DiSTATIS

Richa Singh

11/13/2018

## DiSTATIS

DiSTATIS is a procedure that combines bootstrap estimation (to estimate the variability of the experimental conditions) and a new 3-way extension of MDS, that can be used to integrate the distance matrices generated by the bootstrap procedure and to represent the results as MDS-like maps. Reliability estimates are expressed as (1) tolerance intervals which reflect the accuracy of the assignment of scans to experimental categories and as (2) confidence intervals which generalize standard hypothesis testing.

The purpose of this study is to determine how musically trained and untrained listeners sort Western classical melodies into clusters based on perceived similarities. Listeners at three expertise levels sorted MIDI and natural excerpts from the piano music of Bach, Mozart, and Beethoven. We analyzed the data using DISTATIS1, which showed an effect of composer with both MIDI and natural stimuli, and an effect of pianist with natural stimuli. However, there was only a weak effect of music training.

rm(list = ls())  
graphics.off()

## DataSet

• Piano music from Bach, Mozart, Beethoven 36 excerpts from CD recordings, with 3 from each composer by each of 4 pianists: Arrau, Barenboim, Pirès, Richter - Excerpts were 9 to 15 s long • We presented the stimuli as audio icons arranged randomly on a PowerPoint slide.

for (i in 1:length(Design\_Data)) {  
 Design\_Data[i] <- if (Design\_Data[i] <= 1) 1 else if (Design\_Data[i] <= 4 & Design\_Data[i] >1) 2 else 3  
}

We cateogrize the level of expertise as follows: Musicians • musical training = 5 years and above N = 10 Moderate Musicians • musical training = 1 to 4 years N = 17 Nonmusicians • musical training = less than 1 year N = 10

color4mus <- c(1, 1, 2 ,2, 1 ,1 ,1 ,1, 1, 1 ,1, 1, 3, 2 ,3, 2, 3 ,1, 3 ,1, 2 ,2 ,3 ,2 ,2 ,1 ,1 ,1, 2 ,1 ,2, 3, 3, 3, 3, 3 ,1)  
Judges <- paste0(Design\_Data,1:length(Design\_Data))

## Create the set of distance matrices

DistanceCube <- DistatisR::DistanceFromSort(Sorting\_Data)

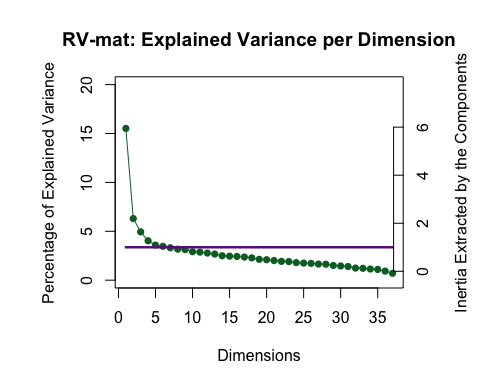
testDistatis <- DistatisR::distatis(DistanceCube)

BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)

## [1] Bootstrap On Factor Scores. Iterations #:   
## [2] 1000

## ScreePlot

ev4C <- testDistatis$res4Cmat$eigValues  
Scree.1 <- PlotScree(ev = ev4C,  
 p.ev = NULL, max.ev = NULL, alpha = 0.05,  
 col.ns = "#006D2C", col.sig = "#54278F",  
 title = "RV-mat: Explained Variance per Dimension",plotKaiser = TRUE)

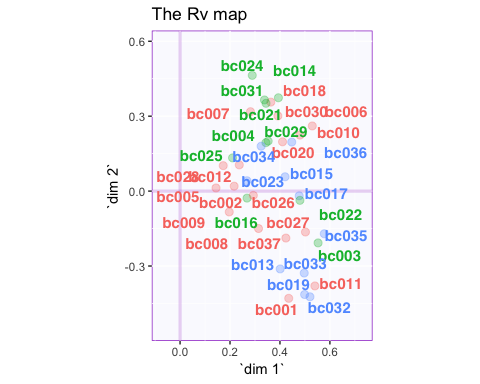


## The Assessor Matrix

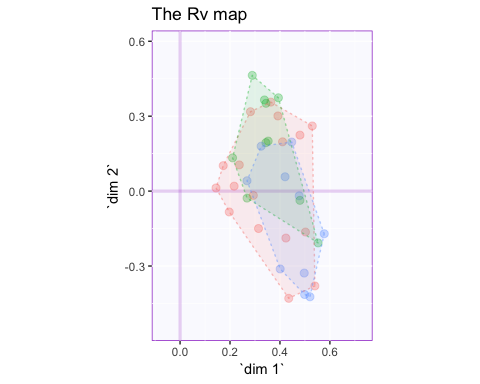
# Plot the assessor matrix  
G <- testDistatis$res4Cmat$G  
# Create a color scheme for the Composers  
col4Non\_musicions <- "#F8766D"   
col4Moderate\_musicions <- "#00BA38"   
col4\_musicions <- "#619CFF"  
Judges[color4mus == 1 ] <- rep(col4Non\_musicions,length(Judges))

## Warning in Judges[color4mus == 1] <- rep(col4Non\_musicions,  
## length(Judges)): number of items to replace is not a multiple of  
## replacement length

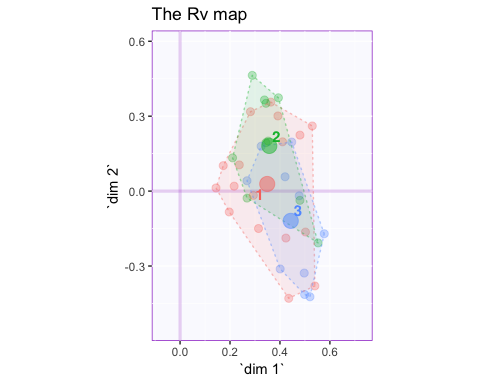
Judges[color4mus == 2 ] <- col4Moderate\_musicions  
Judges[color4mus == 3 ] <- col4\_musicions  
#---------------------------------------------------------------------  
#---------------------------------------------------------------------  
# A graph for the Judges set  
baseMap.j <- PTCA4CATA::createFactorMap(G,  
 title = 'The Rv map',  
 col.points = Judges,  
 alpha.points = .3,  
 col.labels = Judges)  
# A graph for the J-set  
aggMap.j <- baseMap.j$zeMap\_background + # background layer  
 baseMap.j$zeMap\_dots + baseMap.j$zeMap\_text # dots & labels  
# We print this Map with the following code  
print(aggMap.j)



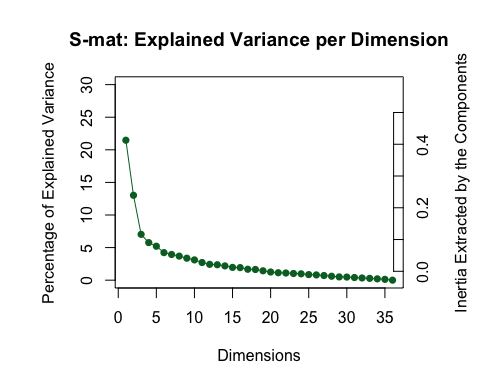
# Create 100% Tolerance interval polygons  
#  
GraphTJ.Hull.100 <- MakeToleranceIntervals(G,  
 as.factor(Design\_Data),  
 names.of.factors = c("Dim1","Dim2"),  
 col = unique(Judges),  
 line.size = .5,  
 line.type = 3,  
 alpha.ellipse = .1,  
 alpha.line = .4,  
 p.level = 1, # full Hulls  
 type = 'hull' #  
 # use 'hull' for convex hull  
)  
#---------------------------------------------------------------------  
# Create the map  
aggMap.j.withHull <- baseMap.j$zeMap\_background + # background layer  
 baseMap.j$zeMap\_dots + GraphTJ.Hull.100  
#---------------------------------------------------------------------  
#---------------------------------------------------------------------  
# Plot it!  
print(aggMap.j.withHull)



JudgesMeans.tmp <- aggregate(G, list(as.numeric(Design\_Data)), mean) # compute the means  
JudgesMeans <- JudgesMeans.tmp[,2:ncol(JudgesMeans.tmp )] # drop var 1  
rownames(JudgesMeans) <- JudgesMeans.tmp[,1] # use var 1 to name the groups  
#---------------------------------------------------------------------  
# a vector of color for the means  
col4Means <- unique(Judges)  
#---------------------------------------------------------------------  
# create the map for the means  
MapGroup <- PTCA4CATA::createFactorMap(JudgesMeans,  
 axis1 = 1, axis2 = 2,  
 constraints = baseMap.j$constraints,  
 title = NULL,  
 col.points = col4Means,  
 display.points = TRUE,  
 pch = 19, cex = 5,  
 display.labels = TRUE,  
 col.labels = col4Means,  
 text.cex = 4,  
 font.face = "bold",  
 font.family = "sans",  
 col.axes = "darkorchid",  
 alpha.axes = 0.2,  
 width.axes = 1.1,  
 col.background = adjustcolor("lavender",  
 alpha.f = 0.2),  
 force = 1, segment.size = 0)  
# The map with observations and group means  
aggMap.j.withMeans <- aggMap.j.withHull +  
 MapGroup$zeMap\_dots + MapGroup$zeMap\_text  
#---------------------------------------------------------------------  
# plot it!  
print(aggMap.j.withMeans)



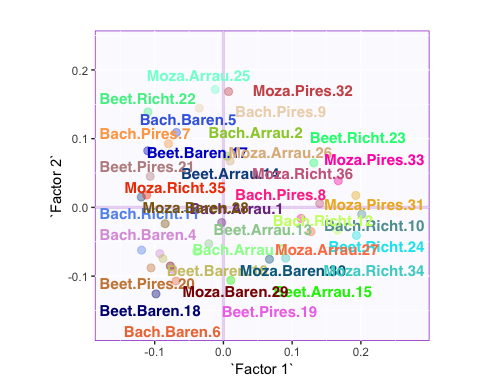
# First we fix a bit of shamefull absentmindness:  
# The eigenvalues of the compromise matrix are not available  
# in DistatisR.  
# So we recompute them here  
ev4S <- eigen(testDistatis$res4Splus$Splus,  
 symmetric = TRUE, only.values = TRUE)$values  
# A scree for the compromise  
Scree.S <- PlotScree(ev = ev4S,  
 p.ev = NULL, max.ev = NULL, alpha = 0.05,  
 col.ns = "#006D2C", col.sig = "#54278F",  
 title = "S-mat: Explained Variance per Dimension")



zeScree.S <- recordPlot()

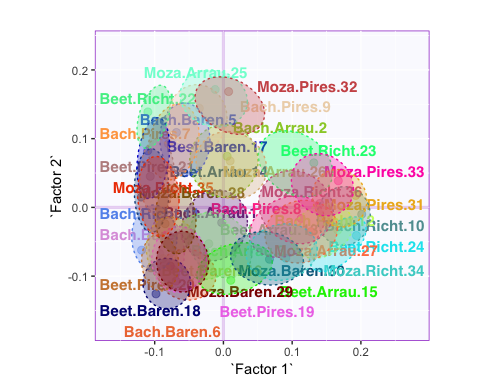
## I-Map

Fi <- testDistatis$res4Splus$F  
col4Beers <- prettyGraphsColorSelection(nrow(Fi))  
# Use colors from prettyGraphs  
#---------------------------------------------------------------------  
# Graphs for the I set  
#---------------------------------------------------------------------  
# Create the base map  
constraints4Fi <- lapply(minmaxHelper(Fi),'\*',1.2)  
baseMap.i <- PTCA4CATA::createFactorMap(Fi,  
 col.points = col4Beers,  
 col.labels = col4Beers,  
 constraints = constraints4Fi,  
 alpha.points = .4)  
#---------------------------------------------------------------------  
# We are interested about the labels here  
# so we will use dots and labels  
#---------------------------------------------------------------------  
# Plain map with color for the I-set  
aggMap.i <- baseMap.i$zeMap\_background + baseMap.i$zeMap\_dots +  
 baseMap.i$zeMap\_text  
#---------------------------------------------------------------------  
# print this Map  
print(aggMap.i)



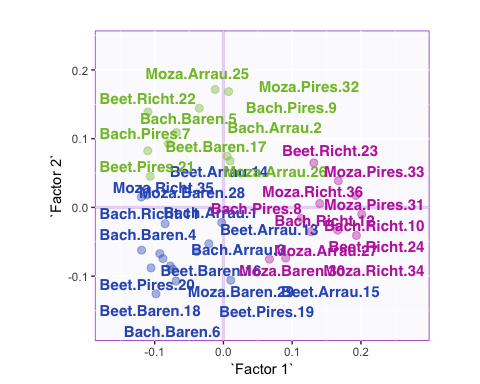
# Create Confidence Interval Plots  
# use function MakeCIEllipses from package PTCA4CATA  
#  
constraints4Fi <- lapply(minmaxHelper(Fi),'\*',1.2)  
GraphElli <- MakeCIEllipses(BootF[,1:2,],  
 names.of.factors = c("Factor 1","Factor 2"),  
 alpha.line = .5,  
 alpha.ellipse = .3,  
 line.size = .5,  
 line.type = 3,  
 col = col4Beers,  
 p.level = .95 )  
#---------------------------------------------------------------------  
# create the I-map with Observations and their confidence intervals  
#  
aggMap.i.withCI <- aggMap.i + GraphElli + MapGroup$zeMap\_text  
#---------------------------------------------------------------------  
# plot it!  
print(aggMap.i.withCI)

## Warning: Removed 3 rows containing missing values (geom\_text\_repel).

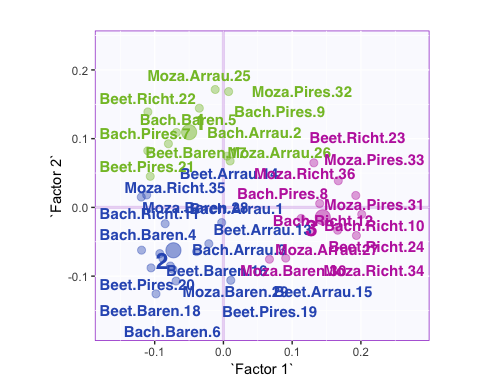


# Old graph with links to partial factor scores  
# Not that informative for sorting tasks  
# Change names of the assessors  
partF <- testDistatis$res4Splus$PartialF  
dimnames(partF)[[3]] <- as.character(1:dim(partF)[3])  
PartialF <- GraphDistatisPartial(FS = testDistatis$res4Splus$F,  
 PartialFS = partF,  
 axis1 = 1, axis2 = 2, constraints = NULL,  
 item.colors = col4Beers,  
 participant.colors = NULL,  
 ZeTitle = "Distatis-Partial",  
 Ctr=NULL, color.by.observations = TRUE,  
 nude = FALSE, lines = TRUE)  
#Plot it !  
F.and.PartialF <- recordPlot()

# Some classification now  
# First plain k-means  
set.seed(42)  
beers.kMeans <- kmeans(x = Fi , centers = 3)  
#---------------------------------------------------------------------  
# Now to get a map by cluster:  
col4Clusters <- createColorVectorsByDesign(  
 makeNominalData(  
 as.data.frame(beers.kMeans$cluster) ))  
  
#=====================================================================  
#---------------------------------------------------------------------  
# Graphs for the I set  
#---------------------------------------------------------------------  
# Create the base map  
# constraints4Fi <- lapply(minmaxHelper(Fi),'\*',1.2)  
baseMap.i.km <- PTCA4CATA::createFactorMap(Fi,  
 col.points = col4Clusters$oc,  
 col.labels = col4Clusters$oc,  
 constraints = constraints4Fi,  
 alpha.points = .4)  
#---------------------------------------------------------------------  
# We are interested about the labels here  
# so we will use dots and labels  
#---------------------------------------------------------------------  
# Plain map with color for the I-set  
aggMap.i.km <- baseMap.i.km$zeMap\_background +  
 baseMap.i.km$zeMap\_dots + baseMap.i.km$zeMap\_text  
# print  
print(aggMap.i.km)

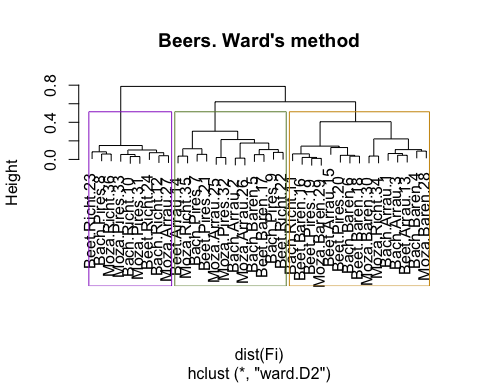


# get the color order in the c=good order  
col4C <- col4Clusters$gc[sort(rownames(col4Clusters$gc),  
 index.return = TRUE)$ix]  
# create the map for the means  
map4Clusters <- PTCA4CATA::createFactorMap(beers.kMeans$centers,  
 axis1 = 1, axis2 = 2,  
 constraints = constraints4Fi,  
 title = NULL,  
 col.points = col4C,  
 display.points = TRUE,  
 pch = 19, cex = 5,  
 display.labels = TRUE,  
 col.labels = col4C,  
 text.cex = 6,  
 font.face = "bold",  
 font.family = "sans",  
 col.axes = "darkorchid",  
 alpha.axes = 0.2,  
 width.axes = 1.1,  
 col.background =  
 adjustcolor("lavender", alpha.f = 0.2),  
 force = 1, segment.size = 0)  
# The map with observations and group means  
aggMap.i.withCenters <- aggMap.i.km +  
 map4Clusters$zeMap\_dots + map4Clusters$zeMap\_text  
#  
print(aggMap.i.withCenters)



## Cluster Analysis

# A cluster analysis  
beer.hc <- hclust(d = dist(Fi),  
 method = 'ward.D2' )  
  
plot.tree <- plot(beer.hc, main = "Beers. Ward's method")  
hc.3.cl <- rect.hclust(beer.hc, k = 3,  
 border = c('darkorchid',  
 'darkolivegreen4','darkgoldenrod3')  
 )



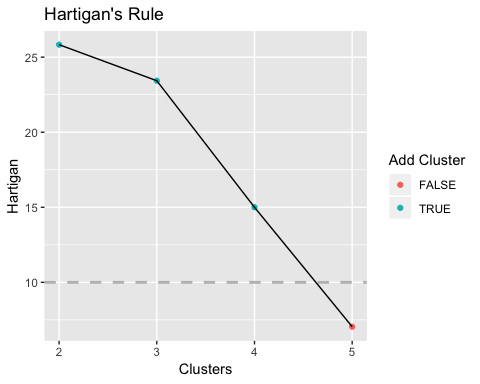
library(useful)

## Loading required package: ggplot2

best.beers <- useful::FitKMeans(Fi, max.clusters = 5,  
 seed = 314)  
print(best.beers) # when Hartigan parameter > 10 => add a cluster

## Clusters Hartigan AddCluster  
## 1 2 25.830051 TRUE  
## 2 3 23.429672 TRUE  
## 3 4 14.998705 TRUE  
## 4 5 7.042848 FALSE

plot.harti <- useful::PlotHartigan(best.beers)  
print(plot.harti)



## Summary

Participants were able to strongly differentiate Mozart’s excerpts from Beethoven’s, with Bach falling in between those two and Richter’s performances of the three composers were clustered relatively close to the Mozart region of the solution, indicating their clarity and balance; in contrast, those of Barenboim were clustered in the Beethoven region, indicating their sumptuousness and passion.