Homework 2 Starter Code

Richard Chen

April 8, 2016

Note

This pdf is produced by R Markdown. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com. When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. One way to produce nice-looking reports for your homework assignments using R Markdown, you need to install the package `knitr` http://yihui.name/knitr/. This document is prepared by R Markdown which is built on the `knitr` package aforementioned.

For any R function which you feel confused about, please utilize the help function in R for detailed instructions. For example, function `cor()` help use to compute correlation (matrices), if you would like to know the details of its usage, input the R command `help(cor)`.

Additional note:

1. In the case where you have installed some package, for example `knitr`, and relaunch R and need to use some functionalities of the R package , you need to run the command `library(knitr)` before you use the functionalities of the corresponding package. If you never installed the package before, you need to run `install.packages("knitr")`, or click the "Packages" button on the top of the right lower window in the R Studio interface.
2. Do not forget to set your working directory at the begining.

Example R Code for Question 1

a

Download and read the data

```
# download data to the current working folder
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/CountryMonthlyReturns2.csv",
              destfile="CountryMonthlyReturns2.csv")
# assign the data to a variable
countryReturn_df = read.csv("CountryMonthlyReturns2.csv")
```

Construct the returns of the equi-weighted portfolio:

```
port_df = countryReturn_df[,c("Canada", "Japan")]
port_df$port1 = 0.5*port_df$Canada + 0.5*port_df$Japan
head(port_df)
```

```
##   Canada  Japan   port1
## 1  0.055  0.048  0.0515
## 2 -0.005 -0.033 -0.0190
## 3  0.010  0.068  0.0390
## 4  0.036  0.030  0.0330
## 5  0.021 -0.004  0.0085
## 6 -0.036  0.026 -0.0050
```

then compute the mean and standard deviation of the return series

```
mean(port_df$port1); sd(port_df$port1)
```

```
## [1] 0.004890625
```

```
## [1] 0.04475644
```

c

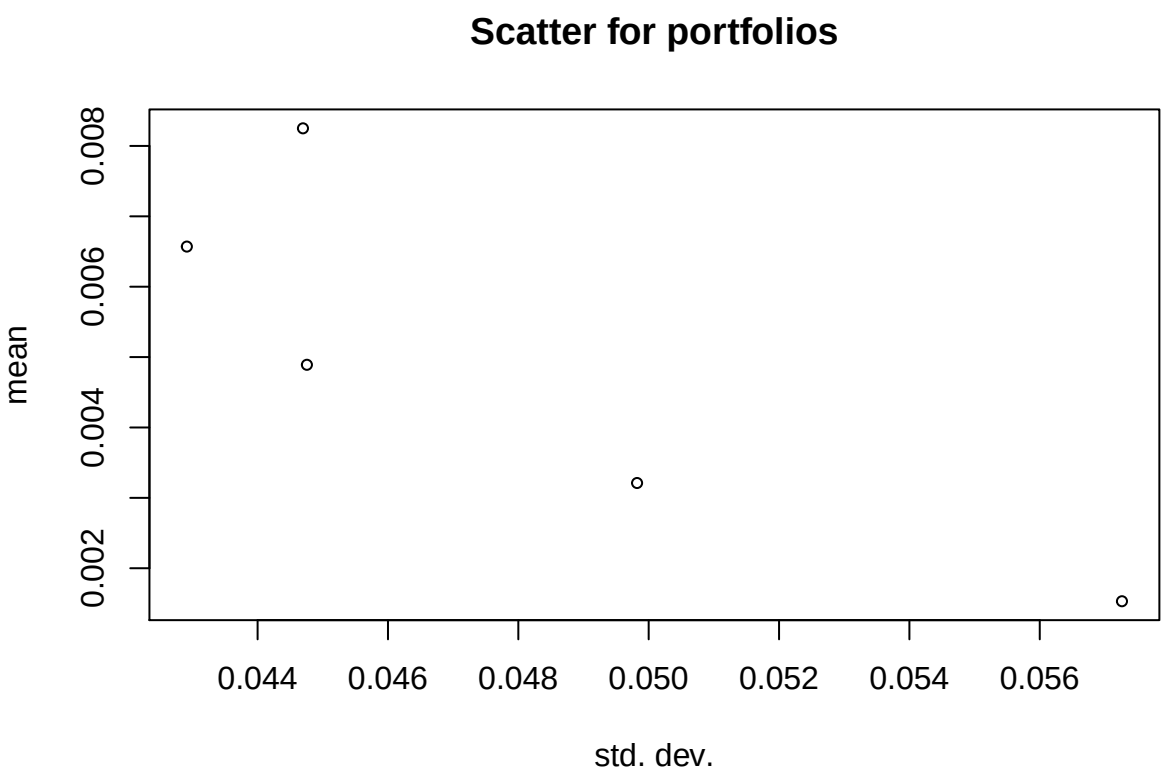
When the weight for the Canada is 25% and 75% respectively,

```
port_df$port2 = 0.25*port_df$Canada + 0.75*port_df$Japan
port_df$port3 = 0.75*port_df$Canada + 0.25*port_df$Japan

mcr = sapply(port_df, mean)            # compute mean for each column
sdcr = sapply(port_df, sd)             # compute standard deviation for each column
```

Now make the standard deviation-mean scatter plot:

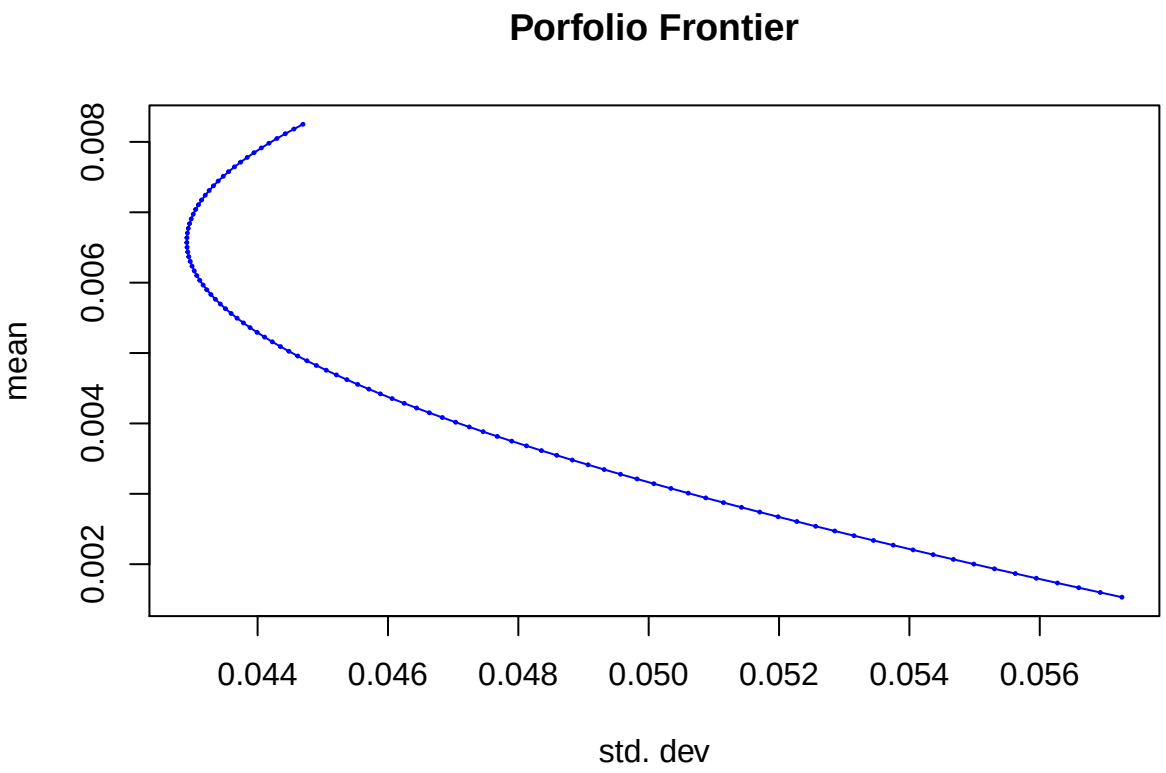
```
plot(sdcr, mcr, type="p", main="Scatter for portfolios", xlab = "std. dev.", ylab = "mean", cex=0.7)
```



```
# pointLabel(sdcr, mcr, labels=names(mcr), cex= 1)
```

d. efficient frontier

```
weights = seq(0,1,0.01)
m = weights*mean(port_df$Canada)+(1-weights)*mean(port_df$Japan)
s = sqrt(weights^2*var(port_df$Canada)+(1-weights)^2*var(port_df$Japan)
               +2*weights*(1-weights)*cov(port_df$Canada,port_df$Japan))
plot(s, m, type="b", col="blue", cex = 0.2,
     xlab="std. dev", ylab="mean", main="Porfolio Frontier")
lines(s,m,type="l", col="blue")
```



Code for Question 3

Download and read the data (reminder: we saw this dataset in lecture01: see 01_core.R)

```
# download data to the current working folder
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/bank.csv",
              destfile="bank.csv")
# assign the data to a variable
bank = read.csv("bank.csv")
int = bank$InterarrivalTime
```

Sample Code for Question 4

Download and read the data

```
# download data to the current working folder
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/studenthw.csv",
              destfile="studenthw.csv")
# assign the data to a variable
studenthw = read.csv("studenthw.csv")
```

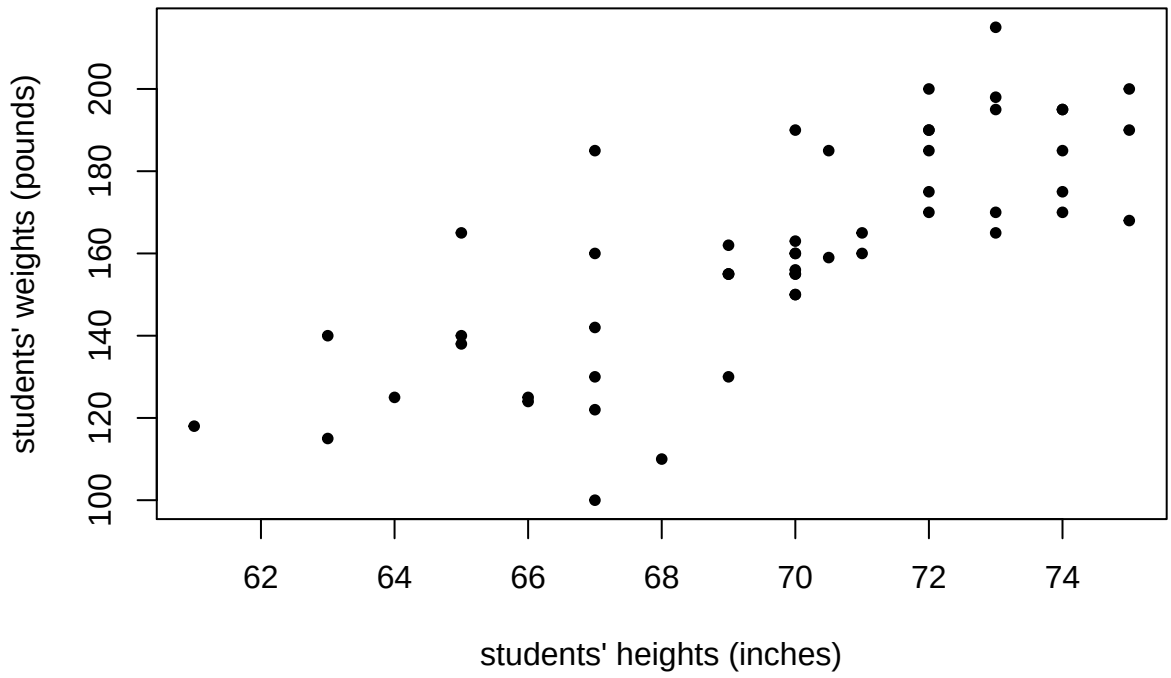
b

Scatter plot of heights and weights:

```
head(studenthw)
```

```
##   height weight
## 1     72    175
## 2     64    125
## 3     65    165
## 4     67    100
## 5     70    155
## 6     75    168
```

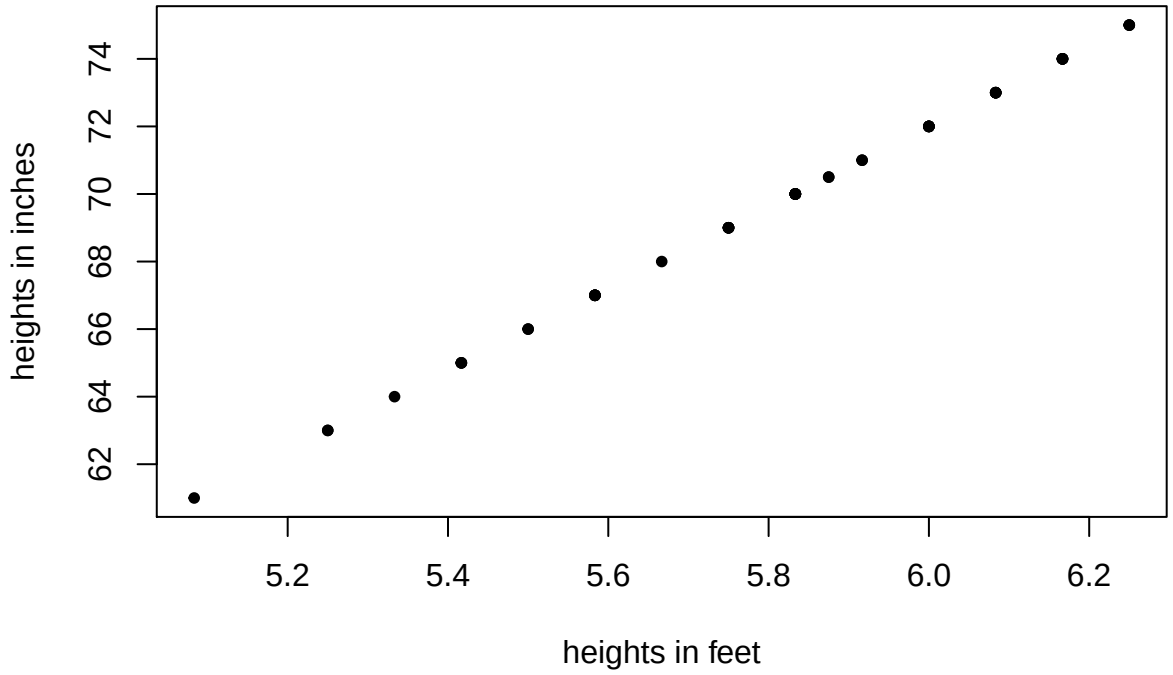
```
plot(studenthw$height, studenthw$weight, pch=20,
     xlab = "students' heights (inches)", ylab = "students' weights (pounds)", main = "")
```



e

Scatter plot of heights in feet and heights in inches

```r
height.inch = studenthw$height
height.foot = studenthw$height/12
plot(height.foot, height.inch, pch=20,
     xlab = "heights in feet", ylab = "heights in inches", main = "")
```



Question 5. First Stab at Linear Regression

Download and read the data

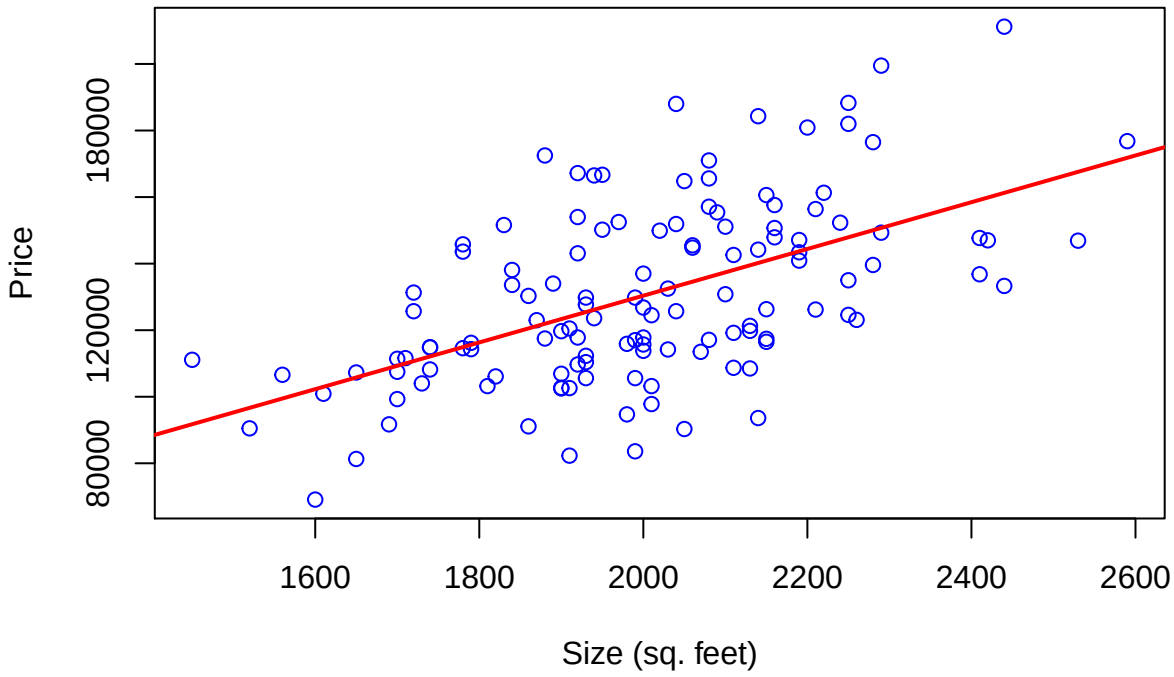
```r
# download data to the current working folder
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/housesp1.csv",
              destfile="housesp1.csv")
# assign the data to a variable
homep_df = read.csv("housesp1.csv")
```

b. linear coefficients by R

The R output about the linear regression estimates is:

```r
# creat scatter plot
plot(homep_df$size, homep_df$price, col = "blue", xlab = "Size (sq. feet)", ylab = "Price")

# add a regression linea
reg = lm(price~size, data = homep_df)
abline(reg, lw = 2, col = "red")
```




```
# summary of linear regression
summary(reg)
```

```
##
## Call:
## lm(formula = price ~ size, data = homep_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -46593 -16644  -1610  15124  54829
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10091.130  18966.104  -0.532    0.596
## size            70.226      9.426   7.450  1.3e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22480 on 126 degrees of freedom
## Multiple R-squared:  0.3058, Adjusted R-squared:  0.3003
## F-statistic:  55.5 on 1 and 126 DF,  p-value: 1.302e-11
```

```
hist(homep_df$price, breaks=20,
     main="Histogram of house prices", xlab="Price", ylab="Frequency")
```

d. prediction baesd on linear regression

```
# download data to the current working folder
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/housesp2.csv",
              destfile="housesp2.csv")
# assign the data to a variable
homep2_df = read.csv("housesp2.csv")

# the new data only contains column sizenew
head(homep_df)
```

```
##   size  price
## 1 1790 114300
## 2 2030 114200
## 3 1740 114800
## 4 1980  94700
## 5 2130 119800
## 6 1780 114600
```

```
# rename the column, it needs to be the same name as in homep_df
colnames(homep2_df) = c("size")
```

Predict prices for new houses

```
yhat = predict(reg, homep2_df)
head(yhat)
```

```
##        1        2        3        4        5        6
## 140193.2 171795.0 123338.9 141597.7 145109.0 126850.2
```

Question 6. Wine Offers

Download data to the current working folder

```
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/Wine_OfferInformation.csv",
              destfile="Wine_OfferInformation.csv")
download.file("https://github.com/mlakolar/BUS41000/raw/master/data/Wine_Transactions.csv",
              destfile="Wine_Transactions.csv")

# load data into R
offers_df       = read.csv("Wine_OfferInformation.csv", row.names = 1)
transactions_df = read.csv("Wine_Transactions.csv")
```

a. cluster analysis

```
# initializa an empty table where rows correspond to customers and columns correspond to offers
customer_by_offer = matrix(0, length(unique(transactions_df[,1])), nrow(offers_df))
rownames(customer_by_offer) = levels(transactions_df[,1])
colnames(customer_by_offer) = 1:nrow(offers_df)

# put transactions into the table
for (i in 1:nrow(transactions_df)) {
  customer_by_offer[ transactions_df[i,1], transactions_df[i,2] ] = 1
}

# customer_by_offer now contains information on which offers did each customer get
# for example, here are offers by "Fisher" or "Smith"
customer_by_offer[ "Fisher", ]
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0
## 26 27 28 29 30 31 32
##  0  0  1  0  1  1  0
```

```
customer_by_offer[ "Smith", ]
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
## 26 27 28 29 30 31 32
##  0  0  0  0  0  0  0
```

```
# find 4 clusters
set.seed(1)       # for reproductivity
grpCustomers = kmeans(customer_by_offer, centers = 4, nstart = 1000)

# take a look at cluster means and try to summarize different groups
grpCustomers
```

```
## K-means clustering with 4 clusters of sizes 24, 21, 15, 40
##
## Cluster means:
##           1         2         3         4          5         6         7
## 1 0.00000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.5000000
## 2 0.23809524 0.0952381 0.1904762 0.1904762 0.04761905 0.2857143 0.1428571
## 3 0.06666667 0.4000000 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000
## 4 0.10000000 0.0500000 0.0500000 0.2000000 0.07500000 0.1500000 0.1000000
##           8         9         10        11        12   13        14
## 1 0.4583333 0.0000000 0.00000000 0.0000000 0.0000000 0.25 0.0000000
## 2 0.1428571 0.0952381 0.09523810 0.2380952 0.0952381 0.00 0.1904762
## 3 0.0000000 0.0000000 0.06666667 0.0000000 0.0000000 0.00 0.0000000
## 4 0.1500000 0.2000000 0.10000000 0.2000000 0.0750000 0.00 0.1250000
##          15    16        17         18        19         20        21 22
## 1 0.0000000 0.000 0.0000000 0.45833333 0.0000000 0.00000000 0.0000000  0
## 2 0.1904762 0.000 0.0000000 0.04761905 0.1428571 0.04761905 0.0952381  1
## 3 0.0000000 0.000 0.4666667 0.00000000 0.0000000 0.00000000 0.0000000  0
## 4 0.0500000 0.125 0.0000000 0.05000000 0.0500000 0.12500000 0.0500000  0
##           23  24        25         26         27        28    29        30
## 1 0.00000000 0.0 0.0000000 0.00000000 0.00000000 0.0000000 0.625 0.6666667
## 2 0.09523810 0.0 0.0952381 0.04761905 0.14285714 0.0952381 0.000 0.1904762
## 3 0.06666667 0.8 0.0000000 0.73333333 0.06666667 0.0000000 0.000 0.0000000
## 4 0.05000000 0.0 0.1000000 0.07500000 0.12500000 0.1000000 0.050 0.0500000
##          31         32
## 1 0.0000000 0.00000000
## 2 0.3333333 0.04761905
## 3 0.0000000 0.00000000
## 4 0.2500000 0.07500000
##
## Clustering vector:
##      Adams      Allen   Anderson     Bailey      Baker     Barnes
##          1          4          3          1          4          2
##       Bell    Bennett     Brooks      Brown     Butler   Campbell
##          3          1          2          1          2          3
##     Carter      Clark    Collins       Cook     Cooper        Cox
##          1          4          4          3          4          3
##       Cruz      Davis       Diaz    Edwards      Evans     Fisher
##          1          2          1          4          2          2
##     Flores     Foster     Garcia      Gomez   Gonzalez       Gray
##          3          2          4          4          4          4
##      Green  Gutierrez       Hall     Harris  Hernandez       Hill
##          4          4          2          2          4          1
##     Howard     Hughes    Jackson      James    Jenkins    Johnson
##          2          1          2          1          3          3
##      Jones      Kelly       King        Lee      Lewis       Long
##          4          4          1          2          1          4
##      Lopez     Martin   Martinez     Miller   Mitchell      Moore
##          4          4          4          2          4          3
##    Morales     Morgan     Morris     Murphy      Myers     Nelson
##          2          1          3          4          1          4
##     Nguyen      Ortiz     Parker      Perez      Perry   Peterson
##          4          4          4          1          1          3
##   Phillips     Powell      Price    Ramirez       Reed      Reyes
##          3          4          2          4          4          4
```

```
## Richardson     Rivera    Roberts   Robinson  Rodriguez     Rogers
##          2          1          4          1          3          2
##       Ross    Russell    Sanchez    Sanders      Scott      Smith
##          4          3          2          4          4          3
##    Stewart   Sullivan     Taylor     Thomas   Thompson     Torres
##          1          1          1          4          4          4
##     Turner     Walker       Ward     Watson      White   Williams
##          4          1          4          1          2          2
##     Wilson       Wood     Wright      Young
##          1          4          4          2
##
## Within cluster sum of squares by cluster:
## [1]  33.37500  62.28571  16.40000 100.60000
##  (between_SS / total_SS =  24.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

b. Wines favored by customers in each cluster

Let check what offers did members of each cluster buy – rank the offers from most popular of offer to least popular offer within each cluster.

```
# find members in each cluster
cluster.1 = rownames(customer_by_offer)[grpCustomers$cluster==1]
cluster.2 = rownames(customer_by_offer)[grpCustomers$cluster==2]
cluster.3 = rownames(customer_by_offer)[grpCustomers$cluster==3]
cluster.4 = rownames(customer_by_offer)[grpCustomers$cluster==4]

# extract the information of the offers in each cluster
cluster_by_offer = matrix(0,4,nrow(offers_df))
rownames(cluster_by_offer) = 1:4
colnames(cluster_by_offer) = 1:nrow(offers_df)

for (i in 1:nrow(offers_df)){
  cluster_by_offer[1,i] = sum(customer_by_offer[cluster.1,i])
  cluster_by_offer[2,i] = sum(customer_by_offer[cluster.2,i])
  cluster_by_offer[3,i] = sum(customer_by_offer[cluster.3,i])
  cluster_by_offer[4,i] = sum(customer_by_offer[cluster.4,i])
}

# Rank the offers by their numbers in each cluster
sort(cluster_by_offer[1,], decreasing = TRUE)
```

```
## 30 29  7  8 18 13  1  2  3  4  5  6  9 10 11 12 14 15 16 17 19 20 21 22 23
## 16 15 12 11 11  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 24 25 26 27 28 31 32
##  0  0  0  0  0  0  0
```

```r
sort(cluster_by_offer[2,], decreasing = TRUE)
```

```
## 22 31  6  1 11  3  4 14 15 30  7  8 19 27  2  9 10 12 21 23 25 28  5 18 20
## 21  7  6  5  5  4  4  4  4  4  3  3  3  3  2  2  2  2  2  2  2  2  1  1  1
## 26 32 13 16 17 24 29
##  1  1  0  0  0  0  0
```

```r
sort(cluster_by_offer[3,], decreasing = TRUE)
```

```
## 24 26 17  2  1 10 23 27  3  4  5  6  7  8  9 11 12 13 14 15 16 18 19 20 21
## 12 11  7  6  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 22 25 28 29 30 31 32
##  0  0  0  0  0  0  0
```

```r
sort(cluster_by_offer[4,], decreasing = TRUE)
```

```
## 31  4  9 11  6  8 14 16 20 27  1  7 10 25 28  5 12 26 32  2  3 15 18 19 21
## 10  8  8  8  6  6  5  5  5  5  4  4  4  4  4  3  3  3  3  2  2  2  2  2  2
## 23 29 30 13 17 22 24
##  2  2  2  0  0  0  0
```