MDS Report

Team 5

2/7/2022

1. Data

To compare our implementation of the multidimensional scaling algorithm to that of the `smacof` package we use the `basket` data that contains co-purchases of 12 grocery store items. The provided data are co-purchases, thus similarities, and hence we first need to re-calculate these into dissimilarities. We choose to do so using the $\delta_{ij} = log\left(\frac{s_{ii}s_{jj}}{s_{ij}s_{ji}}\right)$ formulation in order to get a more uniform distribution of the obtained dissimilarity values. The prepossessing of our data can be seen in a code chunk below:

```r
#------
# DATA |
#------
load("basket.Rdata")
basket <- as.matrix(basket)
rownames(basket) <- colnames(basket)

#transform data into dissimilarities using the log transformation
basketDis <- log(outer(diag(basket), diag(basket)) / (basket * t(basket)))
```

2. Comparison & Discussion

In order to compare our implementation with the `smacof` package we initialize both mds() functions with the same initial matrix of 2-dimensional co-ordinates. This is done to ensure that the same (potentially local) minima are achieved by majorization of the normalized stress function in both instances. Figure 1 shows the final configuration of both functions, with our implementation on the right, and `smacof` package on the left. We can see that the results are the same up to the scale of the dimensions of the two plots. Conceptually, the discrepancy most probably comes from some sort of normalization that is being done to the columns of the configuration matrix in the `smacof` package. However, since the dimensions are not really interpretable, this is not a big issue and the discrepancy can technically be ignored. More importantly, both implementations reach the same minimal stress value thus we can be certain that the solutions to our problem are identical. The full code of our implementation can be seen in the appendix.

```
## The optimized stress from smacof is: 0.2542
```

```
## The optimized stress from our implementation is: 0.2542
```

```
## The difference is: -0.000003544
```

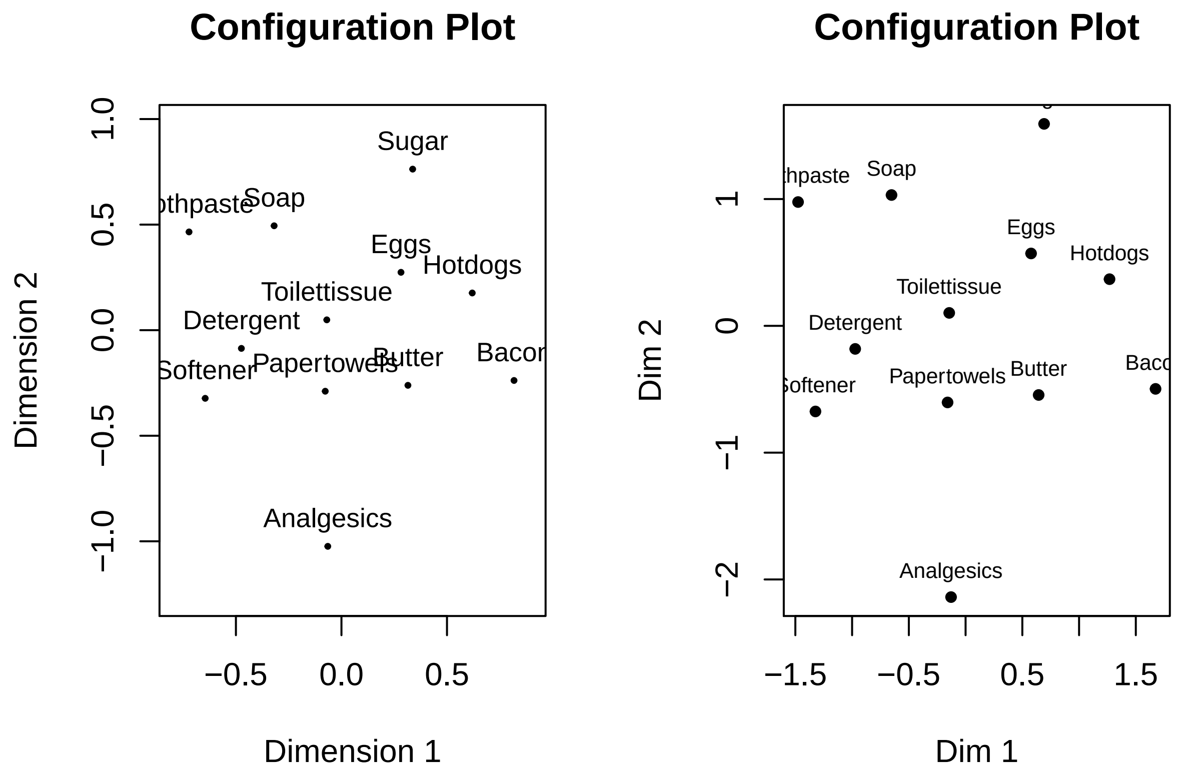

Figure 1: Smacof package (left) vs. our implementation (right)

4. Appendix

```r
## ---------------------------
## Script Name: MDS_Lib
## Author: Tomas Miskov
## Date Created: 2022-02-01
## Purpose: Implement Multi-Dimensional Scaling algorithm by majorization
## ----------------------------------------------------------------------

#' Euclidean Distance Matrix Function
#'
#' Compute distance matrix N x N of input matrix N x p
#'
#' @param mX N x p matrix X
#' @param bLowerTriang If TRUE returns only the lower triangular as a vector
fnEucDistMatrix <- function(mX, bLowerTriang = TRUE){

  mDistX <- apply(mX, 1, function(x) apply(mX, 1, function(y) sqrt(sum((x-y)^2))))

  if(bLowerTriang){
    return(mDistX[lower.tri(mDistX)])
  } else{
    return(mDistX)
  }
}

#--------------------------------------------------------------------------
#' Lower Triangular to symmetric matrix
#'
#' Compute symmetric matrix with 0s on the diagonal given its lower triangular
#' as a vector
#'
#' @param vX Lower triangular as a vector
#' @param iN Dimension of the output matrix
fnLowTriToMatrix <- function(vX, iN){
  mX <- matrix(0, ncol = iN, nrow = iN)   #initialize the output matrix
  mX[lower.tri(mX)] <- vX                 #assign lower triangular
  mX <- t(mX)                             #transpose
  mX[lower.tri(mX)] <- vX                 #assign lower triangular (again)

  return(mX)
}

#--------------------------------------------------------------------------
#' Raw Stress Function
#'
#' Compute raw stress of the dissimilarity matrix and the corresponding
#' distance matrix
#'
#' @param mDS The input dissimilarity matrix
#' @param mX N x p matrix of MDS distances between observations in X
#' @param vW Vector of weights, defaults to vector of 1s
fnRawStress <- function(mDS, mX, vW = rep(1, sum(seq(1,nrow(mX)-1)))){
  #extract the lower triangular as a vector
  mDS.lowTriang <- mDS[lower.tri(mDS)]

  #compute euclidean distances of mX
  mX.eucDist.lowTriang <- fnEucDistMatrix(mX, bLowerTriang = TRUE)

  #compute the raw stress score
  dRawStress <- t(vW) %*% (mDS.lowTriang - mX.eucDist.lowTriang)^2

  return(dRawStress)
}

#---------------------------------------------------------------------------
#' Kruskal's Stress Function
#'
#' Compute normalized Kruskal's stress
#'
#' @param mDS The input dissimilarity matrix
#' @param mX N x p matrix of MDS distances between observations in X
#' @param vW Vector of weights, defaults to vector of 1s
fnKruskalStress <- function(mDS, mX, vW = rep(1, sum(seq(1,nrow(mX)-1)))){

  #compute raw stress
  dRawStress <- fnRawStress(mDS, mX, vW)

  #compute distances in mX
  mX.eucDist.lowTriang <- fnEucDistMatrix(mX, bLowerTriang = TRUE)

  dKruskalStress <- sqrt(dRawStress / t(vW) %*% (mX.eucDist.lowTriang^2))

  return(dKruskalStress)
}

#---------------------------------------------------------------------------
#' Normalized Stress Function
#'
#' Compute normalized Normalized stress
#'
#' @param mDS The input dissimilarity matrix
#' @param mX N x p matrix of MDS distances between observations in X
#' @param vW Vector of weights, defaults to vector of 1s
fnNormalStress <- function(mDS, mX, vW = rep(1, sum(seq(1,nrow(mX)-1)))){

  #compute raw stress
  dRawStress <- fnRawStress(mDS, mX, vW)
  dNormalStress <- sqrt(dRawStress / t(vW) %*% (mDS[lower.tri(mDS)]^2))

  return(dNormalStress)
}

#---------------------------------------------------------------------------
#' MDS Majorization Function
#'
```

```r
#' Majorize the stress loss function to obtain MDS given a dissimilarity matrix
#'
#' @param mDS The input dissimilarity matrix
#' @param vW Vector of weights, defaults to vector of 1s
#' @param iP Number of dimensions for the MDS output
#' @param dEpsilon Precision parameter
#' @param mInit Optional matrix of the initial configuration
#' @param bSilent If FALSE, the iterations are printed, default is TRUE
fnMDS <- function(mDS, vW = rep(1, sum(seq(1,nrow(mX)-1))), iP,
                        dEpsilon = 1e-6, mInit = NULL, bSilent = TRUE){
  iN <- nrow(mDS)
  if(is.null(mInit)){                   #initialize X if none is given
    mInit <- matrix(runif(iN*iP, -1, 1), nrow = iN, ncol = iP)
  }
  mX <- scale(mInit, scale = FALSE)     #column center initial X
  dStress <- fnNormalStress(mDS, mX, vW)   #compute initial stress value
  dStressDelta <- dStress               #temporary delta stress value

  mJ <- diag(iN) - iN^-1 * (rep(1, iN) %*% t(rep(1, iN)))  #centering matrix
  mV <- iN * mJ                                            #Laplacian matrix
  mGenInvV <- matrix(iN^-1, nrow = iN, ncol = iN)          #computing the generalized
  mVinv <- solve(mV + mGenInvV) - mGenInvV                 #inverse of the Laplacian

  k <- 1
  while((k == 1) || (dStressDelta > dEpsilon)){
    k <- k + 1
    mY <- mX

    #COMPUTING THE B MATRIX
    mW <- fnLowTriToMatrix(vW, iN)                  #transforming lower.tri to matrix
    mF <- mW * mDS * fnEucDistMatrix(mY, FALSE)^-1 #computing the intermediate F matrix
    mF[is.na(mF)] <- 0                              #setting NAs to 0
    mBy <- diag(rowSums(mF)) - mF                   #computing B matrix

    #COMPUTING UPDATE FOR X MATRIX
    mX <- mVinv %*% mBy %*% mY                      #updating X

    #COMPUTING NEW STRESS
    dStressY <- fnNormalStress(mDS, mY, vW)        #normalized stress iter (k-1)
    dStressX <- fnNormalStress(mDS, mX, vW)        #normalized stress iter (k)
    dStressDelta <- dStressY - dStressX             #normalized stress delta

    if(!bSilent){
      cat("Normalized stress value in iteration", k, "is:", dStressX)
      cat("\nStress improved from previous interation by:", dStressDelta, "\n\n")
    }
  }
  return(list("conf" = mX, "confdist" = fnEucDistMatrix(mX),
              "stress" = fnNormalStress(mDS, mX, vW)[[1]],
              "iter" = k, "init" = mInit))
}

## ---------------------------
## Script Name: MDS_Main
## Author: Tomas Miskov
## Date Created: 2022-02-01
## Purpose: Compare own MDS package with `smacof`
## ---------------------------------------------

#--------
# SET UP |
#--------
rm(list=ls())                                         # clean the environment
if (!require("pacman")) install.packages("pacman")    # install pacman
pacman::p_load(ggplot2, tidyverse, smacof)            # pre-load packages
source("MDS_Lib.R")                                   # load local libraries

options(scipen = 6, digits = 4)                       # clean numerical notation

#------
# DATA |
#------
load("basket.Rdata")
basket <- as.matrix(basket)
rownames(basket) <- colnames(basket)

#transform data into dissimilarities using the log transformation
basketDis <- log(outer(diag(basket), diag(basket)) / (basket * t(basket)))

#------------
# BASKET MDS |
#------------
set.seed(55)
iN <- nrow(basketDis)
iP <- 2
mInit <- mInit <- matrix(runif(iN*iP, -1, 1), nrow = iN, ncol = iP) #initial X
mds1 <- mds(basketDis, init = mInit)                                #smacof
mds2 <- fnMDS(basketDis, iP = 2, mInit = mInit, bSilent = TRUE)    #ours

cat("The optimized stress from smacof is:", mds1$stress)
cat("The optimized stress from our implementation is:", mds2$stress)
cat("The difference is:", mds2$stress - mds1$stress)

par(mfrow = c(1, 2))
plot(mds1)
plot(scale(mds2$conf), pch = 20, xlab = "Dim 1", ylab = "Dim 2",
     main = "Configuration Plot")
text(scale(mds2$conf), labels=colnames(basket), cex = 0.7, pos = 3)
```