2020/2021 Fall RMSC4002 Assignment 2

Chan Sze Yuen Syngiene 1155127616

11/13/2020

The code used in this assignment is written by R.

1

The stock we are using here is 0151.HK, as the same stock used in assignment 1

From the assignment 1, we obtain the equaiton of the esimated voility computed by the EWMA model with parameter lambda=0.985 as follow:

$$\sigma_i^2 = 0.985\sigma_{i-1}^2 + (1 - 0.985)u_{i-1}^2$$

We set $\sigma_0 = 0.02358510$, which is the fisrt 30-days volatility. Or $\sigma_0^2 = 0.0005562569$ In part(c), since we have 491 simulated scenario, we use the linear interpolation to to find an approximated value of the one day 0.99 VaR, Which is defined as

$$VaR(0.99) = L_{sim}(4) + (491 * 0.01 - 4)(L_{sim}(5) - L_{sim}(4))$$

After the Back Testing step with using most recent 252 days stock price, we found that the total number of exception of the 3 models are all 0, which mean these 3 method over-estimate the risk. Hence, having the smallest VaR value works better, which is hsim2 (L12 in the attached excel file).

The details of the work and the related code please refer to the attached excel file.

2(a)

```
#import the data:
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
PokemonData=read_csv("D:\\CUHKZOOMNOTESANDSOURCE\\RMSC4002\\DATA\\Pokemon.csv")
```

```
## Parsed with column specification:
## cols(
##   `#` = col_double(),
##   Name = col_character(),
##   `Type 1` = col_character(),
##   `Type 2` = col_character(),
##   HP = col_double(),
##   Attack = col_double(),
##   Defense = col_double(),
##   `Sp. Atk` = col_double(),
##   `Sp. Def` = col_double(),
##   Speed = col_double(),
##   Generation = col_double(),
##   Legendary = col_logical()
## )
```

data processing steps:

```
#extracting the columns E to J data
Species_Strength_DataMatrix=PokemonData[,c(seq(5,5+5,1))]
#finding the correlation between different areas of strength
cor(Species_Strength_DataMatrix)
```

```
##                HP    Attack   Defense   Sp. Atk   Sp. Def     Speed
## HP      1.0000000 0.4223860 0.2396223 0.3623799 0.3787181 0.1759521
## Attack  0.4223860 1.0000000 0.4386871 0.3963618 0.2639896 0.3812397
## Defense 0.2396223 0.4386871 1.0000000 0.2235486 0.5107466 0.0152266
## Sp. Atk 0.3623799 0.3963618 0.2235486 1.0000000 0.5061214 0.4730179
## Sp. Def 0.3787181 0.2639896 0.5107466 0.5061214 1.0000000 0.2591331
## Speed   0.1759521 0.3812397 0.0152266 0.4730179 0.2591331 1.0000000
```

```
#perform PCA by corr.matrix
pcaQ2=princomp(Species_Strength_DataMatrix,cor=T)
#display the loadings of the loadings of the first six PCAs
pcaQ2$loadings[,1:6]
```

```
##            Comp.1      Comp.2      Comp.3     Comp.4      Comp.5     Comp.6
## HP      0.3898858  0.08483455  0.47192614  0.7176913  0.21999056  0.2336690
## Attack  0.4392537 -0.01182493  0.59415339 -0.4058359 -0.19025457 -0.5029896
## Defense 0.3637473  0.62878867 -0.06933913 -0.4192373  0.05903197  0.5368986
## Sp. Atk 0.4571623 -0.30541446 -0.30561186  0.1475166 -0.73534497  0.2045304
## Sp. Def 0.4485704  0.23909670 -0.56559403  0.1854448  0.30019970 -0.5451707
## Speed   0.3354405 -0.66846305 -0.07851327 -0.2971625  0.53016082  0.2551400
```

Hence, we will have the first PC is:

$$y_1 = 0.3898858x_{HP} + 0.4392537x_{Attack} + ... + 0.3354405x_{Speed}$$

and the second PC is:

$$y_2 = 0.08483455x_{HP} - 0.01182493x_{Attack} + ... - 0.66846305x_{Speed}$$

2(b)

The first PC telling us that the loadings are all positive and their values are similarly within the range [0.3,0.5], this PC mayber refering to the overall strength of the pokemon as higher basic stat of the pokemon the strong that pokemon is in general.

The second PC telling us that the higher defense stat that pokemon have(like Defense and Sp.Def), the lower power that pokemon have(like Attack, Sp.Atk and Speed) in general.So this PC may refer to the trade-off between defense stat and attacking stat.

2(c)

```
s=pcaQ2$sdev #save the sd of all PC to s
s #display sd
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 1.6466450 1.0457158 0.8824654 0.8489201 0.6546298 0.5168055
```

```
round(s^2,4) #display variance
```

```
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## 2.7114 1.0935 0.7787 0.7207 0.4285 0.2671
```

```
t=sum(s^2) #compute sum of variance of all PCs
round(s^2/t,4) #prop. of var. explained by each PC
```

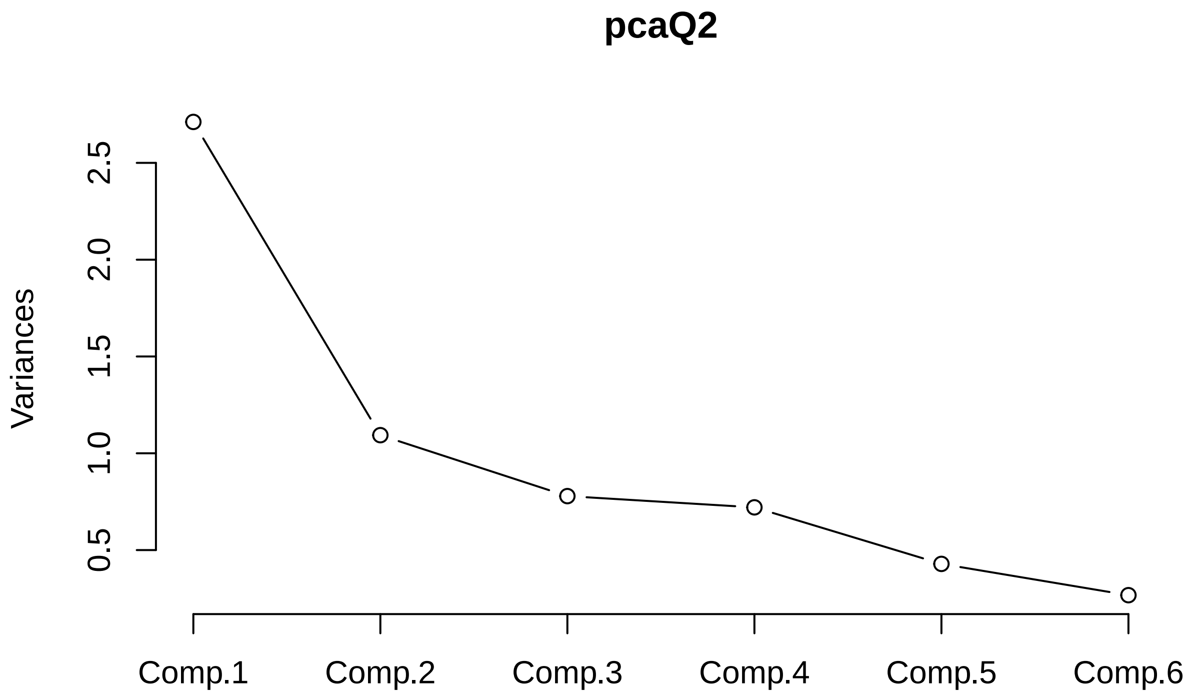
```
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## 0.4519 0.1823 0.1298 0.1201 0.0714 0.0445
```

```
cumsum(s^2/t) #cumulative sum of prop. of var.
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 0.4519067 0.6341602 0.7639511 0.8840620 0.9554853 1.0000000
```

As we can see, the first PCs just explained near 60% of the total variation of different strength area stats. So preheps we should use more PCs in order to have enough power to explain the variation. We can also check the screeplot as below:

```
screeplot(pcaQ2,type="lines")
```



From the plot, there is no clear “elbow”, which means using the first two PCs only is not enough to explain the total variation.

3(a)

```
#import the data:
library(readr)
d=read_csv("D:\\CUHKZOOMNOTESANDSOURCE\\RMSC4002\\DATA\\credit.csv")
```

```
## Parsed with column specification:
## cols(
##   Age = col_double(),
##   Address = col_double(),
##   Employ = col_double(),
##   Bank = col_double(),
##   House = col_double(),
##   Save = col_double(),
##   Result = col_double()
## )
```

3(b)

```
set.seed(27616) #my student id is 1155127616
id=sample(1:690, size=600)
d1=d[id,]
d2=d[-id,]
```

3(c)

```
#fit the logsitc regression and show the ANOVA table of the fitted model
fit0=glm(Result~Age+Address+Employ+Bank+House+Save,data=d1,binomial)
summary(fit0)
```

```
##
## Call:
## glm(formula = Result ~ Age + Address + Employ + Bank + House +
##     Save, family = binomial, data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1983  -0.7424  -0.6048   0.6794   2.0408
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.2969581  0.3364619  -3.855 0.000116 ***
## Age         -0.0063734  0.0094781  -0.672 0.501306
## Address      0.0214496  0.0218908   0.980 0.327162
## Employ       0.2519893  0.0465472   5.414 6.18e-08 ***
## Bank         0.3250025  0.0459385   7.075 1.50e-12 ***
## House       -0.0008385  0.0006572  -1.276 0.202024
## Save         0.0003833  0.0001083   3.539 0.000402 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 819.98  on 599  degrees of freedom
## Residual deviance: 590.02  on 593  degrees of freedom
## AIC: 604.02
##
## Number of Fisher Scoring iterations: 6
```

```
summary(glm(Result~Address+Employ+Bank+House+Save, data=d1, binomial))
```

```
##
## Call:
## glm(formula = Result ~ Address + Employ + Bank + House + Save,
##     family = binomial, data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2539  -0.7344  -0.6155   0.6769   2.0123
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.4743275  0.2115617  -6.969 3.20e-12 ***
## Address      0.0204412  0.0218224   0.937 0.348909
## Employ       0.2427173  0.0442498   5.485 4.13e-08 ***
## Bank         0.3260576  0.0460321   7.083 1.41e-12 ***
## House       -0.0008183  0.0006553  -1.249 0.211762
## Save         0.0003804  0.0001078   3.528 0.000419 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 819.98  on 599  degrees of freedom
## Residual deviance: 590.47  on 594  degrees of freedom
## AIC: 602.47
##
## Number of Fisher Scoring iterations: 6
```

```
#Removed Age, which has the largest p-value
summary(glm(Result~Address+Employ+Bank+Save, data=d1, binomial))
```

```
##
## Call:
## glm(formula = Result ~ Address + Employ + Bank + Save, family = binomial,
##     data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2893  -0.7300  -0.6114   0.7146   1.9075
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.6424682  0.1668634  -9.843  < 2e-16 ***
## Address      0.0247120  0.0214400   1.153  0.24907
## Employ       0.2407727  0.0439654   5.476 4.34e-08 ***
## Bank         0.3308472  0.0460033   7.192 6.39e-13 ***
## Save         0.0003685  0.0001060   3.476  0.00051 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 819.98  on 599  degrees of freedom
## Residual deviance: 592.15  on 595  degrees of freedom
## AIC: 602.15
##
## Number of Fisher Scoring iterations: 6
```

```
#Remove House, which has the largest p-value
summary(glm(Result~Employ+Bank+Save, data=d1, binomial))
```

```
##
```

```
## Call:
## glm(formula = Result ~ Employ + Bank + Save, family = binomial,
##     data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2234  -0.7329  -0.6274   0.7057   1.8652
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5464795  0.1423194 -10.866  < 2e-16 ***
## Employ       0.2461254  0.0438009   5.619 1.92e-08 ***
## Bank         0.3343735  0.0458780   7.288 3.14e-13 ***
## Save         0.0003728  0.0001070   3.485 0.000493 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 819.98  on 599  degrees of freedom
## Residual deviance: 593.46  on 596  degrees of freedom
## AIC: 601.46
##
## Number of Fisher Scoring iterations: 6
```

```r
#Remove Address, p-value=0.24907
#only keep the significance(p-value<0.05) variable to the new model:
fit1=glm(Result~Employ+Bank+Save,data=d1,binomial)
summary(fit1) #new model
```

```
##
## Call:
## glm(formula = Result ~ Employ + Bank + Save, family = binomial,
##     data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2234  -0.7329  -0.6274   0.7057   1.8652
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5464795  0.1423194 -10.866  < 2e-16 ***
## Employ       0.2461254  0.0438009   5.619 1.92e-08 ***
## Bank         0.3343735  0.0458780   7.288 3.14e-13 ***
## Save         0.0003728  0.0001070   3.485 0.000493 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 819.98  on 599  degrees of freedom
## Residual deviance: 593.46  on 596  degrees of freedom
## AIC: 601.46
##
```

```
## Number of Fisher Scoring iterations: 6
```

```
lreg=fit1
names(lreg) # display items in lreg
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"    
##  [4] "effects"           "R"                 "rank"             
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"    
## [13] "iter"              "weights"           "prior.weights"    
## [16] "df.residual"       "df.null"           "y"                
## [19] "converged"         "boundary"          "model"            
## [22] "call"              "formula"           "terms"            
## [25] "data"              "offset"            "control"          
## [28] "method"            "contrasts"         "xlevels"
```

```
pr=(lreg$fitted.values>0.5) # pr=True if fitted >0.
#classification table for this logistic regression on the training dataset d1.
table_d1=table(pr,d1$Result) # tabulation of pr & Result
print(table_d1)
```

```
##        
## pr        0   1
##   FALSE 311 106
##   TRUE   31 152
```

3(d)

```
d2hat=predict(lreg,d2) # predict the value
pt=exp(d2hat)/(1+exp(d2hat)) #Calculate the probability under the def. of log-odd ratio
Prediction=(pt>0.5) #Prediction
table_d2=table(Prediction,d2$Result) #Display the result
```

3(e)

```
##dataset d1 first

#Precision of the model using training dataset d1
Precision_d1=(table_d1[1,1])/sum(table_d1[1,])
print(Precision_d1)
```

```
## [1] 0.7458034
```

```
#Recall of the model using training dataset d1
Recall_d1=(table_d1[1,1])/sum(table_d1[2,1]+table_d1[1,1])
print(Recall_d1)
```

```
## [1] 0.9093567
```

```
#F1 score of the model using training dataset d1
F1_d1=(2*Precision_d1*Recall_d1)/(Precision_d1+Recall_d1)
print(F1_d1)
```

```
## [1] 0.8194993
```

```
##Now we consider the dataset d2
#Precision of the model using training dataset d1
Precision_d2=(table_d2[1,1])/sum(table_d2[1,])
print(Precision_d2)
```

```
## [1] 0.6415094
```

```
#Recall of the model using training dataset d1
Recall_d2=(table_d2[1,1])/sum(table_d2[2,1]+table_d2[1,1])
print(Recall_d2)
```

```
## [1] 0.8292683
```

```
#F1 score of the model using training dataset d1
F1_d2=(2*Precision_d2*Recall_d2)/(Precision_d2+Recall_d2)
print(F1_d2)
```

```
## [1] 0.7234043
```

```
##Comparing different measure of accuracy tools using d1 and d2!
Precision_d1>Precision_d2
```

```
## [1] TRUE
```

```
#Precision of model using d1 is higher than d2
Recall_d1>Recall_d2
```

```
## [1] TRUE
```

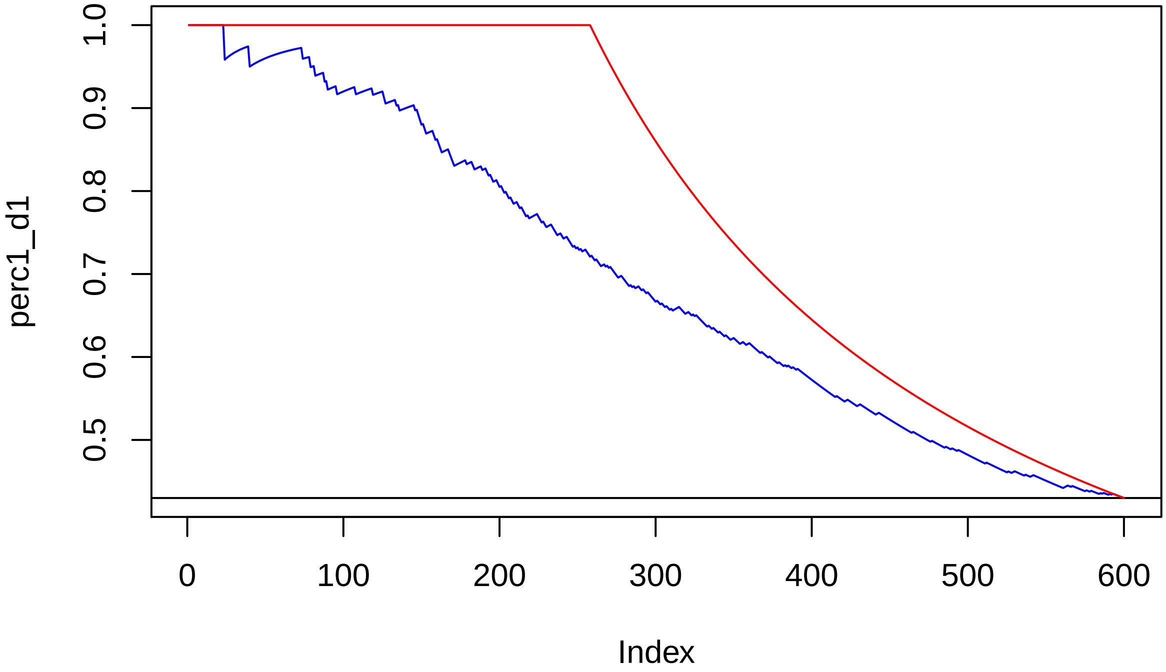
```
#Recall of model using d1 is higher than d2
F1_d1>F1_d2
```

```
## [1] TRUE
```

```
#F1 score of model using d1 is higher than d2
#Hence, The overall accuracy of model using d1 is higher than d2
```

3(f)

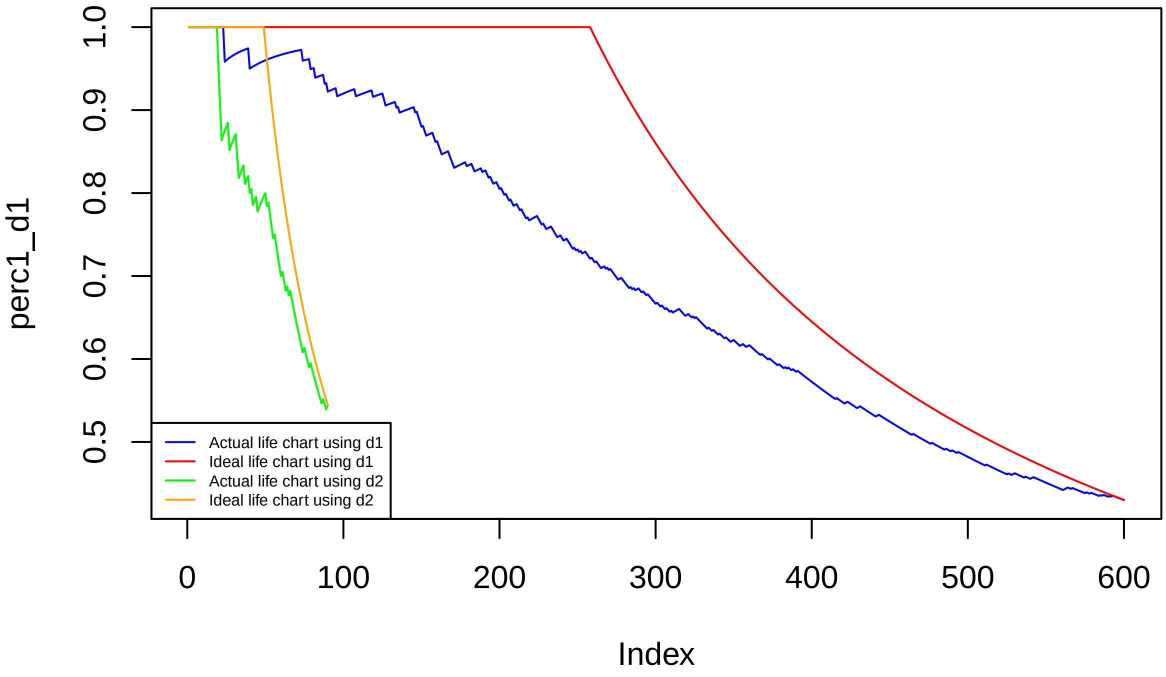
```
ysort_d1<-d1$Result[order(lreg$fit,decreasing=T)] # sort y
n<-length(ysort_d1) # length of ysort
perc1_d1<-cumsum(ysort_d1)/(1:n) # cumulative. perc.
plot(perc1_d1,type="l", col="blue") #plot perc.
abline(h=sum(d1$Result)/n) # add baseline
yideal_d1 <- c(rep(1,sum(d1$Result)),rep(0,length(d1$Result)-sum(d1$Result)))
# the ideal case
perc_ideal_d1 <- cumsum(yideal_d1)/(1:n)
# compute cumulative percentage of ideal case
lines(perc_ideal_d1, type="l", col="red") # plot ideal case
```



3(g)

```
##continuous:
plot(perc1_d1,type="l", col="blue")
lines(perc_ideal_d1, type="l", col="red")
ysort_d2<-d2$Result[order(pt,decreasing=T)] # sort y
n<-length(ysort_d2) # length of ysort
perc1_d2<-cumsum(ysort_d2)/(1:n) # cumulative. perc.
lines(perc1_d2,type="l", col="green") #plot perc.
yideal_d2 <- c(rep(1,sum(d2$Result)),rep(0,length(d2$Result)-sum(d2$Result)))
# the ideal case
perc_ideal_d2 <- cumsum(yideal_d2)/(1:n)
```

```r
# compute cumulative percentage of ideal case
lines(perc_ideal_d2, type="l", col="orange") # plot ideal case
legend("bottomleft",
legend=c("Actual life chart using d1",
         "Ideal life chart using d1",
         "Actual life chart using d2",
         "Ideal life chart using d2"),
       col=c("blue", "red" ,"green" ,"orange"),
lty=1:1, cex=0.5)
```



3(h)

There is more space between the ideal line with actual d1 dataset line, so there is a higher error in d1 dataset compared to the actual d2 dataset.

The accuracy of the d2 dataset is higher in this case.

4(a)

Sorry that I cannot find a suitable dataset for this question(they are all too big and too messy for the data cleaning…really sorry about that). So I will create a 100x20 data matrix R with row equals to the 20 user (say,u1,u2,…,u100) and the colume equals to the 20 movie (say, m1,m2,…,m20). The ranking will be randomly generated using uniform discrete distribuiton,U[0,1,2,3,4,5].

```r
set.seed(27616) #for fixing the data matrix
R=matrix(c(sample(1:5,1000*20,replace=T)),nrow=100,byrow=T) #rating matrix
head(R) #show the head of our data matrix R
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    5    2    4    1    3    3    2    5    3     1     1     2     4     3
## [2,]    4    1    5    3    4    1    3    5    4     5     4     2     2     2
## [3,]    4    2    2    4    4    4    1    4    3     5     3     4     2     5
## [4,]    3    2    3    5    2    2    1    5    3     5     2     2     5     2
## [5,]    3    3    4    5    4    1    3    5    4     5     4     2     2     1
## [6,]    3    2    3    3    4    3    5    1    1     5     1     3     2     5
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]     5     2     3     5     5     4     4     3     5     2     2     3
## [2,]     4     3     2     4     2     1     5     5     3     1     4     2
## [3,]     5     2     5     5     3     5     4     1     1     3     3     3
## [4,]     4     1     1     1     3     2     5     3     4     5     4     5
## [5,]     2     4     4     5     4     5     4     5     4     3     1     2
## [6,]     5     1     4     4     5     3     5     3     3     4     2     3
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]     3     2     4     5     4     2     1     5     4     1     3     4
## [2,]     4     4     5     1     4     3     5     1     2     1     1     2
## [3,]     1     5     5     5     2     3     4     1     2     3     5     4
## [4,]     4     4     5     3     5     3     1     2     3     2     1     4
## [5,]     5     2     5     3     1     1     3     2     5     3     1     4
## [6,]     4     4     1     5     5     4     5     3     1     1     2     3
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]     1     3     1     3     4     1     4     2     4     3     2     4
## [2,]     1     3     1     1     4     1     2     1     1     4     5     2
## [3,]     3     2     2     2     1     5     5     5     3     3     4     1
## [4,]     2     3     1     1     1     2     4     2     2     2     2     3
## [5,]     5     2     5     3     1     2     2     5     4     4     5     5
## [6,]     1     4     2     5     5     5     4     3     2     5     1     2
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]     4     2     4     4     4     3     2     3     2     4     2     3
## [2,]     3     3     2     4     2     2     3     4     4     1     5     3
## [3,]     4     3     5     3     2     5     4     2     4     4     3     1
## [4,]     4     4     1     1     5     4     5     2     3     1     3     2
## [5,]     2     5     3     2     3     4     3     1     2     1     2     5
## [6,]     2     2     4     1     3     5     2     3     4     5     3     3
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]     1     5     2     2     1     4     5     2     4     1     1     3
## [2,]     2     4     1     5     2     3     1     1     4     4     3     1
## [3,]     3     1     1     1     3     3     3     2     2     1     2     5
## [4,]     5     1     4     5     2     1     1     5     2     3     1     4
## [5,]     1     2     4     5     2     5     4     4     2     4     3     4
## [6,]     1     4     4     1     4     4     3     3     1     3     2     1
##      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
## [1,]     1     5     1     5     4     3     5     1     5     2     2     2
## [2,]     2     2     3     5     3     2     3     4     1     5     5     5
## [3,]     1     5     3     5     4     4     2     3     5     3     3     5
## [4,]     3     1     5     2     2     2     4     3     1     3     1     3
## [5,]     5     3     5     3     5     2     1     4     4     5     3     1
## [6,]     5     2     4     5     4     4     4     3     5     5     5     4
##      [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
## [1,]     4     3     5     4     4     5     2     3     4     5     2     1
## [2,]     2     2     4     5     4     4     4     4     2     2     3     5
## [3,]     1     1     3     5     4     1     1     1     2     5     2     4
## [4,]     5     3     2     1     5     3     1     2     1     2     2     4
## [5,]     2     5     5     2     3     3     5     1     1     1     5     4
## [6,]     4     3     1     5     3     4     4     2     3     5     1     2
##      [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## [1,]     4      1      1      5      2      2      3      2      4      2
## [2,]     1      4      3      2      5      4      3      2      5      3
## [3,]     3      3      3      3      2      1      1      1      4      5
## [4,]     4      3      5      1      1      4      4      1      3      4
## [5,]     4      1      5      4      2      4      3      2      1      3
## [6,]     2      5      4      4      1      3      2      2      4      4
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## [1,]      5      1      2      4      1      4      4      2      1      1
## [2,]      1      2      5      5      2      1      4      1      1      1
## [3,]      1      3      5      1      4      3      5      5      1      3
## [4,]      5      3      2      4      1      2      3      2      1      5
## [5,]      1      5      2      3      4      4      3      1      3      4
## [6,]      2      4      1      5      4      5      1      4      4      2
##      [,119] [,120] [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128]
## [1,]      3      2      4      3      3      3      1      2      2      5
## [2,]      4      5      5      2      5      2      5      5      4      1
## [3,]      3      5      5      1      4      3      4      3      5      4
## [4,]      2      2      3      2      5      1      4      2      2      5
## [5,]      2      2      3      2      5      3      5      2      5      2
## [6,]      2      1      5      2      2      3      5      3      2      3
##      [,129] [,130] [,131] [,132] [,133] [,134] [,135] [,136] [,137] [,138]
## [1,]      1      5      2      2      2      2      3      5      1      4
## [2,]      2      4      3      1      2      4      3      2      4      4
## [3,]      1      1      1      4      5      2      4      3      3      1
## [4,]      4      5      1      2      1      5      1      1      5      3
## [5,]      4      1      4      1      3      2      3      1      1      5
## [6,]      2      3      5      4      1      2      1      5      2      3
##      [,139] [,140] [,141] [,142] [,143] [,144] [,145] [,146] [,147] [,148]
## [1,]      1      3      1      5      1      3      3      1      5      3
## [2,]      5      5      3      5      3      2      4      5      2      5
## [3,]      4      2      1      3      5      1      2      3      3      3
## [4,]      4      3      5      3      3      2      5      3      1      3
## [5,]      2      1      5      3      1      2      3      1      4      4
## [6,]      2      3      2      2      3      1      3      5      5      5
##      [,149] [,150] [,151] [,152] [,153] [,154] [,155] [,156] [,157] [,158]
## [1,]      5      3      5      2      2      5      4      4      5      2
## [2,]      2      4      2      2      5      2      2      2      3      5
## [3,]      5      3      3      4      3      5      1      2      1      4
## [4,]      2      2      2      5      4      2      4      1      4      2
## [5,]      2      1      4      5      3      1      3      4      5      1
## [6,]      4      5      2      3      3      4      3      1      1      5
##      [,159] [,160] [,161] [,162] [,163] [,164] [,165] [,166] [,167] [,168]
## [1,]      5      3      1      4      1      5      4      2      4      5
## [2,]      4      3      2      1      1      1      2      3      5      1
## [3,]      1      1      1      5      4      2      2      3      4      2
## [4,]      3      1      5      3      5      4      1      4      1      5
## [5,]      2      4      4      5      4      1      4      2      4      5
## [6,]      3      3      4      4      5      4      3      4      5      2
##      [,169] [,170] [,171] [,172] [,173] [,174] [,175] [,176] [,177] [,178]
## [1,]      5      1      5      4      3      5      3      4      3      3
## [2,]      4      4      1      2      5      3      1      4      3      1
## [3,]      2      1      1      5      5      4      3      5      5      1
## [4,]      1      1      1      3      3      2      1      1      4      5
## [5,]      1      4      4      1      3      5      1      2      1      2
## [6,]      1      4      5      1      3      2      4      3      5      4
##      [,179] [,180] [,181] [,182] [,183] [,184] [,185] [,186] [,187] [,188]
## [1,]      4      1      4      1      4      5      1      3      1      1
## [2,]      2      5      2      4      5      1      1      3      4      3
## [3,]      5      1      3      5      2      2      2      1      3      2
## [4,]      4      2      5      4      1      4      1      4      5      1
## [5,]      5      4      3      4      5      1      4      2      5      5
## [6,]      1      4      3      3      2      1      4      1      4      3
##      [,189] [,190] [,191] [,192] [,193] [,194] [,195] [,196] [,197] [,198]
## [1,]      1      4      3      3      5      5      1      5      4      2
## [2,]      5      2      3      1      2      3      1      1      5      3
## [3,]      2      2      4      4      5      1      3      1      4      2
## [4,]      2      5      3      5      3      5      2      2      3      1
## [5,]      2      2      2      5      2      5      4      3      5      4
## [6,]      2      1      4      5      4      3      5      4      4      2
##      [,199] [,200]
## [1,]      3      2
## [2,]      5      2
## [3,]      1      4
## [4,]      5      2
## [5,]      4      5
## [6,]      4      2
```

```
SVD_R=svd(R,nv=ncol(R)) #singular value decomposition of rating matrix
```

4(b)

We set the latent features be the 10 in this case(assuming there is total 10 types of movie)

```
p=10
```

4(c)

```
P=(SVD_R$u)%*%rbind(diag(p),matrix(c(rep(0,(100-p)*p)),nrow=90))
Q=t((SVD_R$v))%*%rbind(diag(p),matrix(c(rep(0,(200-p)*p)),nrow=190))
```

4(d)

P and Q is the singular value decomposition of the data matrix R,$R \approx PQ^T$.

We can approximate the rating matrix R by just collecting R and Q, they are lower dimesion compared to the rating matrix R. We can lower the cost of the data saving by this decomposition process.

4(e)

```
Rhat=P%*%t(Q)
head(Rhat)
```

```
##              [,1]         [,2]         [,3]         [,4]        [,5]
## [1,] -0.002913404  0.001782938  0.013993177 -0.047024378 -0.04511256
## [2,]  0.032974030 -0.001724192 -0.025750835 -0.056107479 -0.01301996
## [3,]  0.021544820  0.002279342 -0.002084242 -0.009053054 -0.03377583
## [4,]  0.049534628  0.004053506 -0.004395658 -0.029013550 -0.02317689
## [5,]  0.050252270  0.012919428 -0.009801859  0.003552447  0.01760553
## [6,]  0.006429232  0.008342226  0.011397141 -0.004051384 -0.02601130
##              [,6]         [,7]         [,8]         [,9]        [,10]
## [1,] -0.005536853 -0.007451181  0.049303424 -0.029823134  0.035000119
## [2,] -0.001619913  0.011577751  0.005390052  0.027807685 -0.007491767
## [3,]  0.028051316  0.023686324  0.043473435 -0.002599226 -0.002162180
## [4,] -0.007623308 -0.015374042  0.029410923  0.045742193 -0.001685096
## [5,]  0.033003046  0.001506331 -0.017671804  0.064860905 -0.009807093
## [6,]  0.009848331  0.013141066  0.028013403 -0.019019578  0.008965118
##              [,11]        [,12]        [,13]        [,14]        [,15]
## [1,]  0.0067208057  0.017973861 -0.019053989  0.018908843 -0.007817578
## [2,]  0.0059518481  0.004049736 -0.013731124  0.027811936  0.013023361
## [3,]  0.0109615913 -0.022412418 -0.032926151  0.013836070  0.037847880
## [4,] -0.0007624193  0.014889732  0.017354623 -0.022655385 -0.013194621
## [5,]  0.0013193912 -0.010964069  0.005770231 -0.018231223  0.025083814
## [6,]  0.0239084761 -0.020882793 -0.029997703  0.005140451  0.032463019
##            [,16]         [,17]         [,18]         [,19]        [,20]
## [1,]  0.03496676  0.0137969709 -0.0113847805  0.0189603365  0.001926058
## [2,] -0.03520478  0.0253712693 -0.0308931980  0.0004651588  0.050150755
## [3,]  0.02153154  0.0084577116 -0.0132333067 -0.0070259808  0.017128296
## [4,] -0.01691267  0.0242761999 -0.0004718558  0.0315998839  0.011896317
## [5,] -0.01514801 -0.0015796468 -0.0017495984  0.0152870875  0.008663694
## [6,]  0.02792873 -0.0001301773 -0.0159501248  0.0003160542 -0.007907846
##             [,21]        [,22]        [,23]        [,24]        [,25]
## [1,]  0.008521883 0.0207325902  0.008201237  0.032402656 -0.033192338
## [2,] -0.006524356 0.0404939678 -0.023532234 -0.007374531 -0.006076321
## [3,] -0.012315710 0.0361753007 -0.009037620  0.002190940 -0.041111379
## [4,] -0.031521083 0.0149581316  0.017725059  0.014312152  0.032584409
## [5,] -0.042295056 0.0006710312  0.001635324 -0.012725785  0.030493342
## [6,] -0.003159327 0.0287681859 -0.013463310  0.008641089 -0.038621915
##            [,26]         [,27]        [,28]        [,29]         [,30]
## [1,] -0.01384551 -0.0008879703  0.005422448  0.003734197 -0.0002660383
## [2,]  0.02037301 -0.0209361915 -0.009278376 -0.009287993  0.0049948411
## [3,] -0.01084775 -0.0103966785 -0.010586290 -0.001052928 -0.0221514541
## [4,]  0.01039166 -0.0161592816  0.011860139 -0.012050054 -0.0013974659
## [5,]  0.02373080  0.0023222017 -0.021752776 -0.009970222  0.0058913011
## [6,] -0.02284195 -0.0024402195 -0.014123573  0.009914169 -0.0209115278
##             [,31]       [,32]        [,33]       [,34]           [,35]
## [1,] -0.014209576  0.01213634 -0.004487355 -0.02509722  0.0009722281
## [2,] -0.017534315 -0.02105498  0.030481652 -0.01317920  0.0496554684
## [3,] -0.002649241  0.02749842 -0.012080005  0.02291696  0.0206318421
## [4,] -0.033358608 -0.02511916  0.044597800 -0.04295745  0.0382960458
## [5,] -0.038408023 -0.02362826  0.025901473 -0.01433719  0.0043132062
## [6,]  0.005734649  0.02796805 -0.025841092  0.02011353 -0.0108958467
##             [,36]        [,37]        [,38]        [,39]        [,40]
## [1,]  0.033214531  0.002456002 -0.007037286 -0.050197980 -0.045626159
## [2,] -0.010198147 -0.027789929 -0.012650824 -0.025705922 -0.027965792
## [3,]  0.002371606  0.025506884  0.003110853 -0.033035207  0.007001086
## [4,] -0.024067073 -0.049230097  0.004506911 -0.050683649 -0.069404851
## [5,] -0.050980328 -0.037431971  0.026382140 -0.006150334 -0.015353950
## [6,]  0.018559813  0.047283151  0.002910148 -0.026358773  0.016807932
##             [,41]        [,42]         [,43]        [,44]        [,45]
## [1,] -0.026352113  0.032165154 -0.0017737866  0.002599809  0.047933387
## [2,] -0.035576813  0.006252382 -0.0198403698 -0.024317046 -0.001356146
## [3,] -0.011676286  0.036453029 -0.0018040829 -0.014246058 -0.037752985
## [4,]  0.004599920  0.001736822 -0.0418112120 -0.016503846  0.040491635
## [5,]  0.028360402 -0.024983915 -0.0528041987 -0.016951635 -0.004881270
## [6,]  0.008985208  0.040941877  0.0005271815 -0.011335959 -0.017487625
##             [,46]        [,47]        [,48]        [,49]         [,50]
## [1,] -0.015069653  0.010114157  0.050998767  0.016538496  0.0007723074
## [2,] -0.019038919  0.033116193  0.026841075 -0.001523796  0.0072221663
## [3,] -0.010931251  0.003628060  0.031583399 -0.004283128 -0.0012320122
## [4,] -0.032409400 -0.002012410  0.023236906  0.019509191 -0.0222421114
## [5,] -0.027280049 -0.032961946 -0.005611578  0.015999544 -0.0233581455
## [6,] -0.009694956 -0.006644052  0.027526547 -0.000376098 -0.0063555829
##             [,51]        [,52]        [,53]        [,54]        [,55]
## [1,] -0.007422534 -0.022064633  0.032699173  0.018099501  0.037027686
## [2,] -0.031171372 -0.009150349 -0.020461240 -0.038903162 -0.010823493
## [3,] -0.002065672 -0.019242501 -0.006771841 -0.005286802  0.004977566
## [4,] -0.050499645  0.006847942  0.000808800 -0.026976252 -0.013420480
## [5,] -0.007574305 -0.005092150 -0.023806027 -0.038087672 -0.015509310
## [6,]  0.003275711 -0.007076607  0.010940864  0.007094904  0.012062649
##            [,56]        [,57]       [,58]        [,59]        [,60]
## [1,] 0.014014601 -0.002183315 -0.02209855 -0.029115385  0.052386472
## [2,] 0.046470486  0.026991733  0.01323091  0.018469809  0.006115219
## [3,] 0.012092571  0.054343630 -0.02032463  0.011443038  0.007918143
## [4,] 0.015558747  0.015843347  0.02027981 -0.003652493 -0.012490985
## [5,] 0.010859866  0.027491158  0.01720803  0.018691095 -0.027983993
## [6,] 0.005040714  0.038683561 -0.02216008  0.005469449  0.007707168
##             [,61]         [,62]        [,63]         [,64]        [,65]
## [1,]  0.044516289 -0.0066765345 -0.030461086 -0.0304335209  0.011182212
## [2,]  0.022191552 -0.0012346072  0.009194659  0.0363341133 -0.004018400
## [3,] -0.004780635 -0.0161385964 -0.013637794  0.0119416401 -0.023626941
## [4,]  0.031527014  0.0008367375 -0.018130772 -0.0047613810  0.026021416
## [5,]  0.001728856  0.0078653231 -0.013477007  0.0008621911  0.007093581
## [6,] -0.014213183 -0.0191809752 -0.011279821 -0.0008068481 -0.007066715
##             [,66]        [,67]        [,68]        [,69]       [,70]
## [1,] -0.025957317 -0.050659406  0.026631423  0.037836547 -0.02808912
## [2,] -0.023026986  0.020077156 -0.004244519 -0.054326520  0.05519165
## [3,] -0.029320464 -0.011035126  0.013260318 -0.019377730 -0.01569295
## [4,] -0.037086269 -0.004468928 -0.010300533  0.016697734  0.03120443
## [5,]  0.002318227  0.020049203 -0.016664340 -0.025018334  0.04642166
## [6,] -0.005907310 -0.015527884  0.007602138  0.001537568 -0.02507262
##             [,71]        [,72]       [,73]         [,74]         [,75]
## [1,]  0.008435140  0.004626245 -0.01363592  0.0345676847 -0.0318362529
## [2,] -0.025389841  0.009992048  0.03666617 -0.0106827773  0.0008337249
## [3,]  0.032517052 -0.024244554 -0.02179067  0.0009864915 -0.0391687002
## [4,] -0.007743237  0.039428089  0.03927339  0.0063447814  0.0336468034
## [5,]  0.019669101  0.009217147  0.01636778 -0.0194972580  0.0284162871
## [6,]  0.039437603 -0.024457655 -0.02729462  0.0084365745 -0.0363394265
##             [,76]        [,77]        [,78]         [,79]        [,80]
## [1,]  0.005683493 -0.013586829  0.010425981  0.0057592750  0.015936125
## [2,] -0.028441340  0.015562228 -0.001348852  0.0000344401 -0.021625313
## [3,] -0.020936325  0.003594813 -0.019044453 -0.0304689502 -0.009485994
## [4,]  0.028212698 -0.009142217  0.005217788  0.0030187709 -0.007783937
## [5,]  0.015985593  0.028388651 -0.006913756 -0.0090314387 -0.005012225
## [6,] -0.008996554  0.002639266  0.005716025 -0.0291265261  0.003281311
##             [,81]        [,82]        [,83]        [,84]        [,85]
## [1,] -0.030796071 -0.010754768 -0.006355373  0.025143669 -0.041755266
## [2,] -0.001023358  0.037202536 -0.005867742 -0.021884306  0.049736667
## [3,] -0.044199320  0.007746568  0.003713316 -0.006974580  0.017616135
## [4,] -0.005094141 -0.013510989  0.010248168 -0.015258050  0.001496308
## [5,]  0.016938582  0.001982437  0.022962272 -0.008030664  0.024098729
## [6,] -0.035075678 -0.011555934  0.012740877  0.006246382 -0.006663887
##             [,86]        [,87]         [,88]         [,89]        [,90]
## [1,] -0.015240453  0.018722062 -0.0001958148  0.0005825187  0.012525909
## [2,] -0.017725178 -0.027238272 -0.0194024634 -0.0266278079 -0.010142313
## [3,] -0.009160115  0.001111542 -0.0168251182  0.0352535344  0.034945048
## [4,] -0.015928184 -0.022940037  0.0070688055 -0.0386989360 -0.040421506
## [5,]  0.003472883 -0.042487735  0.0227518032 -0.0172751597 -0.008475291
## [6,] -0.002434050 -0.005870427 -0.0058468060  0.0431486715  0.042650822
##              [,91]        [,92]       [,93]         [,94]         [,95]
## [1,]  0.0174824105  0.005746059 -0.02291994 -0.0404313473  0.0293474748
## [2,]  0.0306568925  0.033996517  0.03503121  0.0303076655 -0.0005304532
## [3,]  0.0031854550  0.034477985 -0.04404275  0.0086826465 -0.0132276251
## [4,]  0.0219668480  0.001377230  0.04406369  0.0401707459  0.0156899808
## [5,]  0.0003023204 -0.002789228  0.04632941  0.0504639766 -0.0396075764
## [6,] -0.0080471584  0.013828602 -0.03947254  0.0003524568 -0.0199217084
##            [,96]        [,97]        [,98]         [,99]       [,100]
## [1,] -0.03833110 -0.001459800 -0.005323051  0.0103383321  0.033934424
## [2,]  0.04075999 -0.002631474  0.023671008  0.0111299370  0.022503955
## [3,] -0.01277968 -0.030164853  0.010357325 -0.0195250201  0.018577208
## [4,]  0.01663328  0.010154281 -0.016870069  0.0189646452  0.007334336
## [5,]  0.04582305  0.002848738  0.019387413 -0.0008389214 -0.005325520
## [6,] -0.03285981 -0.032179708  0.007514623 -0.0294406321  0.010847615
##             [,101]       [,102]        [,103]       [,104]        [,105]
## [1,] -0.0315141115  0.012564354  0.0291786478 -0.003704383  0.0003630362
## [2,]  0.0008782615  0.005404468  0.0190443151  0.013501273 -0.0007696080
## [3,] -0.0169254492 -0.019337578  0.0002552162 -0.011369495 -0.0234835772
## [4,]  0.0253692126  0.021396396  0.0451662501 -0.003906636  0.0023735361
## [5,]  0.0262880189 -0.002223133  0.0084701620 -0.005183799  0.0011740523
## [6,] -0.0295404456 -0.012127254 -0.0164301473 -0.007355195 -0.0230184285
##            [,106]       [,107]       [,108]      [,109]      [,110]      [,111]
## [1,]  0.030985522 -0.006295784 -0.006203605 -0.01220836  0.02030656  0.01757136
## [2,] -0.003707792  0.016734779  0.018556851 -0.01108075 -0.02292606  0.01035627
## [3,]  0.007878886 -0.014879889  0.007218087  0.01188499 -0.01091089  0.02430791
## [4,]  0.024705176  0.012258171 -0.008402651 -0.01999680  0.01926303 -0.02655079
## [5,] -0.008178349 -0.014238237 -0.019647961  0.03474561 -0.01374140 -0.02583626
## [6,]  0.002934872 -0.008949025 -0.017305752  0.02199041 -0.01551227  0.02021239
##            [,112]       [,113]        [,114]      [,115]       [,116]
## [1,] -0.012322730  0.043374343 -0.0228594420 0.023207704  0.020332232
## [2,] -0.005295070  0.011821561 -0.0252323085 0.005323749  0.030469580
## [3,] -0.002939575 -0.003761718 -0.0007521651 0.024489109  0.032825711
## [4,]  0.005593592  0.025989644 -0.0345213795 0.011595607 -0.008495094
## [5,]  0.004219917 -0.016270260 -0.0047843003 0.006939067 -0.014823481
## [6,]  0.004525056 -0.013743199 -0.0064785274 0.023673226  0.027817369
##            [,117]      [,118]      [,119]      [,120]       [,121]      [,122]
## [1,]  0.026560682 0.026397856 0.033986587 -0.03364587 -0.015320254 -0.02720814
## [2,]  0.022018279 0.026728663 0.009375485  0.02559283 -0.008629499  0.03508345
## [3,]  0.011604675 0.043008988 0.012102881 -0.01357207  0.006697665 -0.01904766
## [4,]  0.034318606 0.009489369 0.011736816 -0.01141454 -0.025157762  0.01866651
## [5,] -0.009959315 0.009777766 0.017879885  0.03543534  0.005338125  0.02143806
## [6,]  0.015844008 0.030909860 0.014077190 -0.01910264  0.010108620 -0.02436914
##            [,123]       [,124]       [,125]        [,126]       [,127]
## [1,] -0.035849354  0.035957596  0.009857339  1.074784e-02  0.003336203
## [2,]  0.054457238 -0.003735184  0.007547913  7.005239e-05 -0.046712645
## [3,]  0.007625676  0.028611910  0.001536590 -1.637836e-02  0.022003368
## [4,]  0.039160516  0.001832696  0.024832877  2.611187e-02 -0.035037379
## [5,]  0.071217764 -0.036411557 -0.015631099  3.720313e-03 -0.032145817
## [6,] -0.019441535  0.032076343 -0.019278059 -1.632502e-02  0.035442127
##            [,128]       [,129]      [,130]       [,131]       [,132]
## [1,]  0.003204871  0.057212344  0.01531168 -0.011302689 -0.009165661
## [2,] -0.005611467 -0.008216727 -0.01398675  0.002240909  0.016485989
## [3,]  0.008614245  0.006843224 -0.01626959  0.001896518 -0.006383422
## [4,] -0.044464465  0.026934820  0.05010969  0.019097505 -0.004087789
## [5,] -0.020120221 -0.018554732  0.02618067  0.015836881 -0.015152327
## [6,]  0.017933468  0.003778048 -0.01295930  0.002667007 -0.012873540
##            [,133]      [,134]        [,135]      [,136]       [,137]
## [1,]  0.023129255  0.01460946 -0.0143881105  0.04185143  0.037631167
## [2,] -0.020901146 -0.01563627  0.0007797345  0.02965281  0.024823388
## [3,] -0.002386690  0.02720617 -0.0171368637  0.02754929  0.003514232
## [4,]  0.020375636 -0.01702080  0.0247615783  0.01377424  0.002028134
## [5,]  0.001889938 -0.01834746  0.0011440063 -0.02447417 -0.029949429
## [6,]  0.007108608  0.03117554 -0.0287651406  0.01716166 -0.018051154
##            [,138]       [,139]       [,140]      [,141]       [,142]
## [1,] 0.0146068332 -0.005748045 -0.014464042 0.060396583 -0.001815545
## [2,] 0.0481740267  0.021500673  0.012581937 0.001978395 -0.003163757
## [3,] 0.0062741939 -0.003177298 -0.009607041 0.037526466 -0.015565157
## [4,] 0.0249711176  0.030116218  0.005710648 0.046793915 -0.010117948
## [5,] 0.0316782464  0.020516170  0.012459743 0.001717604  0.002494834
## [6,] 0.0004214676 -0.016689773 -0.002542836 0.033781451 -0.008778850
##            [,143]       [,144]      [,145]       [,146]      [,147]
## [1,] -0.027852243 -0.015409782 -0.00579431 -0.014891471  0.01976719
## [2,]  0.009293243 -0.020526661  0.01021419  0.025748244  0.02848239
## [3,] -0.037144804 -0.041480493 -0.01944096 -0.031332135  0.05513751
## [4,]  0.003720365  0.003169795  0.00744533  0.026262271  0.01508329
## [5,]  0.014729964  0.028433808  0.04320759 -0.005993978 -0.01155155
## [6,] -0.032369765 -0.024735949 -0.02005900 -0.029485322  0.04187793
##            [,148]       [,149]        [,150]      [,151]        [,152]
## [1,] -0.022265674  0.005904886 -0.0167509744  0.03253511 -0.0050400962
## [2,] -0.006163116 -0.014754081 -0.0009425319 -0.03316108 -0.0252971353
## [3,]  0.008431761  0.003216717 -0.0140574966  0.02483805  0.0253779471
## [4,]  0.020464765 -0.014586264  0.0009266556 -0.03343806 -0.0225283987
## [5,]  0.027590741 -0.004604571  0.0065784466 -0.03548521  0.0005070954
## [6,]  0.003173845  0.002934616 -0.0104147459  0.03761696  0.0254030340
##           [,153]       [,154]       [,155]       [,156]       [,157]
## [1,] 0.036847245 -0.032123516  0.007088621 -0.022191474 -0.033450008
## [2,] 0.019981942  0.022577214 -0.023548890 -0.047428142  0.002433660
## [3,] 0.007624815 -0.003079890 -0.017621553  0.003222513 -0.001589350
## [4,] 0.032246176 -0.036757523  0.007515215 -0.012258118  0.018059808
## [5,] 0.002076176  0.022792849  0.006649627 -0.006862617  0.052539450
## [6,] 0.005625302  0.001480472 -0.006338150  0.013587261 -0.005642812
##             [,158]      [,159]       [,160]        [,161]        [,162]
## [1,] -0.0106300864 -0.01031738  0.012277777  0.0022798586 -0.0007275331
## [2,]  0.0198917634  0.03042853  0.065211412 -0.0008324443 -0.0214876446
## [3,] -0.0001002928 -0.02194575  0.015535963  0.0224273706 -0.0004803412
## [4,] -0.0056541240  0.02925850  0.067123877  0.0114009937  0.0019254690
## [5,]  0.0002968654  0.04800602  0.035613049 -0.0225626942 -0.0106120466
## [6,] -0.0138342365 -0.02602943 -0.007199716  0.0092631553  0.0033223603
##            [,163]       [,164]      [,165]       [,166]       [,167]
## [1,]  0.026028221 -0.034656389 0.018747268  0.041097701  0.034077036
## [2,]  0.032037799  0.007688101 0.009674490  0.003988646 -0.017313586
## [3,]  0.033970010 -0.022127890 0.002413506  0.019885434 -0.010879138
## [4,]  0.024815315 -0.007120852 0.022863630  0.005021106  0.027193926
## [5,] -0.005071589  0.019642638 0.009863508 -0.033950612 -0.036195318
## [6,]  0.007655609 -0.022722990 0.001847658  0.013315437 -0.003498596
##            [,168]        [,169]       [,170]      [,171]       [,172]
## [1,] -0.030089985 -0.0173789082  0.005931618 -0.03831257  0.020620150
## [2,] -0.010691255  0.0012974391  0.017439735  0.06771597  0.019484580
## [3,]  0.004129041  0.0053209431 -0.003273703 -0.02413392  0.001836653
## [4,] -0.002800230  0.0003652214  0.013107841  0.03745015  0.002455926
## [5,]  0.049041712  0.0195066181  0.003596259  0.07522574 -0.032167979
## [6,]  0.015700295  0.0101761563 -0.008376646 -0.02299989 -0.009055247
##            [,173]      [,174]        [,175]       [,176]        [,177]
## [1,] -0.003429055 -0.01413668  0.0307789901  0.028762819  0.0308438393
## [2,] -0.020552750 -0.03396883 -0.0067087934  0.016410676  0.0052152647
## [3,] -0.009498295  0.01927025 -0.0066746863  0.025435265  0.0027008781
## [4,]  0.009071047 -0.04551431  0.0325934149 -0.006627792  0.0244376111
## [5,]  0.005715620 -0.03303905 -0.0251494201 -0.022979461  0.0010053775
## [6,] -0.003377213  0.02203755  0.0006395844  0.015788312 -0.0006658729
##             [,178]        [,179]       [,180]      [,181]      [,182]
## [1,]  0.0596673299  0.0053227516  0.013186723  0.02637098 0.035355442
## [2,] -0.0072220410 -0.0021521300  0.033013417 -0.01157749 0.036965094
## [3,] -0.0003941522  0.0234813032 -0.012525779  0.00559914 0.019067324
## [4,]  0.0299981495  0.0005915694  0.042290006  0.01226825 0.045556858
## [5,] -0.0090710573 -0.0020647360  0.009785228  0.01947850 0.005258133
## [6,]  0.0226992559  0.0163210807 -0.023803900  0.01181150 0.004230934
##             [,183]      [,184]       [,185]       [,186]      [,187]
## [1,] -0.0070541289 0.004623155  0.036061965  0.027376029 -0.01014202
## [2,] -0.0383114869 0.043409050  0.044245528 -0.035010699  0.01585632
## [3,] -0.0001673481 0.040157463  0.009112050  0.002247962 -0.03680447
## [4,] -0.0165622930 0.020970616  0.029402688  0.011308218  0.01951056
## [5,] -0.0004712245 0.048348966  0.015988079  0.005822252  0.00217398
## [6,]  0.0094942855 0.038548702 -0.003509159  0.020574206 -0.03195782
##          [,188]       [,189]       [,190]       [,191]       [,192]
## [1,]  0.01377571 -0.032913147 -0.016013055  0.013394979  0.030994615
## [2,] -0.01311685 -0.005382463 -0.009426882  0.009334924 -0.001308909
## [3,] -0.02811369 -0.007894647  0.011689991 -0.008831266  0.032094193
## [4,]  0.01456239 -0.038551066  0.014349664  0.008515063 -0.010209637
## [5,] -0.02114006 -0.001337123  0.036857687 -0.002116335 -0.009052358
## [6,] -0.02571061  0.003568281  0.004231103 -0.011478794  0.028270696
##             [,193]       [,194]     [,195]        [,196]       [,197]
## [1,]  0.052380660  0.036019191 0.05019029  0.0007371409 -0.036001775
## [2,] -0.008185913 -0.006862058 0.01958439  0.0499713061 -0.020435762
## [3,]  0.017781148  0.010608476 0.04072782 -0.0079176744 -0.042382513
## [4,]  0.002507273  0.033837527 0.05225206  0.0603298753  0.001528241
## [5,] -0.031634157  0.007849757 0.01011769  0.0593235099  0.011050951
## [6,]  0.025823594  0.020276367 0.01777889 -0.0130625800 -0.025199526
##              [,198]      [,199]       [,200]
## [1,] -0.0001071708 -0.02408570  0.065889339
## [2,]  0.0236206325  0.02024971  0.008340327
## [3,]  0.0142944234 -0.03678787  0.042554133
## [4,] -0.0084651292  0.03973141  0.014237306
## [5,] -0.0030130643  0.04289549 -0.034042512
## [6,]  0.0037833511 -0.03890254  0.035494217
```

4(f)

```
mse.q=sum(as.vector(Rhat-R)^2)
print(mse.q)
```

```
## [1] 220465.9
```

The result is clearly not reasonable as the mse is to big. Also it is because the rating matrix is just simulated by uniform distribuiton. Also, the reason behind this is because the feature is selected wrongly.

5(a)

Notice that Ab should be a 4x1 vector

$$||Ab-c||^2=(Ab-c)(Ab-c)^T$$

to minimize the square norm, we rewrite the form to have $A^T(A\vec{b}-\vec{c})=0$,or

$$A^TA\vec{b}=A^T\vec{c}$$

where $\vec{b}$ is the vector that we want to minimize the norm. Hence, we have the following result:

$$\vec{b}=(A^TA)^{-1}A^T\vec{c}$$

Let solve $\vec{b}$ in the following code

```
A=matrix(c(1,0,2,1,1,0,0,2,1,2,1,1),byrow=T,nrow=4)
c=c(2,1,1,3)
b=solve(t(A)%*%A)%*%t(A)%*%c ; print(b)
```

```
##            [,1]
## [1,] 1.0357143
## [2,] 0.2142857
## [3,] 0.5357143
```

5(b)

$$||Ab-c||^2+\lambda||b||^2=(Ab-c)(Ab-c)^T+\lambda bb^T$$

After some tedious calculation, we have

$$b=(A^TA+\lambda I)^{-1}(A^T)c$$

Hence if we are given $\lambda=0.2$,

```
lambda=0.2
b=solve((t(A)%*%A+0.2*diag(3)))%*%(t(A))%*%c
b
```

```
##           [,1]
## [1,] 0.9942931
## [2,] 0.2254642
## [3,] 0.5397476
```

6(a)

By Q5, we can directly calculate the beta vector by minimizing the least square$||Ab-c||$

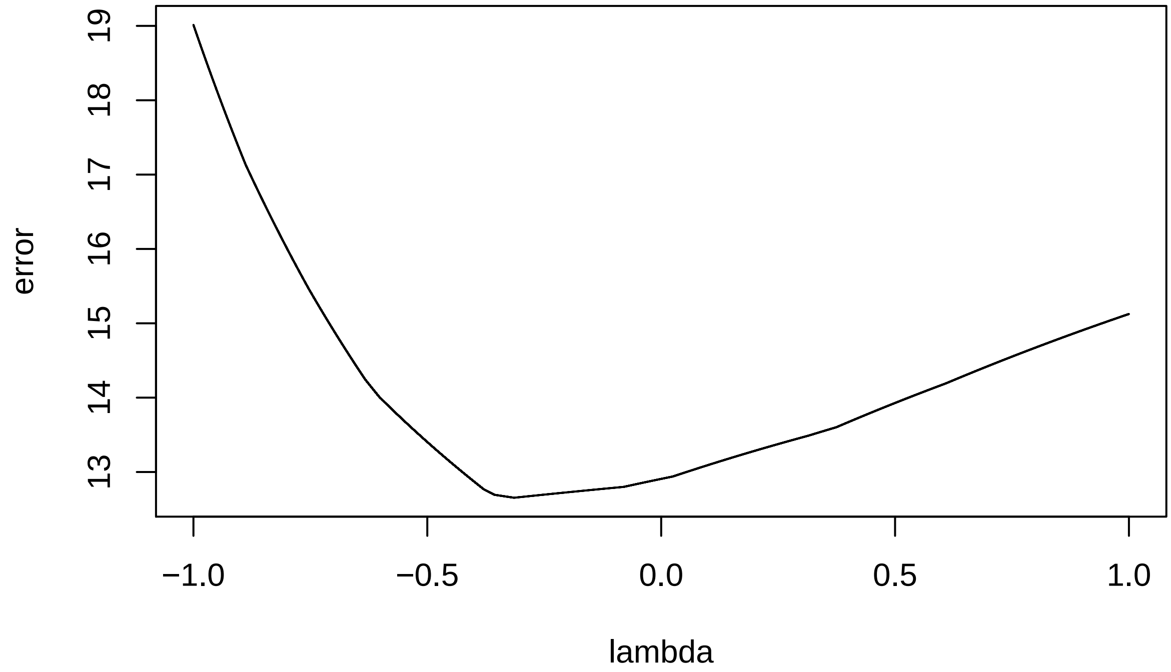
```
#inputing data
A=matrix(
c(1,rep(0,9),1,rep(0,4),
rep(0,2),1,rep(0,7),1,rep(0,4),
rep(0,4),1,rep(0,5),1,rep(0,4),
rep(0,6),1,rep(0,3),1,rep(0,4),
rep(0,7),1,rep(0,2),1,rep(0,4),
1,rep(0,9),rep(0,1),1,rep(0,3),
rep(0,1),1,rep(0,8),rep(0,1),1,rep(0,3),
rep(0,2),1,rep(0,7),rep(0,1),1,rep(0,3),
rep(0,5),1,rep(0,4),rep(0,1),1,rep(0,3),
rep(0,8),1,rep(0,1),rep(0,1),1,rep(0,3),
1,rep(0,9),rep(0,2),1,rep(0,2),
rep(0,1),1,rep(0,8),rep(0,2),1,rep(0,2),
rep(0,3),1,rep(0,6),rep(0,2),1,rep(0,2),
rep(0,6),1,rep(0,3),rep(0,2),1,rep(0,2),
rep(0,7),1,rep(0,2),rep(0,2),1,rep(0,2),
rep(0,8),1,rep(0,1),rep(0,2),1,rep(0,2),
rep(0,9),1,rep(0,2),1,rep(0,2),
rep(0,3),1,rep(0,6),rep(0,3),1,rep(0,1),
rep(0,4),1,rep(0,5),rep(0,3),1,rep(0,1),
rep(0,5),1,rep(0,4),rep(0,3),1,rep(0,1),
rep(0,6),1,rep(0,3),rep(0,3),1,rep(0,1),
rep(0,8),1,rep(0,1),rep(0,3),1,rep(0,1),
rep(0,9),1,rep(0,3),1,rep(0,1),
rep(0,1),1,rep(0,8),rep(0,4),1,
rep(0,2),1,rep(0,7),rep(0,4),1,
rep(0,3),1,rep(0,6),rep(0,4),1,
rep(0,4),1,rep(0,5),rep(0,4),1,
rep(0,5),1,rep(0,4),rep(0,4),1,
rep(0,7),1,rep(0,2),rep(0,4),1,
rep(0,9),1,rep(0,4),1),
byrow=T,ncol=15,nrow=30)
head(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    1    0    0    0    0    0    0    0    0     0     1     0     0     0
## [2,]    0    0    1    0    0    0    0    0    0     0     1     0     0     0
## [3,]    0    0    0    0    1    0    0    0    0     0     1     0     0     0
## [4,]    0    0    0    0    0    0    1    0    0     0     1     0     0     0
## [5,]    0    0    0    0    0    0    0    1    0     0     1     0     0     0
## [6,]    1    0    0    0    0    0    0    0    0     0     0     1     0     0
##      [,15]
## [1,]     0
## [2,]     0
## [3,]     0
## [4,]     0
## [5,]     0
## [6,]     0
```

```
c=c(5,5,4,3,5,
    4,3,2,3,2,
    4,5,3,3,4,5,5,
    1,4,3,2,4,3,
    4,3,2,5,5,5,4
)-3.67
```

We have to use regularization parameter. We may use simulation to help us find the best lambda. In order to do so, we simulate vector c first by diiferent lambda. Then we set the lambda to be the minimum of the error of $|\hat{c} - c|$, where $\hat{c}$ is simulated by different lambda:

```
temp=list()
errfun=c()
temp=list()
lambda=c(seq(-1,1,0.00001))
lambda=lambda[lambda!=0]
for (i in 1:length(lambda)){
  temp[[i]]=solve((t(A)%*%A+lambda[i]*diag(15)))%*%(t(A))%*%c
}
for (j in 1:length(lambda)){
  errfun[j]=sum(abs((A%*%temp[[j]])-c))
}
plot(errfun~lambda,ylab="error",xlab="lambda",type="l")
```



```r
b.lambda=lambda[which(errfun<=min(errfun))] #best lambda
b.lambda
```

```
## [1] -0.31437
```

Hence, we have the optimal user bias b_u^* and the optimal movie bias b_i^* to be:

```r
b=solve((t(A)%*%A+b.lambda*diag(15)))%*%(t(A))%*%c
b
```

```
##             [,1]
##  [1,]  0.7139367
##  [2,]  0.5048608
##  [3,] -0.2864571
##  [4,] -2.0099196
##  [5,]  0.5499316
##  [6,]  0.6011114
##  [7,] -1.4284916
##  [8,]  0.4536810
##  [9,]  0.4844488
## [10,]  0.2241931
## [11,]  0.7784224
## [12,] -1.3590276
## [13,]  0.6532355
## [14,] -0.6052581
## [15,]  0.3399229
```

```
b[c(1:10),]#optimal user bias
```

```
##  [1]  0.7139367  0.5048608 -0.2864571 -2.0099196  0.5499316  0.6011114
##  [7] -1.4284916  0.4536810  0.4844488  0.2241931
```

```
b[-c(1:10),]#optimal movie bias
```

```
## [1]  0.7784224 -1.3590276  0.6532355 -0.6052581  0.3399229
```

6(b)

We directly calculate the RMSE by the formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}$$

```
Rhat=A%*%b+3.67
R=c(5,5,4,3,5,4,3,2,3,2,4,5,3,3,4,5,5,1,4,3,2,4,3,4,3,2,5,5,5,4)
Rhat[which(Rhat>5)]=5 #rounding
Rhat[which(Rhat<1)]=1 #rounding
RMSE=sqrt(sum((Rhat-R)^2)/length(R))
RMSE
```

```
## [1] 0.5222431
```

7(a)

```
d=read.csv("D:\\CUHKZOOMNOTESANDSOURCE\\RMSC4002\\DATA\\credit.csv")
set.seed(27616) #my student id is 1155127616
n=nrow(d)
id=sample(1:n,size=580)#580 random index
d1=d[id,]#training dataset d1
dim(d1) #check dimension
```

```
## [1] 580   7
```

```
d2=d[-id,]#testing dataset d2
dim(d2) #check dimension
```

```
## [1] 110   7
```

7(b)

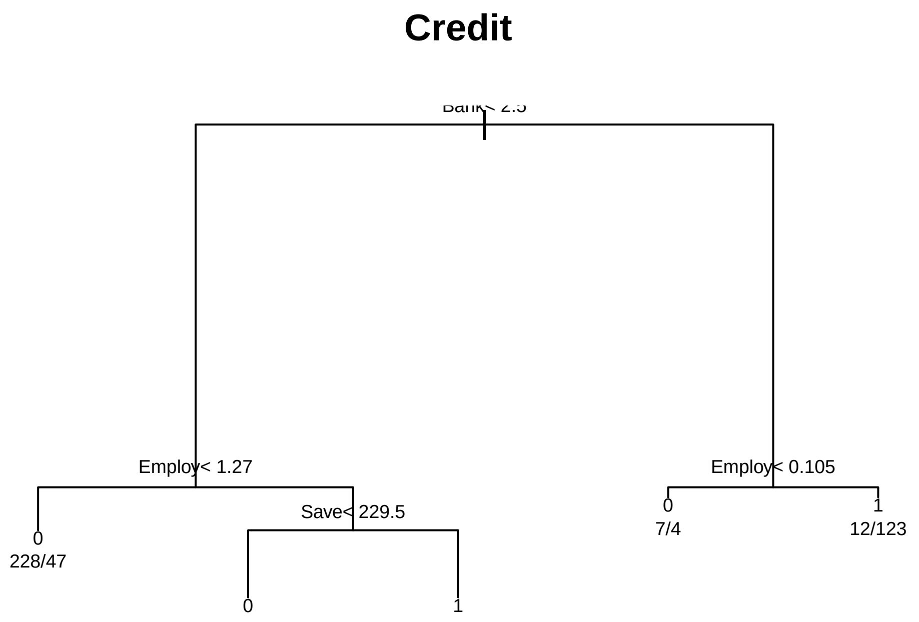
```
library(rpart)
names(d) #Show the variables of credit.csv
```

```
## [1] "Age"     "Address" "Employ"  "Bank"    "House"   "Save"    "Result"
```

```
#Classification tree
ctree=rpart(Result~Age+Address+Employ+Bank+House+Save,data=d1,method="class",control=rpart.control(maxd
```

7(c)

```
plot(ctree,asp=4,main="Credit") #Plot the branch of the tree
text(ctree,use.n=T,cex=0.6) #Add text to the tree
```



```
print(ctree) #Display the nodes
```

```
## n= 580
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 580 250 0 (0.56896552 0.43103448)
##    2) Bank< 2.5 434 123 0 (0.71658986 0.28341014)
##      4) Employ< 1.27 275  47 0 (0.82909091 0.17090909) *
##      5) Employ>=1.27 159  76 0 (0.52201258 0.47798742)
##       10) Save< 229.5 127  50 0 (0.60629921 0.39370079) *
##       11) Save>=229.5 32   6 1 (0.18750000 0.81250000) *
##    3) Bank>=2.5 146  19 1 (0.13013699 0.86986301)
##      6) Employ< 0.105 11   4 0 (0.63636364 0.36363636) *
##      7) Employ>=0.105 135  12 1 (0.08888889 0.91111111) *
```

```
sum(d1["Result"]) #total number of accepted case
```

```
## [1] 250
```

```
nrow(d1)-sum(d1["Result"]) #total number of rejected case
```

```
## [1] 330
```

Rejection Rule: 1. If Bank<2.5 and Employ<1.27, then the person is rejected.

```
Support=275/580 ; print(Support)
```

```
## [1] 0.4741379
```

```
Confidence=(275-47)/275 ; print(Confidence)
```

```
## [1] 0.8290909
```

```
Capture=(275-47)/434 ; print(Capture)
```

```
## [1] 0.5253456
```

2. If Bank<2.5 and Employ>1.27 and Save<229.5, then the person is rejected.

```
Support=127/580 ; print(Support)
```

```
## [1] 0.2189655
```

```
Confidence=(127-50)/127 ; print(Confidence)
```

```
## [1] 0.6062992
```

```
Capture=(127-50)/434 ; print(Capture)
```

```
## [1] 0.1774194
```

3. If Bank>2.5 and Employ< 0.105 then the person is rejected.

```
Support=11/580 ; print(Support)
```

```
## [1] 0.01896552
```

```r
Confidence=(11-4)/11 ; print(Confidence)
```

```
## [1] 0.6363636
```

```r
Capture=(11-4)/146 ; print(Capture)
```

```
## [1] 0.04794521
```

Acceptance rules: 1. If Bank<2.5 and Employ>1.27 and Save>229.5, then the person is accepted

```r
Support=32/580 ; print(Support)
```

```
## [1] 0.05517241
```

```r
Confidence=(32-6)/32 ; print(Confidence)
```

```
## [1] 0.8125
```

```r
Capture=(32-6)/434 ; print(Capture)
```

```
## [1] 0.05990783
```

2. If Bank>2.5 and and Employ> 0.105, then the person is accepted

```r
Support=135/580 ; print(Support)
```

```
## [1] 0.2327586
```

```r
Confidence=(135-12)/135 ; print(Confidence)
```

```
## [1] 0.9111111
```

```r
Capture=(135-12)/146 ; print(Capture)
```

```
## [1] 0.8424658
```

7(d)

```r
pr=predict(ctree) #probability of the sample
head(pr)
```

```
##              0         1
## 101 0.82909091 0.1709091
## 82  0.60629921 0.3937008
## 419 0.82909091 0.1709091
## 483 0.08888889 0.9111111
## 538 0.60629921 0.3937008
## 339 0.60629921 0.3937008
```

```
c1=max.col(pr) #classify the sample with the larger probability with 1:rejected and 2:accepted
head(c1)
```

```
## [1] 1 1 1 2 1 1
```

```
table(c1,d1$Result)#classification table
```

```
##
## c1    0   1
##   1 312 101
##   2  18 149
```

```
Error.rate=(101+18)/580 ; print(Error.rate)
```

```
## [1] 0.2051724
```

```
#error rate
```

7(e)

```
pr2=predict(ctree,d2)#probability of testing data set
head(pr2)
```

```
##             0         1
## 1  0.82909091 0.1709091
## 4  0.08888889 0.9111111
## 14 0.82909091 0.1709091
## 17 0.08888889 0.9111111
## 28 0.08888889 0.9111111
## 48 0.08888889 0.9111111
```

```
c2=max.col(pr2)#classify the sample with the larger probability
head(c2)
```

```
## [1] 1 2 1 2 2 2
```

```
table(c2,d2$Result)#classification table
```

```
##
## c2   0  1
##   1 47 24
##   2  6 33
```

```
Error.rate=(24+6)/(47+33+24+6) ; print(Error.rate) #error rate
```

```
## [1] 0.2727273
```