Homework 3 Code

Jared Yu

April 19, 2019

```r
library(mixtools); library(latex2exp)
# https://stats.stackexchange.com/questions/153564/visualizing-pca-in-r-data-points-eigenvectors-projec
# https://stats.stackexchange.com/questions/9898/how-to-plot-an-ellipse-from-eigenvalues-and-eigenvecto
# 1
mat1 = matrix(c(2,2,2,0,-1,-2,-3,1,-2,1,0,1,-2,1), nrow = 7)
n = 7
apply(mat1, 2, mean)
apply(mat1, 2, var)

sum(mat1[,1]*mat1[,2])/(n-1)

(t(mat1)%*%mat1)/(n-1)

# 3
mixtools::ellipse(mu = c(1,-1),
                  sigma = matrix(c(6,2,2,6), ncol = 2),
                  newplot = TRUE, type = "l",
                  xlim = c(-7, 8), ylim = c(-8, 7),
                  main = TeX('$(x-\\bar{x})^TS^{-1}(x-\\bar{x})\\leq c^2$'),
                  xlab = TeX('$x_1$'), ylab = TeX('$x_2$'),
                  cex.lab=1.5, cex.main=1.5, cex.sub=1.5)
points(1, -1)
abline(h = 0, v = 0)
abline(a = -2, b = 1, lty = 2)
abline(a = 0, b = -1, lty = 2)

mixtools::ellipse(mu = c(0,0),
                  sigma = matrix(c(1,1/3,1/3,1), ncol = 2),
                  newplot = TRUE, type = "l",
                  xlim = c(-3, 3), ylim = c(-3, 3),
                  main = TeX('$x^TR^{-1}x\\leq c^2$'),
                  xlab = TeX('$x_1$'), ylab = TeX('$x_2$'),
                  cex.lab=1.5, cex.main=1.5, cex.sub=1.5)
points(0, 0)
abline(h = 0, v = 0)
abline(a = 0, b = 1, lty = 2)
abline(a = 0, b = -1, lty = 2)

ctr    <- c(1, -1)                               # data centroid -> colMeans(dataMatrix)
A      <- matrix(c(6, 2, 2, 6), nrow=2) # covariance matrix -> cov(dataMatrix)
RR     <- chol(A)                              # Cholesky decomposition
angles <- seq(0, 2*pi, length.out=200)         # angles for ellipse
ell    <- 1 * cbind(cos(angles), sin(angles)) %*% RR  # ellipse scaled with factor 1
ellCtr <- sweep(ell, 2, ctr, "+")              # center ellipse to the data centroid
plot(ellCtr, type="l", lwd=2, asp=1)           # plot ellipse
points(ctr[1], ctr[2], pch=4, lwd=2)           # plot data centroid
```

```r
ellipse(c(1, -1), shape=A, radius=0.98, col="red", lty=2)

eigVal  <- eigen(A)$values
eigVec  <- eigen(A)$vectors
eigScl  <- eigVec %*% diag(sqrt(eigVal))  # scale eigenvectors to length = square-root
xMat    <- rbind(ctr[1] + eigScl[1, ], ctr[1] - eigScl[1, ])
yMat    <- rbind(ctr[2] + eigScl[2, ], ctr[2] - eigScl[2, ])
ellBase <- cbind(sqrt(eigVal[1])*cos(angles), sqrt(eigVal[2])*sin(angles)) # normal ellipse
ellRot  <- eigVec %*% t(ellBase)                                          # rotated ellipse
plot((ellRot+ctr)[1, ], (ellRot+ctr)[2, ], asp=1,
     type="l", lwd=2, main = TeX('$(x-\\bar{x})^TS^{-1}(x-\\bar{x})\\leq c^2$'),
     xlab = TeX('$X_1$'), ylab = TeX('X_2'))
matlines(xMat, yMat, lty=1, lwd=2, col="green")
points(ctr[1], ctr[2], pch=4, col="red", lwd=3)
abline(h = 0, v = 0)

ctr    <- c(0, 0)                               # data centroid -> colMeans(dataMatrix)
A      <- matrix(c(1, 1/3, 1/3, 1), nrow=2) # covariance matrix -> cov(dataMatrix)
RR     <- chol(A)                               # Cholesky decomposition
angles <- seq(0, 2*pi, length.out=200)          # angles for ellipse
ell    <- 1 * cbind(cos(angles), sin(angles)) %*% RR  # ellipse scaled with factor 1
ellCtr <- sweep(ell, 2, ctr, "+")               # center ellipse to the data centroid
plot(ellCtr, type="l", lwd=2, asp=1)            # plot ellipse
points(ctr[1], ctr[2], pch=4, lwd=2)            # plot data centroid


ellipse(c(0, 0), shape=A, radius=0.98, col="red", lty=2)
eigVal  <- eigen(A)$values
eigVec  <- eigen(A)$vectors
eigScl  <- eigVec %*% diag(sqrt(eigVal))  # scale eigenvectors to length = square-root
xMat    <- rbind(ctr[1] + eigScl[1, ], ctr[1] - eigScl[1, ])
yMat    <- rbind(ctr[2] + eigScl[2, ], ctr[2] - eigScl[2, ])
ellBase <- cbind(sqrt(eigVal[1])*cos(angles), sqrt(eigVal[2])*sin(angles)) # normal ellipse
ellRot  <- eigVec %*% t(ellBase)                                          # rotated ellipse
plot((ellRot+ctr)[1, ], (ellRot+ctr)[2, ], asp=1,
     type="l", lwd=2, main = TeX('$x^TR^{-1}x\\leq c^2$'),
     xlab = TeX('$X_1$'), ylab = TeX('$X_2$'))
matlines(xMat, yMat, lty=1, lwd=2, col="green")
points(ctr[1], ctr[2], pch=4, col="red", lwd=3)
abline(h = 0, v = 0)


(1 + 0.05/4)^4 - 1
```