278367 -0.141310977 -0.2063605036
## sulphates            -0.2077642  0.045959499 -0.0772024671
## alcohol               0.2518903 -0.205053085  0.3357018784
```

See some statistics and plots related to the PCA

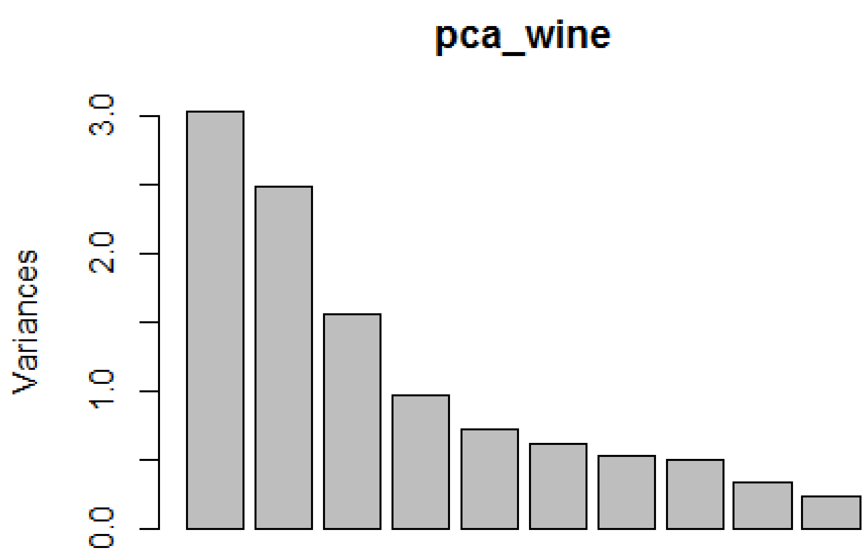
```r
summary(pca_wine)
```

```
## 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
```

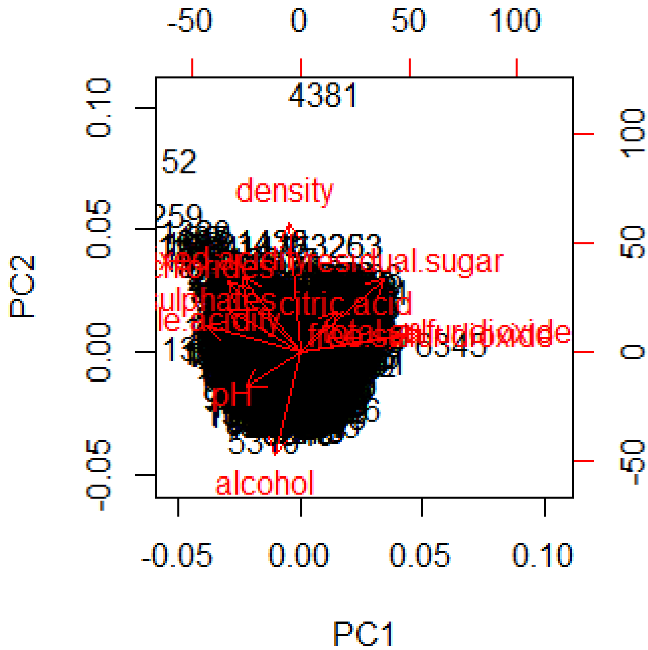
```r
sum((pca_wine$sdev)^2)
```

```
## [1] 11
```

```r
plot(pca_wine)
```

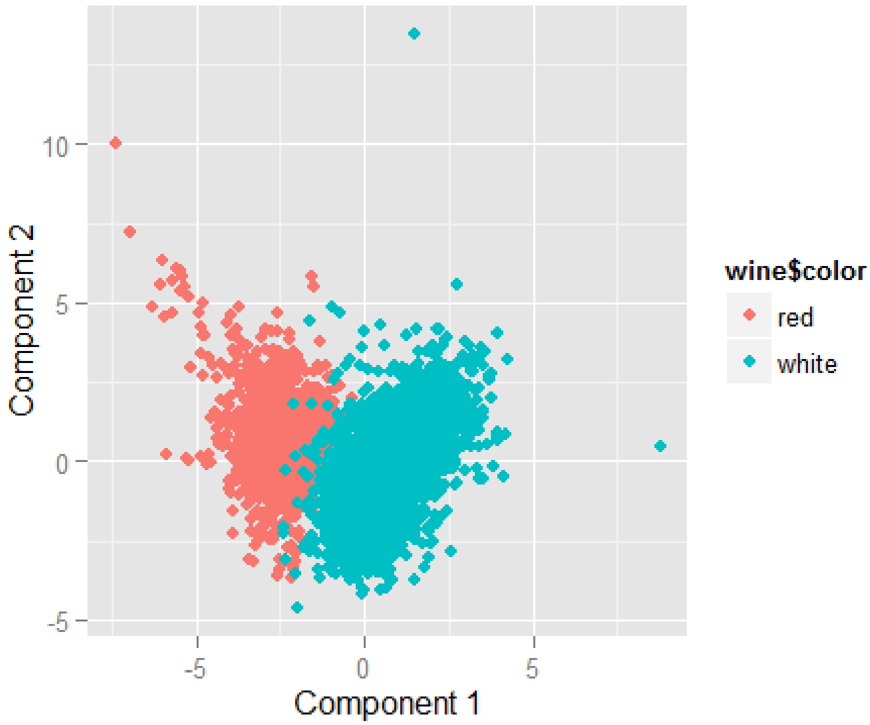



```r
biplot(pca_wine)
```



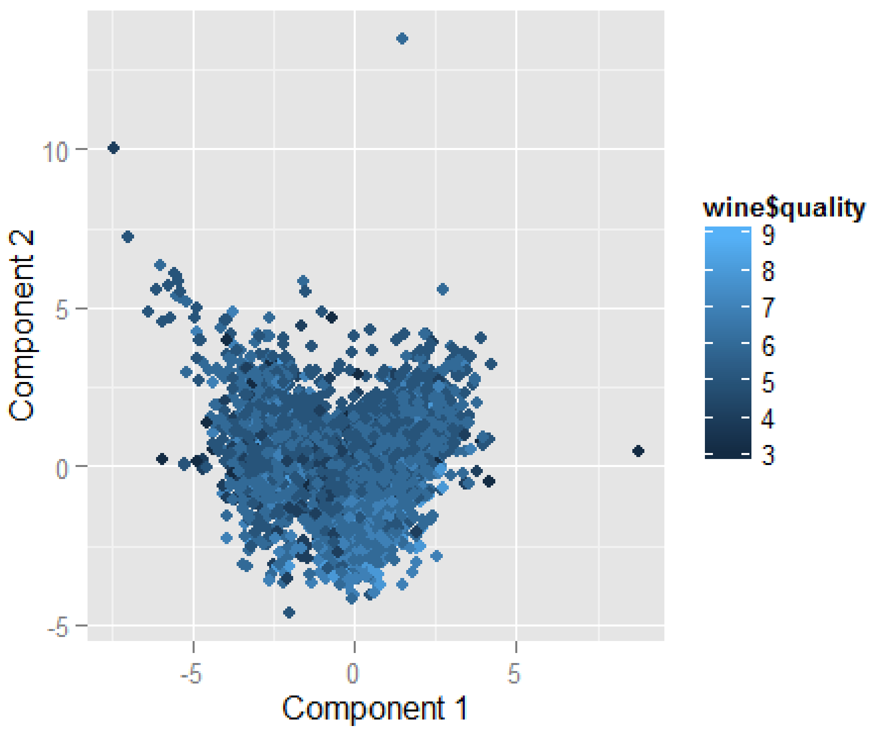
By looking at the score and plotting them, we can again see the distinction from red and white wines

```
scores = pca_wine$x
head(scores)

##             PC1       PC2       PC3       PC4        PC5        PC6
## [1,] -3.205749 0.4164913  2.722027 0.7967162 -0.2028619  0.2273453
## [2,] -3.038817 1.1073769  2.046795 0.7701656  1.3225536 -1.6549941
## [3,] -3.071657 0.8788968  1.742445 0.8021955  0.7620531 -0.8483083
## [4,] -1.571141 2.1123820 -2.592717 0.2927620 -0.6046542  0.7132533
## [5,] -3.205749 0.4164913  2.722027 0.7967162 -0.2028619  0.2273453
## [6,] -3.011934 0.3893675  2.669970 0.6861020 -0.2077957  0.2928529
##              PC7       PC8         PC9        PC10        PC11
## [1,] -0.32552850 0.5672348  0.07122341  0.10803795  0.02745798
## [2,]  0.05955408 0.5145630 -0.42909559  0.26812827 -0.01547017
## [3,]  0.16765673 0.4209200 -0.27101087  0.08682522  0.05414179
## [4,] -0.85115161 0.9295970  0.54936730 -0.11665069 -0.10373671
## [5,] -0.32552850 0.5672348  0.07122341  0.10803795  0.02745798
## [6,] -0.28272294 0.6500382  0.18849800  0.05506242  0.06542536

nrow(scores)

## [1] 6497

qplot(scores[,1], scores[,2], color=wine$color, xlab='Component 1',
ylab='Component 2')
```



Again when plotting to see quality, it is dificult to see any distinction.

```
qplot(scores[,1], scores[,2], color=wine$quality, xlab='Component 1',
ylab='Component 2')
```



In conclusion I beleive clustering is more useful in evaluating this data. The plots formed from clustering claerly seperated red and white wines. Additionally by using the table function we were able to see that clustering distinguished between the wines with a high level of precision.

(4) Market Segmentation

K-Means clustering

Read in the data

```
twitter = read.csv('social_marketing.csv')

head(twitter)

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

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

```r
names(twitter)
```

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

Remove the 1st and last 2 columns of the dataset and scale the data.

```
twitter_num = twitter [,(2:35)]
twitter_scaled = scale(twitter_num, center = TRUE, scale = TRUE)
```

cluster the data using 5 centers, in hopes of distinguishing market segments.

```
cluster_twitter = kmeans(twitter_scaled, centers = 5, nstart = 200)
```

Capture the mean and SD of the scaled data.

```
sigma_twitter = attr(twitter_scaled,"scaled:scale")
mu_twitter = attr(twitter_scaled,"scaled:center")
```

Unscale the data.

```
cluster_twitter$center

##        chatter current_events      travel photo_sharing uncategorized
## 1 -0.086222413     0.11809608 -0.09771495   -0.06537001   -0.07560096
## 2  0.722175675     0.27442218 -0.03596486    1.03832015    0.45534783
## 3 -0.212834402    -0.12948720 -0.23381547   -0.31620751   -0.16643598
## 4 -0.006995531     0.10556173  1.83977588   -0.10723104   -0.04382044
## 5 -0.084492666    -0.01124003 -0.15545173   -0.04798195    0.14853649
##        tv_film sports_fandom   politics        food      family
## 1  0.000746473     2.0470620 -0.1982928  1.81765091  1.49765506
## 2  0.485228283    -0.1645516 -0.1310832 -0.15553966  0.03722871
## 3 -0.149523564    -0.2977550 -0.2727615 -0.36584951 -0.27118944
## 4  0.033096281     0.1920158  2.4282822  0.03297433  0.05045762
## 5 -0.104380041    -0.1939706 -0.1810003  0.43202406 -0.07556437
##   home_and_garden       music        news online_gaming     shopping
## 1       0.1706871  0.04803399 -0.06700095    0.02473461 -0.001135387
## 2       0.2660681  0.57282236 -0.13054926    0.43242891  0.742711633
## 3      -0.1701953 -0.20187397 -0.24809584   -0.12679472 -0.243105271
## 4       0.1185947 -0.05033662  1.93533117   -0.07604167 -0.057068483
## 5       0.1418000  0.02622923 -0.03255397   -0.06324663 -0.007390447
##   health_nutrition college_uni sports_playing     cooking         eco
## 1       -0.1528754 -0.03268345     0.14438408 -0.08800954  0.17248722
## 2       -0.1815311  0.58926171     0.50358625  0.86228797  0.23181896
## 3       -0.3295290 -0.15720684    -0.20697001 -0.33998540 -0.23646305
## 4       -0.2023987 -0.03276814     0.02766946 -0.19240063  0.09727734
## 5        2.1505827 -0.15849832     0.02193048  0.43667699  0.52817655
##     computers   business    outdoors      crafts  automotive         art
## 1  0.07670276  0.1215802 -0.06801627  0.70338638  0.17590291  0.11089822
## 2 -0.02412705  0.3721702 -0.11253202  0.29050312  0.08501058  0.43825776
## 3 -0.25278692 -0.2097674 -0.31512163 -0.25859111 -0.20677952 -0.15790482
## 4  1.64069987  0.3293340  0.10575747  0.10738581  1.07469608 -0.04970671
## 5 -0.07939718  0.0374073  1.66131375  0.08062137 -0.12643014 -0.01768315
##      religion     beauty   parenting      dating      school
## 1  2.25268080  0.3124905  2.12069484 -0.01192335  1.65716669
## 2 -0.16518697  0.8599830 -0.13690144  0.30780026  0.15654677
## 3 -0.30436878 -0.2904203 -0.31199940 -0.17031941 -0.30572877
## 4 -0.03036794 -0.1505644  0.02142103  0.20654026 -0.04530884
## 5 -0.16501662 -0.1669719 -0.10269878  0.16158299 -0.16709797
```

```
##   personal_fitness     fashion small_business
## 1      -0.09784654  0.02696846     0.09556896
## 2      -0.14604173  0.96454926     0.44854802
## 3      -0.34409012 -0.29434889    -0.18012218
## 4      -0.19706439 -0.15869840     0.22739613
## 5       2.11143052 -0.07063281    -0.12811374
```

```
cluster_twitter$center[1,]*sigma_twitter + mu_twitter
```

```
##          chatter   current_events           travel    photo_sharing 
##        4.0944669        1.6761134        1.3616734        2.5182186 
##    uncategorized          tv_film    sports_fandom         politics 
##        0.7422402        1.0715250        6.0175439        1.1875843 
##             food           family  home_and_garden            music 
##        4.6248313        2.5600540        0.6464238        0.7287449 
##             news    online_gaming         shopping health_nutrition 
##        1.0647773        1.2753036        1.3873144        1.8798920 
##      college_uni   sports_playing          cooking              eco 
##        1.4547908        0.7800270        1.6963563        0.6450742 
##        computers         business         outdoors           crafts 
##        0.7395412        0.5074224        0.7004049        1.0904184 
##       automotive              art         religion           beauty 
##        1.0701754        0.9055331        5.4089069        1.1201080 
##        parenting           dating           school personal_fitness 
##        4.1349528        0.6896086        2.7368421        1.2267206 
##          fashion   small_business 
##        1.0458839        0.3954116
```

```
cluster_twitter$center[2,]*sigma_twitter + mu_twitter
```

```
##          chatter   current_events           travel    photo_sharing 
##        6.9474053        1.8744741        1.5028050        5.5329593 
##    uncategorized          tv_film    sports_fandom         politics 
##        1.2391304        1.8751753        1.2384292        1.3913043 
##             food           family  home_and_garden            music 
##        1.1213184        0.9060309        0.7166900        1.2692847 
##             news    online_gaming         shopping health_nutrition 
##        0.9312763        2.3709677        2.7328191        1.7510519 
##      college_uni   sports_playing          cooking              eco 
##        3.2566620        1.1304348        4.9558205        0.6907433 
##        computers         business         outdoors           crafts 
##        0.6206171        0.6809257        0.6465638        0.7531557 
##       automotive              art         religion           beauty 
##        0.9460028        1.4389902        0.7791024        1.8471248 
##        parenting           dating           school personal_fitness 
##        0.7138850        1.2594670        0.9537167        1.1107994 
##          fashion   small_business 
##        2.7601683        0.6136045
```

See attributes are emphasized in each cluster to facilitate targeting

```r
#Young women. Care about fashion, beauty, cooking, shopping, and photo
sharing
rbind(cluster_twitter$center[1,],cluster_twitter$center[1,]*sigma_twitter +
mu_twitter)
```

```
##          chatter current_events      travel photo_sharing uncategorized
## [1,] -0.08622241      0.1180961 -0.09771495   -0.06537001   -0.07560096
## [2,]  4.09446694      1.6761134  1.36167341    2.51821862    0.74224022
##          tv_film sports_fandom   politics     food   family
## [1,] 0.000746473      2.047062 -0.1982928 1.817651 1.497655
## [2,] 1.071524966      6.017544  1.1875843 4.624831 2.560054
##      home_and_garden      music        news online_gaming     shopping
## [1,]       0.1706871 0.04803399 -0.06700095    0.02473461 -0.001135387
## [2,]       0.6464238 0.72874494  1.06477733    1.27530364  1.387314440
##      health_nutrition college_uni sports_playing     cooking       eco
## [1,]       -0.1528754 -0.03268345      0.1443841 -0.08800954 0.1724872
## [2,]        1.8798920  1.45479082      0.7800270  1.69635628 0.6450742
##       computers  business    outdoors    crafts automotive       art
## [1,] 0.07670276 0.1215802 -0.06801627 0.7033864  0.1759029 0.1108982
## [2,] 0.73954116 0.5074224  0.70040486 1.0904184  1.0701754 0.9055331
##      religion    beauty parenting      dating   school personal_fitness
## [1,] 2.252681 0.3124905  2.120695 -0.01192335 1.657167      -0.09784654
## [2,] 5.408907 1.1201080  4.134953  0.68960864 2.736842       1.22672065
##         fashion small_business
## [1,] 0.02696846     0.09556896
## [2,] 1.04588394     0.39541161
```

```r
#Your classic young parent (sports, food, family, religion, and parenting)
rbind(cluster_twitter$center[2,],cluster_twitter$center[2,]*sigma_twitter +
mu_twitter)
```

```
##        chatter current_events      travel photo_sharing uncategorized
## [1,] 0.7221757      0.2744222 -0.03596486      1.038320     0.4553478
## [2,] 6.9474053      1.8744741  1.50280505      5.532959     1.2391304
##        tv_film sports_fandom   politics       food     family
## [1,] 0.4852283    -0.1645516 -0.1310832 -0.1555397 0.03722871
## [2,] 1.8751753     1.2384292  1.3913043  1.1213184 0.90603086
##      home_and_garden     music       news online_gaming  shopping
## [1,]       0.2660681 0.5728224 -0.1305493     0.4324289 0.7427116
## [2,]       0.7166900 1.2692847  0.9312763     2.3709677 2.7328191
##      health_nutrition college_uni sports_playing  cooking       eco
## [1,]       -0.1815311   0.5892617      0.5035862 0.862288 0.2318190
## [2,]        1.7510519   3.2566620      1.1304348 4.955820 0.6907433
##        computers  business   outdoors    crafts automotive       art
## [1,] -0.02412705 0.3721702 -0.1125320 0.2905031 0.08501058 0.4382578
## [2,]  0.62061711 0.6809257  0.6465638 0.7531557 0.94600281 1.4389902
##        religion   beauty  parenting    dating    school personal_fitness
## [1,] -0.1651870 0.859983 -0.1369014 0.3078003 0.1565468       -0.1460417
## [2,]  0.7791024 1.847125  0.7138850 1.2594670 0.9537167        1.1107994
##        fashion small_business
```

```
## [1,] 0.9645493      0.4485480
## [2,] 2.7601683      0.6136045
```

```
#Average No-desciptive cluster
rbind(cluster_twitter$center[3,],cluster_twitter$center[3,]*sigma_twitter +
mu_twitter)
```

```
##         chatter current_events     travel photo_sharing uncategorized
## [1,] -0.2128344     -0.1294872 -0.2338155    -0.3162075     -0.166436
## [2,]  3.6476373      1.3619573  1.0506117     1.8330535      0.657232
##         tv_film sports_fandom   politics       food     family
## [1,] -0.1495236    -0.2977550 -0.2727615 -0.3658495 -0.2711894
## [2,]  0.8222595     0.9505877  0.9618614  0.7479012  0.5567282
##      home_and_garden      music       news online_gaming   shopping
## [1,]      -0.1701953 -0.2018740 -0.2480958    -0.1267947 -0.2431053
## [2,]       0.3952986  0.4713361  0.6843368     0.8680739  0.9496282
##      health_nutrition college_uni sports_playing    cooking       eco
## [1,]        -0.329529  -0.1572068     -0.2069700 -0.3399854 -0.236463
## [2,]         1.085632   1.0940273      0.4372751  0.8320940  0.330295
##       computers   business   outdoors     crafts automotive        art
## [1,] -0.2527869 -0.2097674 -0.3151216 -0.2585911 -0.2067795 -0.1579048
## [2,]  0.3509235  0.2780043  0.4015351  0.3046294  0.5473735  0.4674982
##        religion     beauty  parenting     dating     school
## [1,] -0.3043688 -0.2904203 -0.3119994 -0.1703194 -0.3057288
## [2,]  0.5125929  0.3195011  0.4485488  0.4072919  0.4044135
##      personal_fitness    fashion small_business
## [1,]       -0.3440901 -0.2943489     -0.1801222
## [2,]        0.6344447  0.4583833      0.2249940
```

```
#Athletiic/Healthy Cluster (Nutrition, outdoors, and personal fitness)
rbind(cluster_twitter$center[4,],cluster_twitter$center[4,]*sigma_twitter +
mu_twitter)
```

```
##           chatter current_events   travel photo_sharing uncategorized
## [1,] -0.006995531      0.1055617 1.839776     -0.107231   -0.04382044
## [2,]  4.374068554      1.6602086 5.789866      2.403875    0.77198212
##         tv_film sports_fandom politics       food     family
## [1,] 0.03309628     0.1920158 2.428282 0.03297433 0.05045762
## [2,] 1.12518629     2.0089419 9.149031 1.45603577 0.92101341
##      home_and_garden       music     news online_gaming    shopping
## [1,]       0.1185947 -0.05033662 1.935331   -0.07604167 -0.05706848
## [2,]       0.6080477  0.62742176 5.271237    1.00447094  1.28614009
##      health_nutrition college_uni sports_playing    cooking        eco
## [1,]       -0.2023987 -0.03276814     0.02766946 -0.1924006 0.09727734
## [2,]        1.6572280  1.45454545     0.66616990  1.3383010 0.58718331
##      computers  business  outdoors    crafts automotive         art
## [1,]  1.640700 0.3293340 0.1057575 0.1073858   1.074696 -0.04970671
## [2,]  2.584203 0.6512668 0.9105812 0.6035768   2.298063  0.64381520
##         religion     beauty  parenting    dating      school
## [1,] -0.03036794 -0.1505644 0.02142103 0.2065403 -0.04530884
## [2,]  1.03725782  0.5052161 0.95380030 1.0789866  0.71385991
```

```
##      personal_fitness    fashion small_business
## [1,]       -0.1970644 -0.1586984      0.2273961
## [2,]        0.9880775  0.7064083      0.4769001
```

```r
#worldly and current cluster (travel, politics, news, automotive, computers)
rbind(cluster_twitter$center[5,],cluster_twitter$center[5,]*sigma_twitter +
mu_twitter)
```

```
##          chatter current_events     travel photo_sharing uncategorized
## [1,] -0.08449267    -0.01124003 -0.1554517   -0.04798195     0.1485365
## [2,]  4.10057143     1.51200000  1.2297143    2.56571429     0.9520000
##         tv_film sports_fandom   politics      food      family
## [1,] -0.1043800    -0.1939706 -0.1810003 0.4320241 -0.07556437
## [2,]  0.8971429     1.1748571  1.2400000 2.1645714  0.77828571
##      home_and_garden      music        news online_gaming     shopping
## [1,]       0.1418000 0.02622923 -0.03255397   -0.06324663 -0.007390447
## [2,]       0.6251429 0.70628571  1.13714286    1.03885714  1.376000000
##      health_nutrition college_uni sports_playing  cooking       eco
## [1,]         2.150583  -0.1584983     0.02193048 0.436677 0.5281765
## [2,]        12.236571   1.0902857     0.66057143 3.496000 0.9188571
##        computers  business outdoors     crafts automotive         art
## [1,] -0.07939718 0.0374073 1.661314 0.08062137 -0.1264301 -0.01768315
## [2,]  0.55542857 0.4491429 2.792000 0.58171429  0.6571429  0.69600000
##        religion     beauty  parenting    dating     school
## [1,] -0.1650166 -0.1669719 -0.1026988 0.1615830 -0.1670980
## [2,]  0.7794286  0.4834286  0.7657143 0.9988571  0.5691429
##      personal_fitness     fashion small_business
## [1,]         2.111431 -0.07063281     -0.1281137
## [2,]         6.540571  0.86742857      0.2571429
```

Conclusion

By using K-Menas clustering with 5 centers I was able to successfully group the twitter users into distinct groups. The clusters found were; Young women, Parents, Athletic people, contemporaries, and then a group that was relatively average accross all categories. I think these clusters could provide useful marketing insights to NutrientH20.