Analiza Datelor Seminarii

Seminar1

Prelucrari.R

```r
citireDate = function(fisier){
  tryCatch(
    {
      #citire date utilizand functia read.csv
      tabel = read.csv(file = fisier,row.names = 1)
      return(tabel)
    },
    error = function(e){
      stop(paste("Eroare citire date!",e))
    }
  )
}

calculIndicatori = function(tabel) {
  tryCatch({
    tabelIndicatori = sapply(tabel,FUN = function(variabila){
      m = mean(variabila)
      varianta = var(variabila)
      std = sqrt(varianta)
      mx = median(variabila)
      cv = std/m
      simetrie = moments::skewness(x = variabila)
      aplatizare = moments::kurtosis(variabila)
      return(c(Media=m,Varianta=varianta,AbatereStandard=std,
             Mediana=mx,CoeficVariatie=cv,Simetria=simetrie,
             Aplatizare=aplatizare))


    })
    return(tabelIndicatori)
  },
  error = function(e) {
    stop(paste("Eroare calcul indicatori!", e))
  })
}
```

Seminar1.R

```r
seminar = function() {
  tryCatch({
    source("Prelucrari.R")
    tabel = citireDate("ADN.csv")
    View(tabel)
    indicatori = calculIndicatori(tabel)
    View(indicatori)
    indicator = select.list(choices = names(tabel),
                            graphics = T,
                            title = "Selectie variabila pentru histograma")
    hist(
      x = tabel[[indicator]],
      xlab = indicator[1],
      main = paste("Histograma pentru ", indicator[1])
    )
    i = select.list(
      choices = names(tabel),
      graphics = T,
      multiple = T,
      title = "Alegeti doi indicatori pentru plot"
    )
    if (length(i) == 2) {
      dev.new()
      plot(
        x = tabel[[i[1]]],



     y = tabel[[i[2]]],
        xlab = i[1],
        ylab = i[2]
      )
      text(
        x = tabel[[i[1]]],
        y = tabel[[i[2]]],
        labels = rownames(tabel),
        pos = 1
      )
    }
  },
  error = function(e) {
    print(e)
  })
}
```

s1.R

```r
source("Seminar1.R")
seminar()
```

Seminar2

Prelucrari.R

```r
#'
#' @param X - tabelul de observatii
#' @return Lista principalilor indicatori privind rezultatele
#' PCA
acp = function(X) {
  tryCatch({
    m = length(X)
    #Eliminarea datelor lipsa
    for (j in 1:m) {
      X[is.na(X[, j]),j] = mean(X[,j],na.rm = T)
    }
    #Apelarea functiei princomp si calculul principalelor
    #rezultate ale ACP
    rez = princomp(x = X,cor = T)
    #Preluare abateri standard componente
    std = rez$sdev
    #Preluare componente
    C = rez$scores
    #Preluare corelatii Componente-Variabile
    #standardizate
    a = rez$loadings
    #calcul varianta componente
    alpha = std*std
    #calcul corelatii dintre componente si variabile
    R = matrix(0,m,m)
    for(j in 1:m){
      R[,j] = a[,j]*std[j]
    }
    return(
      list(
        Varianta = alpha,
        Componente = C,
        Corelatii = R
      )
    )
  },
  error = function(e) {
    stop(paste("Eroare PCA!", e))
  })
}
#' Calcul tabel varianta
#' @param Vectorul variantei
#' @return Tabelul cu distributia variantei
#'
tabelareVarianta = function(alpha){
  varCum = cumsum(alpha)
  proc = alpha*100/sum(alpha)
  procCum = cumsum(proc)
  return(
    data.frame(
      Varianta=alpha,
      Varianta.Cumulata=varCum,
      Procent.Varianta=proc,
      Procent.Cumulat=procCum,
      row.names = names(alpha)
    )
  )
}
```

Seminar2_1074.R

```
source("Prelucrari.R")
setDate = read.csv(file = "ADN/ADN.csv",header = T,row.names = 1)
rezultate = acp(X = setDate)
varianta = tabelareVarianta(alpha = rezultate$Varianta)
View(varianta)
plot(x = 1:length(rezultate$Varianta),y = rezultate$Varianta,
     main = "Distributia variantei",xlab = "Index",
     ylab = "Varianta",col="blue")
lines(x = 1:length(rezultate$Varianta),y = rezultate$Varianta,col="red")
View(rezultate$Corelatii)
tabelCorelatii=rezultate$Corelatii
rownames(tabelCorelatii)=names(setDate)
View(tabelCorelatii)
```

Seminar3

Prelucrari.R

```
#'
#' @param X - tabelul de observatii
#' @return Lista principalilor indicatori privind rezultatele
#' PCA
acp = function(X) {
  tryCatch({
    m = length(X)
    #Eliminarea datelor lipsa
    for (j in 1:m) {
      X[is.na(X[, j]),j] = mean(X[,j],na.rm = T)
    }
    #Apelarea functiei princomp si calculul principalelor
    #rezultate ale ACP
    rez = princomp(x = X,cor = T)
    #Preluare abateri standard componente
    std = rez$sdev
    #Preluare componente
    C = rez$scores
    #Preluare corelatii Componente-Variabile
    #standardizate
    a = rez$loadings
    #calcul varianta componente
    alpha = std*std
    #calcul corelatii dintre componente si variabile
    R = matrix(0,m,m)
    for(j in 1:m){
      R[,j] = a[,j]*std[j]
    }
    return(
      list(
        Varianta = alpha,
        Componente = C,
        Corelatii = R
      )
    )
  },
  error = function(e) {
    stop(paste("Eroare PCA!", e))
  })
}
#' Calcul tabel varianta
#' @param Vectorul variantei
#' @return Tabelul cu distributia variantei
#'
tabelareVarianta = function(alpha){
  varCum = cumsum(alpha)
  proc = alpha*100/sum(alpha)
  procCum = cumsum(proc)
  return(
    data.frame(
      Varianta=alpha,
      Varianta.Cumulata=varCum,
      Procent.Varianta=proc,
      Procent.Cumulat=procCum,
      row.names = names(alpha)
    )
  )
}

plotValoriProprii = function(alpha){
  dev.new(noRStudioGD = T)
  plot(x = 1:length(alpha),y = alpha,
       main = "Distributia variantei",xlab = "Index",
       ylab = "Varianta",col="blue")
  lines(x = 1:length(alpha),y = alpha,col="red")
  axis(side = 1,pos = 1,col="green")
}

#' Cercul corelatiilor
#'
cerculCorelatiilor = function(R,i=1,j=2){
  dev.new(noRStudioGD = T)
  t = seq(from=0,to=2*pi,length.out = 100)
  plot(x = cos(t),y = sin(t),type = "l",
       main = "Cercul Corelatiilor",
       xlab = paste("Componenta",i,sep = "-"),
       ylab = paste("Componenta",j,sep = "-")
       )
  points(x = R[,i],y = R[,j],col="red")
  text(x = R[,i],y = R[,j],col="blue",labels = rownames(R),
       pos = 1)
  axis(side = 1,pos = 0,col="green")
  axis(side = 2,pos = 0,col="green")
}
```

```
plotInstante = function(X,tip="Componente",i=1,j=2){
  dev.new(noRStudioGD = T)
  plot(x = X[,i],y = X[,j],main = paste("Plot",tip),
       xlab = paste("Componenta",i),
       ylab = paste("Componenta",j),col="red")
  text(x = X[,i],y = X[,j],labels = rownames(X),col="blue",
       pos = 1)
}

#' Calcul scoruri
calculScoruri = function(C,alpha){
  m = ncol(C)
  n=nrow(C)
  S = matrix(0,nrow = n,ncol = m)
  for(j in 1:m){
    S[,j]=C[,j]/sqrt(alpha[j])
  }
  rownames(S)=rownames(C)
  colnames(S)=colnames(C)
  return(S)
}
```

Seminar3_1074.R

```
#'
#'Functia se apeleaza din consola:
#'> seminar()
#'
seminar = function() {
  source("Prelucrari.R")
  setDate = read.csv(file = "ADN/ADN.csv",
                     header = T,
                     row.names = 1)
  #Apel functie acp
  rezultate = acp(X = setDate)
  #Tabelare varianta si salvare tabel
  varianta = tabelareVarianta(alpha = rezultate$Varianta)
  write.csv(x = varianta,file = "DistributieVarianta.csv",row.names = T)
  #Plot valori proprii
  plotValoriProprii(alpha = rezultate$Varianta)
  #Preluare corelatii
  Corelatii = rezultate$Corelatii
  rownames(Corelatii) = names(setDate)
  colnames(Corelatii) = paste("Comp", 1:ncol(setDate), sep = "")
  #Salvare corelatii
  xlsx::write.xlsx(x = Corelatii, file = "Corelatii.xlsx")
  #Plot corelatii
  for (i in 2:4) {
    cerculCorelatiilor(R = Corelatii, j = i)
  }
  #Salvare componente
  write.csv(x = rezultate$Componente,file = "C.csv",row.names = T)
  #Plot componente
  plotInstante(X = rezultate$Componente)
  #Calcul scoruri
  S = calculScoruri(C = rezultate$Componente, alpha = rezultate$Varianta)
  #Salvare scoruri
  write.csv(x = S,file = "Scoruri.csv",row.names = T)
  #Plot scoruri
  plotInstante(X = S, tip = "Scoruri")
}
```

Seminar4

Functii.R

```
biPlotInstante = function(Z1, Z2, U1, U2, etichete) {
  tryCatch({
    windows(width = 9, height = 7)
    #dev.new(noRStudioGD = T)
    par(mai = c(1, 1, 1, 2), xpd = T)
    xlim = c(min(Z1, U1), max(Z1, U1))
    ylim = c(min(Z2, U2), max(Z2, U2))
    plot(
      x = Z1,
      y = Z2,
      main = "Reprezentare simultana instante (indicativ tara)",
      col = "red",
      xlab = "z1/u1",
      ylab = "z2/u2",
      xlim = xlim,
      ylim = ylim
    )
    text(
      x = Z1,
      y = Z2,
      labels = etichete,
      col = "red",
      pos = 1
    )
    points(x = U1, y = U2, col = "blue")
    text(
      x = U1,
      y = U2,
      col = "blue",
      labels = etichete,
      pos = 1
    )
    legend(
      x = max(Z1, U1) * 1.1,
      y = max(Z2, U2),
      legend = c("Spatiul 1", "Spatiul 2"),
      fill = c("red", "blue")
    )
  },
  error = function(e) {
    stop(paste("Eroare biplot instante!", e))
  })
}

cerculCorelatiilor = function(X,
                              titlu = "",
                              titluX = "",
                              titluY = "") {
  windows(width = 9, height = 7)
  #dev.new(noRStudioGD = T)
  par(mai = c(1, 1, 1, 2), xpd = T)
  t = seq(from = 0,
          to = 2 * pi,
          length.out = 100)
  plot(
    x = cos(t),
    y = sin(t),
    main = titlu,
    xlab = titluX,
    ylab = titluY,
  type = "l"
 )
 m = length(X)
 c = rainbow(m)
 for (i in 1:m) {
  points(x = X[[i]][, 1],
         y = X[[i]][, 2],
         col = c[i])
  text(
   x = X[[i]][, 1],
   y = X[[i]][, 2],
   pos = 1,
   labels = rownames(X[[i]]),
   col = c[i]
  )
 }
 #legend(x = 1.1,y = 1,legend = names(X),fill = c)
 legend(
  x = 1.2,
  y = 1,
  legend = names(X),
  fill = c
 )
}
```

Seminar.R

```
Seminar = function(){
 set1 = xlsx::read.xlsx(file = "EnergieEU/BalantaEnergeticaEuropa.xlsx",
                sheetIndex = 2)
 set2 = xlsx::read.xlsx(file = "EnergieEU/BalantaEnergeticaEuropa.xlsx",
                    sheetIndex = 3)
 X = set1[3:6]
 Y = set2[3:6]
 rez = CCA::cc(X = X,Y = Y)
 #Afisare corelatii canonice
 write.csv(x = rez$cor,file = "CorelatiiCanonice.csv")
 #Variabile canonice (scoruri) prima grupa
 z = rez$scores$xscores
 #Variabile canonice (scoruri) grupa a doua
 u = rez$scores$yscores
 rownames(z) = set1[,1]
 rownames(u) = set1[,1]
 write.csv(x = z,file = "VariabileCanonice_z.csv",row.names = T)
 write.csv(x = u,file = "VariabileCanonice_u.csv",row.names = T)
 biPlotInstante(Z1 = z[,1],Z2 = z[,2],U1 = u[,1],U2 = u[,2],
               etichete = rownames(z))
 #Preluare corelatii dintre variabilele canonice ale grupei X
 #si variabilele X (z-X)
 rzx = rez$scores$corr.X.xscores
 #corelatii z-Y
 rzy = rez$scores$corr.X.yscores
 #corelatii U-Y
 ruy = rez$scores$corr.Y.yscores
 #corelatii U-X
 rux = rez$scores$corr.Y.xscores
 #Salvare corelatii
 corelatii = list(
  CorelatiiZX=rzx,
  CorelatiiZY=rzy,
  CorelatiiUY=ruy,
  CorelatiiUX=rux
 )
 cerculCorelatiilor(X = corelatii,"Corelatii variabile-variabile canonice",
                  titluX = "z1/u1",titluY = "z2/u2")
 write.csv(x = rzx,file = "Corelatii_Z-X.csv")
 write.csv(x = rzy,file = "Corelatii_Z-Y.csv")
 write.csv(x = ruy,file = "Corelatii_U-Y.csv")
 write.csv(x = rux,file = "Corelatii_U-X.csv")
}
```

Seminar5

Functii.R

```
testModel = function(X,g){
 tryCatch(
  {
   #Numar instante
   n = nrow(X)
   #Numar variabile
   m = ncol(X)
   #Determinare grupe
   grupe = levels(g)
   #Numarul de grupe
   q = length(grupe)
   #Testare globala model. Testul Wilks
   testModel = manova(as.matrix(X)~g)
   rezultatTest = summary(object = testModel,test="Wilks")
   frameTest = data.frame(
    c("Wilks","F calculat","PValue"),
    c(rezultatTest$stats[1,"Wilks"],rezultatTest$stats[1,"approx
F"],rezultatTest$stats[1,"Pr(>F)"])
   )
   colnames(frameTest)=c("Indicatori","Valori")
   write.csv(x = frameTest,file = "TestModel.csv",row.names = F)
   #Testare variabile predictor
   W = DiscriMiner::withinCov(variables = X,group = g)
   B = DiscriMiner::betweenCov(variables = X,group = g)
   VW = diag(W)
   VB = diag(B)
   FC = VB/VW
   PValues = pf(q = FC,df1 = q-1,df2 = n-q,lower.tail = F)
   frameTestVariabile = data.frame(Variabile=names(X),PValues=PValues)
   write.csv(x = frameTestVariabile,file = "TestVariabile.csv",row.names = F)

  },
  error = function(e){
   stop(paste("Eroare clasificare!!!",e))
  }
 )
}
```

Seminar5.R

```
seminar = function(){
 source("Functii.R")
 fisier = file.choose()
 setBaza = read.csv(file = fisier,row.names = 1)
 #Variabila de clasificare
 kls = select.list(choices = names(setBaza),
                 title = "Variabila de clasificare",
                 graphics = T)
 if(length(kls)==0){
  tcltk::tk_messageBox(message = "Nu ati selectat variabila de grupare!")
  return(NULL)
 }
 #Selectie variabile predictor
 k = select.list(choices = names(setBaza),
               title = "Variabile predictor",
               multiple = T,
                graphics = T)
  if(length(k)<2){
    tcltk::tk_messageBox(message = "Insuficiente variabile predictor!")
    return(NULL)
  }
  testModel(X = setBaza[k],g = setBaza[,kls])
}
```