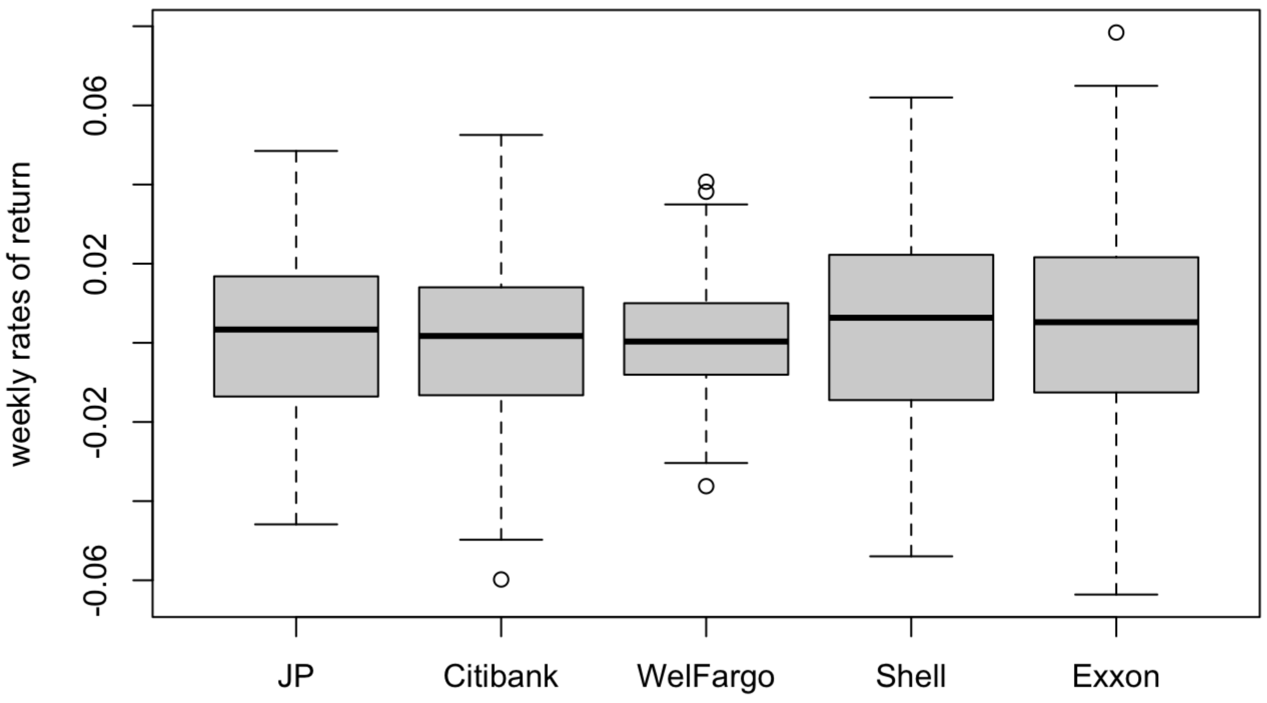
code

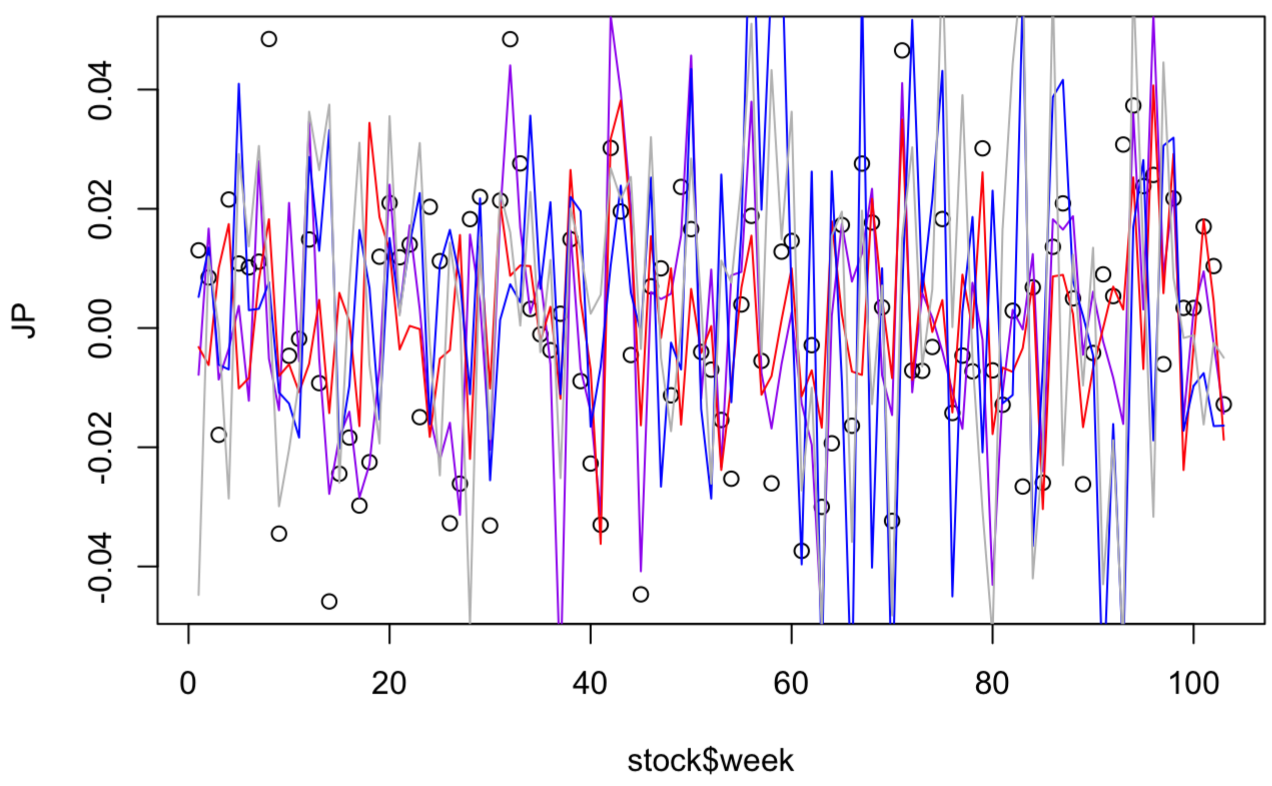
PCA - bank

```
stock <- read.delim("~/Documents/win21/sta135/project/T8-4.DAT", header=FALSE)
names(stock) <- c("JP", "Citibank", "WelFargo", "Shell", "Exxon")
groupId <- colnames(stock)
boxplot(stock, ylab = "weekly rates of return")
```



```
stock$week <- seq.int(nrow(stock))
```

```
# Make a basic graph
plot( data = stock, JP~stock$week)
lines(stock$Citibank~stock$week , col="purple")
lines(stock$WelFargo~stock$week , col="red")
lines(stock$Exxon~stock$week , col="blue")
lines(stock$Shell~stock$week , col="gray")
```



```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     1.5611768 1.1861756 0.7074693 0.63248050 0.50514343
## Proportion of Variance 0.4874546 0.2814025 0.1001025 0.08000632 0.05103398
## Cumulative Proportion  0.4874546 0.7688572 0.8689597 0.94896602 1.00000000
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## JP        0.469  0.368  0.604  0.363  0.384
## Citibank  0.532  0.236  0.136 -0.629 -0.496
## WelFargo  0.465  0.315 -0.772  0.289       
## Shell     0.387 -0.585        -0.381  0.595
## Exxon     0.361 -0.606  0.109  0.493 -0.498
```

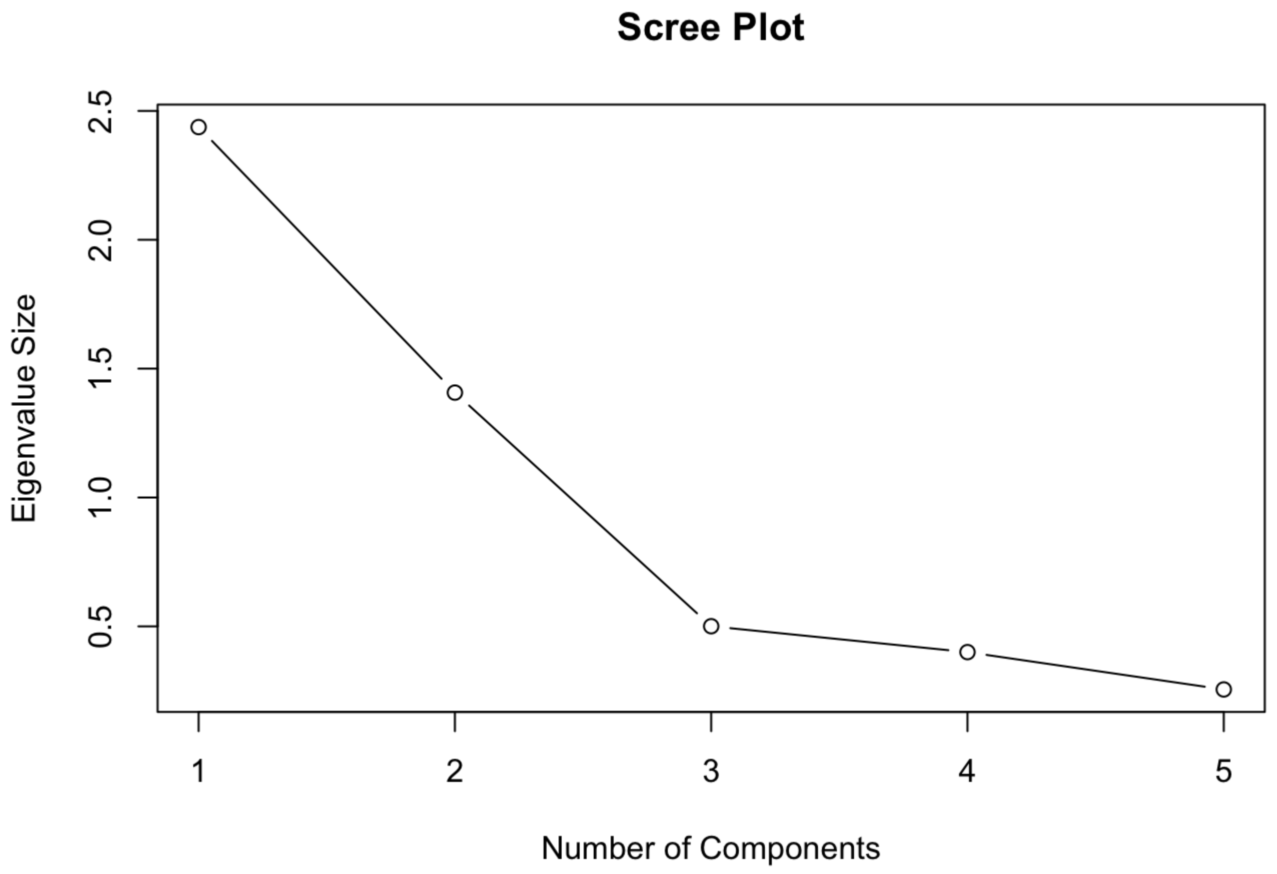
Showing the eigenvalues of the correlation matrix:

```
(X.pca$sdev)^2
```

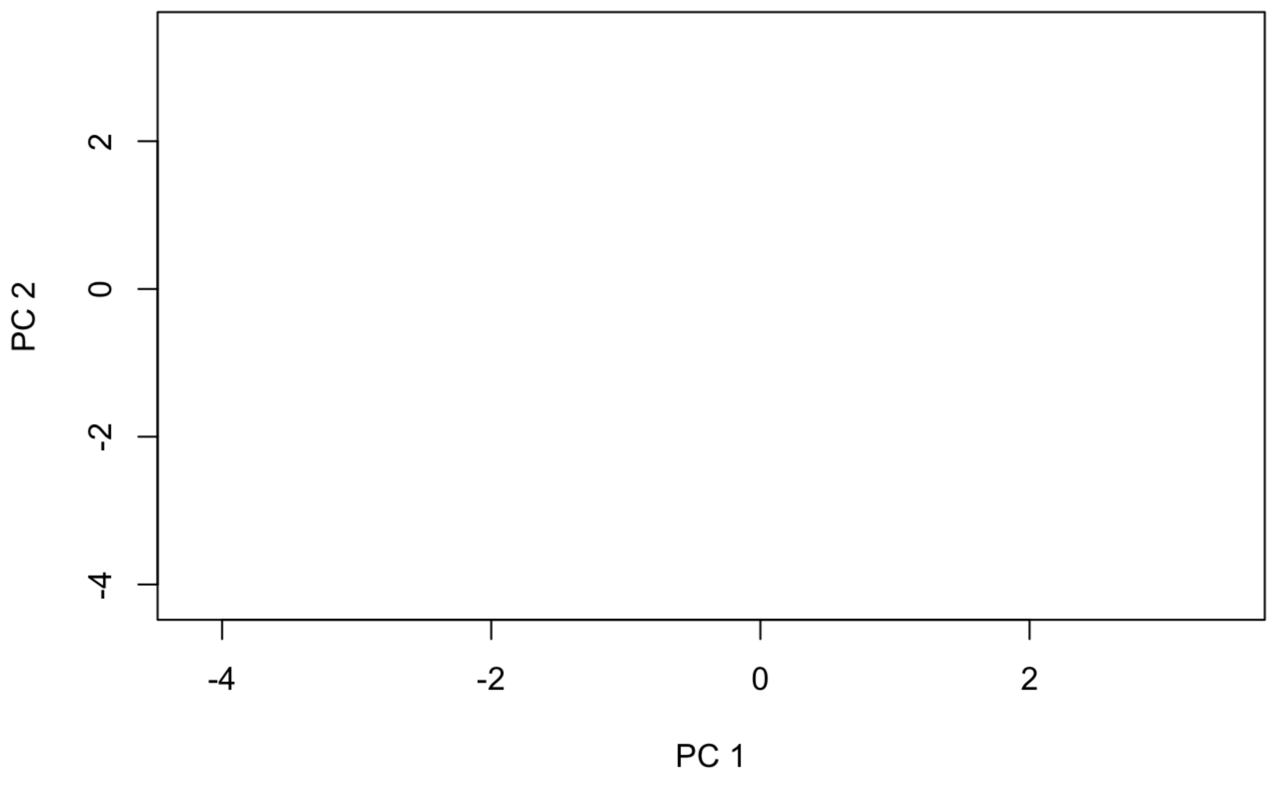
```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## 2.4372731 1.4070127 0.5005127 0.4000316 0.2551699
```

A scree plot:

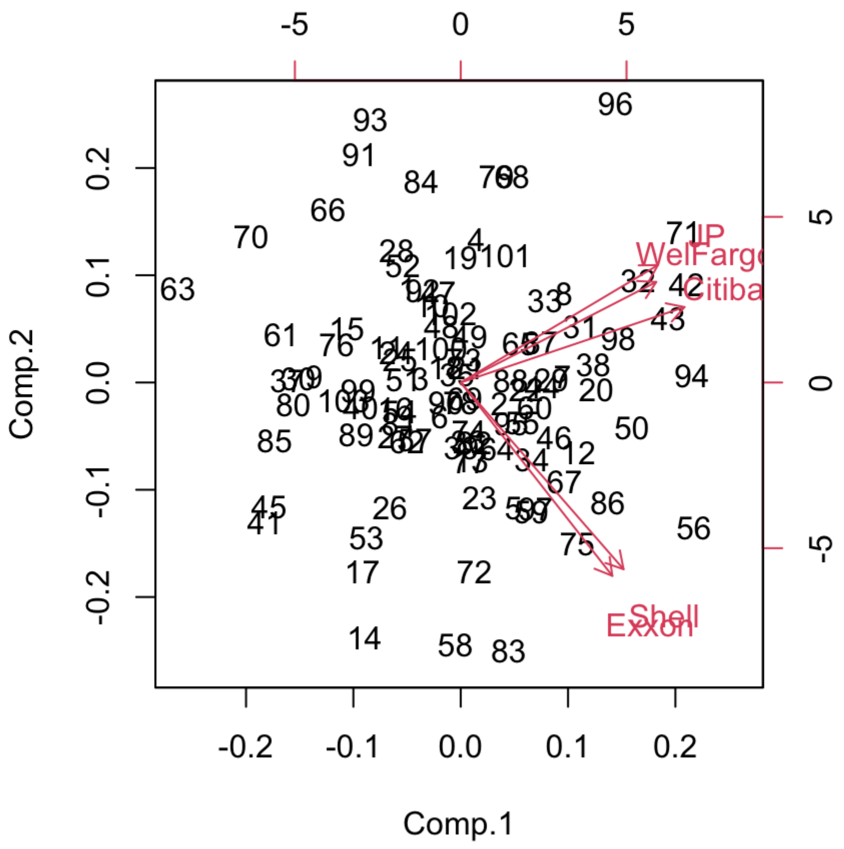
```
plot(1:(length(X.pca$sdev)),  (X.pca$sdev)^2, type='b',
     main="Scree Plot", xlab="Number of Components", ylab="Eigenvalue Size")
```



```
plot(X.pca$scores[,1], X.pca$scores[,2], ylim=range(X.pca$scores[,1]),
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
```



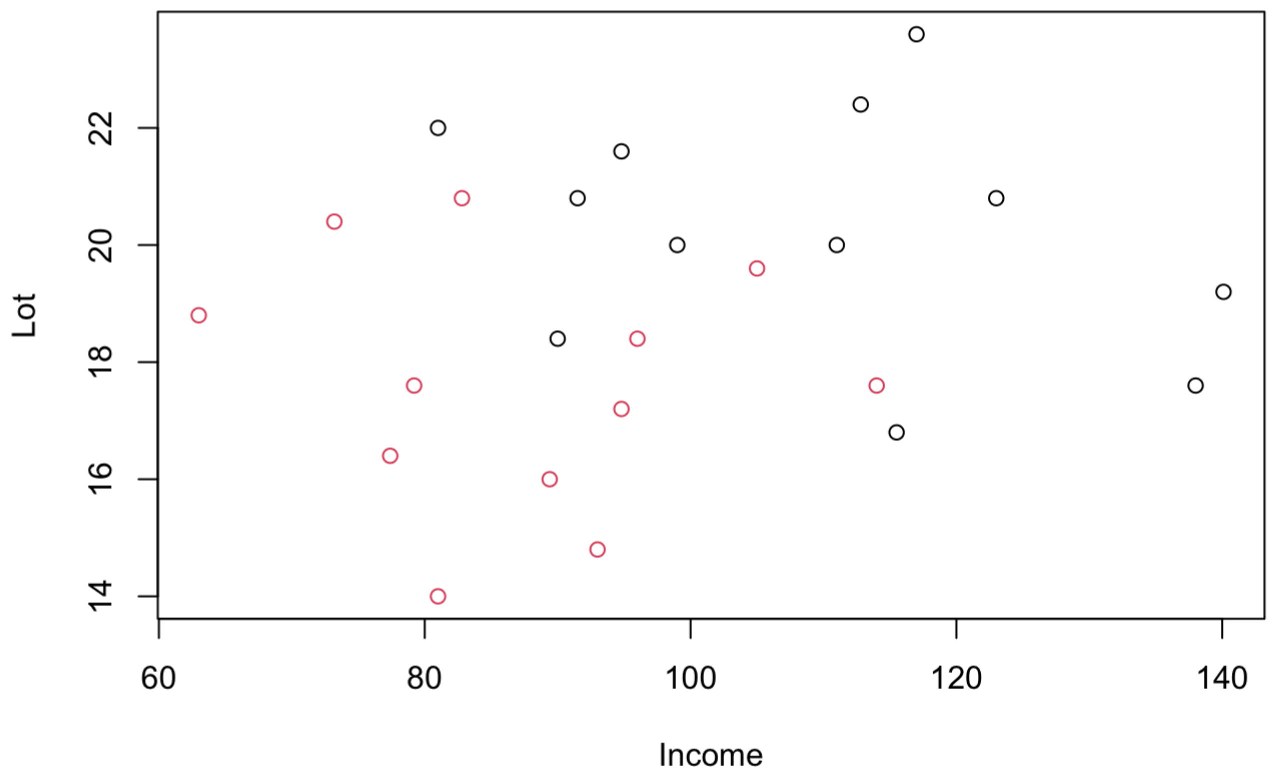
```
biplot(X.pca)
```



LDA - Mover

```
mover <- read.table("~/Documents/win21/sta135/project/T11-1.DAT", quote="\"", commen
t.char="")
names(mover) <- c("Income", "Lot", "owner")
```

```
plot(mover[,1:2],col= mover$owner)
```



```
library(MASS)
(lda.obj<-lda(owner~Income+Lot,data=mover, prior=c(1,1)/2))
```

```
## Call:
## lda(owner ~ Income + Lot, data = mover, prior = c(1, 1)/2)
##
## Prior probabilities of groups:
##   1   2
## 0.5 0.5
##
## Group means:
##    Income      Lot
## 1 109.475 20.26667
## 2  87.400 17.63333
##
## Coefficients of linear discriminants:
##               LD1
## Income -0.0484468
## Lot    -0.3795228
```

```
plda<-predict(object=lda.obj, newdata=mover)
```

```
#determine how well the model fits
true_class <- mover[,3]
table(true_class, plda$class)
```

```
##
## true_class  1  2
##          1 11  1
##          2  2 10
```

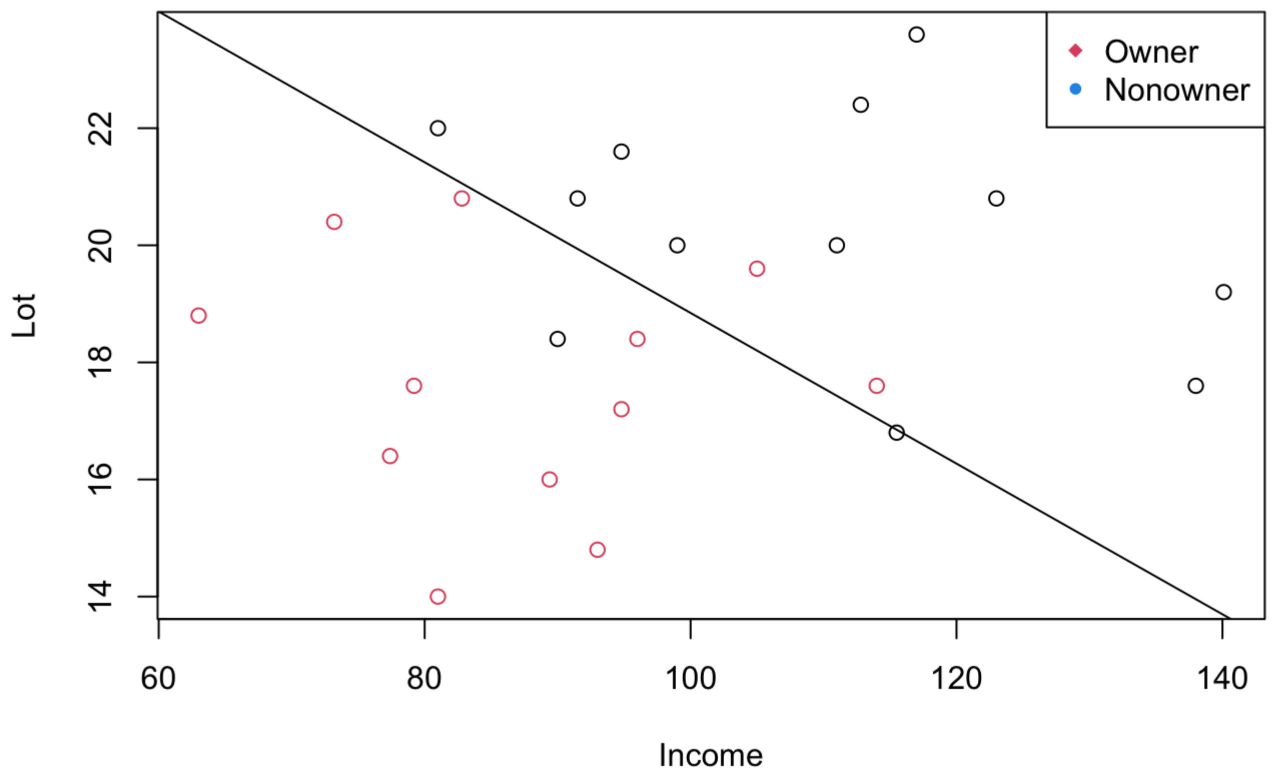
```
mis <- 0
df2 <- mover
for (i in 1:dim(df2)[1]){
  lda.obj<-lda(owner~., data = df2[row.names(df2) != as.character(i),],prior = c(1,1
)/2)
  plda<-predict(object = lda.obj, df2[row.names(df2) == as.character(i),])
  mis <- mis + as.integer(plda$class != df2[i,3])
}
mis
```

```
## [1] 5
```

```
mis / dim(df2)[1]
```

```
## [1] 0.2083333
```

```
#plot the decision line
gmean <- lda.obj$prior %*% lda.obj$means
const <- as.numeric(gmean %*%lda.obj$scaling)
slope <- - lda.obj$scaling[1] / lda.obj$scaling[2]
intercept <- const / lda.obj$scaling[2]
#Plot decision boundary
plot(mover[,1:2],col= mover$owner)
abline(intercept, slope)
legend("topright",legend=c("Owner","Nonowner"),pch=c(18,20),col=c(2,4))
```



#

LRM - SWEAT

```
sweat <- read.table("~/Documents/win21/sta135/project/T5-1.DAT", quote="\"", comment.
char="")
names(sweat) <- c("rate", "Sodium", "Potassium")
sweat$ID <- seq.int(nrow(sweat))
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
fig <- plot_ly(sweat, x = sweat$Potassium, y = sweat$Sodium, z = sweat$rate)
fig <- fig %>% add_markers()
fig <- fig %>% layout(scene = list(zaxis = list(title = 'Rate'),
                     yaxis = list(title = 'Sodium'),
                     xaxis = list(title = 'Potassium')))

fig
```

```
# data
Y <- sweat[,1]
n <- length(Y)
Z <- cbind(rep(1,n),as.matrix(sweat[,2:3]))
r <- dim(Z)[2]-1
summary(lm(Y~Z))
```

```
##
## Call:
## lm(formula = Y ~ Z)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0372 -0.6842 -0.2219  0.8429  2.1846
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.31500    2.18870   3.342  0.00386 **
## Z                 NA         NA      NA       NA
## ZSodium      0.03768    0.02292   1.644  0.11857
## ZPotassium  -0.44010    0.17010  -2.587  0.01918 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.381 on 17 degrees of freedom
## Multiple R-squared:  0.4075, Adjusted R-squared:  0.3378
## F-statistic: 5.846 on 2 and 17 DF,  p-value: 0.01169
```

```
# least square estimates
beta_hat <- solve(t(Z)%*%Z)%*%t(Z)%*%Y

# R^2 statistic
(R_square <- 1 - sum((Y - Z%*%beta_hat)^2)/sum((Y-mean(Y))^2))
```

```
## [1] 0.4074977
```

```
# sigma_hat_square
sigma_hat_square <- sum((Y - Z%*%beta_hat)^2)/(n-r-1)

# estimated covariance of hat{beta}
sigma_hat_square * solve(t(Z)%*%Z)
```

```
##                                Sodium     Potassium
##             4.79040135 -0.0319908667 -0.3254068133
## Sodium     -0.03199087  0.0005253636  0.0008167945
## Potassium -0.32540681  0.0008167945  0.0289337023
```

```r
Omega <- solve(t(Z)%*%Z)

# t-test for single coefficient
# H_0: beta_j = 0, H_a: beta_j != 0

j <- 1
t_stat <- (beta_hat[j+1] - 0)/sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1])
alpha <- 0.05
cval_t <- qt(1-alpha/2, n-r-1)
# One-at-a-time confidence interval for beta_j

j <- 1
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square * Omega[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square * Omega[j+1,j+1]),
    ']')
```

```
## [ -0.01067973 , 0.08603764 ]
```

```r
# Confidence region based simultaneous confidence intervals

j <- 1
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha,r+1,n-r-1))*sqrt(sigma_hat_square * Omega[j
+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha,r+1,n-r-1))*sqrt(sigma_hat_square * Omega[j
+1,j+1]),
    ']')
```

```
## [ -0.03330281 , 0.1086607 ]
```

```r
# Bonferroni correction based confidence intervals

j <- 1
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*sqrt(sigma_hat_square * Omega[j+1,j+
1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*sqrt(sigma_hat_square * Omega[j+1,j+
1]),
    ']')
```

```
## [ -0.0231757 , 0.09853361 ]
```

```
# F-test
# H_0: beta_1 = beta_2 = 0

C <- matrix(c(0,0,1,0,0,1),2,3)

df_1 <- qr(C)$rank # df_1: rank of matrix C
q = 0

Omega_22 = C%*% solve(t(Z)%*%Z) %*%t(C)
f_stat <- t(C%*%beta_hat)%*%solve(Omega_22)%*%(C%*%beta_hat)
f_stat
```

```
##          [,1]
## [1,] 22.29338
```

```
cval_f <- qf(1-alpha, 2, n-r-1)
cval_f * df_1 * sigma_hat_square
```

```
## [1] 13.69625
```

```
# (equivalent) F-test by comparing residuals
# fit the reduced model
Z_1 <- Z[,1:(1+q)]
beta_hat_reduced <- solve(t(Z_1)%*%Z_1)%*%t(Z_1)%*%Y

RSS_red = sum((Y - Z_1%*%beta_hat_reduced)^2)
RSS_full = sum((Y - Z%*%beta_hat)^2)
f_stat_reduced <- (RSS_red - RSS_full)/sigma_hat_square
f_stat_reduced
```

```
## [1] 11.69187
```

```
cval_f * (r-q)
```

```
## [1] 7.183061
```

```
# confidence interval for z_0^T beta

z_0 <- c(1, 4, 3)

cat('[',
    z_0%*%beta_hat - sqrt(sigma_hat_square)*qt(1-alpha/2, n-r-1)*sqrt(t(z_0)%*%solve
(t(Z)%*%Z)%*%z_0),
    ',',
    z_0%*%beta_hat + sqrt(sigma_hat_square)*qt(1-alpha/2, n-r-1)*sqrt(t(z_0)%*%solve
(t(Z)%*%Z)%*%z_0),
    ']')
```

```
## [ 2.570877 , 9.719941 ]
```

```
# prediction interval for Y_0 = z_0^T beta + epsilon_0

cat('[',
    z_0%*%beta_hat - sqrt(sigma_hat_square)*qt(1-alpha/2, n-r-1)*sqrt(1+t(z_0)%*%solv
e(t(Z)%*%Z)%*%z_0),
    ',',
    z_0%*%beta_hat + sqrt(sigma_hat_square)*qt(1-alpha/2, n-r-1)*sqrt(1+t(z_0)%*%solv
e(t(Z)%*%Z)%*%z_0),
    ']')
```

```
## [ 1.534032 , 10.75679 ]
```