Homework 6

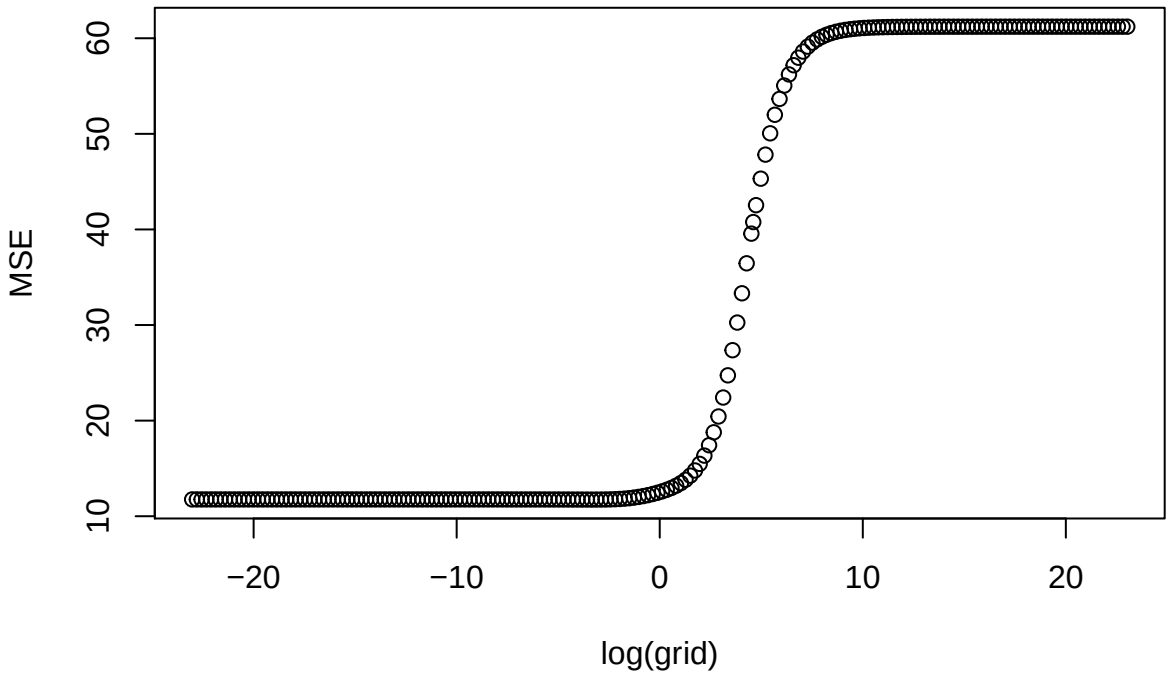
Jiao Qu A20386614, Yuan-An Liu A20375099, Zhenyu Zhang A20287371

1.

a.

```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
grid=c(10^seq(10,-10,length=200),0,100)
cv.errors=matrix(NA,k,length(grid))
for(j in 1:k){
  y = Auto$mpg
  y.train = y[folds != j]
  x = model.matrix(mpg~.-name,data=Auto)[,-1]
  x.train = x[folds != j, ]
  ridge.mod=glmnet(x.train,y.train,alpha=0,lambda=grid)
  c = 1
  for(i in grid){
    pred=predict(ridge.mod,s = i, newx = x[folds==j,])
    cv.errors[j,c]=mean((y[folds==j]-pred)^2)
    c = c + 1
  }
}

MSE = colMeans(cv.errors)
plot(x = log(grid), y = MSE)
```



b.

```
grid[which(MSE == min(MSE))]
```

```
## [1] 0.03489101
```

```
MSE[which(MSE == min(MSE))]
```

```
## [1] 11.74228
```

c. What is the MSE when lamba is 0?

```
MSE[which(grid==0)]
```

```
## [1] 11.7576
```

d. What is the MSE when lambda is 100?

```
MSE[which(grid==100)]
```

```
## [1] 40.7726
```

e. What are the coefficients when lamba is 0, 100, and optimal?

(1).lamda is optimal

```
optimal=grid[which(MSE == min(MSE))]
zero=(grid==0)
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=optimal)[1:9,]
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -10.790584625  -0.371162644  -0.001778458  -0.027854856  -0.003793697
##  acceleration          year       origin2       origin3
##  -0.055735626   0.669730755   1.742289427   2.467967476
```

(2).lambda = 0

```
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=0)[1:9,]
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -10.790584625  -0.371162644  -0.001778458  -0.027854856  -0.003793697
##  acceleration          year       origin2       origin3
##  -0.055735626   0.669730755   1.742289427   2.467967476
```

(3).lambda = 100

```
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=100)[1:9,]
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## 20.5025518407 -0.2030297429 -0.0034275976 -0.0090179149 -0.0004485592
##  acceleration          year       origin2       origin3
##  0.0631472565  0.0771735456  0.2792942264  0.5208241365
```

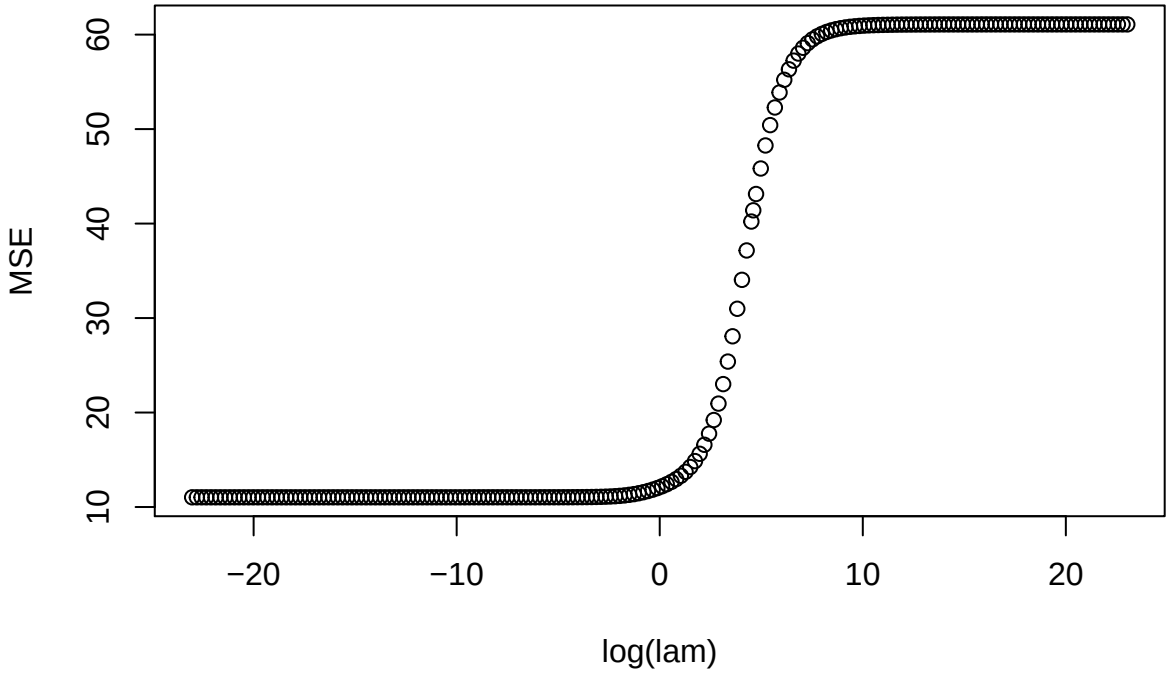
2. Use lasso regression to find a solution to predicting mpg as a function of all features except forname. Remove ???name??? from the data set and treat origin as a categorical variable

a.

```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
lam=c(10^seq(10,-10,length=200),0,100)
cv.errors=matrix(NA,k,length(grid))
```

```
for(j in 1:k){
  y = Auto$mpg
  y.train = y[folds != j]
  x = model.matrix(mpg~.-name,data=Auto)[,-1]
  x.train = x[folds != j, ]
  lasso.mod=cv.glmnet(x.train,y.train,alpha=1,lambda=lam)
  c = 1
  for(i in grid){
    pred=predict(ridge.mod,s = i, newx = x[folds==j,])
    cv.errors[j,c]=mean((y[folds==j]-pred)^2)
    c = c + 1
  }
}

MSE = colMeans(cv.errors)
plot(x = log(lam), y = MSE)
```



b.What is the lambda with the minimum MSE (optimal) and what is the MSE?

```
lam[which(MSE == min(MSE))]
```

```
##  [1] 8.309942e-07 6.593188e-07 5.231099e-07 4.150405e-07 3.292971e-07
##  [6] 2.612675e-07 2.072922e-07 1.644676e-07 1.304902e-07 1.035322e-07
## [11] 8.214344e-08 6.517340e-08 5.170920e-08 4.102658e-08 3.255089e-08
## [16] 2.582619e-08 2.049075e-08 1.625756e-08 1.289890e-08 1.023411e-08
## [21] 8.119845e-09 6.442364e-09 5.111433e-09 4.055461e-09 3.217642e-09
## [26] 2.552908e-09 2.025502e-09 1.607053e-09 1.275051e-09 1.011638e-09
## [31] 8.026434e-10 6.368250e-10 5.052631e-10 4.008806e-10 3.180626e-10
## [36] 2.523539e-10 2.002200e-10 1.588565e-10 1.260383e-10 1.000000e-10
## [41] 0.000000e+00
```

```
MSE[which(MSE == min(MSE))]
```

```
##  [1] 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283
##  [9] 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283
## [17] 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283
## [25] 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283
## [33] 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283 11.0283
## [41] 11.0283
```

c.What is the MSE when lamba is 0?

```
MSE[which(lam==0)]
```

```
## [1] 11.0283
```

d.What is the MSE when lambda is 100?

```
MSE[which(lam==100)]
```

```
## [1] 41.40317
```

e.What are the coefficients when lamba is 0, 100, and optimal?

(1).lamda is optimal

```
optimal=lam[which(MSE == min(MSE))]
out=glmnet(x,y,alpha=1)
predict(out,type="coefficients",s=optimal)[1:9,]
```

```
## 9 x 41 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 41 column names '1', '2', '3' ... ]]
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
```

```
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
```

```
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923 -17.924808923 -17.924808923 -17.924808923
## cylinders     -0.424254454  -0.424254454  -0.424254454  -0.424254454
## displacement   0.021705860   0.021705860   0.021705860   0.021705860
## horsepower    -0.017079208  -0.017079208  -0.017079208  -0.017079208
## weight        -0.006637266  -0.006637266  -0.006637266  -0.006637266
## acceleration   0.075266651   0.075266651   0.075266651   0.075266651
## year           0.774468089   0.774468089   0.774468089   0.774468089
## origin2        2.558842399   2.558842399   2.558842399   2.558842399
## origin3        2.789758687   2.789758687   2.789758687   2.789758687
##
## (Intercept)  -17.924808923
## cylinders     -0.424254454
## displacement   0.021705860
## horsepower    -0.017079208
## weight        -0.006637266
## acceleration   0.075266651
## year           0.774468089
## origin2        2.558842399
## origin3        2.789758687
```

(2).lambda = 0

```
out=glmnet(x,y,alpha=1)
predict(out,type="coefficients",s=0)[1:9,]
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -17.924808923  -0.424254454   0.021705860  -0.017079208  -0.006637266
##  acceleration          year       origin2       origin3
##   0.075266651   0.774468089   2.558842399   2.789758687
```

(3).lambda = 100

```
out=glmnet(x,y,alpha=1)
predict(out,type="coefficients",s=100)[1:9,]
```

```
##  (Intercept)    cylinders displacement   horsepower       weight
##     23.44592      0.00000      0.00000      0.00000      0.00000
## acceleration         year      origin2      origin3
##      0.00000      0.00000      0.00000      0.00000
```

f.Which generated a lower MSE, ridge or lasso?

```
Lasso regression generated a lower MSE.
```

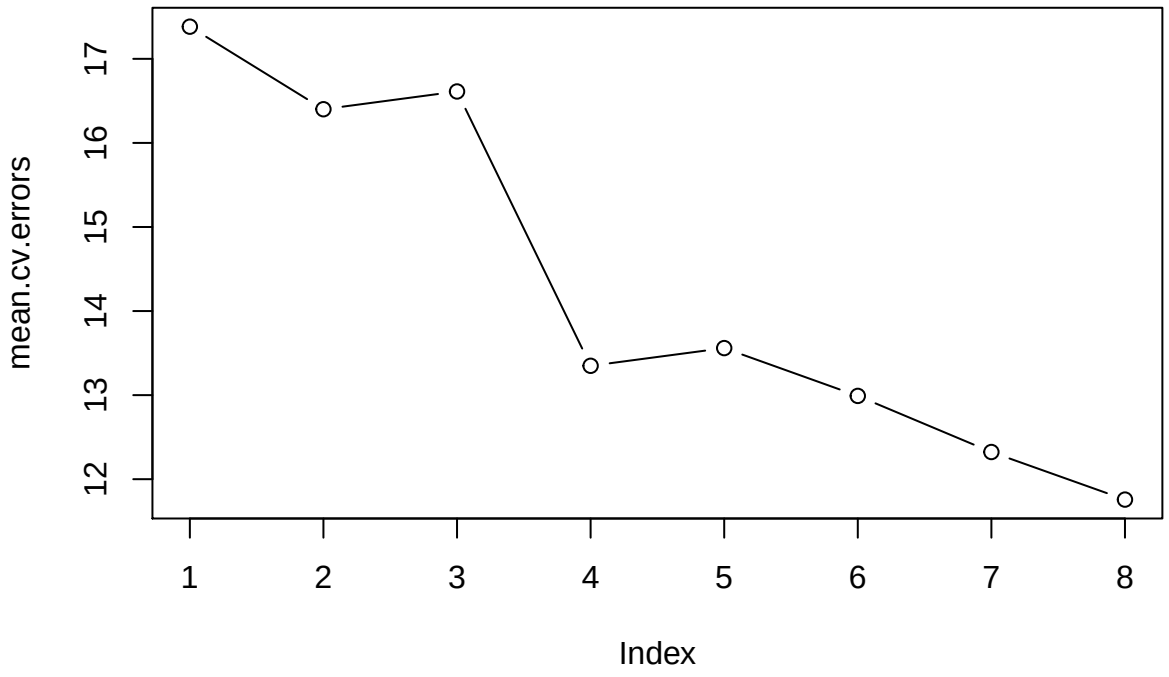
g.Refer to homework 5. Do the MSEs in either ridge or lasso improve over those in homework 5?

```
Yes. Compared with homework5, both ridge and lasso decrease the MSE. In hw5, the best MSE is 11.75048.
```

3.

a. Plot the MSE as a function of number of principle components.

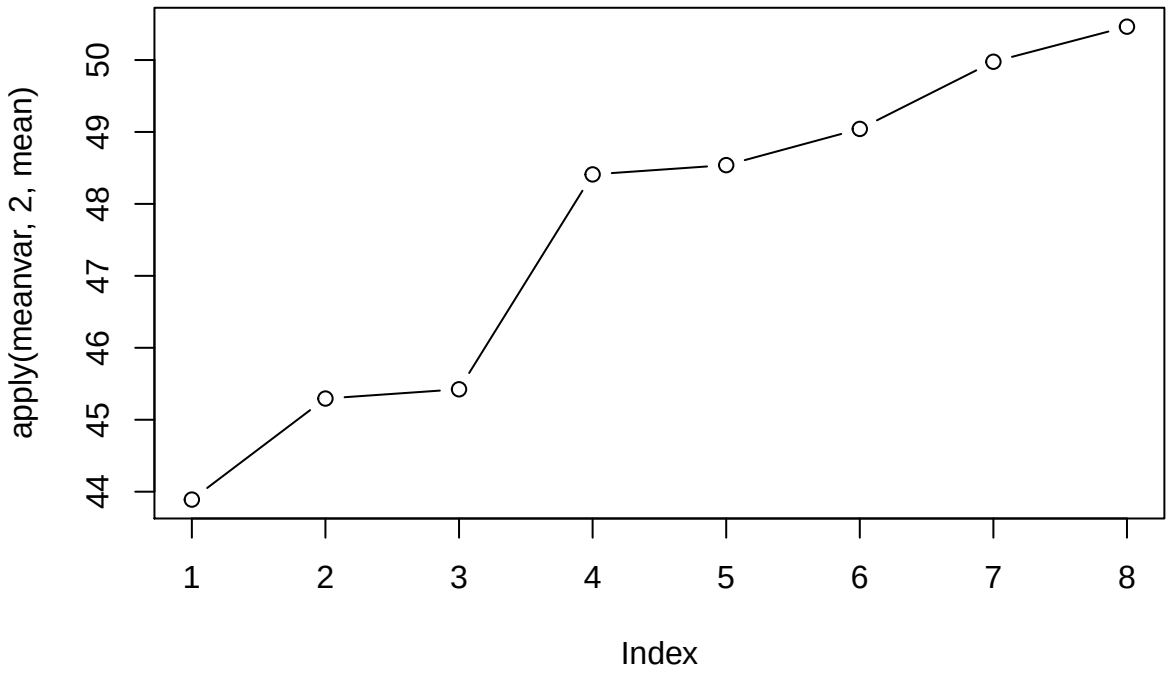
```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
cv.errors=matrix(NA,k,8,dimnames = list(NULL, paste(1:8)))
for(j in 1:k){
  pcr.fit=pcr(mpg~.-name,data=Auto[folds!=j,],scale=TRUE,validation="CV")
  for(i in 1:8){
    pred=predict(pcr.fit,Auto[folds==j,],ncomp=i)
    cv.errors[j,i]=mean((Auto$mpg[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
plot(mean.cv.errors,type="b")
```



b. Plot the variance explained as a function of number of principle components.

```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
meanvar=matrix(NA,k,8,dimnames = list(NULL, paste(1:8)))
for(j in 1:k){
  pcr.fit=pcr(mpg~.-name,data=Auto[folds!=j,],scale=TRUE,validation="CV")
  for(i in 1:8){
    pred=predict(pcr.fit,Auto[folds==j,],ncomp=i)
    meanvar[j,i]=var(pred)
  }
}

plot(apply(meanvar,2,mean),type="b")
```



c. What is the number of principle components in the best (lowest MSE) model?

```
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 17.38255 16.40074 16.61192 13.34885 13.55936 12.99063 12.32323 11.75760
```

The number is 8.

d. What is its MSE?

```
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 17.38255 16.40074 16.61192 13.34885 13.55936 12.99063 12.32323 11.75760
```

11.75760

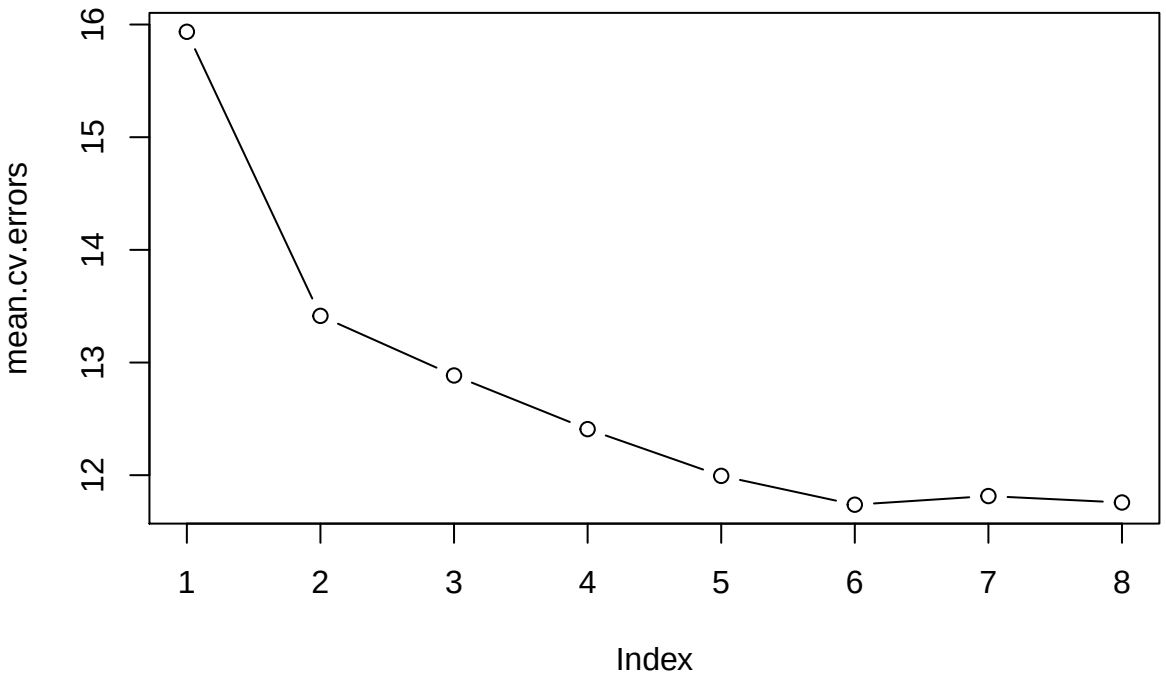
e. Is there another number of principle components you might consider? Why?

```
No,because the difference among each component is obvious.
```

4

. #a. Plot the MSE as a function of number of random segments.

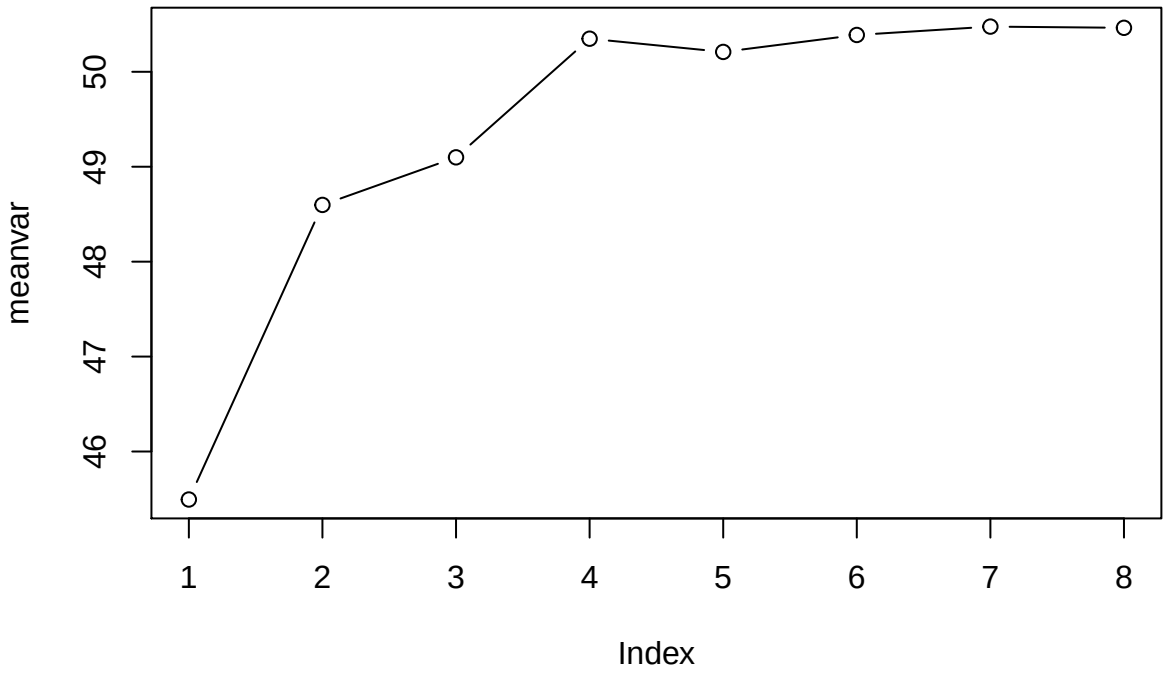
```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
cv.errors=matrix(NA,k,8,dimnames = list(NULL, paste(1:8)))
for(j in 1:k){
  pls.fit=plsr(mpg~.-name,data=Auto[folds!=j,],scale=TRUE,validation="CV")
  for(i in 1:8){
    pred=predict(pls.fit,Auto[folds==j,],ncomp=i)
    cv.errors[j,i]=mean((Auto$mpg[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
plot(mean.cv.errors,type="b")
```



b. Plot the variance explained as a function of number of random segments.

```
k=3
set.seed(1)
Auto$origin=factor(Auto$origin)
folds=sample(1:k,nrow(Auto),replace = TRUE)
meanvar=matrix(NA,k,8,dimnames = list(NULL, paste(1:8)))
for(j in 1:k){
  pls.fit=plsr(mpg~.-name,data=Auto[folds!=j,],scale=TRUE,validation="CV")
  for(i in 1:8){
```

```
    pred=predict(pls.fit,Auto[folds==j,],ncomp=i)
    meanvar[j,i]=var(pred)
  }
}
meanvar=apply(meanvar,2,mean)
plot(meanvar,type="b")
```



c. What is the number of random segments in the best (lowest MSE) model?

```
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 15.93557 13.41406 12.88512 12.40875 11.99390 11.73813 11.81477 11.75760
```

The number is 6.

d. What is its MSE?

```
11.73813
```

e. Is there another number of random segments you might consider? Why?

```
The number 7.Because the difference is very close. Although 8 is much closer than 7, the 7's dimension
```

5.

```
Auto$origin=factor(Auto$origin)
df=Auto[,c(1, 2, 3, 4, 5, 6, 7)]
pca = function (x, retx = TRUE, center = TRUE, scale. = TRUE, tol = NULL,
  ...)
{
  chkDots(...)
  x=as.matrix(x)
  x=scale(x, center = center, scale = scale.)
  cen=attr(x, "scaled:center")
  sc=attr(x, "scaled:scale")
  if (any(sc == 0))
    stop("cannot rescale a constant/zero column to unit variance")
  s=svd(x, nu = 0)
  s$d=s$d/sqrt(max(1, nrow(x) - 1))
  if (!is.null(tol)) {
    rank=sum(s$d > (s$d[1L] * tol))
    if (rank < ncol(x)) {
      s$v=s$v[, 1L:rank, drop = FALSE]
      s$d=s$d[1L:rank]
    }
  }
  dimnames(s$v)=list(colnames(x), paste0("PC", seq_len(ncol(s$v))))
  r=list(sdev = s$d, rotation = s$v, center = if (is.null(cen)) FALSE else cen,
    scale = if (is.null(sc)) FALSE else sc)
  if (retx)
    r$x=x %*% s$v
  class(r)="prcomp"
  r
}

df.pca = pca(df)
```

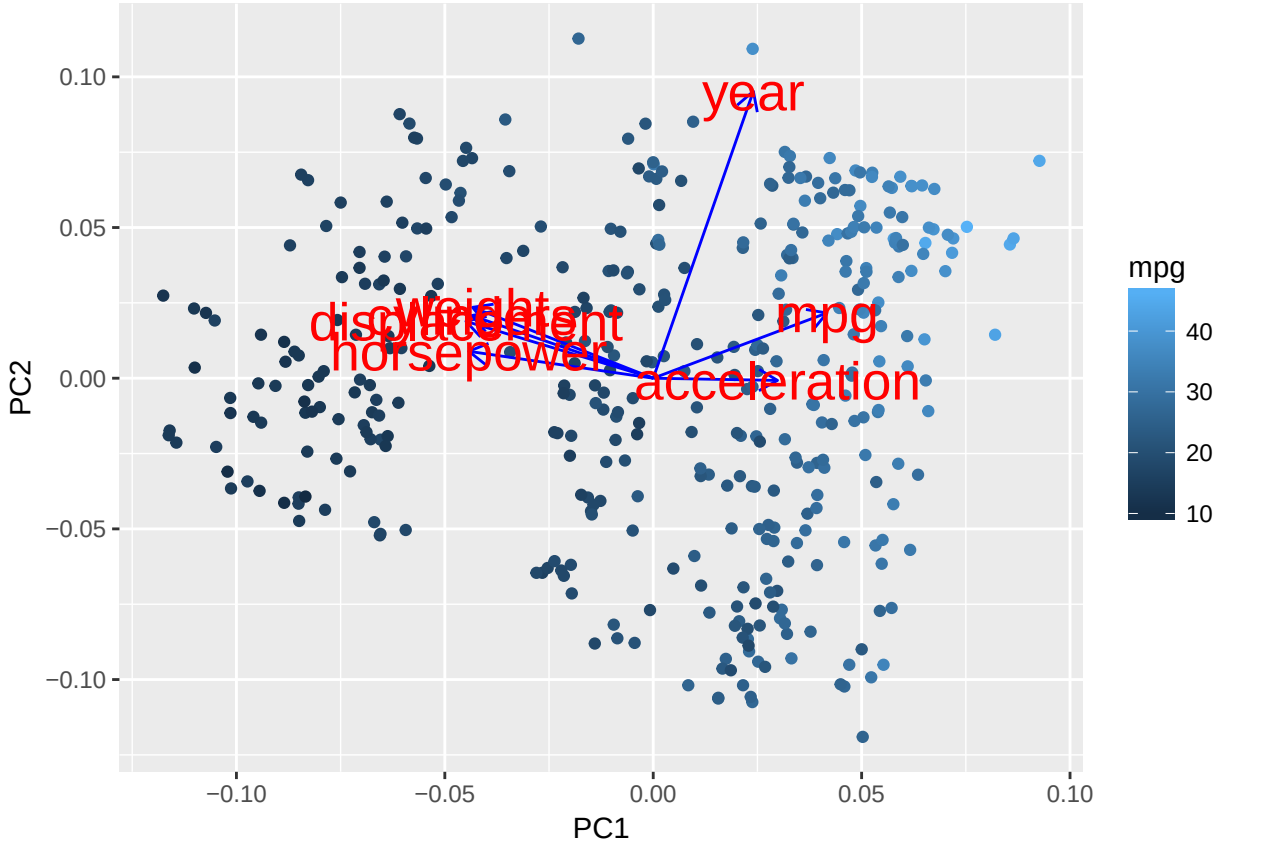
b.

```
summary(df.pca)
```

```
## Importance of components%s:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.2384 0.9304 0.8535 0.42885 0.34917 0.23293
## Proportion of Variance 0.7158 0.1237 0.1041 0.02627 0.01742 0.00775
## Cumulative Proportion  0.7158 0.8395 0.9435 0.96979 0.98721 0.99496
##                           PC7
## Standard deviation     0.18786
## Proportion of Variance 0.00504
## Cumulative Proportion  1.00000
```

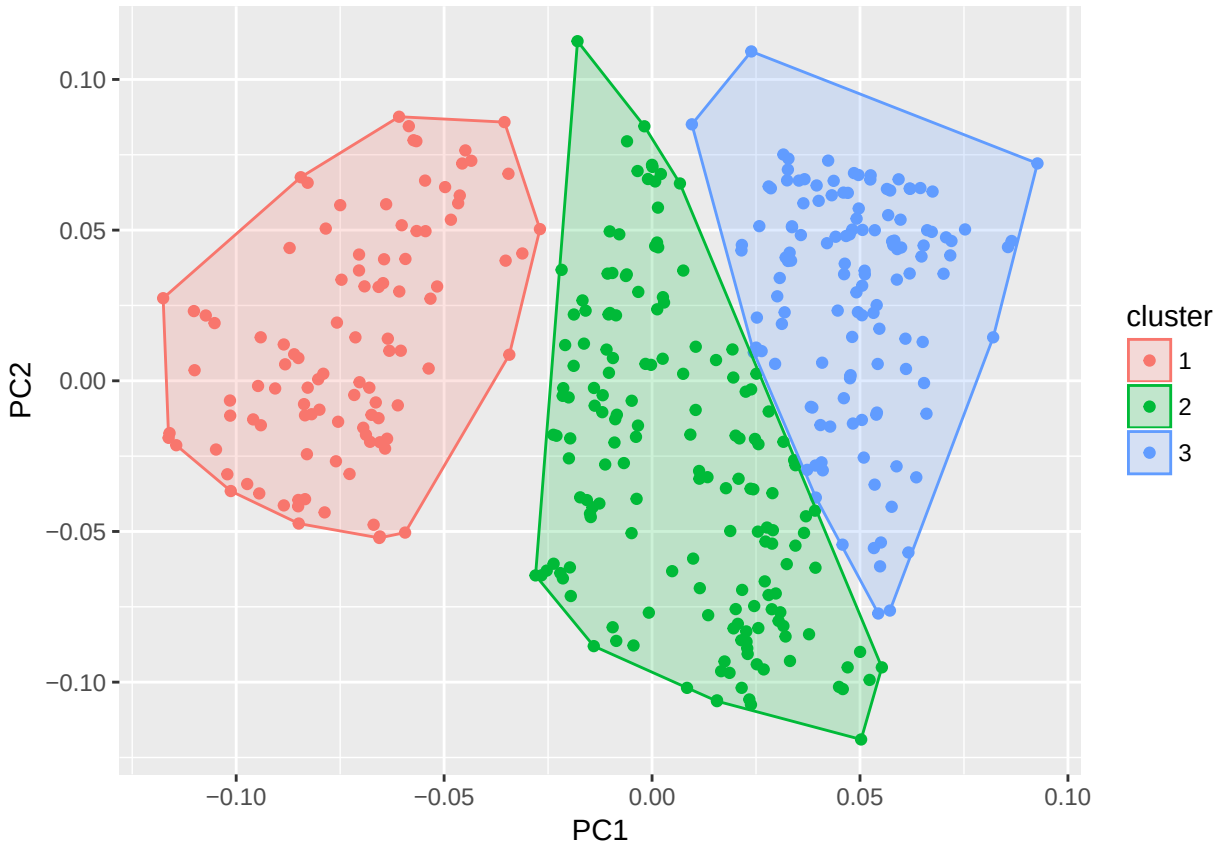
c.

```
autoplot(df.pca,data=df,colour = 'mpg',
         loadings = TRUE,loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 7)
```



d.

```
autoplot(fanny(df.pca$x[,1:2], 3), frame = TRUE)
```



cluster 1: factors in this cluster have a negative correlation with mpg. e.g:High mpg cars with low horsepower,cylinders,displacement,weight

cluster 2: factor in this cluster basically doesn't influnce the mpg.

cluster 3: factor in this cluster have a positive correlation with mpg. e.g:High mpg cars with high acceleration.