R code for CIOS

Haesong Choi

```r
library("psych")
library("car")
data= read.csv("D:/8_1/cios/comp2.csv")
head(data)
data1 <-data[-c(5,13,17)]
head(data1)
 f2 <-fa(data1,3, rotate="promax")
 f2 <-fa(data1,3, rotate="promax",scores="regression")
 x1 <-f2$scores[,1]
x2 <-f2$scores[,2]
 x3 <-f2$scores[,3]
y <-data$c55

df <-data.frame(x1,x2,x3,y)


 lm2 <-lm(y~x1+x2+x3,data=df)
 summary(lm2)
 vif(fit)
sqrt(vif(fit)) > 2

# C13

data=read.csv("D:/8_1/cios/comp5.csv")
head(data)
data1 <-data[-c(5,13,17)]
head(data1)
 f2 <-fa(data1,3, rotate="promax")
 f2 <-fa(data1,3, rotate="promax",scores="regression")
 x1 <-f2$scores[,1]
x2 <-f2$scores[,2]
 x3 <-f2$scores[,3]
y <-data$c1313
df <-data.frame(x1,x2,x3,y)

 lm2 <-lm(y~x1+x2+x3,data=df)
 summary(lm2)

 #Stepwise regression
 lm3 <-regsubsets(c55~c1+c2+c3+c4+s1+s2+s3+i1+i2+i3+i4+i5+i6+i7,
    data=data)
summary(lm3)
plot(lm3, scale="adjr2")
plot(lm3, scale="bic")
null=lm(c55~1,data=data)
full=lm(c55~c1+c2+c3+c4+s1+s2+s3+i1+i2+i3+i4+i5+i6+i7,data=data)
step(null, scope=list(lower=null, upper=full), direction="forward")

#
data=read.csv("C:/Users/haesong/Desktop/8_1/cios/comp4.csv")
data.train <- data[1:4777,]
data.test <- data[4778:9554,]
```

```
m.pls <- plsr(cindep ~ .,data=data.train , validation="LOO")
summary(m.pls)
plot(RMSEP(m.pls), legendpos = "topright")



plot(m.pls, plottype = "coef", ncomp=1:3, legendpos = "bottomleft",
     labels = "numbers", xlab = "nm")
plot(m.pls, plottype = "coef", ncomp=1:3, legendpos = "bottomleft")
explvar(m.pls)
plsr <- plsr(c5 ~ indep, ncomp = 10, data = Train, validation = "
    LOO")
Train <- Train[,c(1:4,6:16,5)]
typeof(ls())

X <- read.csv("data.csv",row.names = 1)

# data.csv : csv file including training data
# data_prediction1.csv : csv file including test data with Y
# data_prediction2.csv : csv file including test data without Y

# X: X of training data
# y: Y of training data
# X_prediction1: X of test data with Y
# y_prediction1: Y of test data with Y
# X_prediction2: X of test data without Y

data = read.csv("data.csv", row.names = 1)
y = as.factor(data[,1])
X = as.matrix(data[c(2:ncol(data))])
data_prediction1 = read.csv("data_prediction1.csv", row.names = 1)
y_prediction1 = as.factor(data_prediction1[,1])
X_prediction1 = as.matrix(data_prediction1[c(2:ncol(data_
    prediction1))])
data_prediction2 = read.csv("data_prediction2.csv", row.names = 1)
X_prediction2 = as.matrix(data_prediction2[c(1:ncol(data_
    prediction2))])
# Delete variables whose numbers are 'Var0Variable' from X, X_
    prediction1 and X_prediction2
if (length(Var0Variable) != 0) {
  X = X[,-Var0Variable]
  X_prediction1 = X_prediction1[,-Var0Variable]
X_prediction2 = X_prediction2[,-Var0Variable]
# data.csv : csv file including training data

# X: X of training data
# y: Y of training data

data = read.csv("data.csv", row.names = 1)
y = as.matrix(data[1])
X = as.matrix(data[c(2:ncol(data))])

#Find variables with zero variance
variables_zerovariance = function(X){
  Var0Variable <- which(apply(X,2,var) == 0)
  if (length(Var0Variable) == 0) {
    print("No variables with zero vairance")
```

```
  } else {
    sprintf("%d variable(s) with zero variance", length(
    Var0Variable))
    print( "Variable number:" )
    print( Var0Variable )
    print( "The variable(s) is(are) deleted." )
  }
  return(Var0Variable)
}


#Calculate r^2
calc_r2 = function( ActualY, EstimatedY ){
  return( 1 - sum( (ActualY-EstimatedY )^2 ) / sum((ActualY-mean(
    ActualY))^2) )
}

#Calculate RMSE
calc_RMSE = function( ActualY, EstimatedY ){
  return( sqrt( sum( (ActualY-EstimatedY )^2 ) / nrow(ActualY)) )
}

#Make YYplot
make_yyplot = function( ActualY, EstimatedY, YMax, YMin,
    EstimatedYName ){
  par(pty = "s")
  plot( ActualY, EstimatedY,
        xlim=c(YMin-0.05*(YMax-YMin),YMax+0.05*(YMax-YMin)),
        ylim=c(YMin-0.05*(YMax-YMin),YMax+0.05*(YMax-YMin)),
        col = "blue", xlab = "Actual Y", ylab = EstimatedYName)
  abline(0,1)
  par(pty = "m")
}


#Make tt plots for PCA with clustering result
make_ttplot_withclustering = function( PcaResult, ClusterNum ){
  pairs(PcaResult$x[,1:3], col = ClusterNum)
  plot( PcaResult$x[,1], pca_result$x[,2], col = ClusterNum, xlab =
     "First principal component", ylab = "Second  principal
    component")
  text( PcaResult$x[,1], pca_result$x[,2], labels = rownames(X),
    pos=3, offset = 0.1)
}

#Make Threshold for T^2 and SPE
make_threshold_t2spe = function( Index, NumOfIndexThreshold ){
  SortedIndex = sort(Index)
  return( SortedIndex[NumOfIndexThreshold] )
}
#Save vector
savevectorcsv = function( X, ColName, FileName ){
  ClusterNum <- as.matrix( X, col = length(X), row = 1 )
  colnames(X) <- c(ColName)
  write.table(X, FileName, quote=FALSE, sep = ",", col.names=NA)
}

```

```r
#Optimize gamma to maximize variance of Gaussian gram matrix
optimize_gamma_grammatrix = function(X, CandidatesOfGamma){
  # Calculate gram matrix of Gaussian kernel and its variance for
    each gamma candidate
  VarianceOfKernelMatrix <- NULL
  DistanceMatrics <- dist(X, diag = TRUE, upper = TRUE)^2
  for (CandidateOfGamma in CandidatesOfGamma) {
    KernelMatrix <- exp(-CandidateOfGamma*DistanceMatrics)
    VarianceOfKernelMatrix <- c(VarianceOfKernelMatrix,var(c(as.
    vector(KernelMatrix),as.vector(KernelMatrix),rep(1,nrow(X)))))
  }
  # Decide the optimal gamma with the maximum variance value
  OptimalGamma = CandidatesOfGamma[which(VarianceOfKernelMatrix ==
    max(VarianceOfKernelMatrix))]
  return( OptimalGamma[1] )
}
lm2 = lm(c5~s1+s2+s3+c1+c2+c3+c4+i1+i2+i3+i4+i5+i6+i7, data=data)
vif(lm2)
sqrt(vif(lm2))>2
data=read.csv("C:/Users/haesong/Desktop/8_1/cios/comp2.csv")
```