Transformations to Normality
Principal Component Analysis (PCA)

Assignments:

Perform PCA for data set USArrests from the R data collection. Alternatively, you can use a data set from your research.

1. Find the best Box-Cox transformation for each variable;
2. Construct the Normal probability plots for the original and transformed variables. Discuss the quality and appropriateness of the transformations;
3. Perform PCA and find the loadings;
4. Compute principal components and find the correlations between the first PC and the original variables;
5. Discuss and interpret your findings;
6. Illustrate your analysis.

Reports: Printed reports are due on Thursday, April 6, 2017.

Report preparation: Consider each report as a mini-paper. It should not be long, but it should provide a reader with all background information about the data, problem, and methods you are using. Review the necessary theoretical material and describe the data. Do not insert the R-output in your report; instead, summarize it in tables or text in a nice readable form. If you feel some parts of the R-output or R-code should be included, put them in Appendix. Put your name on the title page. Illustrations should support your conclusions and make it easier to read a report.
# Transformations to Normality:

# Successful case 1

x <- rexp(1000)  # exponential sample with parameter 1

qqnorm(x)        # Normal probability plot for original variable
grid()

boxcox(x ~ 1)    # Illustration of Log-Likelihood profile

# Functions from package "car"
p <- powerTransform(x)  # Estimation of Box-Cox lambda
y <- bcPower(x, p$lambda)  # Box-Cox transformation
# ... or by hands
lambda <- 0.28
y <- (x^lambda - 1)/lambda

qqnorm(y)        # Normal probability plot for transformed variable
grid()

# Successful case 2

x <- rnorm(1000)^4  # sample

qqnorm(x)        # Normal probability plot for original variable
grid()

boxcox(x ~ 1)    # Illustration of Log-Likelihood profile
# Functions from package "car"
p <- powerTransform(x)  # Estimation of Box-Cox lambda
y <- bcPower(x, p$lambda)  # Box-Cox transformation
# ... or by hands
lambda <- 0.1
y <- (x^lambda - 1)/lambda

qqnorm(y)  # Normal probability plot for transformed variable
grid()

#==============================================
# Transformation to Normality: Unsuccessful case
#==============================================
x <- rnorm(1000)^3  # sample
x <- x - min(x) + sd(x)

qqnorm(x)  # Normal probability plot for original variable
grid()

boxcox(x~1)  # Illustration of Log-Likelihood profile

# Functions from package "car"
p <- powerTransform(x)  # Estimation of Box-Cox lambda
y <- bcPower(x, p$lambda)  # Box-Cox transformation
# ... or by hands
lambda <- 0.6
y <- (x^lambda - 1)/lambda

qqnorm(y)  # Normal probability plot for transformed variable
grid()

#==============================================
# Principal Component Analysis (PCA)
#==============================================
# This example shows how to perform PCA...
#==============================================
Sigma1 <- matrix(c(10, 2, 2, 2), 2, 2)
X1 <- mvrnorm(n = 1000, c(5, 9), Sigma1, empirical = TRUE)
S <- var(X1)
SA(X1)

p <- princomp(X1)  # PCA
summary(p)  # summary
p$sdev  # st.dev. of components
p$loadings  # coefficients of linear transformations
plot(p)  # scree plot

# .. and illustrates that PCA is equivalent to
# the spectral decomposition of the variance matrix
S <- var(X1)
e <- eigen(S)
e$values^ .5
e$vectors

# A simple 4D PCA example
Sigma1 <- diag(c(10, 6, 4, 1)^2)
X <- mvrnorm(n = 10000, c(0, 0, 0, 0), Sigma1, empirical = TRUE)
p <- princomp(X)
p$loadings
p$sdev
plot(p)

# A bit more elaborate 4D PCA example
Sigma1 <- matrix(c(10, 2, 2, 2), 2, 2)
X1 <- mvrnorm(n = 1000, c(0, 0), Sigma1)
X2 <- mvrnorm(n = 1000, c(0, 0), 2 * Sigma1)
X <- cbind(X1, X2)
p <- princomp(X)
p$loadings
p$sdev
plot(p)

# Real data PCA example
T <- read.table('Lab3_close.csv', sep = ',', header = TRUE)
len <- dim(T)[1]  # length
Itime <- seq(len, 1, by = -1)  # inverse index for plots

time <- 1989 + (31 + 28 + 31 + 30 + 4) / 365.25 + seq(1, len) / 250

names(T)  # names of variables
P<-T[,seq(2,11)]     # remove the dates
P<-log10(P)          # log transform
names(P)             # names of stocks
plot(time,P[Itime,1],type='l',ylim=c(-2,2))# plot several log-time series
#plot(time,P[Itime,1],type='l',ylim=c(0,100)) # plot several time series
points(time,P[Itime,3],type='l',col=3)     # ...
points(time,P[Itime,4],type='l',col=4)     # ...
points(time,P[Itime,5],type='l',col=5)     # ...

do=princomp(P,cor=TRUE)                   # PCA
do$loadings                      # loadings
do$sdev                          # st. dev. of components
plot(do)                          # scree plot

P1<-as.matrix(P)**diag(1/do$scale)     # normalize data
L1<-as.matrix(do$loadings)[,1]        # loadings for PC1
L2<-as.matrix(do$loadings)[,2]        # loadings for PC2
L3<-as.matrix(do$loadings)[,3]        # loadings for PC3
Y1<-P1%*%L1                          # PC1
Y2<-P1%*%L2                          # PC2
Y3<-P1%*%L3                          # PC3
plot(time,-Y1[Itime],type='l',ylim=c(-5,20),col=1) # plot PC1
points(time,-Y2[Itime],type='l',col=2)     # ... and PC2
points(time,Y3[Itime],type='l',col=3)     # ... and PC3

SA(cbind(Y3-mean(Y3),-Y2+mean(Y2)))

# Function that illustrates spectral decomposition
# and statistical distance ellipses
#===============================================
SA <- function(X,add=FALSE,data.plot=TRUE)
{
  # Vector of means
  n<-dim(X)[1]
ones<-matrix(rep(1,n),ncol=1)
u<-as.vector(t(X) ** ones / n)

  # Variance
  Sigma<-var(X)

e<-eigen(Sigma)
par(bg='yellow')

}}
```r
ellipse(mu, Sigma, 3, add = TRUE, xlim = range(X), ylim = range(X))
ellipse(mu, Sigma, 2, add = TRUE)
ellipse(mu, Sigma, 1, add = TRUE)

if (data.plot)
  points(X[,1], X[,2], pch = 20, col = 4)
  arrows(mu[1], mu[2], mu[1] + e$vectors[1,1] * sqrt(e$values[1]),
         mu[2] + e$vectors[2,1] * sqrt(e$values[1]), length = .1, col = 'green', lwd = 2)
  arrows(mu[1], mu[2], mu[1] + e$vectors[1,2] * sqrt(e$values[2]),

e <- ellipse

function(mu, Sigma, R, col = 'red', add = FALSE, xlim = NULL, ylim = NULL, N = 1000)
{
  # Find coordinates of a circle
  t <- seq(0, 2*pi, length.out = N)
  x <- R * cos(t)
  y <- R * sin(t)

  # Spectral decomposition of Sigma
  e <- eigen(Sigma)  # spectral decomposition
  P <- e$vectors     # eigenvectors
  L <- e$values

  # Square root matrix
  S05 <- P %*% sqrt(diag(L)) %*% t(P)

  # Ellipse coordinates
  vec <- cbind(x, y)
  vec <- t(vec %*% S05)
  x <- vec[1,] + mu[1]
  y <- vec[2,] + mu[2]

  if (add)
    points(x, y, type = 'l', col = col)
  else
    plot(x, y, type = 'l', col = col, xlim = xlim, ylim = ylim)
}
```