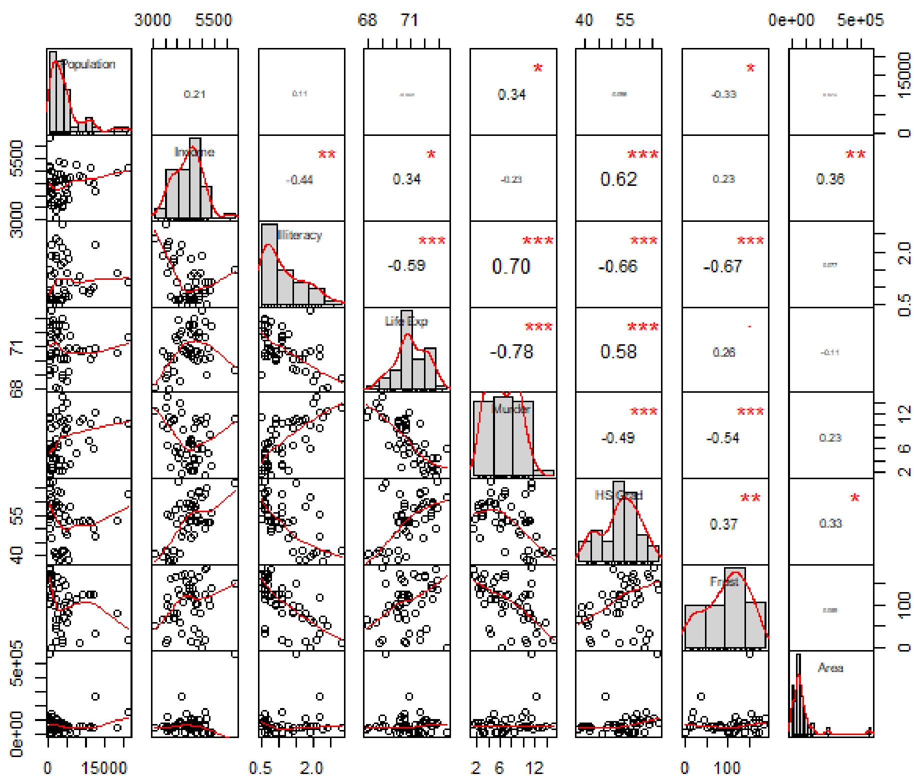
Problem 1

a) `state.x77` dataset has 50 observations on 8 variables. There is no respond variable and all variables are considered as predictor variables. All the variables are quantitative. Bellow is a summary of these variables.

```
   Population        Income       Illiteracy       Life Exp         Murder          HS Grad     
 Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96   Min.   : 1.400   Min.   :37.80  
 1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12   1st Qu.: 4.350   1st Qu.:48.05  
 Median : 2838   Median :4519   Median :0.950   Median :70.67   Median : 6.850   Median :53.25  
 Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88   Mean   : 7.378   Mean   :53.11  
 3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89   3rd Qu.:10.675   3rd Qu.:59.15  
 Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60   Max.   :15.100   Max.   :67.30  
     Frost             Area       
 Min.   :  0.00   Min.   :  1049  
 1st Qu.: 66.25   1st Qu.: 36985  
 Median :114.50   Median : 54277  
 Mean   :104.46   Mean   : 70736  
 3rd Qu.:139.75   3rd Qu.: 81163  
 Max.   :188.00   Max.   :566432  
```

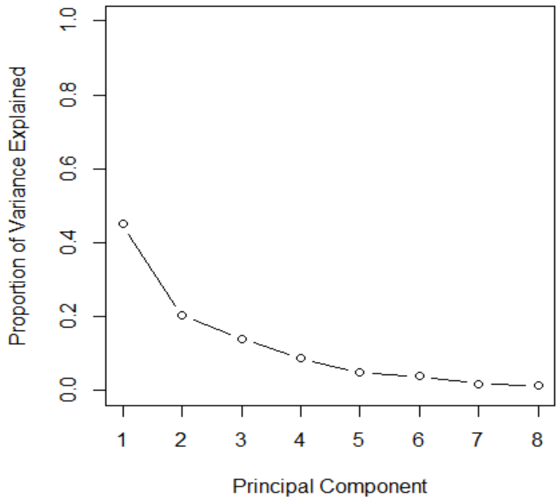
Following figure shows how most of these variables, specially `Life Exp`, `Murder`, `HS Grad` and `Frost`, are highly correlates with other variables.

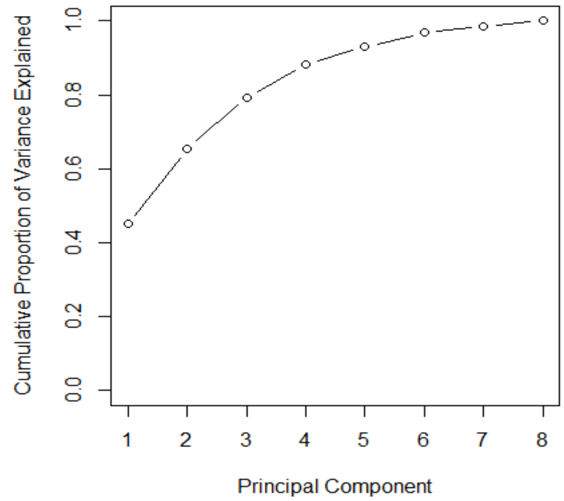


b) Data needs to be standardized in order to make the sample mean of principle components = 0 and variance equal to the eigen values of them. Thus make them uncorrelated.

```
> apply(state.x77, 2, mean)
Population     Income Illiteracy   Life Exp     Murder    HS Grad      Frost       Area 
 4246.4200  4435.8000     1.1700    70.8786     7.3780    53.1080   104.4600 70735.8800 
> apply(state.x77, 2, sd)
  Population       Income   Illiteracy     Life Exp       Murder      HS Grad        Frost 
4.464491e+03 6.144699e+02 6.095331e-01 1.342394e+00 3.691540e+00 8.076998e+00 5.198085e+01 
        Area 
8.532730e+04 
```

c) PCA was preformed on the data and proportion of variance explained (PVE) was computed to find the number of PCs needed. Bellow are the graphs of PVE and cumulative of PVE against the number of PCs.





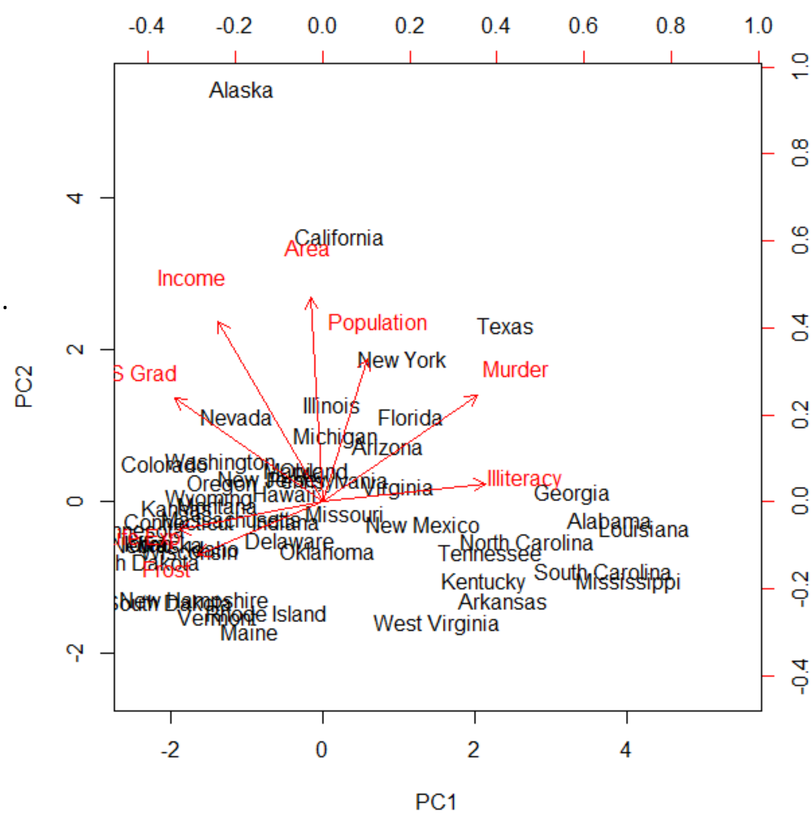

```
> cumsum(pve)
[1] 0.4498619 0.6538519 0.7928445 0.8812825 0.9293627 0.9677954 0.9858515 1.0000000
```

In the first graph there is a big drop from 1PC to 2PCs and a small drop from 6PCs to 7PCs. Also by looking at the Cumulative of PVE, we have 65% with 2PCs and 92% with 5PCs. So I would recommend to have more than 5PCs for this data set.

d)

```
    Population     Income Illiteracy   Life Exp    Murder    HS Grad      Frost        Area
PC1  0.2398436 -0.5669029 0.88720374 -0.7808560 0.8427885 -0.8056584 -0.6780384 -0.06333314
PC2  0.5248778  0.6629778 0.06766573 -0.1043129 0.3921173  0.3816642 -0.1961984  0.75067024
    cumulative.PVE
PC1      0.4498619
PC2      0.6538519
```

With the figure, its clear that the two components are related to most of the states. This can also be seen with the cumulative percentage of the total variability explained. When you have the first two PCs cumulative.PVE = `0.6538519` and thus they were enough to explain more than half of the data. Most of the unexplained states are southern. Thus there is a southern component in the figure.



Problem 2

a)

```
> signif(sort(train.latent.sem[, 1], decreasing = TRUE)[1:30], 2)
    theater        music            m    theaters    composers      matinee        opera
      0.180        0.130        0.080       0.079        0.078        0.077        0.075
> signif(sort(train.latent.sem[, 1], decreasing = FALSE)[1:30], 2)
        her          she       ms     painting    paintings       mother       cooper      artists
     -0.150       -0.140   -0.130       -0.110       -0.100       -0.092       -0.090       -0.086
> signif(sort(train.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
        she          her     theater         said            i           ms       mother       cooper
      0.240        0.240       0.200        0.170        0.120        0.110        0.110        0.110
> signif(sort(train.latent.sem[, 2], decreasing = FALSE)[1:30], 2)
   patterns      chinese          feb      chelsea    computers      diamond    europeans
     -0.065       -0.051       -0.046       -0.046       -0.046       -0.045       -0.044
```

With the 1st PC, words with positive projections are mostly associated with music, those with negative components with the visual arts and with the 2nd PC, the positive words are about art and the negative words are musical.

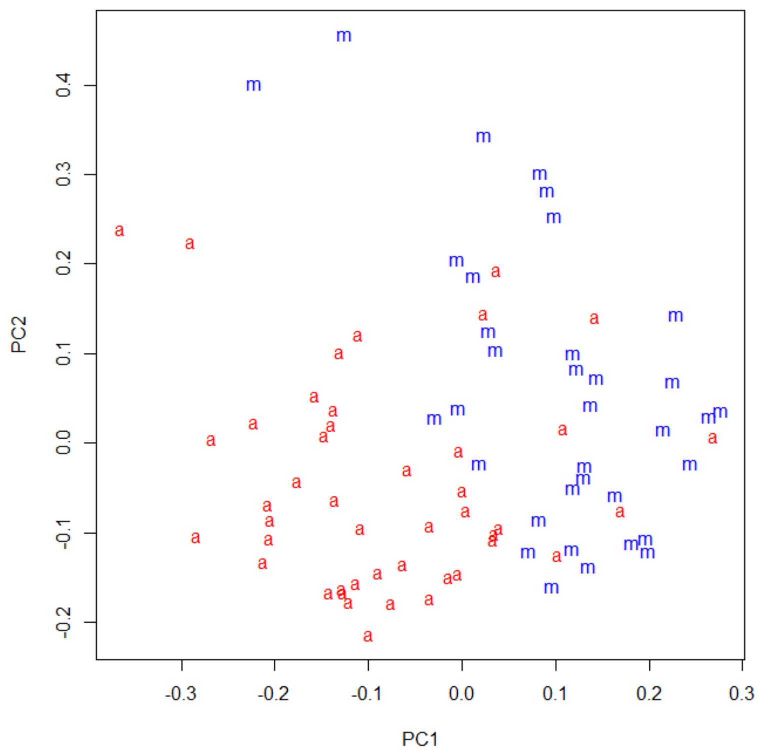
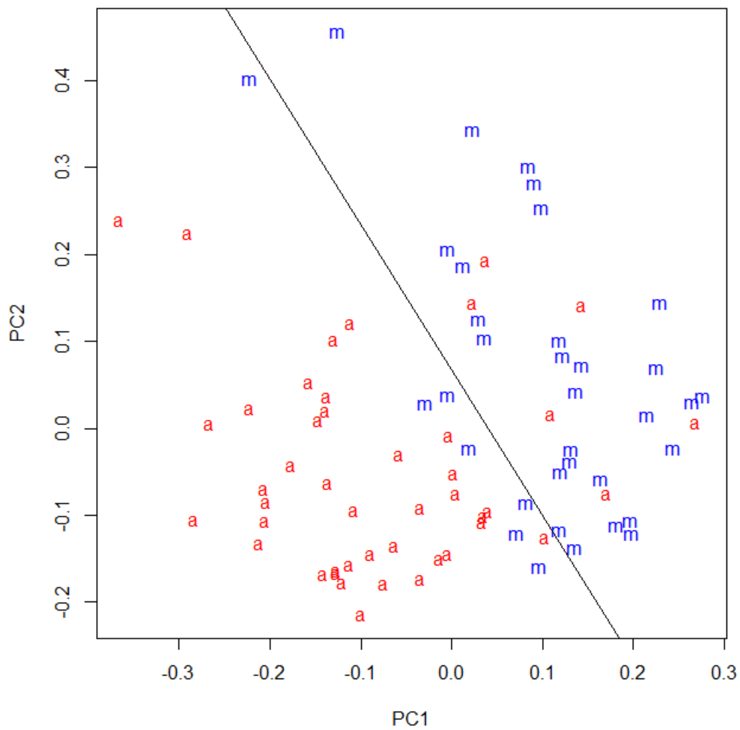


Figure : Projection of the Times stories on to the first two principal components. Music stories are marked with a blue "m", art stories with a red "a".

Total number of PCs is 80 and this is what we would expect since the training data set consist of 80 observations and 4432 predictors.

b) By looking at the above figure, we can see that almost all the words are separated into two groups, arts and music, with few been scattered around. So we can use the first two PCs for a good prediction.

c) estimate of the training error rate = `0.1625`



With the glm fit, we were able give a decision boundary for the separation of the words into arts and music. Above graph shows this linear boundary and most of the words can be identify as arts or music with an error rate of `0.1625`.

d)

```
      test.predicted.classes
       art music
art     10     2
music    5     5
```

test error rate = `(5+2)/22 = 0.3181818`

class specific error rates

art = `2/12 = 0.1666667`

music = `5/10 = 0.5`

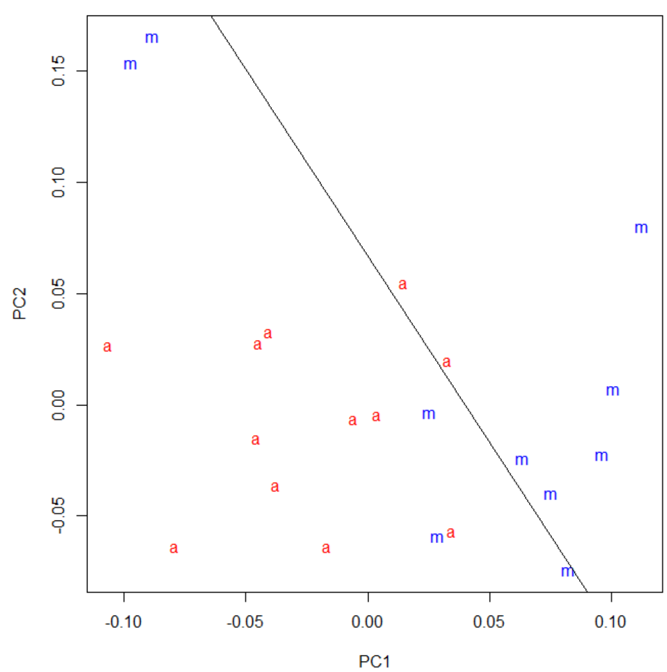


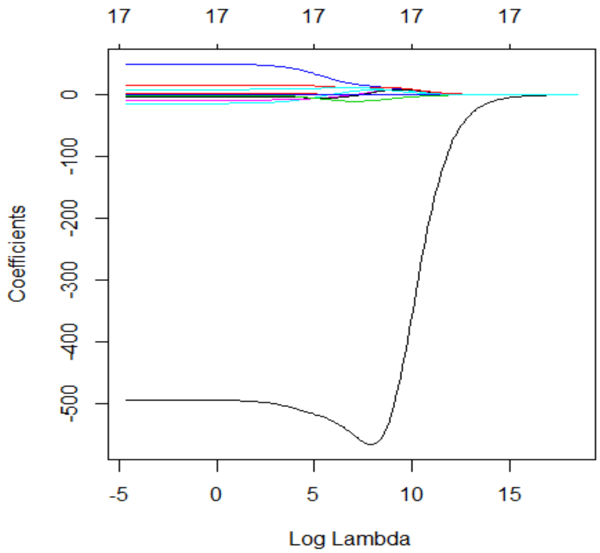
Figure shows how the 22 words in the test data set is scattered around and how the decision boundary from the glm fit is able to classify them.

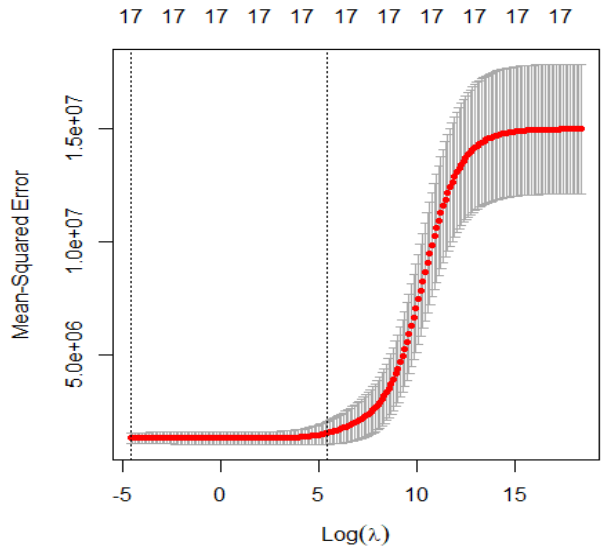
e) We can increase the number of PCs considered for the glm fit and check the test error rate with the test data set so than we can pick M as the number of PCs that give the minimum test error rate. PCR method is used when the respond variable is qualitative. Thus its best to use PCA instead of PCR.

Problem 3

a) Test error = 1276986

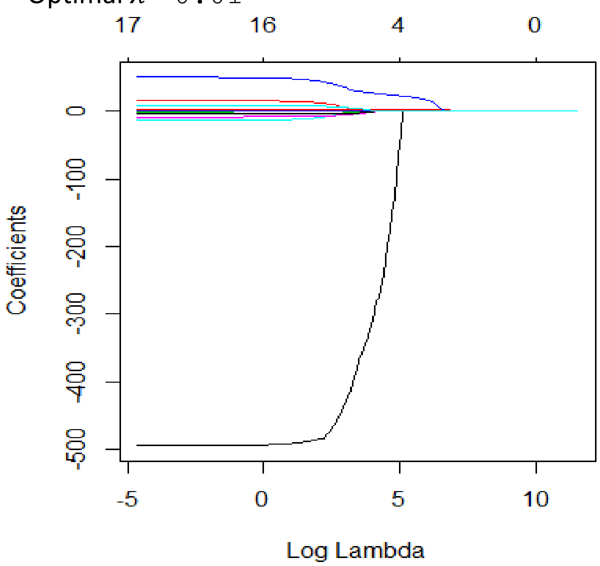
b) Optimal λ = 0.01

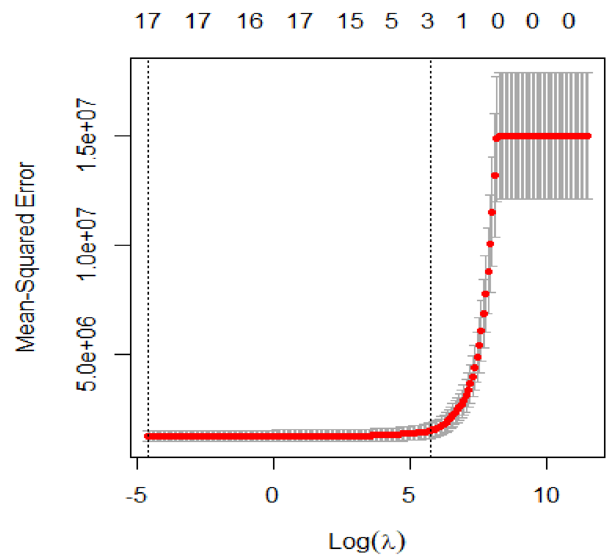




Test error = 1277006

c) Optimal λ = 0.01



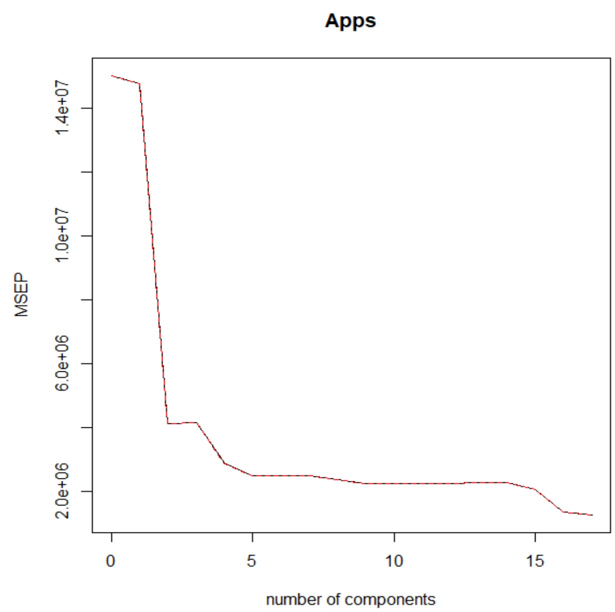


Test error = 1277099

d)

```
> MSEP(pcr.fit)
       (Intercept)    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
CV         1.5e+07   14744421  4123358  4157777  2883584  2506292  2510774  2492036  2376917  2245553
adjCV      1.5e+07   14744466  4123261  4157770  2879623  2505933  2510639  2492027  2376694  2245437
       10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV      2237022   2251874   2251604   2267449   2267624   2067302   1357315   1276987
adjCV   2236914   2251762   2251488   2267334   2267512   2066763   1357131   1276825
```

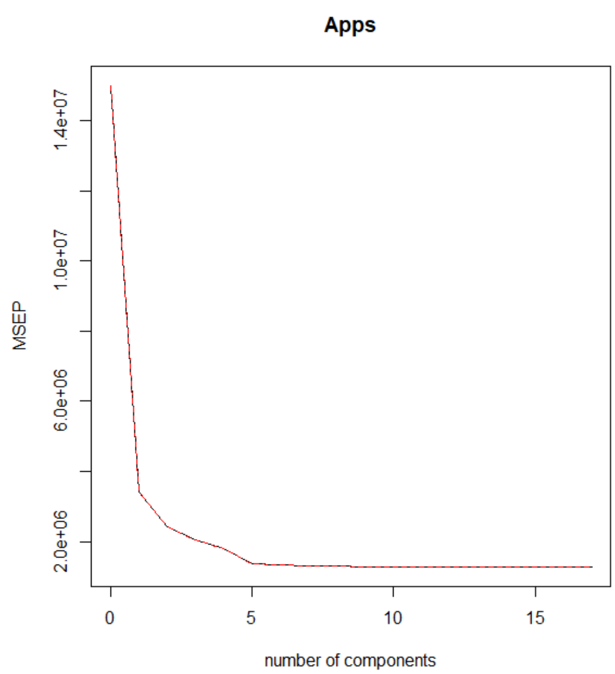
17PCs gives them minimum MSEP. So the test error = 1276987



e)

```
> MSEP(pls.fit)
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
CV         1.5e+07  3418008  2414639  2049701  1806343  1371151  1325548  1301942  1297801  1283053
adjCV      1.5e+07  3417887  2414598  2049563  1805870  1370529  1325245  1301788  1297869  1282936
       10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV      1282432   1280546   1279732   1278057   1277875   1277232   1276966   1276987
adjCV   1282273   1280398   1279576   1277894   1277712   1277070   1276804   1276825
```

16PCs gives them minimum MSEP. So the test error = 1276966



	Linear model	Ridge	Lasso	PCR	PLS
Test error (MSEP)	1276986	1277006	1277099	1276987	1276966

Although all the models gives approximately closer test errors, lowest test error gives from PLS model.

```
# Problem 1

head(state.x77)
str(state.x77)
# Extract the names of states
states <- row.names(state.x77)

# part a)
summary(state.x77)
chart.Correlation(state.x77) #matrix of scatterplot and correlation after transformation

# part b)
# Look at mean and sd
apply(state.x77, 2, mean)
apply(state.x77, 2, sd)

# part c)
# Perform PCA
pca <- prcomp(state.x77, center = T, scale = T)
names(pca)

# Get the loading matrix
pca$rotation

# Get the score matrix
dim(pca$x)
head(pca$x)

# Check the covariance matrix of the scores
round(cov(pca$x), 4)

# Display a biplot the results (shows both pc scores and loading vectors)
pca$rotation
biplot(pca, scale=0)

# Display the biplot after changing the signs of loadings and scores
pca$rotation <- -pca$rotation
pca$x <- -pca$x
pca$rotation
biplot(pca, scale=0)

# Compute the proportion of variance explained (PVE)
pc.var <- pca$sdev^2
pve <- pc.var/sum(pc.var)
pve
cumsum(pve)

#Correlations
rbind(PC1 = c(pca$rotation[,1]*pca$sdev[1], cumalative.PVE = cumsum(pve)[1]),
      PC2 = c(pca$rotation[,2]*pca$sdev[2], cumalative.PVE = cumsum(pve)[2]))

par(mfrow=c(1,2))
# Scree plot
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim =
c(0,1), type = 'b')

# Plot of cumulative PVE
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance
Explained", ylim = c(0,1), type = 'b')
par(mfrow=c(1,1))
```

```
#Correlations
rbind(PC1 = c(pca$rotation[,1]*pca$sdev[1], cumalative.PVE = cumsum(pve)[1]),
      PC2 = c(pca$rotation[,2]*pca$sdev[2], cumalative.PVE = cumsum(pve)[2]))


# problem 2

train.data <- read.csv("nyt.train.csv", header = T)
test.data <- read.csv("nyt.test.csv", header = T)

train.pca <- prcomp(train.data[, -1])
# We need to omit the first column because it contains categorical variables, and PCA
doesn't apply to them.
train.latent.sem <- train.pca$rotation
signif(sort(train.latent.sem[, 1], decreasing = TRUE)[1:30], 2)
#     theater        music            m     theaters    composers      matinee        opera
#       0.180        0.130        0.080        0.079        0.078        0.077        0.075
#      sunday      musical       jersey            p    orchestra         band    committee
#       0.067        0.065        0.064        0.064        0.062        0.061        0.060
# performance performances         east        organ        dance         hour      program
#       0.059        0.056        0.056        0.053        0.052        0.051        0.051
#      events    yesterday         will     recitals       ballet     purchase          X.d
#       0.050        0.049        0.049        0.048        0.048        0.048        0.047
#   guitarist        calif
#       0.045        0.044

signif(sort(train.latent.sem[, 1], decreasing = FALSE)[1:30], 2)
#    her         she           ms    painting   paintings      mother      cooper     artists
# -0.150      -0.140      -0.130      -0.110      -0.100      -0.092      -0.090      -0.086
#  white      images           i        said     process   sculpture     picasso    gagosian
# -0.078      -0.077      -0.071      -0.070      -0.070      -0.070      -0.068      -0.065
#    art          my      nature       image       color  sculptures        work         red
# -0.064      -0.064      -0.064      -0.061      -0.061      -0.059      -0.059      -0.058
# artist      rothko       paint photographs       paper      figure
# -0.056      -0.055      -0.055      -0.055      -0.054      -0.054


signif(sort(train.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
#   she        her     theater        said           i          ms      mother      cooper
# 0.240      0.240       0.200       0.170       0.120       0.110       0.110       0.110
#  says      opera          my        hour          id          im  production         was
# 0.089      0.084       0.084       0.082       0.081       0.079       0.075       0.075
#   mrs       play         sir    broadway      awards         you    national       garde
# 0.074      0.074       0.071       0.070       0.066       0.066       0.065       0.063
#    me     season    jonathan        week        baby    networks
# 0.062      0.062       0.062       0.060       0.059       0.059


signif(sort(train.latent.sem[, 2], decreasing = FALSE)[1:30], 2)
#   patterns      chinese          feb      chelsea    computers      diamond    europeans
#     -0.065       -0.051       -0.046       -0.046       -0.046       -0.045       -0.044
#    gallery       museum          art        heads        white        stone        views
#     -0.042       -0.041       -0.040       -0.039       -0.039       -0.039       -0.039
#    painted    recalling         soho      artists        pills       statue       newman
#     -0.039       -0.039       -0.039       -0.038       -0.037       -0.037       -0.037
#   computer compositions         grid   landscapes      spatial       images         wood
#     -0.037       -0.037       -0.037       -0.037       -0.037       -0.036       -0.035
# technology     personal
#     -0.035       -0.035
```

```
plot(train.pca$x[, 1:2],
     pch = ifelse(train.data[, "class.labels"] == "music", "m", "a"),
     col = ifelse(train.data[, "class.labels"] == "music", "blue","red"))


# part c)
train.set <- data.frame(class.labels=train.data$class.labels,
                              CP1 = train.pca$x[, 1],
                              CP2 = train.pca$x[, 2])
predictors <- train.pca$x[, 1:2]
lm.fit <- glm(class.labels ~ CP1 + CP2, data = train.set, family = "binomial")

lm.pred <- predict(lm.fit, train.set, type = 'response')
predicted.classes <- as.factor(ifelse(lm.pred < 0.5, 'art', 'music'))
mean(train.data$class.labels != predicted.classes) # training error
# [1] 0.1625



slope <- coef(lm.fit)[2]/(-coef(lm.fit)[3])
intercept <- coef(lm.fit)[1]/(-coef(lm.fit)[3])

abline(intercept , slope)


# part d)
pca.scores <- predict(train.pca, test.data)[,1:2]
test.set <- cbind.data.frame(class.labels = test.data$class.labels,
                            CP1 = pca.scores[, 1],
                            CP2 = pca.scores[, 2])



plot(pca.scores[, 1:2],
     pch = ifelse(test.data[, "class.labels"] == "music", "m", "a"),
     col = ifelse(test.data[, "class.labels"] == "music", "blue","red"))

test.lm.pred <- predict(lm.fit, test.set , type = 'response')
test.predicted.classes <- as.factor(ifelse(test.lm.pred < 0.5, 'art', 'music'))
mean(test.data$class.labels != test.predicted.classes) # test error
# [1] 0.3181818
abline(intercept , slope)

# confusion matrix for test and training data
table(test.data$class.labels, test.predicted.classes)



# problem 3

library(caret) # for cross-validation
library(ISLR)
library(glmnet) # for ridge and lasso
library(pls) # for pcr and pls
train.data <- College
str(train.data)
attach(train.data)

# part a)
lm.fit <- train(Apps ~ .,
```

```
                method = "lm",
                data = train.data,
                trControl = trainControl(method = "LOOCV"))

print(lm.fit)
# Resampling results:
#
#     RMSE      Rsquared  MAE
# 1130.038  0.91468   630.0335

1130.038^2 # = 1276986

lm.pred <- predict(lm.fit, train)
mean((Apps - lm.pred)^2) # training error
# [1] 1059279


# part b)
# Create response vector and the design matrix (without the first column of 1s)
y <- Apps
x <- model.matrix(Apps ~ ., train.data)[, -1]

grid <- 10^seq(8, -2, length = 200)

# Fit ridge regression for each lambda on the grid
ridge.out <- glmnet(x, y, alpha = 0, lambda = grid)
plot(ridge.out, xvar = "lambda")

# leave one out cross-validation
set.seed(1)
ridge.cv.out <- cv.glmnet(x, y, alpha = 0, lambda = grid, nfolds = dim(train.data)[1])
plot(ridge.cv.out)

print(ridge.cv.out)
# Measure: Mean-Squared Error
#
#     Lambda Measure     SE Nonzero
# min   0.01 1277006 258548      17
# 1se 235.43 1516697 508117      17

# Find the best value of lambda
ridge.bestlam <- ridge.cv.out$lambda.min
# [1] 0.01
# log(ridge.bestlam) = -4.60517


coef.ridge <- predict(ridge.out, type = "coefficients", s = ridge.bestlam)
#                          1
# (Intercept) -445.26830402
# PrivateYes  -494.15980608
# Accept         1.58570739
# Enroll        -0.88022903
# Top10perc     49.92174051
# Top25perc    -14.23153321
# F.Undergrad    0.05734870
# P.Undergrad    0.04444658
# Outstate      -0.08586349
# Room.Board     0.15104313
# Books          0.02090569
# Personal       0.03109799
```

```
# PhD           -8.67805769
# Terminal      -3.33091855
# S.F.Ratio     15.38992988
# perc.alumni    0.17691111
# Expend         0.07789878
# Grad.Rate      8.66799704



ridge.pred <- predict(ridge.out, s = ridge.bestlam, newx = x, type='response')
train.error.ridge <- mean((ridge.pred - y)^2) # training error
# [1] 1059279



# part c)
grid <- 10^seq(5, -2, length = 200)

# Fit lasso regression for each lambda on the grid
lasso.out <- glmnet(x, y, alpha = 1, lambda = grid)
plot(lasso.out, xvar = "lambda")

# leave one out cross-validation
set.seed(1)
lasso.cv.out <- cv.glmnet(x, y, alpha = 1, lambda = grid, nfolds = dim(train.data)[1])
plot(lasso.cv.out)

print(lasso.cv.out)
# Measure: Mean-Squared Error
#
#     Lambda Measure     SE Nonzero
# min   0.01 1277099 258652      17
# 1se  318.1 1524047 355598       3

# Find the best value of lambda
lasso.bestlam <- lasso.cv.out$lambda.min
# [1] 0.01
# log(lasso.bestlam) = -4.60517


coef.lasso <- predict(lasso.out, type = "coefficients", s = lasso.bestlam)
#                       1
# (Intercept) -445.26830402
# PrivateYes  -494.15980608
# Accept         1.58570739
# Enroll        -0.88022903
# Top10perc     49.92174051
# Top25perc    -14.23153321
# F.Undergrad    0.05734870
# P.Undergrad    0.04444658
# Outstate      -0.08586349
# Room.Board     0.15104313
# Books          0.02090569
# Personal       0.03109799
# PhD           -8.67805769
# Terminal      -3.33091855
# S.F.Ratio     15.38992988
# perc.alumni    0.17691111
# Expend         0.07789878
# Grad.Rate      8.66799704
```

```
lasso.pred <- predict(lasso.out, s = lasso.bestlam, newx = x, type='response')
train.error.lasso <- mean((lasso.pred - y)^2) # training error
# [1] 1059279


# part d)
pcr.fit <- pcr(Apps~., data=train.data, scale=T, validation="LOO")

# Scree plot
validationplot(pcr.fit, val.type="MSEP")

MSEP(pcr.fit)
#        (Intercept)   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps  9 comps  10 comps
# CV         1.5e+07  14744421  4123358  4157777  2883584  2506292  2510774  2492036
2376917  2245553   2237022
# adjCV      1.5e+07  14744466  4123261  4157770  2879623  2505933  2510639  2492027
2376694  2245437   2236914
#        11 comps  12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
# CV      2251874   2251604   2267449   2267624   2067302   1357315   1276987
# adjCV   2251762   2251488   2267334   2267512   2066763   1357131   1276825

pcr.pred <- predict(pcr.fit, train.data, ncomp=which.min(MSEP(pcr.fit)$val[1,1,]) - 1)
mean((Apps - pcr.pred)^2) # training error
# [1] 1059279



# part e)
pls.fit <- plsr(Apps~., data=train.data, scale=T, validation="LOO")

# Scree plot
validationplot(pls.fit, val.type="MSEP")

MSEP(pls.fit)
#        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps  9 comps  10 comps
# CV         1.5e+07  3418008  2414639  2049701  1806343  1371151  1325548  1301942
1297801  1283053   1282432
# adjCV      1.5e+07  3417887  2414598  2049563  1805870  1370529  1325245  1301788
1297869  1282936   1282273
#        11 comps  12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
# CV      1280546   1279732   1278057   1277875   1277232   1276966   1276987
# adjCV   1280398   1279576   1277894   1277712   1277070   1276804   1276825



pls.pred <- predict(pls.fit, train.data, ncomp=which.min(MSEP(pls.fit)$val[1,1,]) - 1)
mean((Apps - pls.pred)^2) # training error
# [1] 1059279
```