HW 2

Carlyle Morgan

2/19/2022

Problem 4

1.

```
digit_3 <- read.table("train_3.txt",header = F,sep=',')
digit_5 <- read.table("train_5.txt",header = F,sep=',')
digit_8 <- read.table("train_8.txt",header = F,sep=',')
digit_3$digit<-'3'
digit_5$digit<-'5'
digit_8$digit<-'8'

X_temp <- rbind(digit_3,digit_5,digit_8)

trainind<-sample(nrow(X_temp), size = floor(0.75*1756))
train<-X_temp[trainind,]
test<-X_temp[-trainind,]

f<-paste("digit","~", paste(names(train)[-257], collapse = " + "))
lda256<-lda(as.formula(paste(f)), data = train)

lda256trainpred<-predict(lda256, train)
lda256trainerror<-sum(ifelse(as.numeric(train$digit)==lda256trainpred$class, 0, 1))/nrow(train)

lda256testpred<-predict(lda256, test)
lda256testerror<-sum(ifelse(as.numeric(test$digit)==lda256testpred$class, 0, 1))/nrow(test)
```

2. ***

```
digits_pca<-prcomp(train[,-257])
train_pca<-as.data.frame(digits_pca$x[,1:49])
train_pca$digit<-train$digit
test_pca<-as.data.frame(as.matrix(scale(test[, -257], center = TRUE, scale = TRUE)) %*% digits_pca$rota
test_pca$digit<-test$digit

lda49<-lda(digit~., data = train_pca)

lda49trainpred<-predict(lda49, train_pca)
lda49trainerror<-sum(ifelse(as.numeric(train_pca$digit)==lda49trainpred$class, 0, 1))/nrow(train_pca)
lda49testpred<-predict(lda49, test_pca)
lda49testerror<-sum(ifelse(as.numeric(test$digit)==lda49testpred$class, 0, 1))/nrow(test)
```

3.

```r
multinomreg<-multinom(digit~., train)
```

```
## # weights:  774 (514 variable)
## initial  value 1446.872384
## iter  10 value 116.149745
## iter  20 value 36.001414
## iter  30 value 13.188504
## iter  40 value 4.450002
## iter  50 value 1.643305
## iter  60 value 0.609351
## iter  70 value 0.201638
## iter  80 value 0.065107
## iter  90 value 0.018878
## iter 100 value 0.008294
## final  value 0.008294
## stopped after 100 iterations
```

```r
multinomtrainpred<-predict(multinomreg, train)
multinomtrainerror<-sum(ifelse(as.numeric(train$digit)==multinomtrainpred, 0, 1))/nrow(train)
multinomtestpred<-predict(multinomreg, test)
multinomtesterror<-sum(ifelse(as.numeric(test$digit)==multinomtestpred, 0, 1))/nrow(test)


training_error<-c(lda256trainerror,lda49trainerror, multinomtrainerror)
test_error<-c(lda256testerror, lda49testerror, multinomtesterror)
models<-c("LDA on R_256", "LDA on 49 PCs", "Logistic Multinomial")

results<-data.frame(models, training_error, test_error)
head(results)
```

```
##                 models training_error test_error
## 1         LDA on R_256     0.01214882  0.0501139
## 2        LDA on 49 PCs     0.04479879  0.0523918
## 3 Logistic Multinomial     0.00000000  0.1184510
```

Problem 5

1.

```r
stock_data <- getSymbols("AAPL", auto.assign = F, from ="2021-01-01", to = "2022-01-01")[,4]
```

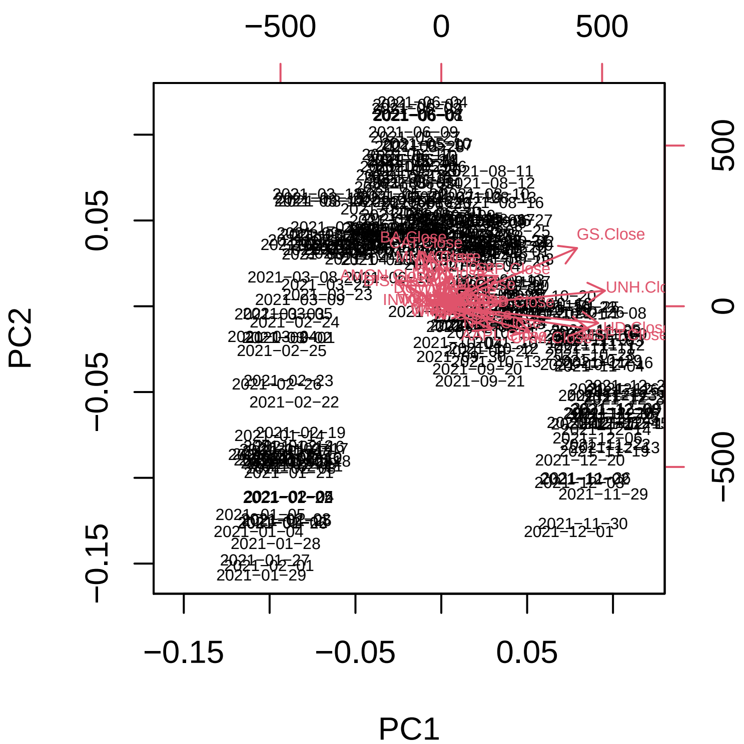
```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.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

```r
for (i in 2:length(comps)){
  temp_data = getSymbols(comps[i], auto.assign = F, from ="2021-01-01", to = "2022-01-01")
  stock_data = cbind(stock_data, temp_data[,4])
```

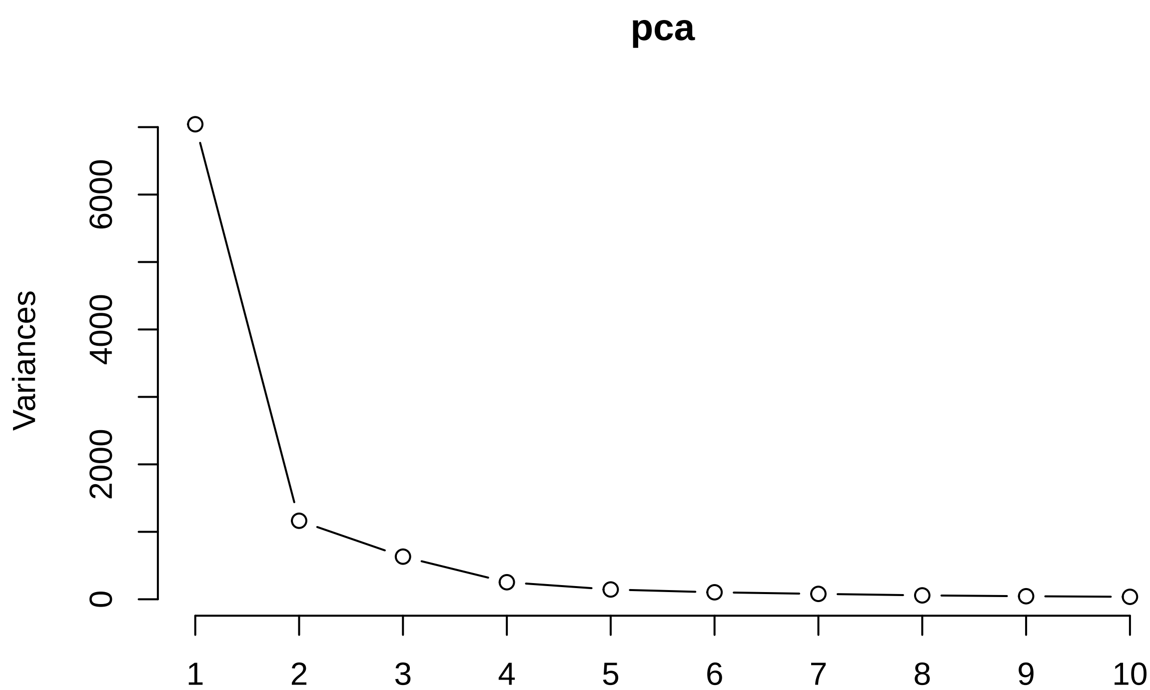
```
}
stock_data<-data.frame(stock_data)
```

2.

```
pca<-prcomp(stock_data)
biplot(pca, cex = 0.5)
```



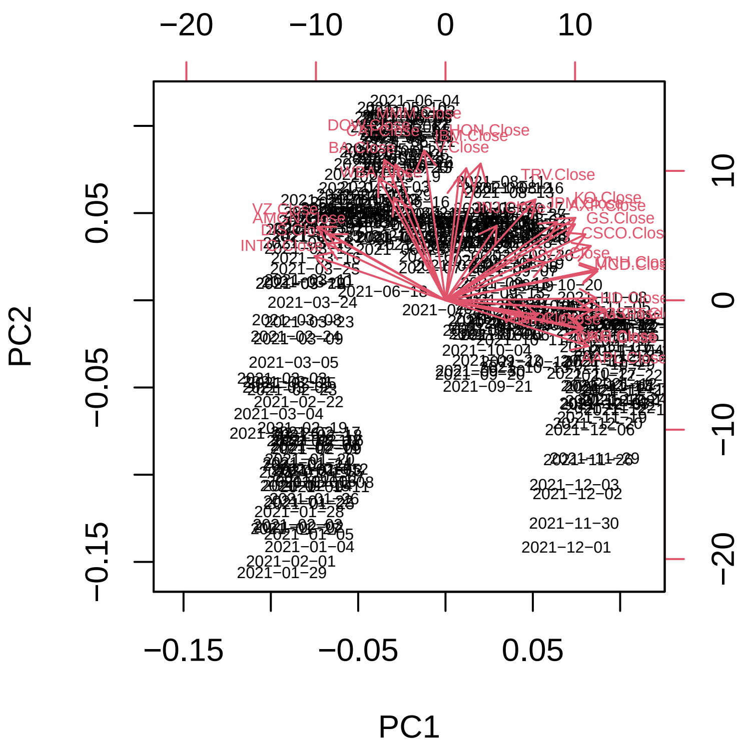

```
screeplot(pca,type="l")
```



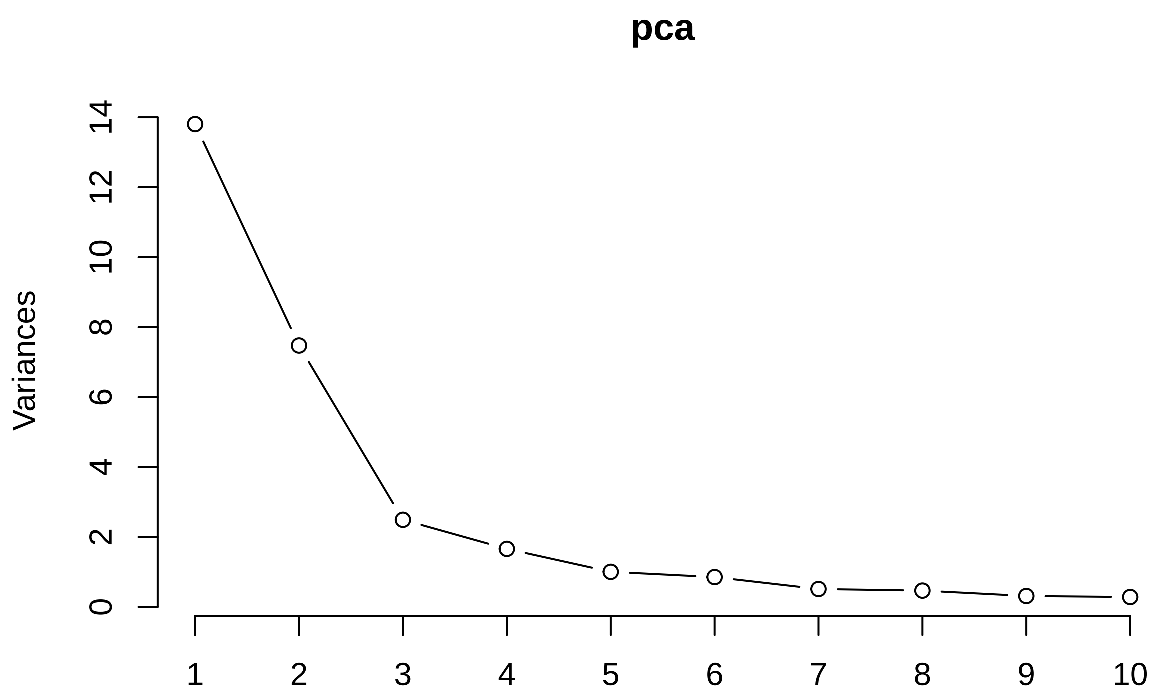
From this plot, it is difficult to interpret exactly what types of stocks are correlated with each other. However, one can see that stocks like GS and UNH have high variability, and that in general all stocks in the Dow Jones Industrial had lower prices in early 2021.

3.

```
stock_data.c <- scale(stock_data)
pca<-prcomp(stock_data.c)
biplot(pca, cex = 0.5)
```



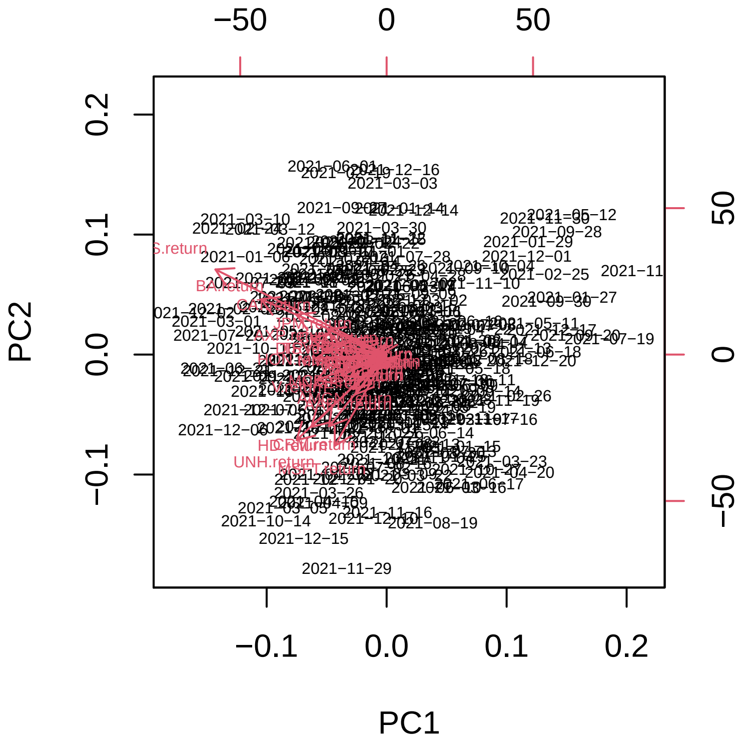

```r
screeplot(pca,type="l")
```



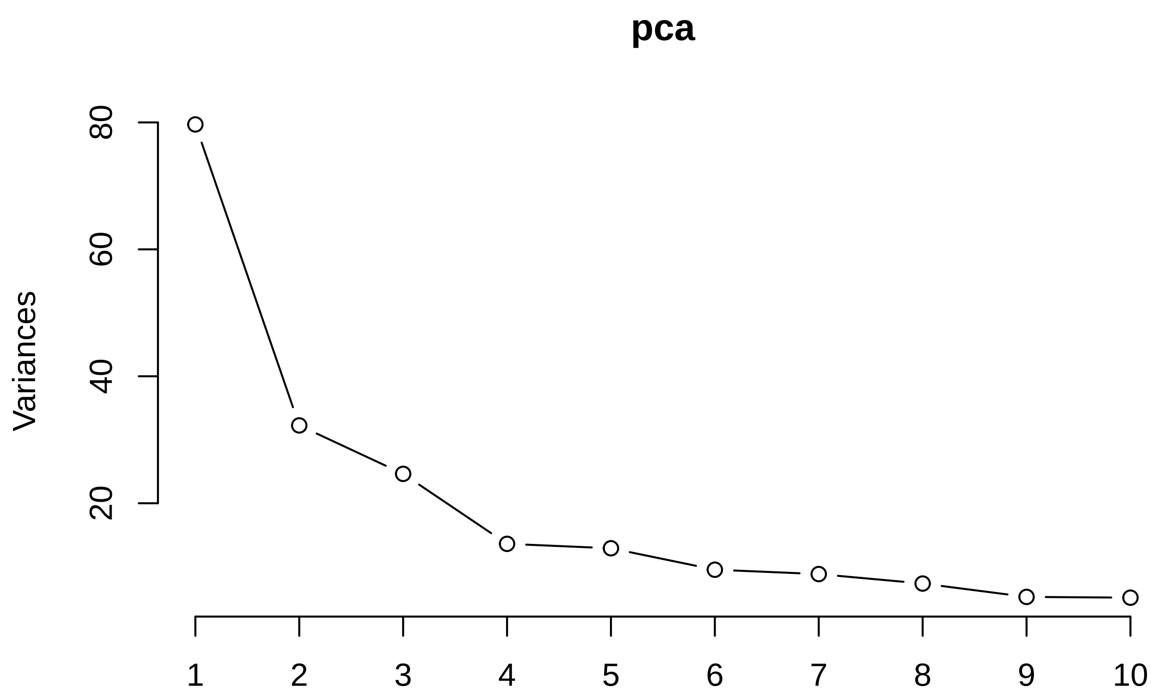
From the scaled plot, we can see that some stocks are not particularly correlated with each other in terms of their prices, but there is not a neat divide across industries. For example, Intel and Apple seem to have negatively correlated stock prices. This migtht make sense as Apple and Intel are competitors. We also see that in the first few days of the year, stock prices are lower for all of the Dow Jones 30.

4.

```
returns<-data.frame(apply(stock_data,2,diff))
colnames(returns)<-sapply(comps, function(x) paste(x,".return", sep = ""))
pca<-prcomp(returns)
biplot(pca, cex = 0.5)
```




```r
screeplot(pca,type="l")
```



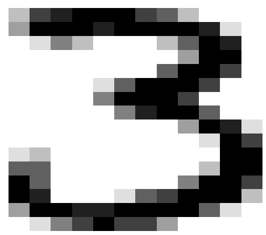
From this plot, there is not much to see in terms of industry divides. However, like the first biplot, those firms that had high variability in terms of stock prices also had high variability in terms of stock returns. Firms like GS and UNH have vectors with high magnitudes. From our screeplot, we would likely benefit the most by using 8 or 9 principal components, as the amount of variance explained by the first couple components is a lot less than those in our previous examples. If the stocks were completely uncorrelated in terms of returns, we'd see a much flatter screeplot without as many large jumps. We need a lot of principle components to model several uncorrelated variables.

Problem 6

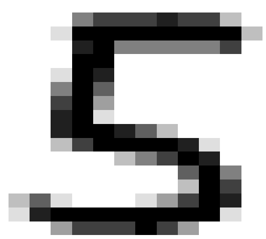
a.

```
# Define output.image function (Lab 2)
output.image<-function(vector) {
  digit<-matrix(vector, nrow=16, ncol=16)
  index= seq(from=16, to =1, by=-1)
  sym_digit = digit[,index]
  image(sym_digit, col= gray((8:0)/8), axes=FALSE)
}


# Define temporary data matrix
digit_3 <- read.table("train_3.txt",header = F,sep=',')
digit_5 <- read.table("train_5.txt",header = F,sep=',')
digit_8 <- read.table("train_8.txt",header = F,sep=',')
```

```r
X_temp <- rbind(digit_3,digit_5,digit_8)

# Extract "test" cases
# Test case 1
ConstructCase_1 <- X_temp[20,]
output.image(as.matrix(ConstructCase_1))
```

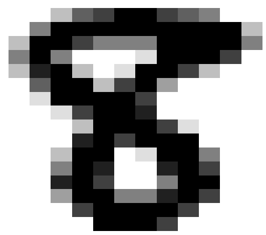


```r
ConstructCase_1 <- unlist(ConstructCase_1) # Not needed but might be helpful

# Test case 2
ConstructCase_2 <- X_temp[735,]
output.image(as.matrix(ConstructCase_2))
```



```r
ConstructCase_2 <- unlist(ConstructCase_2) # Not needed but might be helpful

# Test case 3
```

```r
ConstructCase_3 <- X_temp[1260,]
output.image(as.matrix(ConstructCase_3))
```

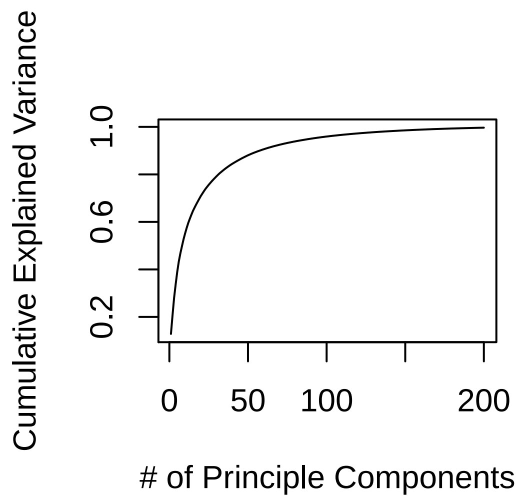


```r
ConstructCase_3 <- unlist(ConstructCase_3) # Not needed but might be helpful

# Remove cases 20, 735, 1260 from original dataframe
X <- X_temp[-c(20,735,1260),]
dim(X)
```

```
## [1] 1753  256
```

b.

```r
pca<-prcomp(X)
cumlvar<-function(x){
  pctexplvariance<-(pca$sdev)^2/sum(pca$sdev^2)
  sum(pctexplvariance[1:x])
}
y<-sapply(seq(1,200,1), cumlvar)
plot(seq(1,200,1), y, type = "l",
     xlab = "# of Principle Components", ylab = "Cumulative Explained Variance")
```



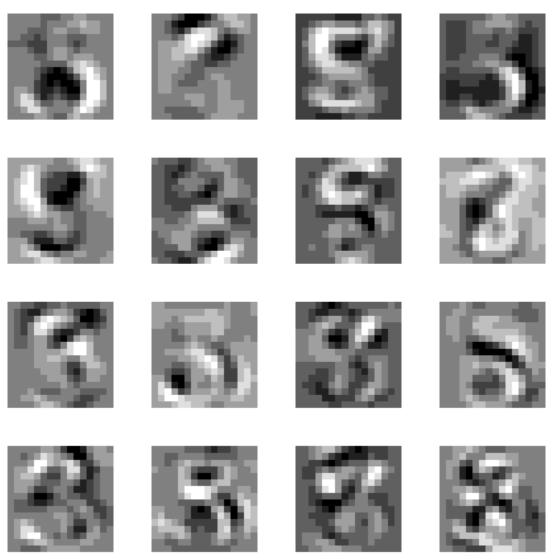

```
length(y[y<.9])
```

```
## [1] 57
```

Therefore, 58 principle components are needed to reach our desired threshold.

c.

```
par(mfrow=c(4,4), mai = c(0.1,0.1,0.1,0.1))
for (i in 1:16){
  output.image(as.matrix(pca$rotation[,i]))
}
```



d.

```
par(mfrow=c(3,3), mai = c(0.1,0.1,0.1,0.1))
c1<-c()
for (i in 1:256){
  c1[i]<-sum(pca$rotation[,i]*ConstructCase_1)
}

c1_estimate<-c1[1]*pca$rotation[,1]
for (i in 2:3){
  c1_estimate<-c1_estimate+c1[i]*pca$rotation[,i]
}
c1_estimate3<-c1_estimate
output.image(as.matrix(c1_estimate3))
for (i in 4:58){
  c1_estimate<-c1_estimate+c1[i]*pca$rotation[,i]
}
c1_estimate58<-c1_estimate
output.image(c1_estimate58)
for (i in 59:256){
  c1_estimate<-c1_estimate+c1[i]*pca$rotation[,i]
}
c1_estimate256<-c1_estimate
output.image(c1_estimate256)

c2<-c()
for (i in 1:256){
  c2[i]<-sum(pca$rotation[,i]*ConstructCase_2)
}

c2_estimate<-c2[1]*pca$rotation[,1]
for (i in 2:3){
  c2_estimate<-c2_estimate+c2[i]*pca$rotation[,i]
}
c2_estimate3<-c2_estimate
output.image(as.matrix(c2_estimate3))
for (i in 4:58){
  c2_estimate<-c2_estimate+c2[i]*pca$rotation[,i]
}
c2_estimate58<-c2_estimate
output.image(c2_estimate58)
for (i in 59:256){
  c2_estimate<-c2_estimate+c2[i]*pca$rotation[,i]
}
c2_estimate256<-c2_estimate
output.image(c2_estimate256)

c3<-c()
for (i in 1:256){
  c3[i]<-sum(pca$rotation[,i]*ConstructCase_3)
}

c3_estimate<-c3[1]*pca$rotation[,1]
for (i in 2:3){
  c3_estimate<-c3_estimate+c3[i]*pca$rotation[,i]
```

```
}
c3_estimate3<-c3_estimate
output.image(as.matrix(c3_estimate3))
for (i in 4:58){
  c3_estimate<-c3_estimate+c3[i]*pca$rotation[,i]
}
c3_estimate58<-c3_estimate
output.image(c3_estimate58)
for (i in 59:256){
  c3_estimate<-c3_estimate+c3[i]*pca$rotation[,i]
}
c3_estimate256<-c3_estimate
output.image(c3_estimate256)
```