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

# Object shape and the peeling plot

# JV

# Object shape

# D = distance matrix

# is.squared = whether the elements of D are squared distances or not

msphe <- function(D, is.squared = FALSE){

if(!is.squared){

D <- D^2

}

n <- nrow(D)

sum(D%\*%D)/(n\*sum(D^2))

}

# Peeling estimator

# D = distance matrix (with elements being non-squared distances)

# min = TRUE/FALSE, whether we want to minimize (find lines) or maximize (find sphericity)

peel <- function(D, min = TRUE){

n <- nrow(D)

D <- D^2

res <- matrix(0, n, 2)

list\_of\_elem <- 1:n

removed\_elem <- NULL

n\_tail <- floor(0.1\*n)

for(i in 1:(n - n\_tail)){

remaining\_elem <- setdiff(list\_of\_elem, removed\_elem)

# length(remaining\_elem) = n - i + 1

temp <- cbind(remaining\_elem, rep(0, n - i + 1))

for(j in 1:(n - i + 1)){

temp[j, 2] <- msphe(D[remaining\_elem[-j], remaining\_elem[-j]], is.squared = TRUE)

}

if(min){

min\_ind <- which.min(temp[, 2])[1]

}

if(!min){

min\_ind <- which.max(temp[, 2])[1]

}

res[i, ] <- temp[min\_ind, ]

removed\_elem <- c(removed\_elem, remaining\_elem[min\_ind])

}

remaining\_elem <- setdiff(list\_of\_elem, removed\_elem)

for(j in 0:(n\_tail - 1)){

res[n - j, ] <- c(remaining\_elem[j + 1], res[n - n\_tail, 2])

}

res

}

# Peeling plot

#

# peel\_obj = an object returned by the function peel()

peel\_plot <- function(peel\_obj){

n <- nrow(peel\_obj)

plot(1:n, peel\_obj[, 2], type = "l")

}