#---------------

#

# SURE methods

# JV + NL

#

#---------------

library(ICSNP)

library(ICtest)

library(changepoint)

# R1k-version of the robust SURE from the paper

# Can estimate the dimension if it is between 0 and p - 1

#

# x = n x p data matrix

# type = either "Cov" or "SSCM", signifying the scatter matrix to be used.

# full = if FALSE, return the estimated latent dimension. If TRUE, returns the "SURE-curve"

sure\_1 <- function(x, type, full = FALSE){

if(type == "Cov"){

return(sure\_2(x, "Cov", full))

}

if(type == "SSCM"){

n <- nrow(x)

p <- ncol(x)

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

inverse\_norms <- apply(x\_centered, 1, function(v) 1/sqrt(sum(v^2)))

z <- sweep(x\_centered, 1, inverse\_norms, "\*")

eig\_SSCM <- eigen(cov(z)\*(n - 1)/n)

lambda <- eig\_SSCM$values

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- sum(lambda[(k + 1):p])

}

res <- res - p\*lambda[p]

# Derivative computation

Z <- rowMeans(sapply(1:n, function(i){

AA <- tcrossprod(z[i, ])

AA%x%AA

}))

dim(Z) <- c(p^2, p^2)

U <- eig\_SSCM$vectors

G1 <- rowMeans(sapply(1:n, function(i) z[i, ]%\*%t(z[i, ]%x%z[i, ])))

dim(G1) <- c(p, p^2)

A <- array(0, dim = c(p, p, n))

for(i in 1:n){

A[, , i] <- inverse\_norms[i]\*(diag(p) - tcrossprod(z[i, ]))

}

G2 <- rowMeans(sapply(1:n, function(i) z[i, ]%x%A[, , i]))

dim(G2) <- c(p^2, p)

Gtemp <- G2%\*%solve(apply(A, 1:2, sum))%\*%G1

for(k in 0:(p - 1)){

deriv\_sum <- 0

deriv\_sum <- deriv\_sum + n\*k + p - k

if(k != 0){

ellm\_sum <- 0

for(ell in 1:k){

for(m in (k + 1):p){

temp <- lambda[ell] + lambda[m]

ull <- U[, ell]%x%U[, ell]

ulm <- U[, ell]%x%U[, m]

uml <- U[, m]%x%U[, ell]

umm <- U[, m]%x%U[, m]

temp <- temp - 4\*t(ull)%\*%Z%\*%umm

temp <- temp + (2/n)\*t(uml)%\*%Gtemp%\*%uml + (2/n)\*t(ulm)%\*%Gtemp%\*%ulm

temp <- temp/(lambda[ell] - lambda[m])

ellm\_sum <- ellm\_sum + temp

}

}

deriv\_sum <- deriv\_sum + ellm\_sum

}

deriv\_sum <- 2\*lambda[p]\*deriv\_sum/n

res[k + 1] <- res[k + 1] + deriv\_sum

}

if(full){

return(cbind(0:(p - 1), res))

}

if(!full){

return(which.min(res) - 1)

}

}

}

# R2k-version of the robust SURE from the paper

# Can estimate the dimension if it is between 0 and p - 1

#

# x = n x p data matrix

# type = either "Cov", "SSCM", "Tyler" or "HR", signifying the scatter matrix to be used.

# full = if FALSE, return the estimated latent dimension. If TRUE, returns the "SURE-curve"

sure\_2 <- function(x, type, full = FALSE){

n <- nrow(x)

p <- ncol(x)

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "SSCM"){

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

x\_no\_signs <- sweep(x\_centered, 1, apply(x\_centered, 1, function(v) sqrt(sum(v^2))), "/")

lambda <- eigen(cov(x\_no\_signs)\*(n - 1)/n)$values

}

if(type == "Tyler"){

x\_median <- spatial.median(x)

lambda <- eigen(tyler.shape(x, location = x\_median))$values

}

if(type == "HR"){

lambda <- eigen(HR.Mest(x)$scatter)$values

}

# Estimate the noise variance

s2 <- rep(lambda[p], p)

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- sum(lambda[(k + 1):p])

}

res <- res + s2\*(2\*p + 2\*(n - 1)\*(0:(p - 1)) - n\*p)/n

diff\_mat <- matrix(0, p, p)

for(i in 1:(p - 1)){

for(j in (i + 1):p){

diff\_mat[i, j] <- (lambda[i] + lambda[j])/(lambda[i] - lambda[j])

}

}

for(k in 1:(p - 1)){

res[k + 1] <- res[k + 1] + 2\*s2[k + 1]\*sum(diff\_mat[1:k, (k + 1):p])/n

}

if(full){

return(cbind(0:(p - 1), res))

}

if(!full){

return(which.min(res) - 1)

}

}

# R3k-version of the robust SURE from the paper

# Can estimate the dimension if it is between 0 and p - 1

#

# x = n x p data matrix

# type = either "Cov", "SSCM", "Tyler" or "HR", signifying the scatter matrix to be used.

# full = if FALSE, return the estimated latent dimension. If TRUE, returns the "SURE-curve"

sure\_3 <- function(x, type, full = FALSE){

p <- ncol(x)

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "SSCM"){

n <- nrow(x)

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

x\_no\_signs <- sweep(x\_centered, 1, apply(x\_centered, 1, function(v) sqrt(sum(v^2))), "/")

lambda <- eigen(cov(x\_no\_signs)\*(n - 1)/n)$values

}

if(type == "Tyler"){

x\_median <- spatial.median(x)

lambda <- eigen(tyler.shape(x, location = x\_median))$values

}

if(type == "HR"){

lambda <- eigen(HR.Mest(x)$scatter)$values

}

# Estimate the noise variance

s2 <- rep(lambda[p], p)

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- sum(lambda[(k + 1):p])

}

res <- res + s2\*(2\*(0:(p - 1)) - p)

if(full){

return(cbind(0:(p - 1), res))

}

if(!full){

return(which.min(res) - 1)

}

}

# Asymptotic test for the determination of the latent dimension

# Note: the code is partially taken from the function PCAasymp in the package ICtest written by Klaus Nordhausen

h0\_asymp <- function(x, type){

if(type == "Cov"){

scatter <- "cov"

}

if(type == "HR"){

scatter <- "tyler"

}

X <- as.matrix(x)

p <- ncol(X)

n <- nrow(X)

scatter <- match.arg(scatter, c("cov", "tyler"))

if (scatter == "cov") {

MEAN <- colMeans(X)

Xc <- sweep(X, 2, MEAN, "-")

COV <- crossprod(Xc)/n

MAHA <- rowSums(Xc %\*% solve(COV) \* Xc)

sigma1 <- (mean(MAHA^2))/(p \* (p + 2))

}

else {

TYLER <- HR.Mest(X)

MEAN <- TYLER$center

COV <- TYLER$scatter

Xc <- sweep(X, 2, MEAN, "-")

sigma1 <- (p + 2)/p

}

EV <- eigen(COV, symmetric = TRUE)

EV.VALUES <- EV$values

k <- -1

found <- FALSE

while(!found){

k <- k + 1

ind <- (k + 1):p

MEANev <- mean(EV.VALUES[ind])

VAR.K <- sum((EV.VALUES[ind] - MEANev)^2)/(p - k)

TESTSTATISTIC <- n \* VAR.K \* (p - k)/(2 \* sigma1 \* MEANev^2)

names(TESTSTATISTIC) = "T"

PARAMETER <- (p - k - 1) \* (p - k + 2)/2

PVAL <- 1 - pchisq(TESTSTATISTIC, df = PARAMETER)

if(PVAL >= 0.05){

found <- TRUE

}

}

return(k)

}

# Forward stepwise hypothesis testing with bootstrapped null distributions

# Can estimate the dimension if it is between 1 and p - 2

#

# x = n x p data matrix

# type = either "Cov", "SSCM", "Tyler" or "HR", signifying the scatter matrix to be used.

# n.boot = number of bootstrap sample, default is 200

h0\_boot <- function(x, type, n.boot = 200){

if(type == "Cov"){

LS <- function(x){

list(colMeans(x), cov(x))

}

}

if(type == "SSCM"){

LS <- function(x){

n <- nrow(x)

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

x\_no\_signs <- sweep(x\_centered, 1, apply(x\_centered, 1, function(v) sqrt(sum(v^2))), "/")

list(x\_median, cov(x\_no\_signs)\*(n - 1)/n)

}

}

if(type == "Tyler"){

LS <- function(x){

x\_median <- spatial.median(x)

list(x\_median, tyler.shape(x, location = x\_median))

}

}

if(type == "HR"){

LS <- function(x){

HR <- HR.Mest(x)

list(HR$center, HR$scatter)

}

}

p <- ncol(x)

for(k in 1:(p - 2)){

if(PCAboot(x, k, n.boot = n.boot, S = LS)$p.value >= 0.05){

break

}

}

return(k)

}

# Ladle estimator with a chosen scatter matrix

# Can estimate the dimension if it is between 0 and p

#

# x = n x p data matrix

# type = either "Cov", "SSCM", "Tyler" or "HR", signifying the scatter matrix to be used.

# n.boot = number of bootstrap sample, default is 200

ladle\_wrapper <- function(x, type, n.boot = 200){

if(type == "Cov"){

S <- function(x){

cov(x)

}

}

if(type == "SSCM"){

S <- function(x){

n <- nrow(x)

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

x\_no\_signs <- sweep(x\_centered, 1, apply(x\_centered, 1, function(v) sqrt(sum(v^2))), "/")

cov(x\_no\_signs)\*(n - 1)/n

}

}

if(type == "Tyler"){

S <- function(x){

x\_median <- spatial.median(x)

tyler.shape(x, location = x\_median)

}

}

if(type == "HR"){

S <- function(x){

HR <- HR.Mest(x)

HR$scatter

}

}

ladle(x, S, n.boots = n.boot)$k

}

# Changepoint version of SURE2

sure\_2\_cp <- function(x, type, full = FALSE){

n <- nrow(x)

p <- ncol(x)

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "SSCM"){

x\_median <- spatial.median(x)

x\_centered <- sweep(x, 2, x\_median, "-")

x\_no\_signs <- sweep(x\_centered, 1, apply(x\_centered, 1, function(v) sqrt(sum(v^2))), "/")

lambda <- eigen(cov(x\_no\_signs)\*(n - 1)/n)$values

}

if(type == "Tyler"){

x\_median <- spatial.median(x)

lambda <- eigen(tyler.shape(x, location = x\_median))$values

}

if(type == "HR"){

lambda <- eigen(HR.Mest(x)$scatter)$values

}

# Estimate the noise variance

s2 <- rep(lambda[p], p)

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- sum(lambda[(k + 1):p])

}

res <- res + s2\*(2\*p + 2\*(n - 1)\*(0:(p - 1)) - n\*p)/n

diff\_mat <- matrix(0, p, p)

for(i in 1:(p - 1)){

for(j in (i + 1):p){

diff\_mat[i, j] <- (lambda[i] + lambda[j])/(lambda[i] - lambda[j])

}

}

for(k in 1:(p - 1)){

res[k + 1] <- res[k + 1] + 2\*s2[k + 1]\*sum(diff\_mat[1:k, (k + 1):p])/n

}

if(full){

return(cbind(0:(p - 1), res))

}

if(!full){

temp <- diff(res)

cp\_test <- suppressWarnings(cpt.meanvar(temp, method = "BinSeg", Q = 1))

return(c(cp\_test@cpts.full))

}

}

# AIC-estimate from Wax (1985)

#

# x = n x p data matrix

# type = either "Cov" or "HR", signifying the scatter matrix to be used.

wax\_aic <- function(x, type){

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "HR"){

HR <- HR.Mest(x)

lambda <- eigen(HR$scatter)$values

}

n <- nrow(x)

p <- ncol(x)

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- -2\*(p - k)\*n\*(mean(log(lambda[(k + 1):p])) - log(mean(lambda[(k + 1):p]))) + 2\*k\*(2\*p - k)

}

return(which.min(res) - 1)

}

# MDL-estimate from Wax (1985)

#

# x = n x p data matrix

# type = either "Cov" or "HR", signifying the scatter matrix to be used.

wax\_mdl <- function(x, type){

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "HR"){

HR <- HR.Mest(x)

lambda <- eigen(HR$scatter)$values

}

n <- nrow(x)

p <- ncol(x)

res <- rep(0, p)

for(k in 0:(p - 1)){

res[k + 1] <- -1\*(p - k)\*n\*(mean(log(lambda[(k + 1):p])) - log(mean(lambda[(k + 1):p]))) + 0.5\*k\*(2\*p - k)\*log(n)

}

return(which.min(res) - 1)

}

# Laplace-estimate from Minka (2000)

#

# x = n x p data matrix

# type = either "Cov" or "HR", signifying the scatter matrix to be used.

minka <- function(x, type){

if(type == "Cov"){

lambda <- eigen(cov(x))$values

}

if(type == "HR"){

HR <- HR.Mest(x)

lambda <- eigen(HR$scatter)$values

}

n <- nrow(x)

p <- ncol(x)

res <- rep(0, p - 1)

for(k in 1:(p - 1)){

logpU <- -1\*k\*log(2) + sum(lgamma(0.5\*(p - 1:k + 1))) - sum(0.5\*(p - 1:k + 1))\*log(pi)

logprod <- -1\*(n/2)\*sum(log(lambda[1:k]))

v <- mean(lambda[(k + 1):p])

logv <- -1\*n\*0.5\*(p - k)\*log(v)

m <- p\*k - k\*(k + 1)/2

logpi <- 0.5\*(m + k)\*log(2\*pi)

lambda\_hat <- lambda

lambda\_hat[(k + 1):p] <- v

l\_sum <- 0

for(i in 1:k){

for(j in (i + 1):p){

l\_sum <- l\_sum + log((lambda[i] - lambda[j])\*(1/lambda\_hat[j] - 1/lambda\_hat[i])) + log(n)

}

}

logAz <- -0.5\*l\_sum

logN <- -1\*0.5\*k\*log(n)

res[k] <- logpU + logprod + logv + logpi + logAz + logN

}

return(which.max(res))

}

#-------

# An example

#-------

# Simulates data from multivariate t-distribution

# n = sample size

# p = dimension

# d = latent dimension

# s2\_signal = d-vector of signal variances

# s2\_noise = noise variance (scalar)

# df = degrees of freedom

simulate\_t\_data <- function(n, p, d, s2\_signal, s2\_noise, df){

y <- rmvt(n, sigma = diag(p), df = df)

dvec <- rep(sqrt(s2\_noise), p)

dvec[1:d] <- sqrt(s2\_signal)

y <- sweep(y, 2, dvec, "\*")

v <- rorth(p)

y%\*%t(v)

}