R Exam

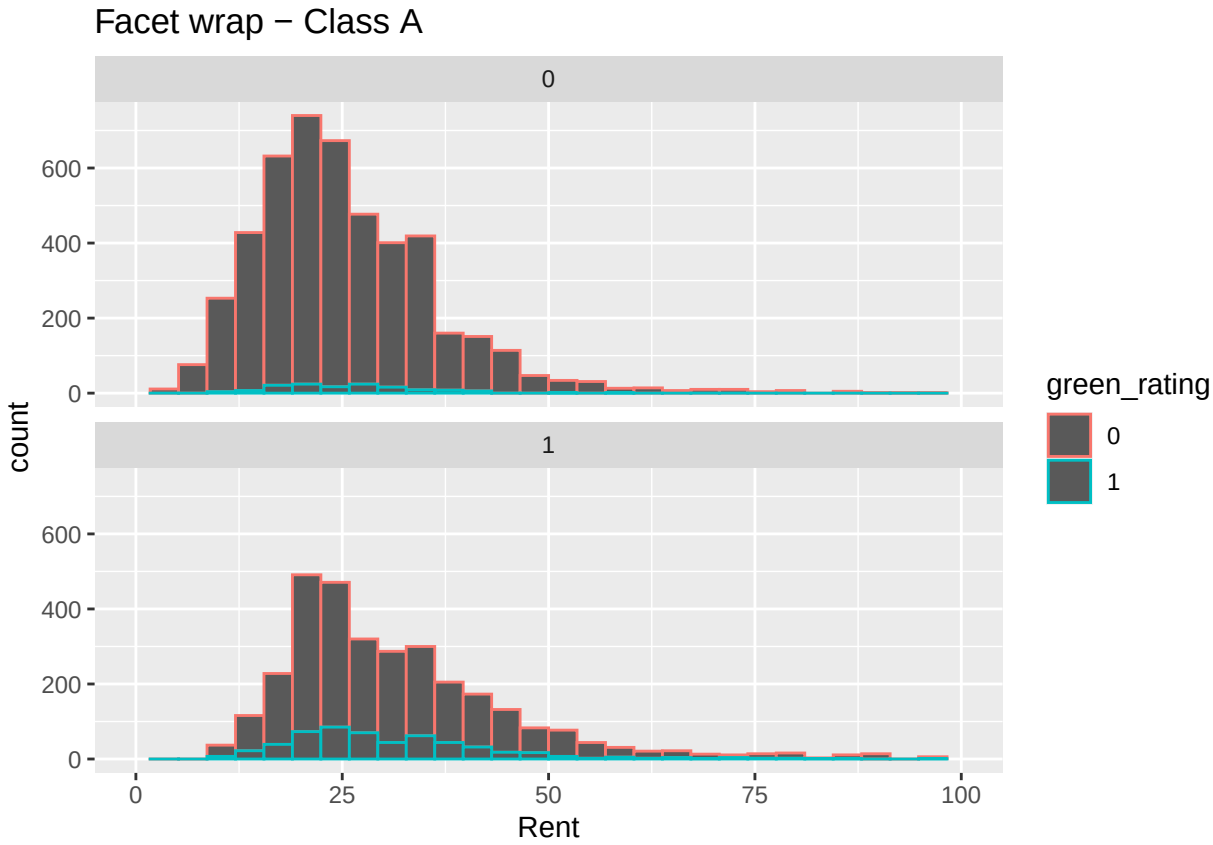
Question 1

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

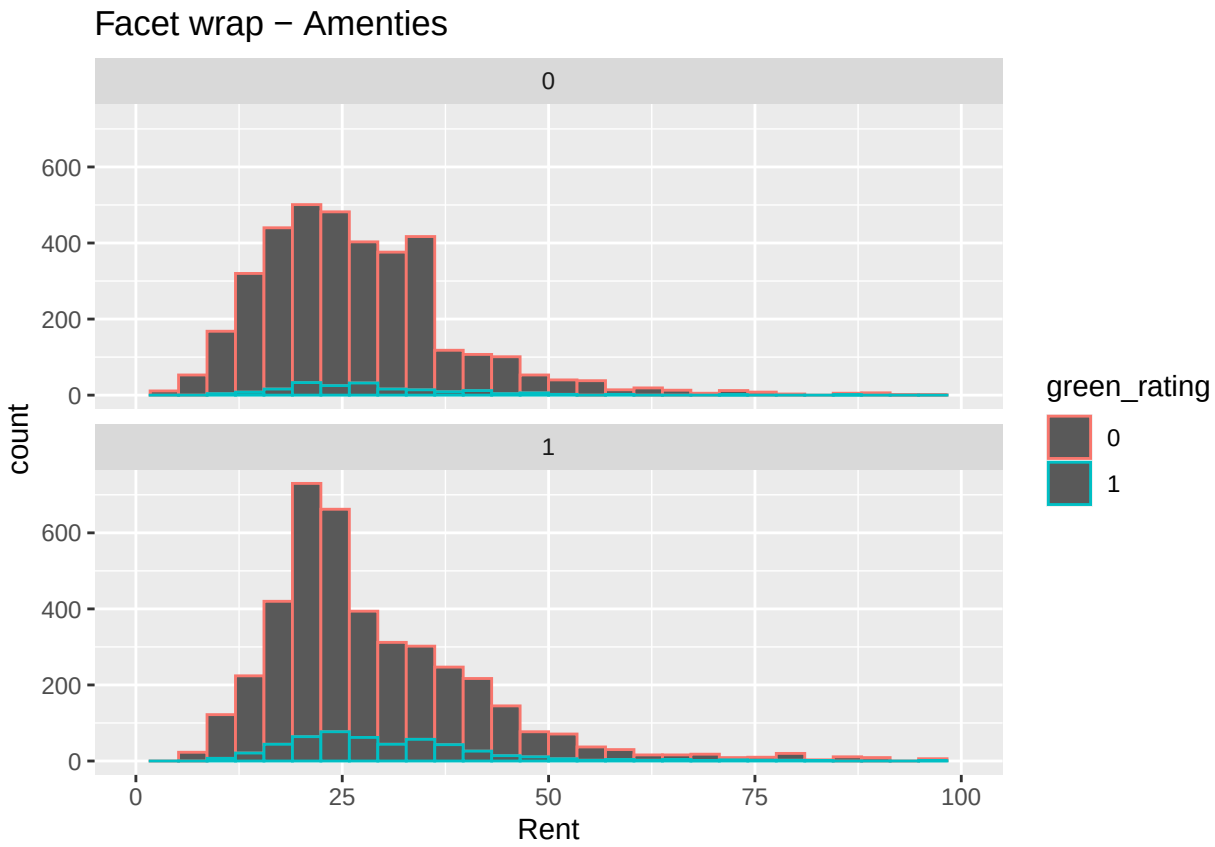
Rents of Green housing is, on average, 0.0833333% greater than non-green

Additional Pay from Green - 6.7708333×10^{6}

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 26 rows containing non-finite values (stat_bin).

## Warning: Removed 8 rows containing missing values (geom_bar).
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 26 rows containing non-finite values (stat_bin).

## Warning: Removed 8 rows containing missing values (geom_bar).
```



The East Cesar Chavez area seems to be high profile and seems like that it has all the amenities. Let's rerun the analysis for class a and amenties

```
df1 = df[df$class_a == 1,]
additional_rent_pre_sqft = median(df1[df1$green_rating  == 1,]$Rent) - median(df1[df1$green_rating  ==
area = 250000 #sqft
additional_pay_from_green = additional_rent_pre_sqft * area
```

Additional Pay from Green - 6×10^4

```
df2 = df[df$amenities == 1,]
additional_rent_pre_sqft = median(df2[df2$green_rating  == 1,]$Rent) - median(df2[df2$green_rating  ==

area = 250000 #sqft
additional_pay_from_green = additional_rent_pre_sqft * area
```

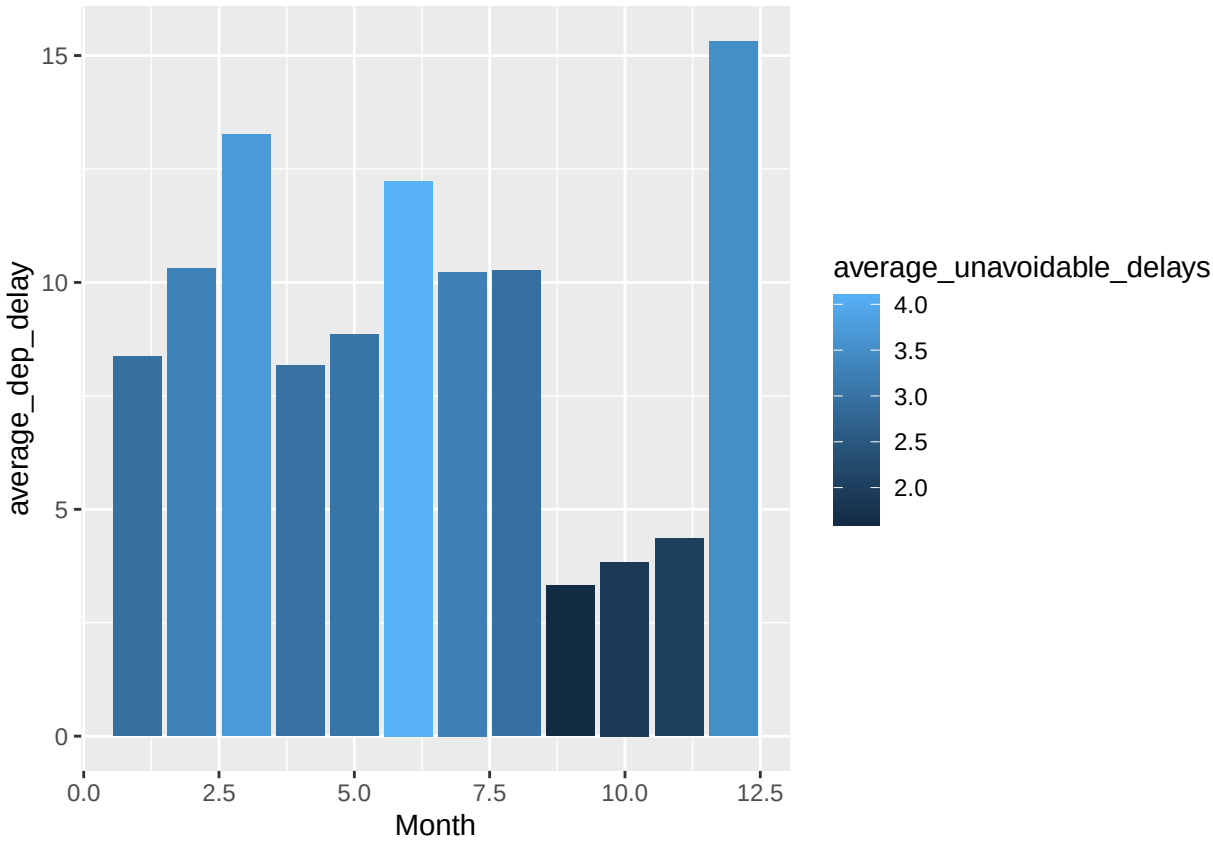
Additional Pay from Green - 6.9375×10^5

```
df3 = df[df$amenities == 1 & df$class_a == 1,]
additional_rent_pre_sqft = median(df3[df3$green_rating  == 1,]$Rent) - median(df3[df3$green_rating  ==

area = 250000 #sqft
additional_pay_from_green = additional_rent_pre_sqft * area
```

Additional Pay from Green - 1.175×10^5

Question 2


Arrival Delay vs Dept Delay by months
1
2
3
4
5
6
7
8
9
10
11
12
750
500
250
0
Departure Delay
0 250500750 1000
Arrival Delay
Delay caused due to Airport Authorities
0
1



Question 3

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following object is masked from 'package:DescTools':
##
##     MAD

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, 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

## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "AAPL"  "GOOGL" "TXN"   "TSLA"
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/GOOGL?
## period1=-2208988800&period2=1628899200&interval=1d&events=div&crumb=grFS8wO7Z0Q'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOOGL?
## period1=-2208988800&period2=1628899200&interval=1d&events=split&crumb=grFS8wO7Z0Q'
```

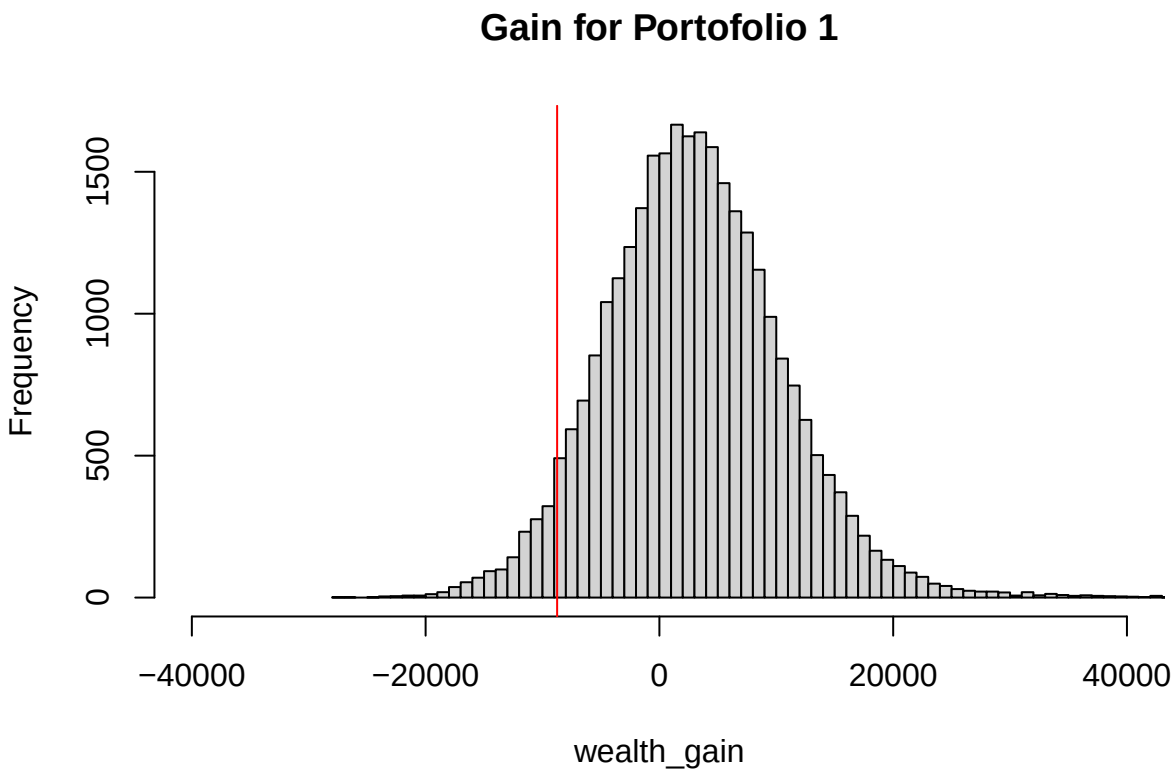
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOOGL?
## period1=-2208988800&period2=1628899200&interval=1d&events=split&crumb=grFS8wO7Z0Q'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/TSLA?
## period1=-2208988800&period2=1628899200&interval=1d&events=div&crumb=grFS8wO7Z0Q'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/TSLA?
## period1=-2208988800&period2=1628899200&interval=1d&events=split&crumb=grFS8wO7Z0Q'
```

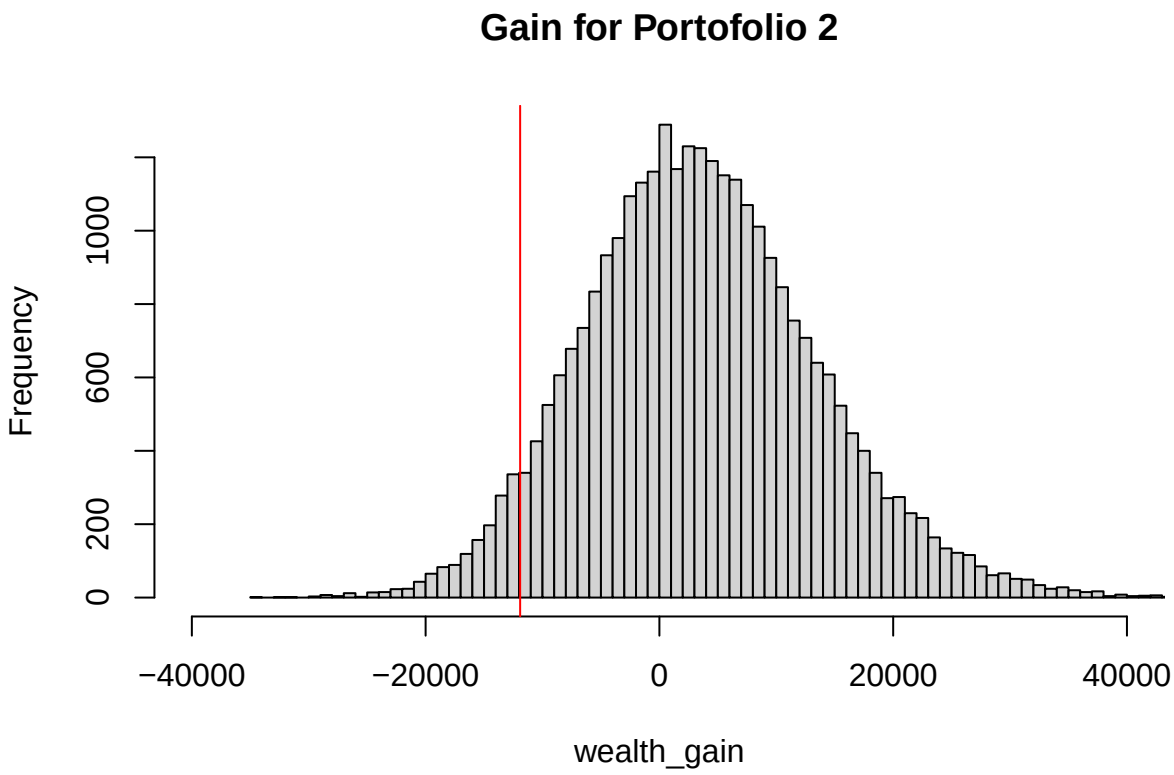
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/TSLA?
## period1=-2208988800&period2=1628899200&interval=1d&events=split&crumb=grFS8wO7Z0Q'
```

Portofolio 1



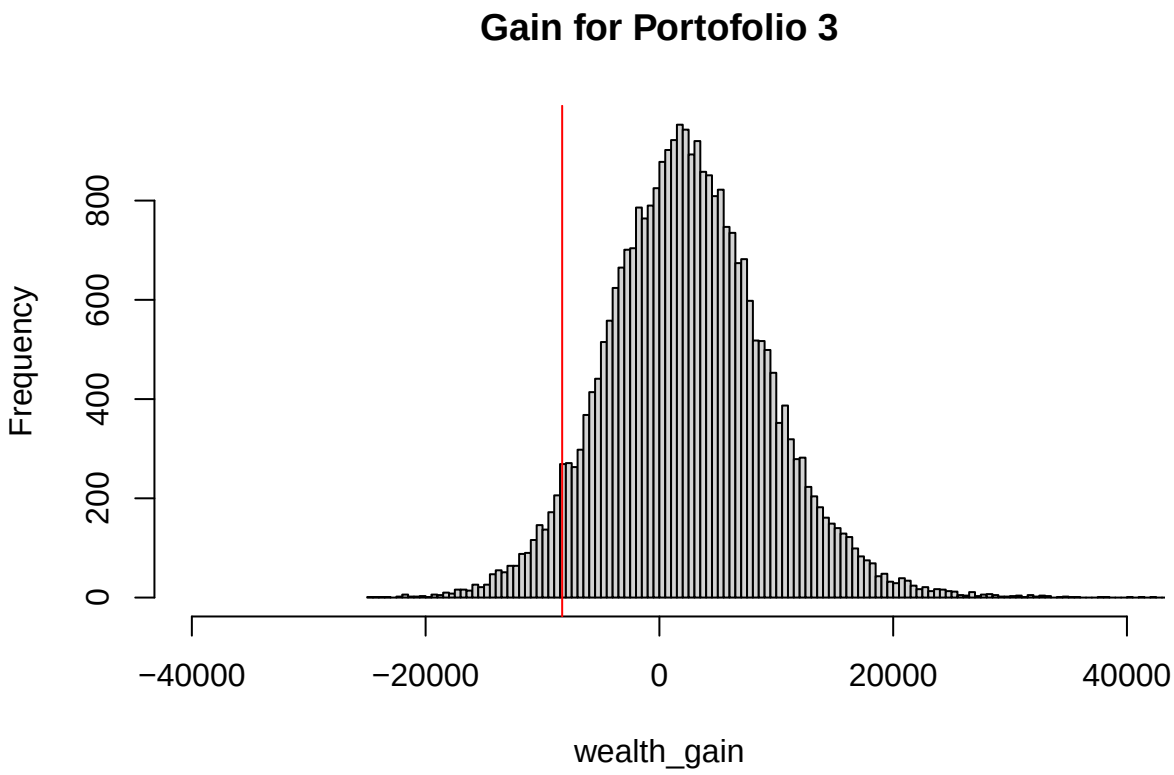
-8750.0196677

Portofolio 2



-1.19141×10^4

Portofolio 3



-8318.3911908

Question 4

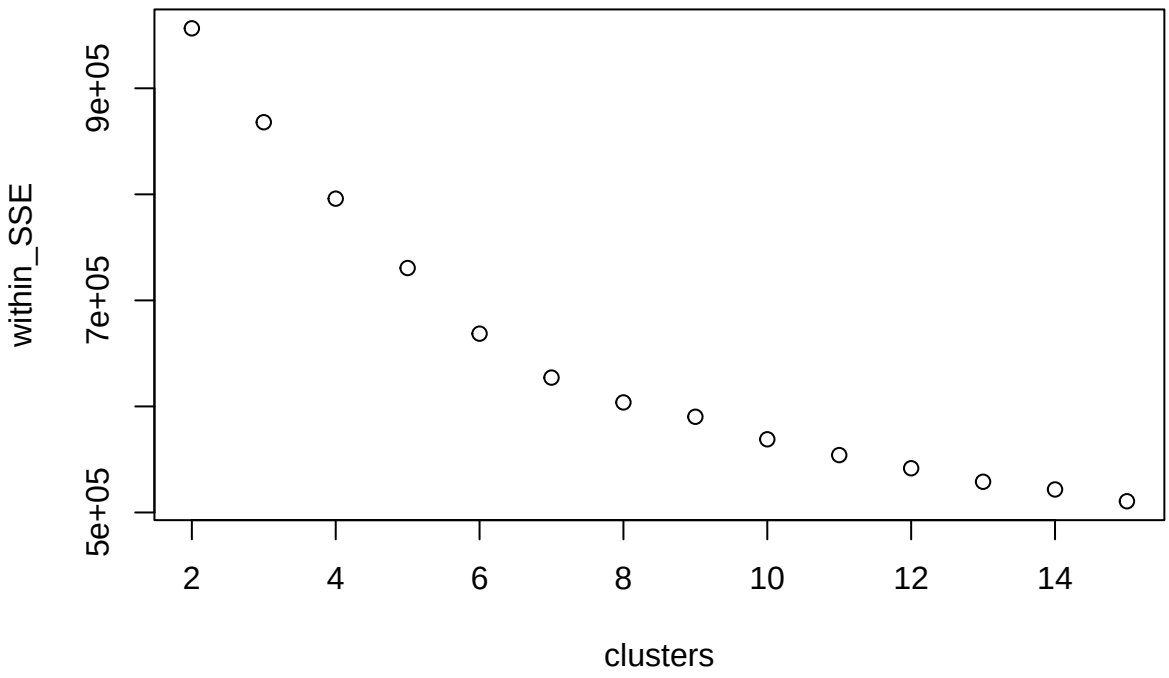
number of followers - 7882

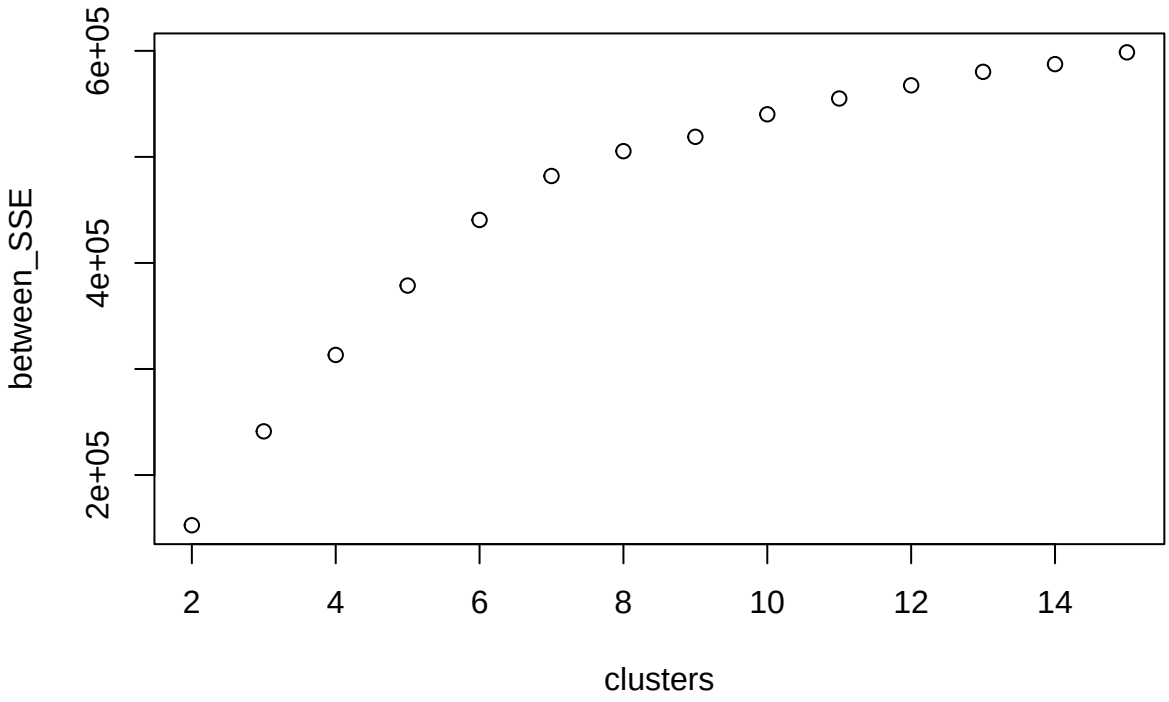
K-Mean clustering

K-Means good when the number of samples is huge

No scaling will be done because the unit of all the columns is same.

Different K and different distance metrics will be explored, starting with K = 5 and Euclidean distance metric.


within_SSE
5e+05
7e+05
9e+05
2
4
6
8
10
12
14
clusters




```
##       chatter current_events    travel photo_sharing uncategorized   tv_film
## 1 -0.08868426     0.71841989 0.1232091    0.71557797    -0.2309572 0.6764819
## 2  5.50651991     0.47553995 0.7917713    1.78522778     0.3704899 0.9890781
## 3  0.64285961    -0.08586167 1.3854028    0.08999201    -0.3776038 0.1971119
## 4  0.47298582    -0.08222480 0.7858856   -0.00702703     0.6006889 0.6799747
##   sports_fandom   politics      food     family home_and_garden       music
## 1     0.5904936 0.41091153  0.267310 -0.9221228      -0.2584652 -0.32684509
## 2     0.6260885 0.82745476  1.641534 -0.3789840      -0.2777545  0.05799701
## 3     0.2393073 0.80204191 14.435906 -0.3600817      -0.5249874  0.04242885
## 4     0.6112907 0.04307279  1.026870 -0.6538126      -0.3824259  0.10737656
##          news online_gaming    shopping health_nutrition college_uni
## 1 -0.05592945     2.9791671 0.479351851       0.54918523   6.3962789
## 2  0.02092842     0.1581264 3.945788574       0.38428947   0.3156359
## 3 -0.11994078    -0.4151491 0.005811054      -0.08284061   0.1141709
## 4 -0.28034858     0.5421898 0.996939360       8.56158891  -0.1655863
##   sports_playing    cooking        eco   computers   business    outdoors
## 1      0.7932445 -0.6713876 -0.3501236 -0.41202485 -0.3782863 -0.31043114
## 2      0.5540270  2.0111639 -0.2291664 -0.01949209 -0.2116605 -0.01935319
## 3      5.2891988 -0.3208730 -0.5648810  0.08143519 -0.5805220 -0.44139980
## 4      0.1484944  0.3929663 -0.3075304 -0.03743274 -0.4691700  2.46290734
##        crafts  automotive         art    religion      beauty    parenting
## 1 -0.02591343  0.08653198  0.19640360  0.03767655 -1.06476038  0.182942139
## 2  0.36824648  0.24946818  0.02857516  1.35565269 -0.02893089  0.006215571
## 3 -0.30076300 -0.18822232 -0.10634196 12.68527902 -0.50210843 -0.348864281
## 4  0.15859074  0.01505361 -0.26309062  0.39886759 -0.73864016 -0.259250626
##        dating      school personal_fitness    fashion small_business       spam
```

```
## 1 -0.35862858 -0.24826904       -0.1725205  0.4161937     -0.2935046 -0.6020426
## 2  0.22457222 -0.04149517        0.4029258  2.6187610     -0.1516985 -0.3974940
## 3 -0.03616171 -0.40142016       -0.3990760 -0.2025607     -0.4223298 -0.6398003
## 4  0.31684914 -0.40532539        7.1013913  0.6064756     -0.3092033 -0.6047456
##         adult
## 1 -0.31665025
## 2  0.02765828
## 3  5.13074792
## 4 -0.03411253
```

Groups identified - 1. Teen girls? - High shopping, fashion, school, chatter, photosharing 2. Fitness freaks - high - Outdoors, personal_fitness , health_nutirition 3. Old person - religion, adult, food 4. college students - high - college_uni, online_gaming, sport_playing

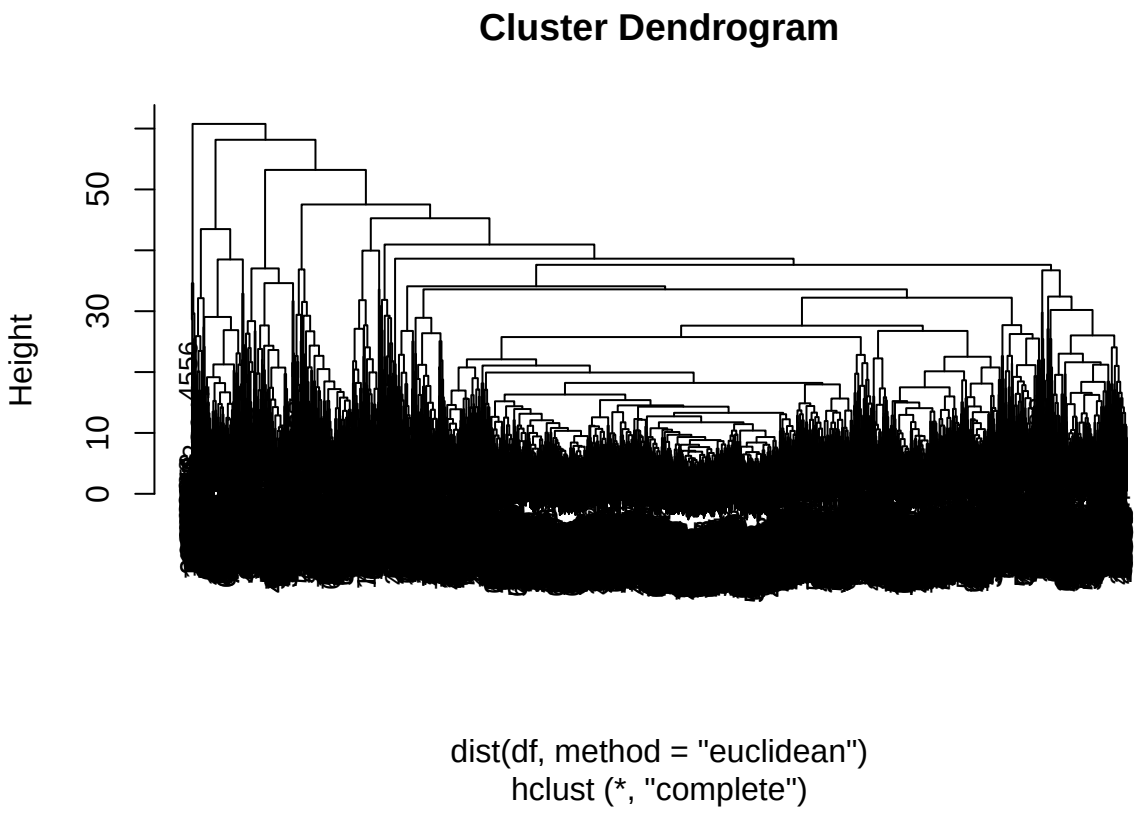
```
## [1] 443
```

```
## [1] 1675
```

```
## [1] 4698
```

```
## [1] 1066
```

Heirarchial Clustering



```
##    1    2    3    4
##  417 7014  420   31
```

I think KMeans clustering provides good enough insights.

Question 5

```
rm(list = ls())
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
```

```
## The following object is masked from 'package:mosaic':
##
##     inspect
```

```
library(tidyverse)
library(slam)
library(proxy)
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(plyr)
```

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

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following object is masked from 'package:mosaic':
##
##     count

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(caret)
```

```
##
## Attaching package: 'caret'

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

## The following object is masked from 'package:purrr':
##
##     lift

## The following objects are masked from 'package:DescTools':
##
##     MAE, RMSE
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(arules)
```

```
##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:mosaic':
##
##     inspect, lhs, rhs

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)

# function for reading text as plain data
readerPlain = function(fname){
                readPlain(elem=list(content=readLines(fname)),
                          id=fname, language='en') }

# fetching all directory
file_list = Sys.glob('./ReutersC50/C50train/*/*.txt')

# read text
text = lapply(file_list, readerPlain)

# Create a corpus from the text above
documents_raw = Corpus(VectorSource(text))

# modifying corpus
my_documents = documents_raw %>%
  tm_map(content_transformer(tolower))  %>%             # make everything lowercase
  tm_map(content_transformer(removeNumbers)) %>%        # remove numbers
  tm_map(content_transformer(removePunctuation)) %>%    # remove punctuation
  tm_map(content_transformer(stripWhitespace))          # remove excess white-space
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(tolower)): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(removeNumbers)):
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(removePunctuation)):
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(stripWhitespace)):
## transformation drops documents
```

```
# removing stop words
my_documents = tm_map(my_documents,
                      content_transformer(removeWords),
                      stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(removeWords), :
## transformation drops documents
```

```
# Create Term frequency matrix. Each row is term frequncy vector of single  document
DTM = DocumentTermMatrix(my_documents)
# Current Sparisty is 99%. Let's remove long tail of infrequently used words
DTM = removeSparseTerms(DTM, 0.95)
train_X = as.matrix(DTM)

# Let's create target variable
train_Y = rep(1, 50)
for (i in 2:50)
{
  train_Y = c(train_Y, rep(i, 50))
}
```

```
# fetching all directory
file_list = Sys.glob('./ReutersC50/C50test/*/*.txt')

# read text
text = lapply(file_list, readerPlain)

# Create a corpus from the text above
documents_raw = Corpus(VectorSource(text))

# modifying corpus
my_documents = documents_raw %>%
  tm_map(content_transformer(tolower))  %>%             # make everything lowercase
  tm_map(content_transformer(removeNumbers)) %>%        # remove numbers
  tm_map(content_transformer(removePunctuation)) %>%    # remove punctuation
  tm_map(content_transformer(stripWhitespace))          # remove excess white-space
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(tolower)): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(removeNumbers)):
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(removePunctuation)):
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(stripWhitespace)):
## transformation drops documents
```

```
# removing stop words
my_documents = tm_map(my_documents,
                      content_transformer(removeWords),
                      stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(removeWords), :
## transformation drops documents
```

```
# Create Term frequency matrix. Each row is term frequncy vector of single  document
DTM = DocumentTermMatrix(my_documents)
# Current Sparisty is 99%. Let's remove long tail of infrequently used words
DTM = removeSparseTerms(DTM, 0.95)
test_X = as.matrix(DTM)

# Let's create target variable
test_Y = rep(1, 50)
for (i in 2:50)
{
  test_Y = c(test_Y, rep(i, 50))
}
```

```
scrub_cols = which(colSums(train_X) == 0)
scrub_cols = which(sapply(as.data.frame(train_X), function(x) var(x, na.rm = T)) == 0)

train_X = train_X[,-scrub_cols]
```

```
pr.out=prcomp(train_X , scale=TRUE, rank = 200)
```

```
rf = randomForest(x = pr.out$x, y = as.factor(train_Y),
                  type = 'classification', maxnodes = 300)
confusionMatrix(rf$predicted, as.factor(train_Y),
                positive = NULL,
                dnn = c("Prediction", "Reference"))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##          1  44  0  0  0  2  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0
##          2   0 38  0  1  0  0  1  0  0  0  0  0  0  2  0  0  0  0  1  0  1  0  0
##          3   0  0 28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1
##          4   0  0  0 33  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  1  0  0
```

```
##          5   0  0  0  0 32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          6   0  0  0  0  1 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3
##          7   0  0  0  0  0  0 37  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0
##          8   0  0  0  0  0  0  0 43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          9   0  0  2  0  0  0  0  0 26  0  0  0  2  0  0  0  1  3  1  3  0  4  0
##          10  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0  0  1
##          11  0  1  1  0  1  0  0  0  0  0 45  2  0  0  2  0  0  0  0  0  0  0  0
##          12  0  1  0  0  0  0  0  0  0  0  1 37  0  0  1  0  0  0  0  0  0  0  0
##          13  0  0  0  0  0  0  2  0  0  0  0  1 35  0  2  0  1  0  0  0  1  0  0
##          14  0  3  0  0  0  0  1  0  0  0  0  0  0 35  0  0  0  0  7  0  0  0  0
##          15  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0
##          16  2  2  0  0  2  0  0  0  6  2  0  1  2  2  0 48  1  2  1  1  0  1  2
##          17  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0 39  0  0  2  0  0  0
##          18  0  0  6  0  0  1  0  0  2  0  1  1  0  0  0  0  1 25  1  1  1  1  0
##          19  0  1  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0 28  0  0  0  0
##          20  0  0  2  0  0  1  1  0  8  0  0  0  0  0  0  0  2  6  0 35  0  5  1
##          21  2  4  1  4  2  2  2  2  2  1  0  0  3  6  1  1  0  4 10  2 42  1  1
##          22  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 33  1
##          23  0  0  0  0  0  9  0  0  0  6  0  0  3  0  0  0  0  1  0  0  1  0 36
##          24  0  0  1  0  3  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##          25  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  2  0
##          26  0  0  0  0  0  3  0  1  0  4  0  0  0  0  0  0  0  0  0  0  0  0  1
##          27  0  0  0  0  0  0  3  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##          28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          29  0  0  0  1  0  0  1  0  0  0  0  2  0  0  1  0  0  0  0  0  1  0  0
##          30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          31  0  0  0  0  5  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##          32  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          33  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0
##          34  1  0  1  1  0  2  2  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0
##          35  0  0  0  4  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0
##          36  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##          37  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##          39  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##          40  0  0  0  0  1  3  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          41  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          42  0  0  0  0  1  1  0  0  0  4  0  2  0  0  1  0  0  2  0  0  0  0  0
##          43  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  2  0  0  0  0  0
##          44  0  0  0  3  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0
##          45  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##          47  0  0  0  0  0  1  0  1  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0
##          48  0  0  2  0  0  0  0  0  5  0  1  0  0  0  0  0  5  1  0  4  0  3  0
##          49  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          50  0  0  0  2  0  0  0  0  0  0  0  2  0  0  2  0  0  0  1  0  0  0  0
##            Reference
## Prediction 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
##          1  0  0  0  0  0  0  1  0  0  0  3  0  0  0  0  0  0  4  4  0  0  0  0
##          2  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  1  0  0  0  1  1  0  0
##          3  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  3  0  0  0
##          4  0  0  0  0  0  0  0  0  0  1  0 13  0  0  0  0  0  0  0  0 13  0  0
##          5  4  0  0  0  0  0  0  3  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##          6  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
```

```
##          7   0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0
##          8   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          9   2  2  0  0  0  0  0  1  1  0  1  0  1  0  0  1  1  0  0  0  0  0  0
##          10  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##          11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##          12  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          13  0  1  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##          14  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          15  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##          16  2  0  2  0  0  0  2  2  0  0  3  0  0  0  0  3  1  0  1  2  0  2  4
##          17  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          18  0  0  0  0  0  1  0  1  0  0  2  0  0  0  0  0  0  0  0  0  0  6  0
##          19  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##          20  1  6  0  0  0  1  0  1  0  0  0  1  0  1  0  1  0  0  0  0  0  3  1
##          21  0  1  0  3  0  2  1  4  0  4  4  2  1  4  0  2  0  1  0  1  3  3  1
##          22  1  4  0  0  0  0  1  0  1  0  3  0  0  0  0  1  1  0  1  0  0  2  0
##          23  0  0  2  1  0  0  0  0  2  0  3  0  1 13  0  0  5  0  2  0  1  0  0
##          24 28  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          25  0 25  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0
##          26  0  1 40  0  0  0  0  0  5  0  0  0  0  3  0  0  0  0  3  0  1  0  0
##          27  0  0  0 43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          28  0  0  0  0 49  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  4
##          29  0  0  0  0  0 43  0  1  1  2  0  2  0  0  0  0  0  0  0  0  4  1  2
##          30  0  0  0  0  0  0 42  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0
##          31 10  0  0  0  0  0  0 30  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0
##          32  0  0  0  0  0  0  0  0 33  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##          33  0  0  0  0  0  0  0  0  0 39  0  0  0  0  0  0  0  0  0  0  0  0  0
##          34  1  0  2  1  0  0  0  0  3  0 28  1  1  1  0  0  4  1  3  0  0  0  0
##          35  0  0  0  0  0  0  0  0  0  0  0 15  0  0  2  0  0  0  0  0  5  0  1
##          36  0  3  0  0  0  0  0  0  0  0  2  0 38  1  0  0  0  1  1  1  0  0  0
##          37  0  0  0  0  0  0  0  0  1  0  0  0  0 23  0  0  0  0  0  0  0  0  0
##          38  0  0  0  0  1  0  0  0  0  0  0  2  0  0 44  0  0  0  0  1  4  0 12
##          39  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0 34  0  0  0  0  0  0  0
##          40  0  0  0  0  0  0  0  0  1  0  0  0  2  1  0  0 38  0  0  1  0  0  0
##          41  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 41  0  0  0  0  0
##          42  1  1  1  1  0  0  1  0  2  0  0  1  1  0  0  0  0  2 32  0  0  0  0
##          43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 31  0  0  0
##          44  0  0  0  0  0  0  0  0  0  1  0  7  0  0  0  0  0  0  0  0 13  0  1
##          45  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0 32  0
##          46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  6  0  0 24
##          47  0  0  3  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  2  0  0  0  0
##          48  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0
##          49  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          50  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  4  0  0
##            Reference
## Prediction 47 48 49 50
##          1  0  0  0  0
##          2  0  1  0  0
##          3  0  1  0  0
##          4  0  0  0  6
##          5  0  0  0  0
##          6  0  0  0  0
##          7  0  0  0  0
##          8  0  0  5  0
```

```
##         9   0  1  0  0
##         10  5  0  0  0
##         11  0  0  1  0
##         12  0  0  0  1
##         13  0  0  0  0
##         14  0  0  0  0
##         15  0  0  0  2
##         16  2  1  3  1
##         17  0  0  0  0
##         18  0  0  0  1
##         19  0  0  0  0
##         20  0  7  1  1
##         21  0  0  5  6
##         22  0  1  0  0
##         23  1  0  3  0
##         24  0  0  0  0
##         25  0  0  0  0
##         26  6  0  0  0
##         27  0  0  0  0
##         28  0  0  0  0
##         29  0  0  0  6
##         30  0  0  0  0
##         31  0  0  0  0
##         32  0  0  0  0
##         33  0  0  0  0
##         34  0  0  3  0
##         35  0  0  0  3
##         36  1  0  0  1
##         37  0  0  0  0
##         38  0  0  0  0
##         39  0  0  0  0
##         40  0  1  1  0
##         41  0  0  0  0
##         42  8  0  0  0
##         43  0  0  0  0
##         44  0  0  0  5
##         45  0  2  0  0
##         46  0  0  0  0
##         47 27  0  0  0
##         48  0 35  1  0
##         49  0  0 27  0
##         50  0  0  0 17
##
## Overall Statistics
##
##                Accuracy : 0.6652
##                  95% CI : (0.6463, 0.6837)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6584
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.8800   0.7600   0.5600   0.6600   0.6400   0.4800
## Specificity            0.9931   0.9951   0.9971   0.9824   0.9967   0.9980
## Pos Pred Value         0.7213   0.7600   0.8000   0.4342   0.8000   0.8276
## Neg Pred Value         0.9975   0.9951   0.9911   0.9930   0.9927   0.9895
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0176   0.0152   0.0112   0.0132   0.0128   0.0096
## Detection Prevalence   0.0244   0.0200   0.0140   0.0304   0.0160   0.0116
## Balanced Accuracy      0.9365   0.8776   0.7786   0.8212   0.8184   0.7390
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
## Sensitivity            0.7400   0.8600   0.5200    0.5800    0.9000    0.7400
## Specificity            0.9980   0.9980   0.9890    0.9971    0.9963    0.9984
## Pos Pred Value         0.8810   0.8958   0.4906    0.8056    0.8333    0.9024
## Neg Pred Value         0.9947   0.9971   0.9902    0.9915    0.9980    0.9947
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0148   0.0172   0.0104    0.0116    0.0180    0.0148
## Detection Prevalence   0.0168   0.0192   0.0212    0.0144    0.0216    0.0164
## Balanced Accuracy      0.8690   0.9290   0.7545    0.7886    0.9482    0.8692
##                      Class: 13 Class: 14 Class: 15 Class: 16 Class: 17
## Sensitivity             0.7000    0.7000    0.4000    0.9600    0.7800
## Specificity             0.9959    0.9955    0.9988    0.9763    0.9967
## Pos Pred Value          0.7778    0.7609    0.8696    0.4528    0.8298
## Neg Pred Value          0.9939    0.9939    0.9879    0.9992    0.9955
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0140    0.0140    0.0080    0.0192    0.0156
## Detection Prevalence    0.0180    0.0184    0.0092    0.0424    0.0188
## Balanced Accuracy       0.8480    0.8478    0.6994    0.9682    0.8884
##                      Class: 18 Class: 19 Class: 20 Class: 21 Class: 22
## Sensitivity             0.5000    0.5600    0.7000    0.8400    0.6600
## Specificity             0.9890    0.9971    0.9792    0.9596    0.9927
## Pos Pred Value          0.4808    0.8000    0.4070    0.2979    0.6471
## Neg Pred Value          0.9898    0.9911    0.9938    0.9966    0.9931
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0100    0.0112    0.0140    0.0168    0.0132
## Detection Prevalence    0.0208    0.0140    0.0344    0.0564    0.0204
## Balanced Accuracy       0.7445    0.7786    0.8396    0.8998    0.8263
##                      Class: 23 Class: 24 Class: 25 Class: 26 Class: 27
## Sensitivity             0.7200    0.5600    0.5000    0.8000    0.8600
## Specificity             0.9780    0.9955    0.9976    0.9886    0.9984
## Pos Pred Value          0.4000    0.7179    0.8065    0.5882    0.9149
## Neg Pred Value          0.9942    0.9911    0.9899    0.9959    0.9971
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0144    0.0112    0.0100    0.0160    0.0172
## Detection Prevalence    0.0360    0.0156    0.0124    0.0272    0.0188
## Balanced Accuracy       0.8490    0.7778    0.7488    0.8943    0.9292
##                      Class: 28 Class: 29 Class: 30 Class: 31 Class: 32
## Sensitivity             0.9800    0.8600    0.8400    0.6000    0.6600
## Specificity             0.9976    0.9898    0.9976    0.9927    0.9992
## Pos Pred Value          0.8909    0.6324    0.8750    0.6250    0.9429
## Neg Pred Value          0.9996    0.9971    0.9967    0.9918    0.9931
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0196    0.0172    0.0168    0.0120    0.0132
```

```
## Detection Prevalence    0.0220    0.0272    0.0192    0.0192    0.0140
## Balanced Accuracy       0.9888    0.9249    0.9188    0.7963    0.8296
##                      Class: 33 Class: 34 Class: 35 Class: 36 Class: 37
## Sensitivity             0.7800    0.5600    0.3000    0.7600    0.4600
## Specificity             0.9988    0.9878    0.9918    0.9947    0.9992
## Pos Pred Value          0.9286    0.4828    0.4286    0.7451    0.9200
## Neg Pred Value          0.9955    0.9910    0.9858    0.9951    0.9891
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0156    0.0112    0.0060    0.0152    0.0092
## Detection Prevalence    0.0168    0.0232    0.0140    0.0204    0.0100
## Balanced Accuracy       0.8894    0.7739    0.6459    0.8773    0.7296
##                      Class: 38 Class: 39 Class: 40 Class: 41 Class: 42
## Sensitivity             0.8800    0.6800    0.7600    0.8200    0.6400
## Specificity             0.9914    0.9980    0.9947    0.9988    0.9878
## Pos Pred Value          0.6769    0.8718    0.7451    0.9318    0.5161
## Neg Pred Value          0.9975    0.9935    0.9951    0.9963    0.9926
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0176    0.0136    0.0152    0.0164    0.0128
## Detection Prevalence    0.0260    0.0156    0.0204    0.0176    0.0248
## Balanced Accuracy       0.9357    0.8390    0.8773    0.9094    0.8139
##                      Class: 43 Class: 44 Class: 45 Class: 46 Class: 47
## Sensitivity             0.6200    0.2600    0.6400    0.4800    0.5400
## Specificity             0.9984    0.9910    0.9971    0.9959    0.9955
## Pos Pred Value          0.8857    0.3714    0.8205    0.7059    0.7105
## Neg Pred Value          0.9923    0.9850    0.9927    0.9895    0.9907
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0052    0.0128    0.0096    0.0108
## Detection Prevalence    0.0140    0.0140    0.0156    0.0136    0.0152
## Balanced Accuracy       0.8092    0.6255    0.8186    0.7380    0.7678
##                      Class: 48 Class: 49 Class: 50
## Sensitivity             0.7000    0.5400    0.3400
## Specificity             0.9894    0.9992    0.9943
## Pos Pred Value          0.5738    0.9310    0.5484
## Neg Pred Value          0.9938    0.9907    0.9866
## Prevalence              0.0200    0.0200    0.0200
## Detection Rate          0.0140    0.0108    0.0068
## Detection Prevalence    0.0244    0.0116    0.0124
## Balanced Accuracy       0.8447    0.7696    0.6671
```

```
cols = c()
for (i in c(1: length(test_Y)))
{
  if(colnames(test_X)[i] %in% colnames(train_X))
    cols = c(cols, i)
}
test_scrubbed = scale(test_X)[,cols]

yy <- read.table(textConnection(""), col.names = colnames(train_X),
                 colClasses = "integer")

test_data = rbind.fill(yy, as.data.frame(test_scrubbed))
test_data[is.na(test_data)] = 0

test_pca = predict(pr.out, newdata = test_data)
```

```r
test_result = predict(rf, newdata = test_pca, type = 'class')

confusionMatrix(test_result, as.factor(train_Y),
                positive = NULL,
                dnn = c("Prediction", "Reference"))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##          1 27  0  2  0  2  1  2  1  0  1  0  0  0  0  0  0  0  3  0  0  1  0  0
##          2  0  7  0  0  1  3  0  0  0  0  0  1  1  2  0  0  0  0  1  0  0  0  1
##          3  0  0 15  0  0  0  0  0  1  0  0  0  0  0  0  0  0  5  0  0  0  1  0
##          4  0  0  0  8  0  0  0  0  0  0  0  2  0  0  7  0  0  0  0  0  2  0  0
##          5  0  0  0  0 13  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0
##          6  0  0  0  0  0  3  0  2  0  0  0  1  0  0  0  0  0  0  0  0  0  0  2
##          7  1  2  2  0  1  0  2  2  0  0  0  2  5  0  0  1  1  1  0  0  0  0  1
##          8  0  0  0  0  0  1  4  8  1  1  0  0  2  0  2  0  0  0  1  1  4  0  1
##          9  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##          10 0  0  1  0  0  0  0  0  0 18  0  0  0  0  0  1  0  0  1  0  0  0  0
##          11 7  2 12  3  5 11  0  1  2  0 43  5  0  5  2  2  0  4  0  2  1  0  5
##          12 0  2  0  5  0  0  1  0  0  0  0 17  0  0  0  0  0  0  0  0  0  0  0
##          13 2  0  0  0  0  1  9  0  0  0  0  0 13  0  2  0  0  1  0  0  1  0  1
##          14 1  9  0  0  4  0  2  0  0  0  1  0  9 14  2  0  0  0 12  0  0  0  1
##          15 0  0  0  5  0  0  0  0  0  0  0  0  8  0  4  0  0  0  0  0  2  0  0
##          16 0  2  0  0  0  1  1  1  2  0  0  0  1  1  0 26  0  1  0  1  0  2  3
##          17 0  3  2  1  1  0  1  1  3  0  0  0  0  0  0  0 42  4  0 11  0 12  1
##          18 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1
##          19 0  3  0  1  2  0  0  0  0  0  1  0  2 12  2  0  0  0 14  0  5  0  0
##          20 0  0  1  0  0  3  0  1  5  0  0  0  0  1  0  0  1  7  0 15  1 10  0
##          21 0  0  0  0  0  0  4  2  0  0  0  1  1  0  1  0  0  1  7  0  7  0  0
##          22 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0
##          23 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5
##          24 0  0  1  0  7  0  0  0  2  0  0  0  2  1  0  0  1  3  0  2  1  0  0
##          25 0  0  0  0  2  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0
##          26 0  0  0  0  0  0  0  3  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##          27 3  6  1  0  1 10 21  2  3  7  3  6  2  0  1  1  2  1  0  1  1  3  7
##          28 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          29 0  1  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  1
##          30 0  0  0  0  0  0  0  2  4  0  0  0  0  4  0  0  0  0  0  0  0  0  0
##          31 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          32 0  0  0  0  0  0  0  4  0  4  0  0  0  0  0  0  0  0  0  0  0  0  1
##          33 1 12  0  1  3  2  1  3  2  1  0  2  2  0  0  7  0  0  4  4  6  0  5
##          34 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##          35 0  0  0  9  0  0  1  0  0  0  0  2  0  0 13  0  0  0  0  0  5  0  0
##          36 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##          37 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  4
##          38 2  0  0  1  0  0  0  0  0  0  0  0  0  1  3  2  0  0  0  0  1  0  0
##          39 0  0  2  0  1  1  0  0  5  2  0  1  0  1  0  1  1  0  2  1  0  3  1
##          40 0  0  0  0  5  9  0  7  0  4  0  0  0  2  0  0  0  0  1  2  0  1  6
##          41 3  0  1  1  2  0  0  0  2  4  0  0  0  0  0  0  0  0  0  0  0  0  0
##          42 1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##          43 1  1  9  1  0  1  0  1  1  0  2  7  0  0  0  2  0  4  1  3  0  0  1
```

```
##         44  0  0  0  6  0  0  0  0  0  0  0  1  0  2  5  0  0  0  6  0  4  0  0
##         45  0  0  0  0  0  0  0  2  2  0  0  0  0  0  0  0  0  5  0  1  0  2  0
##         46  1  0  0  6  0  0  0  0  1  1  0  1  0  0  5  7  0  1  0  0  7  1  0
##         47  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0
##         48  0  0  1  0  0  0  0  1 11  0  0  0  0  3  0  0  2  8  0  5  0  5  1
##         49  0  0  0  0  0  1  0  4  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0
##         50  0  0  0  1  0  0  0  0  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0
##           Reference
## Prediction 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
##         1   0  2  2  0  0  0  1  0  6  0  5  0  2  1  0  0  0 10  4  0  0  2  1
##         2   0  0  0  0  0  0  1  1  0  0  1  1  1  0  0  2  0  0  0  1  0  1  0
##         3   0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  3  0
##         4   0  0  0  0  0  1  0  0  0  0  1  3  0  0  0  0  0  0  0  0  1  0  1
##         5   6  3  0  0  0  0  0  4  0  0  0  0  0  2  0  0  1  0  1  1  0  0  1
##         6   0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##         7   0  0  1  1  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##         8   0  0  2  0  0  0  1  0  3  0  1  6  2  0  0  0  0  2  0  0  0  0  0
##         9   0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0
##         10  0  0 12  0  0  1  0  0  1  0  0  0  5  1  0  0  0  0  9  0  0  0  0
##         11  1  1  5  2  1  3  1  2  4  0  4  1  5  3  1  2  3  0  3  3  0  8  1
##         12  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0
##         13  0  0  0  0  1  3  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         14  2  0  0  0  1  5  0  1  0  1  1  0  0  0  0  0  0  0  0  2  0  1  1
##         15  2  0  0  0  3  2  0  0  0  1  0  0  0  0  1  0  0  1  0  1  5  0  0
##         16  0  0  1  2  0  0  0  2  0  1  0  0  3  3  0  0  0  0  2  0  0  1  0
##         17  0  9  0  0  0  3  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  9  0
##         18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##         19  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0
##         20  0  2  0  0  0  0  3  1  1  0  2  0  0  4  0  1  0  0  1  0  0  2  0
##         21  0  0  0  0  2  0  0  0  0  1  2  1  0  0  0  0  0  0  0  0  3  0  0
##         22  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         23  0  0  2  1  0  0  0  0  2  0  0  0  0  2  0  0  0  0  1  0  0  0  0
##         24 23  1  0  0  0  0  0 11  1  0  3  0  2  0  0  2  1  0  1  0  0  0  2
##         25  2 13  0  0  0  1  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0
##         26  0  0 13  1  0  0  0  0  5  0  0  0  0  2  0  0  0  0  3  0  0  0  0
##         27  3  1  5 38  0  1  0 10  4  0  4  4  9  6  0  0  1  0  1  1  3  2  1
##         28  0  0  0  0 26  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0
##         29  1  0  0  0  1 12  0  1  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0
##         30  3  2  1  0  0  1 32  1  0  0  1  0  1  0  0 17  1  0  1  0  0  0  1
##         31  2  0  0  0  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         32  0  0  0  0  0  0  0  0 10  0  0  0  1  0  0  0  0  0  1  0  0  0  0
##         33  1  0  2  1 10  6  1  4  0 38  0  0  3  1  0  0  1  0  3  0  1  0  2
##         34  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
##         35  0  0  0  0  0  0  0  1  0  1  1 20  0  0  1  0  0  2  0  0 18  0  0
##         36  1  1  0  0  0  0  0  0  0  1  0  0  9  0  0  0  0  0  0  0  0  0  0
##         37  0  0  0  0  0  0  0  0  0  0  1  0  0  8  0  0  0  0  1  0  0  0  0
##         38  0  0  0  0  0  0  1  0  0  0  3  3  0  0 12  0  0  0  0  7  2  2 12
##         39  0  7  0  0  1  5  4  3  4  1  0  0  0  5  0 22  1  2  1  0  0  1  0
##         40  0  3  3  2  0  4  0  0  2  2  5  2  1  5  0  2 40  1  0  0  0  2  0
##         41  3  0  0  0  0  0  0  0  1  0  5  0  1  2  1  0  0 32  2  0  0  0  0
##         42  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0
##         43  0  0  1  2  0  0  1  1  2  2  0  4  0  2  1  0  0  0  0 19  0  4  1
##         44  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  1  0  0  0  0  1  2  0  1
##         45  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0
```

```
##         46  0  0  0  0  1  0  0  2  0  0  1  2  0  0 31  0  0  0  0 13  5  2 24
##         47  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  5  0  0  0  0
##         48  0  1  0  0  0  0  4  1  0  0  0  0  0  1  0  1  1  0  0  0  0  0  0
##         49  0  0  0  0  0  0  0  0  0  0  0  2  0  1  0  0  0  0  0  0  0  0  0
##         50  0  0  0  0  2  0  0  0  0  0  3  1  1  0  0  0  0  0  0  0  6  0  0
##           Reference
## Prediction 47 48 49 50
##         1  11  0  2  0
##         2   0  0  0  0
##         3   0  2  0  0
##         4   0  0  0  5
##         5   0  0  0  0
##         6   0  0  0  0
##         7   0  0  0  0
##         8   0  0 14  0
##         9   0  1  0  0
##         10 14  0  1  1
##         11  2  1  5  7
##         12  0  0  0  0
##         13  0  0  0  0
##         14  0  0  0  0
##         15  0  0  0  2
##         16  0  1  3  0
##         17  0  2  0  0
##         18  0  0  1  0
##         19  0  0  0  0
##         20  0  4  1  0
##         21  0  0  5  3
##         22  0  0  0  0
##         23  0  0  1  0
##         24  0  2  1  0
##         25  0  0  0  0
##         26  2  0  0  0
##         27  5  4  3  3
##         28  0  0  0  0
##         29  0  0  0  0
##         30  0  0  0  0
##         31  0  0  0  0
##         32  1  0  0  0
##         33  0  8  1  2
##         34  0  0  1  0
##         35  0  0  0 10
##         36  0  0  0  0
##         37  1  0  0  0
##         38  0  0  0  3
##         39  0  1  2  2
##         40  2  1  1  0
##         41  2  1  0  0
##         42  2  0  0  0
##         43  0  0  1  1
##         44  0  0  0  6
##         45  0  3  0  0
##         46  0  0  1  3
##         47  7  0  0  0
```

```
##         48  0 19  1  0
##         49  1  0  5  0
##         50  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.3028
##                  95% CI : (0.2848, 0.3212)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2886
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.5400   0.1400   0.3000   0.1600   0.2600   0.0600
## Specificity            0.9747   0.9922   0.9947   0.9906   0.9910   0.9976
## Pos Pred Value         0.3034   0.2692   0.5357   0.2581   0.3714   0.3333
## Neg Pred Value         0.9905   0.9826   0.9858   0.9830   0.9850   0.9811
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0108   0.0028   0.0060   0.0032   0.0052   0.0012
## Detection Prevalence   0.0356   0.0104   0.0112   0.0124   0.0140   0.0036
## Balanced Accuracy      0.7573   0.5661   0.6473   0.5753   0.6255   0.5288
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
## Sensitivity           0.04000   0.1600   0.0400    0.3600    0.8600    0.3400
## Specificity           0.99020   0.9800   0.9984    0.9804    0.9437    0.9955
## Pos Pred Value        0.07692   0.1404   0.3333    0.2727    0.2376    0.6071
## Neg Pred Value        0.98060   0.9828   0.9808    0.9869    0.9970    0.9867
## Prevalence            0.02000   0.0200   0.0200    0.0200    0.0200    0.0200
## Detection Rate        0.00080   0.0032   0.0008    0.0072    0.0172    0.0068
## Detection Prevalence  0.01040   0.0228   0.0024    0.0264    0.0724    0.0112
## Balanced Accuracy     0.51510   0.5700   0.5192    0.6702    0.9018    0.6678
##                      Class: 13 Class: 14 Class: 15 Class: 16 Class: 17
## Sensitivity             0.2600    0.2800    0.0800    0.5200    0.8400
## Specificity             0.9910    0.9771    0.9865    0.9857    0.9739
## Pos Pred Value          0.3714    0.2000    0.1081    0.4262    0.3962
## Neg Pred Value          0.9850    0.9852    0.9813    0.9902    0.9967
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0052    0.0056    0.0016    0.0104    0.0168
## Detection Prevalence    0.0140    0.0280    0.0148    0.0244    0.0424
## Balanced Accuracy       0.6255    0.6286    0.5333    0.7529    0.9069
##                      Class: 18 Class: 19 Class: 20 Class: 21 Class: 22
## Sensitivity             0.0200    0.2800    0.3000    0.1400    0.1000
## Specificity             0.9984    0.9873    0.9788    0.9861    0.9996
## Pos Pred Value          0.2000    0.3111    0.2239    0.1707    0.8333
## Neg Pred Value          0.9804    0.9853    0.9856    0.9825    0.9820
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0004    0.0056    0.0060    0.0028    0.0020
## Detection Prevalence    0.0020    0.0180    0.0268    0.0164    0.0024
## Balanced Accuracy       0.5092    0.6337    0.6394    0.5631    0.5498
##                      Class: 23 Class: 24 Class: 25 Class: 26 Class: 27
```

```
## Sensitivity             0.1000    0.4600    0.2600    0.2600    0.7600
## Specificity             0.9959    0.9808    0.9963    0.9931    0.9376
## Pos Pred Value          0.3333    0.3286    0.5909    0.4333    0.1990
## Neg Pred Value          0.9819    0.9889    0.9851    0.9850    0.9948
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0020    0.0092    0.0052    0.0052    0.0152
## Detection Prevalence    0.0060    0.0280    0.0088    0.0120    0.0764
## Balanced Accuracy       0.5480    0.7204    0.6282    0.6265    0.8488
##                       Class: 28 Class: 29 Class: 30 Class: 31 Class: 32
## Sensitivity             0.5200    0.2400    0.6400    0.0400    0.2000
## Specificity             0.9980    0.9963    0.9837    0.9984    0.9951
## Pos Pred Value          0.8387    0.5714    0.4444    0.3333    0.4545
## Neg Pred Value          0.9903    0.9847    0.9926    0.9808    0.9839
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0104    0.0048    0.0128    0.0008    0.0040
## Detection Prevalence    0.0124    0.0084    0.0288    0.0024    0.0088
## Balanced Accuracy       0.7590    0.6182    0.8118    0.5192    0.5976
##                       Class: 33 Class: 34 Class: 35 Class: 36 Class: 37
## Sensitivity             0.7600    0.0200    0.4000    0.1800    0.1600
## Specificity             0.9580    0.9988    0.9739    0.9984    0.9963
## Pos Pred Value          0.2695    0.2500    0.2381    0.6923    0.4706
## Neg Pred Value          0.9949    0.9804    0.9876    0.9835    0.9831
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0152    0.0004    0.0080    0.0036    0.0032
## Detection Prevalence    0.0564    0.0016    0.0336    0.0052    0.0068
## Balanced Accuracy       0.8590    0.5094    0.6869    0.5892    0.5782
##                       Class: 38 Class: 39 Class: 40 Class: 41 Class: 42
## Sensitivity             0.2400    0.4400    0.8000    0.6400    0.2000
## Specificity             0.9824    0.9747    0.9694    0.9873    0.9980
## Pos Pred Value          0.2182    0.2619    0.3478    0.5079    0.6667
## Neg Pred Value          0.9845    0.9884    0.9958    0.9926    0.9839
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0048    0.0088    0.0160    0.0128    0.0040
## Detection Prevalence    0.0220    0.0336    0.0460    0.0252    0.0060
## Balanced Accuracy       0.6112    0.7073    0.8847    0.8137    0.5990
##                       Class: 43 Class: 44 Class: 45 Class: 46 Class: 47
## Sensitivity             0.3800   0.04000    0.1800    0.4800    0.1400
## Specificity             0.9763   0.98571    0.9931    0.9624    0.9959
## Pos Pred Value          0.2468   0.05405    0.3462    0.2069    0.4118
## Neg Pred Value          0.9872   0.98051    0.9834    0.9891    0.9827
## Prevalence              0.0200   0.02000    0.0200    0.0200    0.0200
## Detection Rate          0.0076   0.00080    0.0036    0.0096    0.0028
## Detection Prevalence    0.0308   0.01480    0.0104    0.0464    0.0068
## Balanced Accuracy       0.6782   0.51286    0.5865    0.7212    0.5680
##                       Class: 48 Class: 49 Class: 50
## Sensitivity             0.3800    0.1000    0.0400
## Specificity             0.9808    0.9955    0.9931
## Pos Pred Value          0.2879    0.3125    0.1053
## Neg Pred Value          0.9873    0.9819    0.9807
## Prevalence              0.0200    0.0200    0.0200
## Detection Rate          0.0076    0.0020    0.0008
## Detection Prevalence    0.0264    0.0064    0.0076
## Balanced Accuracy       0.6804    0.5478    0.5165
```

Question 6

```
rm(list = ls())

groceries = read.transactions('groceries.txt', sep = ',',
                              rm.duplicates = T, header = F,
                              format = 'basket')
summary(groceries)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
grocery_rules = apriori(groceries,
    parameter=list(support=.005, confidence=.1, maxlen=5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##       5  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
```

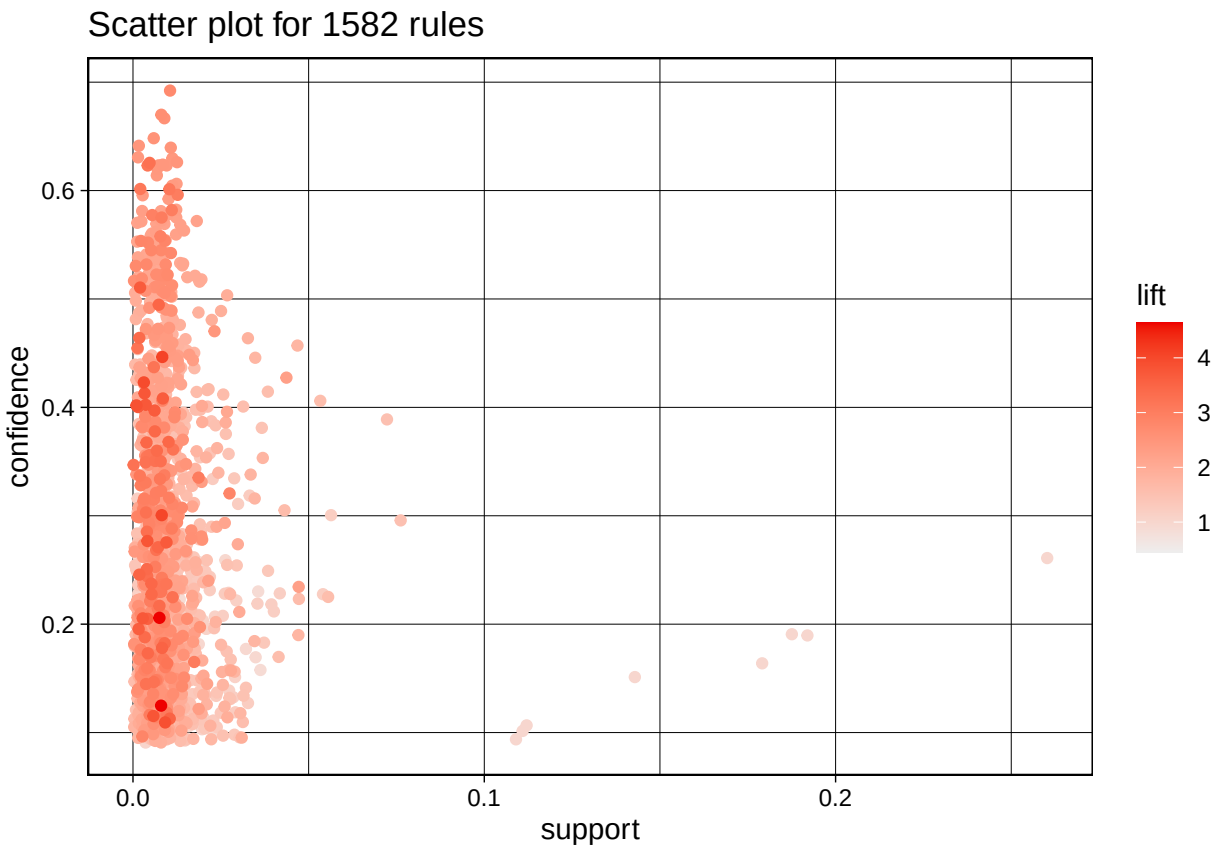
```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [1582 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(subset(grocery_rules, subset=lift > 4))
```

```
##     lhs                    rhs                      support confidence    coverage     lift count
## [1] {ham}               => {white bread}        0.005083884  0.1953125 0.02602949 4.639851    50
## [2] {white bread}       => {ham}                0.005083884  0.1207729 0.04209456 4.639851    50
## [3] {butter,
##      other vegetables} => {whipped/sour cream} 0.005795628  0.2893401 0.02003050 4.036397    57
## [4] {citrus fruit,
##      other vegetables,
##      whole milk}       => {root vegetables}    0.005795628  0.4453125 0.01301474 4.085493    57
```

```
plot(grocery_rules, measure = c("support", "confidence"), shading = 'lift')
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



```
saveAsGraph(head(grocery_rules, n = 1000, by = "lift"), file = "musicrules.graphml")
```