Hamed\_HW2

# Load the cpus dataset from the MASS package

library(MASS)  
library(leaps)  
data(cpus)  
attach(cpus)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

# Use syct, mmin , mmax , cach , chmin, chmax as the predictors (independent variables) to predict performance (perf)

# Perform best subset selection in order to choose the best predictors from the above predictors

regfit.full <- regsubsets(perf ~ syct + mmin + mmax + cach + chmin + chmax,data=cpus,nvmax = 6)  
summary(regfit.full)

## Subset selection object  
## Call: regsubsets.formula(perf ~ syct + mmin + mmax + cach + chmin +   
## chmax, data = cpus, nvmax = 6)  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 6  
## Selection Algorithm: exhaustive  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " "\*" "\*" " " " " "\*"   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

# What is the best model obtained according to Cp, BIC, and adjusted R2?

reg.summary\_sub <- summary(regfit.full)  
reg.summary\_sub$cp

## [1] 176.563616 95.808585 28.225948 10.977588 5.099604 7.000000

reg.summary\_sub$bic

## [1] -274.7146 -320.4675 -370.5300 -383.5185 -386.1684 -380.9290

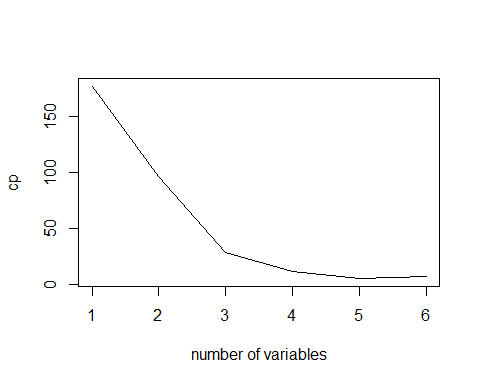
reg.summary\_sub$adjr2

## [1] 0.7435259 0.7981760 0.8444189 0.8567846 0.8614788 0.8608616

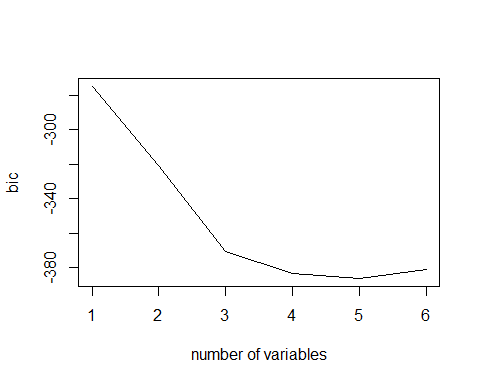
# Show some plots to provide evidence for your answer,

# and report the coefficients of the best model obtained for each criteria.

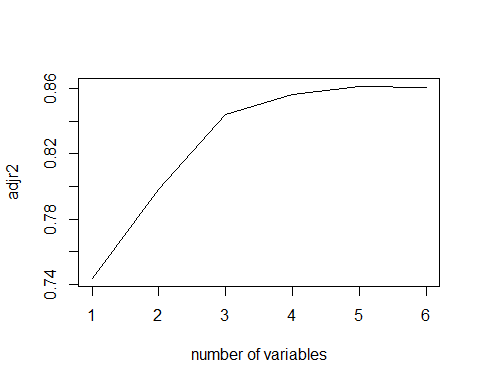
plot(reg.summary\_sub$cp,xlab="number of variables" ,ylab="cp" ,type="l")



plot(reg.summary\_sub$bic,xlab="number of variables" ,ylab="bic" ,type="l")



plot(reg.summary\_sub$adjr2,xlab="number of variables" ,ylab="adjr2" ,type="l")



# Repeat using forward stepwise selection and also using backwards stepwise selection.

# How does your answer compare to the best subset results?

# Forward Stepwise Selection  
regfit.fwd <- regsubsets(perf ~ syct + mmin + mmax + cach + chmin + chmax,data=cpus,nvmax = 6, method = "forward")  
summary(regfit.fwd)

## Subset selection object  
## Call: regsubsets.formula(perf ~ syct + mmin + mmax + cach + chmin +   
## chmax, data = cpus, nvmax = 6, method = "forward")  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 6  
## Selection Algorithm: forward  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " "\*" "\*" "\*" " " " "   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

reg.summary\_fwd <- summary(regfit.fwd)  
reg.summary\_fwd$cp

## [1] 176.563616 95.808585 56.684812 10.977588 5.099604 7.000000

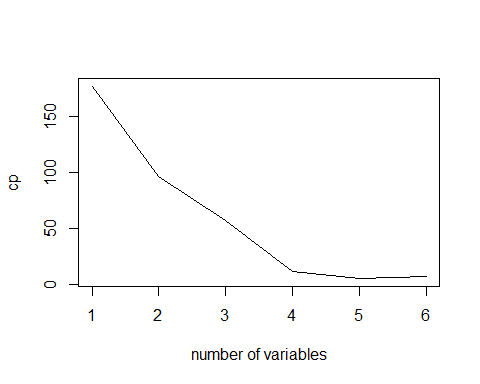
reg.summary\_fwd$bic

## [1] -274.7146 -320.4675 -346.0709 -383.5185 -386.1684 -380.9290

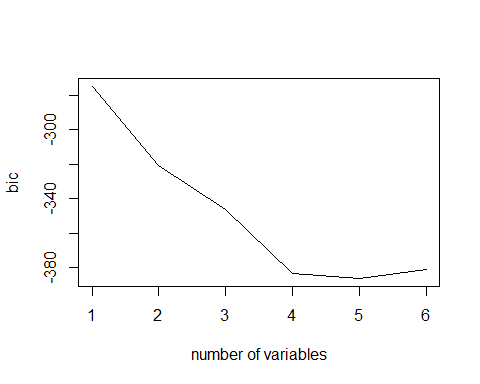
reg.summary\_fwd$adjr2

## [1] 0.7435259 0.7981760 0.8251032 0.8567846 0.8614788 0.8608616

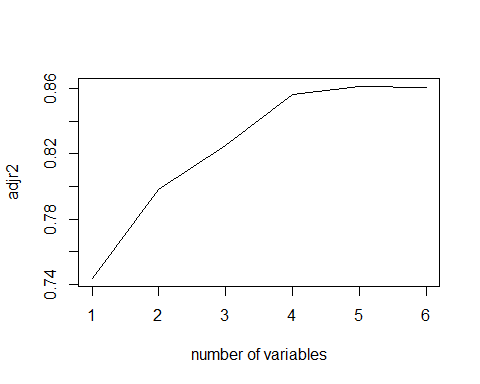
plot(reg.summary\_fwd$cp,xlab="number of variables" ,ylab="cp" ,type="l")



plot(reg.summary\_fwd$bic,xlab="number of variables" ,ylab="bic" ,type="l")



plot(reg.summary\_fwd$adjr2,xlab="number of variables" ,ylab="adjr2" ,type="l")



# Backward Stepwise Selection  
regfit.bwd <- regsubsets(perf ~ syct + mmin + mmax + cach + chmin + chmax,data=cpus,nvmax = 6, method = "backward")  
summary(regfit.bwd)

## Subset selection object  
## Call: regsubsets.formula(perf ~ syct + mmin + mmax + cach + chmin +   
## chmax, data = cpus, nvmax = 6, method = "backward")  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 6  
## Selection Algorithm: backward  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " "\*" " " " " " " " "   
## 2 ( 1 ) " " "\*" " " " " " " "\*"   
## 3 ( 1 ) " " "\*" "\*" " " " " "\*"   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

reg.summary\_bwd <- summary(regfit.fwd)  
reg.summary\_bwd$cp

## [1] 176.563616 95.808585 56.684812 10.977588 5.099604 7.000000

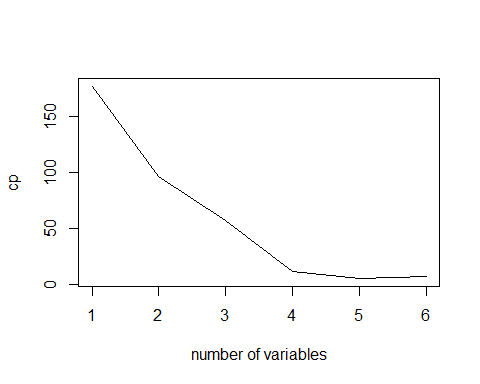
reg.summary\_bwd$bic

## [1] -274.7146 -320.4675 -346.0709 -383.5185 -386.1684 -380.9290

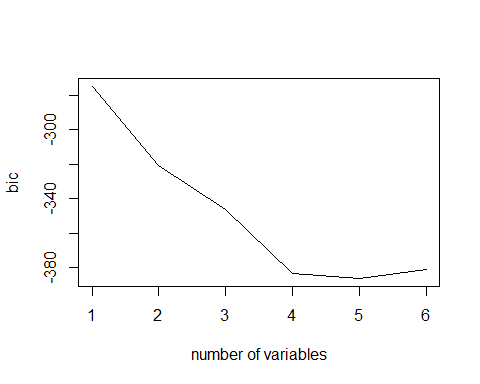
reg.summary\_bwd$adjr2

## [1] 0.7435259 0.7981760 0.8251032 0.8567846 0.8614788 0.8608616

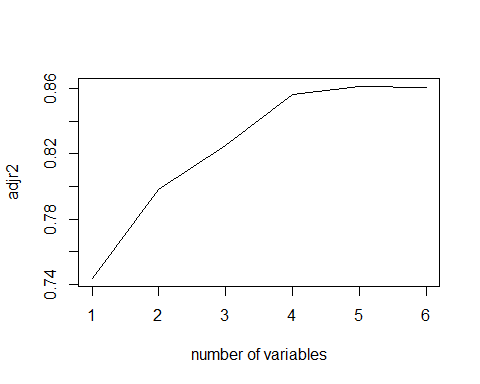
plot(reg.summary\_bwd$cp,xlab="number of variables" ,ylab="cp" ,type="l")



plot(reg.summary\_bwd$bic,xlab="number of variables" ,ylab="bic" ,type="l")



plot(reg.summary\_bwd$adjr2,xlab="number of variables" ,ylab="adjr2" ,type="l")



library(ISLR)  
data(College)  
attach(College)  
head(College)

## Private Apps Accept Enroll Top10perc Top25perc  
## Abilene Christian University Yes 1660 1232 721 23 52  
## Adelphi University Yes 2186 1924 512 16 29  
## Adrian College Yes 1428 1097 336 22 50  
## Agnes Scott College Yes 417 349 137 60 89  
## Alaska Pacific University Yes 193 146 55 16 44  
## Albertson College Yes 587 479 158 38 62  
## F.Undergrad P.Undergrad Outstate Room.Board Books  
## Abilene Christian University 2885 537 7440 3300 450  
## Adelphi University 2683 1227 12280 6450 750  
## Adrian College 1036 99 11250 3750 400  
## Agnes Scott College 510 63 12960 5450 450  
## Alaska Pacific University 249 869 7560 4120 800  
## Albertson College 678 41 13500 3335 500  
## Personal PhD Terminal S.F.Ratio perc.alumni Expend  
## Abilene Christian University 2200 70 78 18.1 12 7041  
## Adelphi University 1500 29 30 12.2 16 10527  
## Adrian College 1165 53 66 12.9 30 8735  
## Agnes Scott College 875 92 97 7.7 37 19016  
## Alaska Pacific University 1500 76 72 11.9 2 10922  
## Albertson College 675 67 73 9.4 11 9727  
## Grad.Rate  
## Abilene Christian University 60  
## Adelphi University 56  
## Adrian College 54  
## Agnes Scott College 59  
## Alaska Pacific University 15  
## Albertson College 55

library(caret)

# Predict the number of applications received using the other variables in the College data set in library ISLR

colnames(College)

## [1] "Private" "Apps" "Accept" "Enroll" "Top10perc"   
## [6] "Top25perc" "F.Undergrad" "P.Undergrad" "Outstate" "Room.Board"   
## [11] "Books" "Personal" "PhD" "Terminal" "S.F.Ratio"   
## [16] "perc.alumni" "Expend" "Grad.Rate"

# (a) Split the data set into a training set and a test set using caret library and

# fit each of the following models using caret and ten fold cross validation.

intrain <- createDataPartition(College$Apps,p=0.75,list = FALSE)  
train1 <- College[intrain,]  
test1 <- College[-intrain,]  
  
trctrl <- trainControl(method= "cv", number=10)  
  
nrow(train1)

## [1] 585

nrow(test1)

## [1] 192

# (b) Fit a linear model using ordinary least squares on the training set,

# and report the test mean squared error obtained.

ols <- train(Apps ~., data = train1,   
 trControl=trctrl,  
 preProcess=c('scale','center'))  
ols

## Random Forest   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: scaled (17), centered (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 528, 527, 525, 526, 527, 527, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 2 1498.896 0.8799545 716.3014  
## 9 1268.750 0.9032436 522.0455  
## 17 1268.218 0.8974584 517.7527  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 17.

# Report test mean squared  
ols$results

## mtry RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 2 1498.896 0.8799545 716.3014 1116.489 0.08867297 183.5196  
## 2 9 1268.750 0.9032436 522.0455 1141.621 0.10361832 162.3887  
## 3 17 1268.218 0.8974584 517.7527 1145.685 0.11315247 176.5862

ols$results$RMSE^2

## [1] 2246690 1609727 1608376

cat("test mean squared error")

## test mean squared error

# Predict for the test dataset  
ln\_pred <-predictions <- predict(ols, newdata= test1)  
# Mean squared error in the test dataset  
ln\_mse <- mean(( predictions - test1$Apps)^2)  
ln\_mse

## [1] 1195690

# (c) Fit a ridge regression model on the training set, with λ chosen by cross-validation.

# Report the test mean squared error obtained. Report the value of λ used in the mode

ridge\_fit <- train(Apps ~., data = train1,  
 method="ridge",  
 trControl=trctrl,  
 preProcess=c('scale','center'))  
ridge\_fit

## Ridge Regression   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: scaled (17), centered (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 527, 527, 526, 526, 526, 527, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0e+00 1036.263 0.9248218 606.4225  
## 1e-04 1036.334 0.9248477 606.3501  
## 1e-01 1197.058 0.9188929 700.9011  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.

ridgeGrid <- data.frame(lambda = seq(0, .2, length = 15))  
ridge\_fit <- train(Apps~.,data=train1,  
 method="ridge",  
 tuneGrid=ridgeGrid,   
 trControl=trctrl,  
 preProcess=c('scale','center'))   
ridge\_fit

## Ridge Regression   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: scaled (17), centered (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 529, 527, 527, 525, 526, 527, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.00000000 1067.551 0.9281832 618.2244  
## 0.01428571 1065.748 0.9311674 614.8620  
## 0.02857143 1076.910 0.9315776 620.8149  
## 0.04285714 1093.498 0.9309338 633.0227  
## 0.05714286 1112.704 0.9297919 649.3933  
## 0.07142857 1133.205 0.9284120 666.8707  
## 0.08571429 1154.331 0.9269287 685.0552  
## 0.10000000 1175.742 0.9254146 703.2052  
## 0.11428571 1197.265 0.9239095 721.0767  
## 0.12857143 1218.820 0.9224348 739.2864  
## 0.14285714 1240.375 0.9210017 757.5451  
## 0.15714286 1261.919 0.9196151 775.3353  
## 0.17142857 1283.457 0.9182766 793.0985  
## 0.18571429 1304.996 0.9169857 810.6974  
## 0.20000000 1326.546 0.9157409 828.3609  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.01428571.

# Generate predictions with test dataset  
ridge\_pred <- predict(ridge\_fit, newdata = test1)   
head(ridge\_pred)

## Adrian College Albertson College   
## 1264.7578 689.1371   
## Alderson-Broaddus College Alma College   
## 638.8626 1795.4662   
## American International College Amherst College   
## 1144.8377 3424.4456

# Calculate Mean Square Error (MSE) ridge\_mse  
ridge\_mse <- mean((ridge\_pred - test1$Apps)^2)   
ridge\_mse

## [1] 1469828

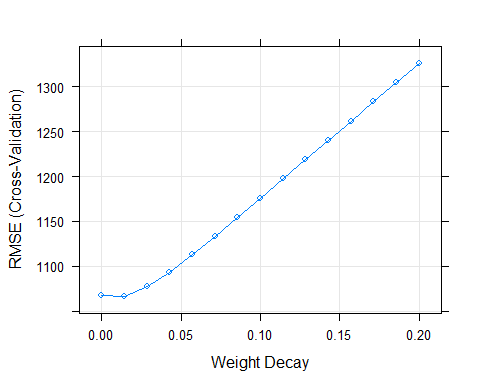
# Predicting the ridge\_fit   
predict(ridge\_fit$finalModel, type='coef', mode='norm')$coefficients

## PrivateYes Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad  
## 0 0.00000 0.000 0.0000 0.0000 0.00000 0.00000 0.00000  
## 1 0.00000 3107.494 0.0000 0.0000 0.00000 0.00000 0.00000  
## 2 0.00000 3406.305 0.0000 298.8110 0.00000 0.00000 0.00000  
## 3 0.00000 3516.470 0.0000 355.4996 0.00000 0.00000 0.00000  
## 4 -85.10238 3532.268 0.0000 393.1734 0.00000 0.00000 0.00000  
## 5 -123.23259 3533.393 0.0000 405.9061 0.00000 0.00000 0.00000  
## 6 -132.88931 3532.487 0.0000 415.1257 0.00000 0.00000 0.00000  
## 7 -134.14308 3532.690 0.0000 416.3824 0.00000 0.00000 0.00000  
## 8 -142.35585 3536.760 0.0000 438.9648 0.00000 0.00000 0.00000  
## 9 -145.07089 3537.104 0.0000 442.9592 0.00000 0.00000 0.00000  
## 10 -159.86816 3534.901 0.0000 469.7317 0.00000 0.00000 14.08922  
## 11 -166.94460 3534.157 0.0000 481.2025 0.00000 0.00000 20.28294  
## 12 -186.33993 3613.729 -104.3296 508.8359 0.00000 0.00000 40.47082  
## 13 -201.04461 3675.025 -180.6745 571.8337 -50.46454 0.00000 54.97060  
## 14 -214.09929 3730.100 -250.4806 627.3609 -94.90267 0.00000 66.62224  
## 15 -221.36345 3760.528 -289.0791 657.8563 -119.50862 0.00000 73.03312  
## 16 -223.09537 3772.408 -326.0164 670.9452 -131.10416 25.06694 74.22332  
## 17 -226.55307 3803.837 -424.9786 706.0117 -161.82271 91.40697 77.57179  
## Outstate Room.Board Books Personal PhD Terminal S.F.Ratio  
## 0 0.00000 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 1 0.00000 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 2 0.00000 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 3 0.00000 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 4 0.00000 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000  
## 5 0.00000 26.59073 0.000000 0.000000 0.000000 0.000000 0.000000  
## 6 0.00000 36.28786 0.000000 0.000000 0.000000 0.000000 0.000000  
## 7 0.00000 37.25613 0.000000 0.000000 -2.073079 0.000000 0.000000  
## 8 -34.84148 63.45019 0.000000 0.000000 -30.770448 0.000000 0.000000  
## 9 -44.72034 69.28797 0.000000 0.000000 -37.773772 0.000000 0.000000  
## 10 -105.78909 105.18079 0.000000 0.000000 -83.405011 0.000000 0.000000  
## 11 -131.38425 121.11433 0.000000 0.000000 -96.816664 -8.138322 0.000000  
## 12 -174.83996 134.30637 0.000000 0.000000 -114.322952 -17.710458 0.000000  
## 13 -206.03288 144.10892 0.000000 0.000000 -126.019576 -18.751976 0.000000  
## 14 -232.94781 153.21363 0.000000 7.027112 -136.107651 -19.685471 0.000000  
## 15 -247.70460 158.01594 1.275852 10.647493 -141.450186 -20.425112 0.000000  
## 16 -252.83583 159.82638 1.832312 11.633432 -143.868318 -21.301852 0.000000  
## 17 -266.30652 164.98313 3.093920 14.757475 -150.815095 -23.655286 6.612105  
## perc.alumni Expend Grad.Rate  
## 0 0.00000 0.0000 0.000000  
## 1 0.00000 0.0000 0.000000  
## 2 0.00000 0.0000 0.000000  
## 3 0.00000 81.9782 0.000000  
## 4 0.00000 148.0035 0.000000  
## 5 0.00000 160.2378 0.000000  
## 6 -14.68213 167.1304 0.000000  
## 7 -15.29255 167.9205 0.000000  
## 8 -19.41647 189.2174 0.000000  
## 9 -21.44036 195.3512 5.159169  
## 10 -33.98853 233.3089 40.423469  
## 11 -38.90375 249.7533 55.454746  
## 12 -41.64644 271.4519 73.750186  
## 13 -42.08003 280.5936 88.472860  
## 14 -41.60042 288.2754 102.198266  
## 15 -41.24444 292.4828 109.825551  
## 16 -40.77802 294.7074 113.448015  
## 17 -39.04298 303.8768 123.081749

# See what lambda was used in the ridge   
ridge\_fit$bestTune$lambda

## [1] 0.01428571

# Plot the lambdas to see how the lambda was chosen (minimum RMSE)   
plot(ridge\_fit)



# (d) Fit a lasso model on the training set, with fraction chosen by cross validation.

# Report the test mean squared error obtained, along with the number of non-zero coefficient estimates and the fraction.

lasso <- train(Apps ~., data = train1,   
 method= 'lasso',  
 preProc=c('scale','center'),  
 trControl=trctrl)  
lasso

## The lasso   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: scaled (17), centered (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 526, 528, 525, 527, 525, 526, ...   
## Resampling results across tuning parameters:  
##   
## fraction RMSE Rsquared MAE   
## 0.1 3099.302 0.9140627 2076.3624  
## 0.5 1082.136 0.9314001 576.6874  
## 0.9 1026.635 0.9365021 602.9177  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was fraction = 0.9.

lassoGrid <- data.frame(fraction = seq(0.05, .8, length = 15))  
lasso <- train(Apps ~., data = train1,  
 method='lasso',  
 preProc=c('scale','center'),  
 tuneGrid = lassoGrid,  
 trControl=trctrl)  
# Generate predictions with test dataset  
lasso\_pred <- predict(lasso, newdata = test1)  
  
# Calculate Mean Square Error (MSE)  
lasso\_mse <- mean(lasso\_pred - test1$Apps)^2  
lasso\_mse

## [1] 5.282411

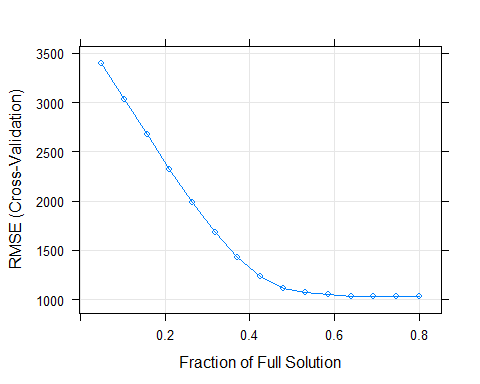
# See the coefficients of the regression (the last row is the final coeffcients)  
predict(lasso$finalModel, type='coefficients', mode='norm')$coefficients

## PrivateYes Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad  
## 0 0.00000 0.000 0.00000 0.0000 0.00000 0.0000 0.00000  
## 1 0.00000 3118.519 0.00000 0.0000 0.00000 0.0000 0.00000  
## 2 0.00000 3412.191 0.00000 293.6716 0.00000 0.0000 0.00000  
## 3 0.00000 3543.425 0.00000 360.0307 0.00000 0.0000 0.00000  
## 4 -64.34795 3554.396 0.00000 387.8549 0.00000 0.0000 0.00000  
## 5 -87.18885 3554.697 0.00000 395.3741 0.00000 0.0000 0.00000  
## 6 -89.18293 3554.464 0.00000 397.2842 0.00000 0.0000 0.00000  
## 7 -96.90407 3555.585 0.00000 404.9843 0.00000 0.0000 0.00000  
## 8 -108.77504 3561.492 0.00000 439.3036 0.00000 0.0000 0.00000  
## 9 -116.84280 3600.314 -45.62726 451.9002 0.00000 0.0000 0.00000  
## 10 -130.87969 3659.902 -117.03541 469.0630 0.00000 0.0000 0.00000  
## 11 -143.87277 3723.532 -198.92921 489.3963 0.00000 0.0000 14.66885  
## 12 -163.70017 3816.901 -318.84036 519.1336 0.00000 0.0000 36.32149  
## 13 -176.62501 3878.790 -394.52176 581.4808 -49.81138 0.0000 49.74382  
## 14 -210.30132 4041.844 -597.11480 742.3108 -178.14841 0.0000 81.37388  
## 15 -216.10470 4112.781 -835.29127 812.1246 -239.96087 171.7969 85.18803  
## 16 -217.03504 4123.959 -872.83523 823.0672 -249.71417 198.8725 85.78119  
## 17 -217.11401 4124.669 -875.19759 823.7565 -250.33417 200.5906 85.81556  
## Outstate Room.Board Books Personal PhD Terminal S.F.Ratio  
## 0 0.00000 0.00000 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 1 0.00000 0.00000 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 2 0.00000 0.00000 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 3 0.00000 0.00000 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 4 0.00000 0.00000 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 5 0.00000 15.89804 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 6 0.00000 17.89628 0.0000000 0.00000 0.00000 0.000000 0.0000000  
## 7 0.00000 23.79990 0.0000000 0.00000 -12.67536 0.000000 0.0000000  
## 8 -54.47427 64.11197 0.0000000 0.00000 -55.99901 0.000000 0.0000000  
## 9 -72.85521 70.46131 0.0000000 0.00000 -65.49958 0.000000 0.0000000  
## 10 -104.76370 79.75910 0.0000000 0.00000 -80.74236 0.000000 0.0000000  
## 11 -137.29498 87.83562 0.0000000 0.00000 -97.90737 0.000000 0.0000000  
## 12 -184.37387 100.31231 0.0000000 0.00000 -116.06260 -9.082052 0.0000000  
## 13 -213.21537 108.31070 0.0000000 0.00000 -126.87369 -8.555821 0.0000000  
## 14 -286.05657 130.11651 0.0000000 18.63705 -154.12468 -7.272111 0.0000000  
## 15 -309.32808 136.44430 0.0000000 23.44908 -166.25414 -10.463126 0.0000000  
## 16 -312.95887 137.37273 0.3717955 24.12972 -168.09089 -11.038567 0.0000000  
## 17 -313.19058 137.42593 0.3991982 24.16412 -168.19789 -11.074925 -0.1120441  
## perc.alumni Expend Grad.Rate  
## 0 0.000000 0.0000 0.000000  
## 1 0.000000 0.0000 0.000000  
## 2 0.000000 0.0000 0.000000  
## 3 0.000000 97.7062 0.000000  
## 4 0.000000 147.6031 0.000000  
## 5 0.000000 154.7367 0.000000  
## 6 -3.029819 156.1379 0.000000  
## 7 -6.633261 160.8712 0.000000  
## 8 -12.169565 193.3602 0.000000  
## 9 -12.077931 201.7602 0.000000  
## 10 -13.868335 217.0680 9.492627  
## 11 -15.232503 232.1583 21.884916  
## 12 -16.730465 254.4692 39.965940  
## 13 -16.194739 260.9267 52.254455  
## 14 -12.440556 276.5087 85.898496  
## 15 -8.154701 285.7632 102.264835  
## 16 -7.452478 287.2115 104.856871  
## 17 -7.416487 287.2462 105.020853

# See what fraction was used in the lasso regression  
lasso$bestTune$fraction

## [1] 0.8

# Plot the fractions to see how the fraction was chosen (minimum RMSE)  
plot(lasso)



# (e) Fit a PCR model on the training set, with no. of principal components M chosen by cross-validation.

# Report the test mean squared error obtained, along with the value of M selected by cross-validation.

pcr\_fit <- train(Apps ~., data=train1,  
 preProc = c('center', 'scale'),  
 method='pcr',  
 trControl=trctrl)  
pcr\_fit

## Principal Component Analysis   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: centered (17), scaled (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 525, 526, 528, 525, 525, 526, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 3805.440 0.01645192 2585.426  
## 2 1992.716 0.75646996 1297.173  
## 3 2000.603 0.75407555 1312.703  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 2.

# Generate predictions with test dataset  
pcr\_pred <- predict(pcr\_fit, test1)  
  
# Calculate Mean Square Error (MSE)  
pcr\_mse <- mean(pcr\_pred - test1$Apps)^2  
pcr\_mse

## [1] 35092.46

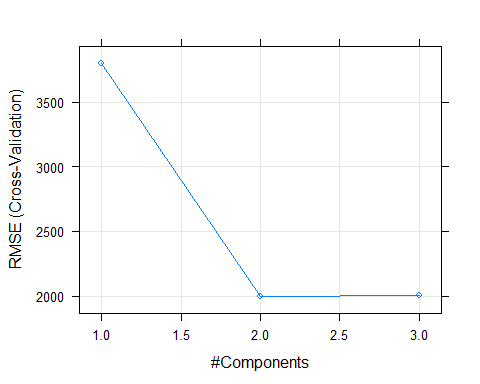
# See the coefficients of the regression   
pcr\_fit$finalModel$coefficients

## , , 1 comps  
##   
## .outcome  
## PrivateYes 29.495042  
## Accept -1.143878  
## Enroll -7.574264  
## Top10perc 46.421089  
## Top25perc 42.358269  
## F.Undergrad -11.009675  
## P.Undergrad -18.015828  
## Outstate 51.955779  
## Room.Board 39.793887  
## Books 4.213103  
## Personal -22.830621  
## PhD 34.234778  
## Terminal 33.988896  
## S.F.Ratio -37.327140  
## perc.alumni 40.258632  
## Expend 46.885412  
## Grad.Rate 40.424021  
##   
## , , 2 comps  
##   
## .outcome  
## PrivateYes -460.68162  
## Accept 652.47172  
## Enroll 683.66489  
## Top10perc 297.06783  
## Top25perc 346.76339  
## F.Undergrad 683.94502  
## P.Undergrad 417.65542  
## Outstate -40.58215  
## Room.Board 57.14744  
## Books 175.76323  
## Personal 265.54569  
## PhD 454.86023  
## Terminal 440.62044  
## S.F.Ratio 139.05725  
## perc.alumni -96.85047  
## Expend 171.49869  
## Grad.Rate 26.33576

# See the number of principal components used in the regression  
pcr\_fit$bestTune$ncomp

## [1] 2

# Plot principal components vs RMSE to see how the no of components was chosen  
plot(pcr\_fit)



# (f) Fit a PLS model on the training set, with M chosen by cross validation.

# Report the test error obtained, along with the value of M selected by cross-validation.

pls\_fit <- train(Apps ~., data=train1,  
 preProc = c('center', 'scale'),  
 method='kernelpls',  
 trControl=trctrl)  
pls\_fit

## Partial Least Squares   
##   
## 585 samples  
## 17 predictor  
##   
## Pre-processing: centered (17), scaled (17)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 525, 528, 526, 526, 526, 527, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 1800.981 0.8093431 1165.4165  
## 2 1415.929 0.8874875 801.8750  
## 3 1347.589 0.8969791 801.1473  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 3.

pls.pred <- predict(pls\_fit, test1)  
  
# Calculate Mean Square Error (MSE)  
pls\_fit\_mse <-mean(pcr\_pred - test1$Apps)^2  
pls\_fit\_mse

## [1] 35092.46

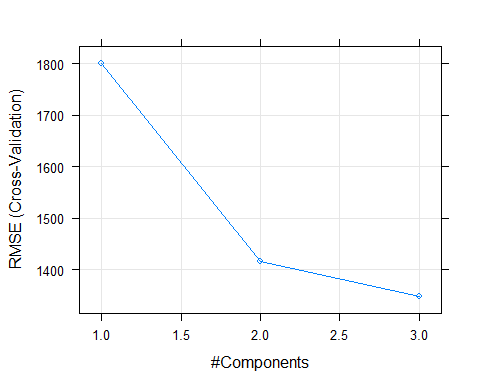
# Plot principal components vs RMSE to see how the no of components was chosen  
pls\_fit$finalModel$coefficients

## , , 1 comps  
##   
## .outcome  
## PrivateYes -410.350479  
## Accept 883.505125  
## Enroll 774.103805  
## Top10perc 298.515791  
## Top25perc 323.306950  
## F.Undergrad 743.040277  
## P.Undergrad 353.787168  
## Outstate 3.524867  
## Room.Board 129.390197  
## Books 135.726697  
## Personal 196.589653  
## PhD 349.585400  
## Terminal 333.685650  
## S.F.Ratio 113.301372  
## perc.alumni -105.320445  
## Expend 206.700110  
## Grad.Rate 110.737131  
##   
## , , 2 comps  
##   
## .outcome  
## PrivateYes -235.78288  
## Accept 1712.81438  
## Enroll 1006.09523  
## Top10perc 167.91993  
## Top25perc 91.62511  
## F.Undergrad 861.49776  
## P.Undergrad 85.48709  
## Outstate 46.34354  
## Room.Board 294.53788  
## Books -33.72818  
## Personal -41.53068  
## PhD -166.65869  
## Terminal -183.18808  
## S.F.Ratio 76.51053  
## perc.alumni -219.25455  
## Expend 222.43450  
## Grad.Rate 306.05485  
##   
## , , 3 comps  
##   
## .outcome  
## PrivateYes -165.027643  
## Accept 1879.814879  
## Enroll 984.395684  
## Top10perc 219.852877  
## Top25perc 109.962439  
## F.Undergrad 813.214636  
## P.Undergrad -2.954522  
## Outstate 119.359042  
## Room.Board 369.151457  
## Books -45.066065  
## Personal -112.849103  
## PhD -189.220502  
## Terminal -202.784951  
## S.F.Ratio 7.853934  
## perc.alumni -178.371574  
## Expend 302.654354  
## Grad.Rate 370.889345

# See the number of principal components used in the regression  
pls\_fit$bestTune$ncomp

## [1] 3

plot(pls\_fit)



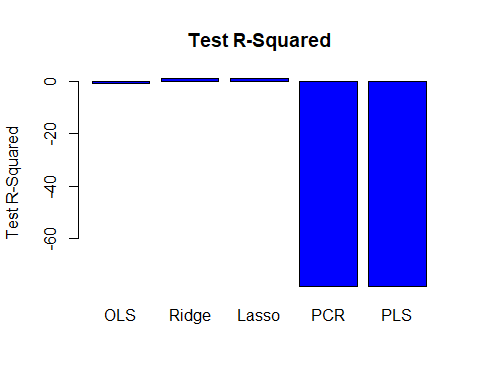
# (g) Comment on the results obtained.

# Is there much difference among the test errors resulting from these five approaches?

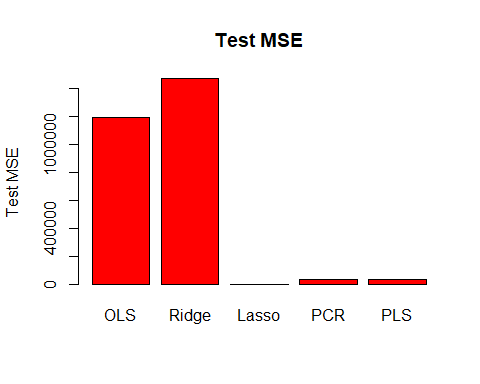
avg\_test <- mean(test1[,"Apps"])   
  
linear\_r2 <- -1-mean((test1[,"Apps"]-ln\_pred)^2)/mean((test1[,"Apps"]-avg\_test)^2)   
  
ridge\_r2 <- 1 - mean((test1[, "Apps"] -ridge\_pred)^2) /mean((test1[, "Apps"] - avg\_test)^2)   
lasso\_r2 <- 1 - mean((test1[, "Apps"] -lasso\_pred)^2) /mean((test1[, "Apps"] - avg\_test)^2)   
  
pcr\_r2 <- 1 - mean((test1[, "Apps"] -(pcr\_mse))^2) /mean((test1[, "Apps"] -avg\_test)^2)   
  
pls\_r2 <- 1 - mean((test1[, "Apps"] -(pls\_fit\_mse))^2) /mean((test1[, "Apps"] -avg\_test)^2)   
  
par(mfrow <- c(1,2))

## NULL

barplot(c(linear\_r2,ridge\_r2,lasso\_r2,pcr\_r2,pls\_r2),col="blue",  
 names.arg = c("OLS","Ridge","Lasso","PCR","PLS"),main="Test R-Squared", ylab="Test R-Squared")



barplot(c(ln\_mse,ridge\_mse, lasso\_mse, pcr\_mse, pls\_fit\_mse), col="red",   
  
names.arg=c("OLS","Ridge", "Lasso", "PCR", "PLS"), main = "Test MSE", ylab = "Test MSE")



# 1.Load the Cars93 dataset in the MASS package in R

library(MASS)  
data(Cars93)  
attach(Cars93)  
head(Cars93)

## Manufacturer Model Type Min.Price Price Max.Price MPG.city MPG.highway  
## 1 Acura Integra Small 12.9 15.9 18.8 25 31  
## 2 Acura Legend Midsize 29.2 33.9 38.7 18 25  
## 3 Audi 90 Compact 25.9 29.1 32.3 20 26  
## 4 Audi 100 Midsize 30.8 37.7 44.6 19 26  
## 5 BMW 535i Midsize 23.7 30.0 36.2 22 30  
## 6 Buick Century Midsize 14.2 15.7 17.3 22 31  
## AirBags DriveTrain Cylinders EngineSize Horsepower RPM  
## 1 None Front 4 1.8 140 6300  
## 2 Driver & Passenger Front 6 3.2 200 5500  
## 3 Driver only Front 6 2.8 172 5500  
## 4 Driver & Passenger Front 6 2.8 172 5500  
## 5 Driver only Rear 4 3.5 208 5700  
## 6 Driver only Front 4 2.2 110 5200  
## Rev.per.mile Man.trans.avail Fuel.tank.capacity Passengers Length Wheelbase  
## 1 2890 Yes 13.2 5 177 102  
## 2 2335 Yes 18.0 5 195 115  
## 3 2280 Yes 16.9 5 180 102  
## 4 2535 Yes 21.1 6 193 106  
## 5 2545 Yes 21.1 4 186 109  
## 6 2565 No 16.4 6 189 105  
## Width Turn.circle Rear.seat.room Luggage.room Weight Origin Make  
## 1 68 37 26.5 11 2705 non-USA Acura Integra  
## 2 71 38 30.0 15 3560 non-USA Acura Legend  
## 3 67 37 28.0 14 3375 non-USA Audi 90  
## 4 70 37 31.0 17 3405 non-USA Audi 100  
## 5 69 39 27.0 13 3640 non-USA BMW 535i  
## 6 69 41 28.0 16 2880 USA Buick Century

cars\_data <- na.omit(Cars93)

# 2.Run a principal component analysis on columns 4 through 8 in the dataset

cars\_pca <- prcomp(cars\_data[4:8], scale=TRUE)  
summary(cars\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5  
## Standard deviation 2.0045 0.9198 0.28510 0.23339 0.002231  
## Proportion of Variance 0.8036 0.1692 0.01626 0.01089 0.000000  
## Cumulative Proportion 0.8036 0.9728 0.98910 1.00000 1.000000

cars\_pca$center

## Min.Price Price Max.Price MPG.city MPG.highway   
## 16.82683 19.17073 21.51585 23.08537 29.97561

# The scaling factor that was used to scale the variables  
cars\_pca$scale

## Min.Price Price Max.Price MPG.city MPG.highway   
## 8.921953 9.959318 11.446337 5.593650 5.011039

# To see the eigen vectors (loadings for each variable for each PC)  
cars\_pca$rotation

## PC1 PC2 PC3 PC4 PC5  
## Min.Price 0.4721737 -0.2602795 -0.756860888 0.07049906 -0.3626265737  
## Price 0.4721263 -0.3510487 0.027649698 -0.00453180 0.8081314972  
## Max.Price 0.4534056 -0.4075026 0.639508057 -0.06300006 -0.4641394246  
## MPG.city -0.4167658 -0.5698611 0.002012706 0.70820942 -0.0001597383  
## MPG.highway -0.4181846 -0.5641094 -0.131994257 -0.69962830 -0.0001421337

# To see the square root of the eigen values  
cars\_pca$sdev

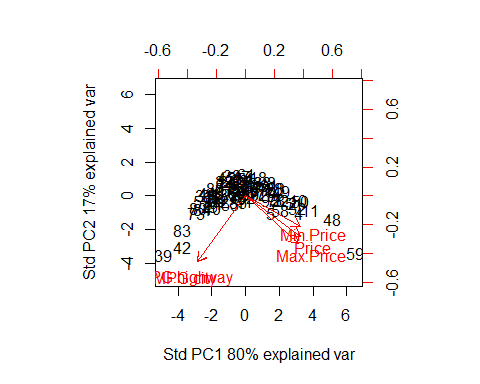
## [1] 2.004548487 0.919797562 0.285098882 0.233391212 0.002231334

# To see the rotated data (i.e the centered/scaled data multiplied by the rotation matrix)  
cars\_pca$x

## PC1 PC2 PC3 PC4 PC5  
## 1 -0.69858943 0.01615709 0.14600836 0.0847955922 4.247297e-03  
## 2 2.82788093 -0.41371802 0.08057144 0.0473127678 -4.234604e-03  
## 3 1.93971012 -0.23673460 -0.03600106 0.1722480637 -1.668366e-04  
## 4 3.16844579 -1.01883459 0.25904390 0.0127452685 -2.181258e-04  
## 5 1.53760504 -0.99716825 0.26637670 -0.1722613299 3.967243e-03  
## 6 -0.47516733 0.34431293 -0.04971220 -0.2764138097 -3.909529e-03  
## 7 0.71642906 0.48496538 -0.19532057 -0.2188882064 2.967178e-06  
## 8 1.59743162 0.83340417 -0.15958695 -0.1774620528 -4.007785e-03  
## 9 1.58152961 0.05320129 -0.43962827 -0.0565204087 -3.304811e-04  
## 10 3.12085802 -0.26357817 -0.37437438 -0.1630341927 3.606756e-03  
## 11 3.86851278 -0.81304426 -0.38355474 -0.1651587508 -6.337031e-04  
## 12 -1.48703209 -0.31242749 0.35268689 -0.6441696675 3.568628e-04  
## 13 -1.53478066 0.14426208 -0.23169874 -0.3031324315 4.784595e-05  
## 14 -0.09187560 1.04995010 0.06649467 -0.2406865680 3.653089e-04  
## 15 -0.22303952 0.64846347 0.13248679 -0.1362552586 3.157684e-04  
## 18 0.75380027 1.11455240 -0.10505909 -0.1954174133 2.077278e-04  
## 20 0.31804725 0.62892793 -0.25874818 -0.0848761905 9.304125e-06  
## 21 -0.28662115 0.57500004 -0.00667607 0.2724738004 1.778741e-04  
## 22 2.03828202 -0.25617326 -0.49671953 0.2159234186 -4.913652e-04  
## 23 -2.07056635 0.05746204 0.04218692 0.3206712304 -3.858780e-03  
## 24 -1.02110001 0.90224205 0.30993768 0.1026647834 5.277031e-04  
## 25 -0.47984162 1.03886281 0.09883728 0.2792841838 3.698491e-04  
## 27 -0.07548687 0.91454480 -0.04616873 0.1651864436 2.258265e-04  
## 28 1.73925252 0.49588297 0.67924674 0.1368893053 5.027852e-04  
## 29 -1.69464255 -0.25832947 0.38014933 0.2868328355 3.309032e-04  
## 30 0.42399373 0.52377695 -0.02346508 -0.1078083542 -3.919558e-03  
## 31 -2.46478362 0.04245206 -0.02810870 0.5816682710 1.533007e-04  
## 32 -1.25254553 0.91384925 0.15176422 -0.0237475159 -3.609484e-03  
## 33 -0.75306516 1.24212158 0.08085644 0.2821014909 4.227120e-04  
## 34 -0.33215577 0.52987370 0.49867026 -0.0445005467 5.345706e-04  
## 35 -0.77859359 0.42865995 -0.02593491 0.1176920775 1.734638e-04  
## 37 0.26729435 0.09229487 0.28902383 -0.2956707335 2.712657e-04  
## 38 0.97316645 0.80262907 -0.15968763 -0.0647280554 7.371709e-05  
## 39 -4.88107845 -3.50361720 -0.33343352 0.0937215038 3.361621e-03  
## 40 -1.93359847 -0.70671952 -0.17068306 0.0394117594 -1.117671e-04  
## 41 -0.06773357 -0.27788816 0.02656105 -0.0326567835 -4.048849e-03  
## 42 -3.75412489 -3.03231728 -0.03940443 0.1255794949 -4.611339e-04  
## 43 -0.40553590 -0.05006207 0.20783065 -0.0486398958 2.071891e-04  
## 44 -2.24112387 0.18169177 0.05395169 0.3202308692 2.468820e-04  
## 45 -1.10322311 1.14636111 0.07628560 -0.0009995234 4.407784e-04  
## 46 -1.81322203 0.17307141 -0.06246174 -0.1918583021 -3.879701e-03  
## 47 -0.25215625 1.18551956 0.09088981 0.0264405981 4.461264e-03  
## 48 5.13722378 -1.35673203 -0.52248544 0.3967972381 -9.783664e-04  
## 49 2.13367512 0.32310578 -0.34071669 0.2328725715 3.800287e-03  
## 50 3.22469282 -0.28448166 -0.50290750 0.3864779498 3.467096e-03  
## 51 2.92020159 -0.43707937 -0.48278526 -0.1679855885 -5.515926e-04  
## 52 3.08826801 -0.72349574 -0.43106734 -0.0472630535 -6.039417e-04  
## 53 -2.53292048 -0.29311970 -0.10706401 -0.2330838747 4.144736e-03  
## 54 -1.90653683 -0.41101810 -0.19004668 -0.2115340985 -3.800002e-05  
## 55 -0.92488051 -0.48187030 -0.05533899 -0.1961073019 -2.750871e-05  
## 58 2.08913743 -0.85614879 -0.22496438 -0.2376931200 -4.475992e-03  
## 59 6.48935020 -3.39879761 1.22756499 0.0492339085 -3.394412e-04  
## 60 -0.34699631 0.96983867 0.02575602 0.5545583847 -3.782516e-03  
## 61 0.06967101 1.30602920 -0.11477934 0.0609494113 2.703455e-04  
## 62 -1.93789860 -0.05735908 0.19070831 0.3059312620 2.645786e-04  
## 63 1.83311612 0.48545771 0.17045320 0.1851822688 -3.909694e-03  
## 64 -1.73464486 -0.21060668 0.22178162 0.3021425690 2.368237e-04  
## 65 -0.56462434 0.25253978 0.13501580 0.1014366318 4.285720e-03  
## 67 0.83760275 0.43891485 -0.21652955 0.3202879289 -7.062521e-05  
## 68 -0.92269817 0.37059752 -0.13767412 -0.0135127374 1.036109e-04  
## 69 -0.47765843 0.18212629 0.01377049 -0.1561316359 1.437843e-04  
## 71 0.69844172 0.49303915 -0.15049166 -0.2231041942 3.653023e-05  
## 72 -0.67207110 0.47895667 0.21649383 -0.0322706140 3.531832e-04  
## 73 -2.90853398 -1.02365896 -0.23293274 -0.5367347614 -4.180333e-03  
## 74 -1.20001976 0.70481369 0.09365158 -0.1608717692 3.668522e-04  
## 75 0.24534478 0.77703562 0.27981665 -0.2624467648 4.250237e-04  
## 76 0.44873589 0.81344802 0.20078856 -0.1132317058 3.561321e-04  
## 77 1.16563535 0.09852933 0.28728944 -0.2668576052 2.122562e-04  
## 78 1.81451566 -0.23015236 0.70612022 0.1017613050 3.479684e-04  
## 79 -2.16334665 -0.59030735 -0.06638094 -0.5072767723 4.099462e-03  
## 80 -2.81565541 -0.71547341 -0.07451594 0.2703170145 -1.040308e-05  
## 81 -1.27626423 0.64199531 -0.05696292 0.2490035828 2.244657e-04  
## 82 0.03895853 -0.03244286 0.11109096 -0.0250436175 1.161092e-04  
## 83 -3.73412636 -2.02702223 -0.20191090 0.1894267330 -4.397862e-03  
## 84 -2.55721319 -0.75941363 0.01509660 0.1343622267 3.347597e-05  
## 85 -0.44420554 -0.35774604 0.22863403 -0.0666021354 1.526426e-04  
## 86 0.01765895 0.31332158 0.14297196 -0.0118801972 2.192836e-04  
## 88 -1.77851353 0.48431549 -0.08885214 -0.1733456675 2.297622e-04  
## 90 0.26859123 0.12644140 -0.01528227 -0.2665667437 7.200484e-05  
## 91 1.39779865 0.67772010 -0.25247133 0.0849142983 -5.649245e-05  
## 92 0.82933924 0.09473021 -0.24993932 0.0385725289 3.905359e-03  
## 93 1.45029326 -0.20991090 -0.21433746 -0.0936716221 3.828190e-03

# 3.Plot the biplot

biplot(cars\_pca,xlab="Std PC1 80% explained var",ylab="Std PC2 17% explained var", scale=0)



# 4.Calculate the percent of variance explained

cars\_pca$sdev

## [1] 2.004548487 0.919797562 0.285098882 0.233391212 0.002231334

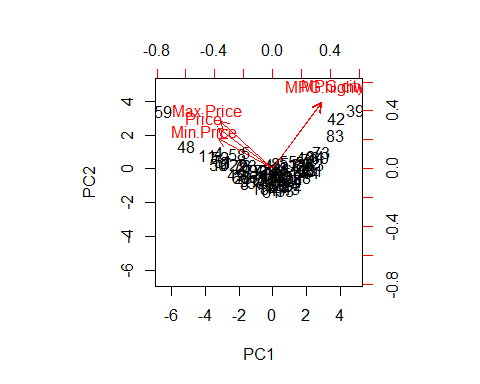
cars\_var <- cars\_pca$sdev^2   
cars\_var

## [1] 4.018215e+00 8.460276e-01 8.128137e-02 5.447146e-02 4.978852e-06

cars\_ve <- cars\_var/sum(cars\_var)   
cars\_ve

## [1] 8.036429e-01 1.692055e-01 1.625627e-02 1.089429e-02 9.957703e-07

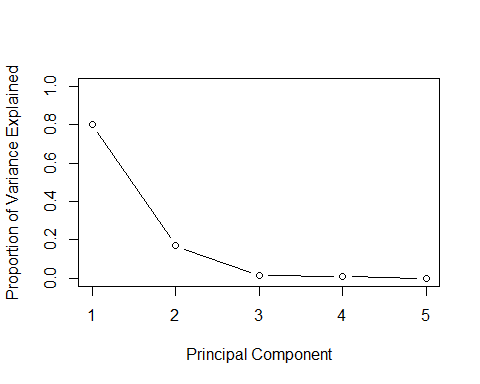
# Change the direction of the plot. Does not alter the values.  
cars\_pca$rotation=-cars\_pca$rotation  
cars\_pca$x=-cars\_pca$x  
biplot(cars\_pca, scale=0)



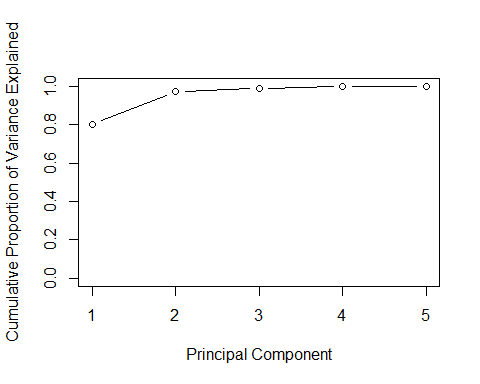
# 5.Plot the percent of variance explained

# Compute percentage variance explained

plot(cars\_ve, xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1),type='b')



# Compute percentage variance explained  
plot(cumsum(cars\_ve), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained", ylim=c(0,1),type='b')



# 6.How much of the variance does the first two principal components explain?

summary(cars\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5  
## Standard deviation 2.0045 0.9198 0.28510 0.23339 0.002231  
## Proportion of Variance 0.8036 0.1692 0.01626 0.01089 0.000000  
## Cumulative Proportion 0.8036 0.9728 0.98910 1.00000 1.000000