Multivariate Statistical Analysis Cookbook with R

Yile Wang

2020-12-10

Contents

Chapter 1

Introduction

Figure 1.1: UTD Library

Welcome to Yile's Multivariate Statistical Analysis Cookbook! This cookbook is designed for beginners and future me who might need **Multivariate Statistical Analysis (MSA)** as tools to analyze neuroimaging or behavioral data. The original idea of writing it is coming from my final project in Advanced Research Methods class (RM3),instructed by Dr. Herve Abdi, which is the most useful and hardcore (if you allow me to use this word lol) class I have ever taken at UTD. In this class, we are required to write a book to introduce all **Eight** analysis with R codes skills we learned from this class. After this class, I plan to keep updating this book as a source base for myself and anyone in need. This book will be accessible on My Personal Website. Also, it is welcomed to send email to me if you have any comment to this book ylwwayne@gmail.com.

1.1 Main Packages

For happy computing, I listed the packages required for our analysis in the chunk below. Before conducting any analysis, we should keep all these packages installed and updated in your local R environment.

```
# Clean Start----
rm(list = ls())
graphics.off()
# Packages----
library(bookdown)
library(MExPosition)
library(RCurl)
library(dplyr)
library(ggplot2)
library(ggplotify)
library(grid)
library(gridExtra)
library(PTCA4CATA)
library(ExPosition)
library(data4PCCAR)
library(stringr)
library(readxl)
library(RColorBrewer)
library(DistatisR)
library(TExPosition)
library(InPosition)
library(TInPosition)
library(corrplot)
library(tidyverse)
library(gtable)
library(prettyGraphs)
library(superheat)
library(knitr)
library(psych)
library(pheatmap)
library(factoextra)
```

1.2 MSA Introduction

Let's cut right to the chase now. What's MSA? If only one sentence is allowed to give a definition about MSA, probably I will say it is a tool that helps us explore the linear relationship between observations and **multiple**variables. Compared to analysis of variance (ANOVA), which is designed to analyze the differences among groups means based on the law of total variance, MSA will provide me more details of how these groups are different and which variables contribute more to the group separation.

Well, it still looks like an abstract concept. What should we know from the ground? When I am studying a new concept, my favorite part is when instructor gave example on it. Let me try to give an example here: As we know, if doing research is to cut down the tree then observe its inside texture, the statistical analysis should be the tool (saw), our data should be the tree and the answer of the research question should be the texture of this tree.

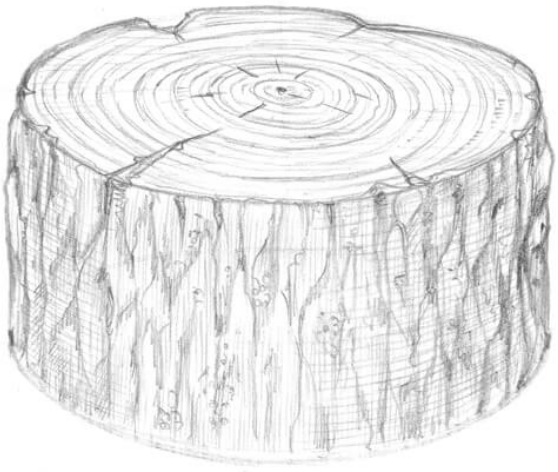


Figure 1.2: Beautiful Tree Texture Here!

Thus, our question is: how to find out the best saw for different trees? There are different kinds of data sets in various studies. In the table below, it shows that different methods is corresponding to different data types (quantitative data and qualitative data) and data tables. Still, when I am talking about two or more data table, generally it points to data tables in different modalities (brain imaging vs behavioral test, or IQ test vs Physical activity).

Data Types	One Data Table	Two Data Table	Multiple Data Table
Quantitative	PCA	BADA, PLS-C	MFA
Qualitative	CA, MCA	DiCA, PLS-C	DiSTATIS

Table.1: Different MSA methods corresponding to number of different Data Tables.

1.3 MSA data set

In this book, we will go through all these methods with vivid examples, which include graphs, codes and illustration for each methods. I will use a public data set called "collective action" as our main example, and use two food-related data sets as special examples for CA and DiSTATIS. All these

data sets are accessible in my github, so it is welcomed if anyone want to practice MSA with these data sets following this book.

1.4 Whoo-hoo!

Let's begin our journey at MSA!

Chapter 2

Functions Introduction

In this Chapter, I list some of important functions for plotting. These functions are mainly based on packages `PTCA4CATA` and `data4PCCAR`, which are developed by Dr. Herve Abdi and his team at The University of Texas at Dallas. These functions will provide computational convenience to my plotting and visualization of my results. It is worth to mention that these functions are designed to generate plots so users should conduct MSA before using these functions. For how to conduct MSA in R, please see each chapter in this book: PCA[4], CA[5], MCA[8], BADA[7], DiCA[6], PLS-C[9], DiSTATIS[10] and MFA[11]. For further information about the two basic packages, please visit Dr. Abdi's github

2.1 Plotting Scree plot

The first function is designed to plot scree plot:

Required parameters: eigs(eigenvalue), p.vals(inference p values)

Output: Scree plot

```
plot.scree <- function(eigs, p.vals){
  PlotScree(ev = eigs,
  p.ev = p.vals,
  plotKaiser = TRUE,
  title = "Explained Variance per Dimension")
}
```

2.2 Plotting Permutation test results

The second function is designed to plot the permutation histogram:

Required parameters: eigs.perm(eigenvalue.permutation), eigs(eigenvalue), para1(x_lim for X axis, default = 5), para2(intervals of distribution, default = 20), Dim(number of dimension, default = 1).

Output: Permutation Plots

```
plot.permutation <- function(eigs.perm, eigs,
                             para1 = 5,  para2 = 20,
                             Dim = 1){
  zeDim = Dim
  pH <- prettyHist(
    distribution = eigs.perm[,zeDim],
    observed = eigs[zeDim],
    xlim = c(0,para1), # needs to be set by hand
    breaks = para2,
    border = "gray",
    main = paste0("Permutation Test for Eigenvalue ",zeDim),
    xlab = paste0("Eigenvalue ",zeDim),
    ylab = "",
    counts = FALSE,
    cutoffs = c( 0.975))
}
```

2.3 Plotting Heatmap

Required parameters: data1(data set 1), data2(data set 2), xcol(color of labels in X axis), ycol(color of labels in X axis), textsize(font size of the labels, default = 5)

Output: Heatmap of correlation between two data sets.

```
plot.heatmap <- function(data1, data2,
                         DATA=NULL, xcol = NULL,
                         ycol = NULL, textsize = 5,
                         scale = TRUE ){
  if(is.null(DATA)){
  data1 <- as.matrix(data1)
  data2 <- as.matrix(data2)
  DATA = cor(data1, data2)
  }
  if(is.null(xcol)){
    xcol <- prettyGraphsColorSelection(
      n.colors = ncol(DATA))
  }
  if(is.null(ycol)){
    ycol <- prettyGraphsColorSelection(
      n.colors = nrow(DATA))
  }
  hmap <- superheat(DATA,
            heat.pal = c("brown", "white","red4"),
            left.label.size = 0.5,
            bottom.label.size = 0.5,
            bottom.label.text.angle = 90,
```

```
            left.label.text.alignment = "right",
            bottom.label.text.alignment = "right",
            bottom.label.text.col = xcol,
            left.label.text.col = ycol,
            grid.vline = FALSE,
            grid.hline = FALSE,
            bottom.label.col = "white",
            left.label.col = "white",
            scale = scale,
            left.label.text.size = textsize,
            bottom.label.text.size = textsize,
            )
}
```

2.4 Plotting Row Factor Scores

Required parameters: DESIGN(the grouping design of observation, such as Lakers is group1 and Celtics is group2), fs(factor scores), eigenvalues, tau(Kendall rank correlation coefficient), d(dimension, default = 1), mode(do you want "hull" or "CI"?), method(The string you want to add on your graph's title).

Output: Global row factor scores map for each observation

```
plot.fs <- function(DESIGN, fs, eigs,
                    tau, d = 1, mode="CI",
                    method = " ")
{
  fs <- as.matrix(fs)
  # GETMEANS
  fm.tmp <- aggregate(fs, list(DESIGN), mean)
  fm <- fm.tmp[,2:ncol(fm.tmp)]
  rownames(fm) <- fm.tmp[,1]
  tau <- round(tau)
  color.dot <- as.matrix(DESIGN)
  na.de <- as.matrix(levels(DESIGN))
  num.de <- length(na.de)
  for (i in 1: num.de){
    color.dot[which(color.dot == na.de[i])] <- index[i]
  }
  color.dot <- as.matrix(color.dot)
  rownames(color.dot) <- DESIGN
  fs.plot <- createFactorMap(fs, # data
                             title = paste0("Factor Scores_",
                                            method),
                             alpha.points = 0.3,
                             axis1 = d, axis2 = (d+1),
```

```
                          pch = 19,
                          cex = 2,
                          text.cex = 2.5,
                          display.labels = FALSE,
                          col.points = color.dot,
                          col.labels =  color.dot
)
 fs.label <- createxyLabels.gen(d,(d+1),
                              lambda = eigs,
                              tau = tau,
                              axisName = "Component "
)
 ind <- c(1:num.de)
 for (i in 1:num.de){
   inde <- match(rownames(color.dot),
                 sort(rownames(fm)))
   ind[i] <- which(inde == i)[1]
 }
 grp.ind <- ind
 rownames(fm[sort(rownames(fm)),]) <- sub("[[:punct:]]","",
                                      rownames(fm[sort(rownames(fm)),]))

 graphs.means <- PTCA4CATA:: createFactorMap(fm[sort(rownames(fm)),],
                                 axis1 = d, axis2 =(d+1),
                                 constraints = fs.plot$constraints,
                                 col.points = color.dot[grp.ind],
                                 col.labels = color.dot[grp.ind],
                                 alpha.points = 0.9)

 BootCube <- PTCA4CATA::Boot4Mean(fs,
                                 design = as.factor(DESIGN),
                                 niter = 100)

 dimnames(BootCube$BootCube)[[2]] <- paste0("dim ",
                                      1:dim(BootCube$BootCube)[[2]])

 boot.elli <- MakeCIEllipses(data =
                               BootCube$BootCube[,d:(d+1),][sort(rownames(BootCube$BootCube[,
                 names.of.factors =
                 c(paste0("Dimension ", d),paste0("Dimension ",
                 (d+1))),
                 col = color.dot[grp.ind],
                 alpha.line = 0.3,
                 alpha.ellipse = 0.3
)

 # with Hull
```

```
  colnames(fs) <- paste0('Dimension ', 1:ncol(fs))
  # getting the color correct: an ugly trick
  #col.groups <- as.data.frame(col.groups)
  DESIGN <- factor(DESIGN, levels = DESIGN[grp.ind])
  GraphHull <- PTCA4CATA::MakeToleranceIntervals(fs,
                axis1 = d,
                axis2 = (d+1),
                design = DESIGN,
                col = color.dot[grp.ind],
                names.of.factors =  c(paste0("Dim ", d),
                                      paste0("Dim 2", (d+1))),
                p.level = 1.00)

  if (mode == "hull"){
  factor.map <- fs.plot$zeMap_background +
    GraphHull +
    fs.plot$zeMap_dots +
    fs.label +
    graphs.means$zeMap_dots +
    graphs.means$zeMap_text}
  if (mode == "CI"){
    factor.map <- fs.plot$zeMap_background +
    boot.elli +
    fs.plot$zeMap_dots +
    fs.label +
    graphs.means$zeMap_dots +
    graphs.means$zeMap_text}
  factor.map
}
```

2.5 Plotting Column Plot

Required parameters: fj(column factor scores), eigs(eigenvalue), tau(Kendall rank correlation coefficient), d(dimension, default = 1), col(color coding of variables), method(title names)

Output: column factor scores plot.

```
plot.cfs <- function(fj, eigs,
                     tau, d = 1,
                     col = NULL,
                     method = " ",
                     colrow = "row")
  {
  if(colrow == "row"){
    cr <- nrow(fj)
  }
```

```
  if(colrow == "col"){
    cr <- ncol(fj)
  }
  if(is.null(col)){
    col <- prettyGraphs::prettyGraphsColorSelection(cr)
  }
  fj.labels <- createxyLabels.gen(d,(d+1),
                              lambda = eigs,
                              tau = round(tau),
                              axisName = "Component "
                              )
  fj.plot <- createFactorMap(fj, # data
                title = paste0("Factor Scores", method),
                axis1 = d,
                axis2 = (d+1),
                pch = 19,
                cex = 2,
                text.cex = 3,
                col.points = col,
                col.labels = col
                             )
  column.factor.map <- fj.plot$zeMap_background +
    fj.plot$zeMap_dots +
    fj.plot$zeMap_text +
    fj.labels
  column.factor.map
}
```

2.6 Plotting Loading Plot

Required parameters: data(raw data), col(color coding), fs(factor scores), eigs(eigenvalue), tau(Kendall rank correlation coefficient), d(dimension, default = 1)

Output: Loading plot for each variable.

```
plot.loading <- function(data, col=NULL, fs, eigs, tau, d=1){
  if (is.null(col)){
    col <- prettyGraphsColorSelection(n.colors = ncol(data))
  }
  cor.loading <- cor(data, fs)
  rownames(cor.loading) <- rownames(cor.loading)
  fi.labels <- createxyLabels.gen(d,(d+1),
                      lambda = eigs,
                      tau =  round(tau),
                      axisName = "Component "
  )
```

```
  loading.plot <- createFactorMap(cor.loading,
                      axis1 = d,
                      axis2 = (d+1),
                      constraints = list(minx = -1, miny = -1,
                                         maxx = 1, maxy = 1),
                      col.points = col,
                      col.labels = col)
  LoadingMapWithCircles <-  loading.plot$zeMap_background +
    loading.plot$zeMap_dots +
    addArrows(cor.loading[,d:(d+1)], color = col) +
    addCircleOfCor() +
    loading.plot$zeMap_text +
    fi.labels
  LoadingMapWithCircles
}
```

2.7 Plotting Contribution Barplot

Required parameters: cj(contribution of each variable), fj(factor scores of each variable), col(color coding), boot.ratios(bootstrap ratio), signiOnly(choose to only display sigificant contribution or not, default = FALSE), fig(how many plots to display, default = 2)

Output: Combination of contribution barplots.

```
plot.cb <- function(cj, fj, col = NULL,
                    boot.ratios, signifOnly = FALSE,
                    fig = 2, horizontal = FALSE,
                    colrow = "col", fontsize = 3)
  {
  if (colrow == "col"){
    cr <- ncol(cj)
  }
  if (colrow == "row"){
    cr <- nrow(cj)
  }
  # Contribution Plot
  ### plot contributions for component 1
  if (is.null(col)){
      col <- prettyGraphs::prettyGraphsColorSelection(cr)
  }
  signed.ctrJ <- cj * sign(fj)
  laDim = 1
  ctrJ.1 <- PrettyBarPlot2(signed.ctrJ[,laDim],
            threshold = 1 / NROW(signed.ctrJ),
            font.size = fontsize,
            signifOnly = signifOnly,
```

```
           horizontal = horizontal,
           color4bar = col, # we need hex code
           main = 'Variable Contributions (Signed)',
           ylab = paste0('Contributions Barplot',
                         laDim),
           ylim = c(1.2*min(signed.ctrJ),
           1.2*max(signed.ctrJ))
  ) + ggtitle("Contribution",subtitle = paste0('Component ', laDim))

### plot contributions for component 2
laDim =2
ctrJ.2 <- PrettyBarPlot2(signed.ctrJ[,laDim],
             threshold = 1 / NROW(signed.ctrJ),
             font.size = fontsize,
             color4bar = col, # we need hex code
             signifOnly =signifOnly,
             horizontal = horizontal,
             main = 'Variable Contributions (Signed)',
             ylab = paste0('Contributions Barplot',laDim),
             ylim = c(1.2*min(signed.ctrJ),1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))
laDim =3
ctrJ.3 <- PrettyBarPlot2(signed.ctrJ[,laDim],
               threshold = 1 / NROW(signed.ctrJ),
               font.size = fontsize,
               color4bar = col, # we need hex code
               signifOnly = signifOnly,
               horizontal = horizontal,
               main = 'Variable Contributions (Signed)',
               ylab = paste0('Contributions Barplot',laDim),
               ylim = c(1.2*min(signed.ctrJ),
                                1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))



### Bootstrap of columns vectors
BR <- boot.ratios
laDim = 1

# Plot the bootstrap ratios for Dimension 1
ba001.BR1 <- PrettyBarPlot2(BR[,laDim],
               threshold = 2,
               font.size = fontsize,
               signifOnly = signifOnly,
               horizontal = horizontal,
               color4bar = col, # we need hex code
```

```
                ylab = 'Bootstrap ratios',
                ylim = c(1.2*min(BR[,laDim]),
                                    1.2*max(BR[,laDim]))
) + ggtitle("Bootstrap ratios",
            subtitle = paste0('Component ', laDim))

# Plot the bootstrap ratios for Dimension 2
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR[,laDim],
                threshold = 2,
                font.size = fontsize,
                signifOnly = signifOnly,
                horizontal = horizontal,
                color4bar = col, # we need hex code
                ylab = 'Bootstrap ratios',
                ylim = c(1.2*min(BR[,laDim]),
                         1.2*max(BR[,laDim]))
) + ggtitle("",subtitle = paste0('Component ', laDim))

laDim = 3
ba003.BR3 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = fontsize,
                            signifOnly = signifOnly,
                            horizontal = horizontal,
                            color4bar = col,
                            ylab = 'Bootstrap ratios',
                            ylim = c(1.2*min(BR[,laDim]),
                                     1.2*max(BR[,laDim]))
) + ggtitle("",subtitle = paste0('Component ', laDim))

if(fig == 2){
  grid.arrange(
  as.grob(ctrJ.1),
  as.grob(ctrJ.2),
  as.grob(ba001.BR1),
  as.grob(ba002.BR2),
  ncol = 2,nrow = 2,
  top = textGrob("Barplots for variables",
                 gp = gpar(fontsize = 18, font = 3))
)
}
if(fig==3){
  grid.arrange(
  as.grob(ctrJ.1),
  as.grob(ctrJ.2),
  as.grob(ctrJ.3),
```

```
    as.grob(ba001.BR1),
    as.grob(ba002.BR2),
    as.grob(ba003.BR3),
    ncol = 2,nrow = 3,
    top = textGrob("Barplots for variables",
                   gp = gpar(fontsize = 18, font = 3))
  )
  }
}
```

2.8 Bins Helper Functions

Required parameters: quantitative data

Output: qualitative data with trisection

```
bins_helper <- function(DATA, name = "Variable") {
  data <- as.data.frame(DATA)
  tp <- quantile(as.numeric(data$DATA),seq(0,1,1/3))
  ac = as.data.frame(table(data))
  if (tp[3] < tp[4]){
    if (ac[1,"Freq"] > 3){low <- which(data< tp[2])
    high <- which(data > tp[3])
    medium <- which(data  >= tp[2] & data <=tp[3])
    data[low,] <- 0
    data[medium,] <- 1
    data[high,] <-2
    }else{ low <- which(data <= tp[2])
      high <- which(data > tp[3])
      medium <- which(data > tp[2] & data <=tp[3])
      data[low,] <- 0
      data[medium,] <- 1
      data[high,] <-2}
  }else{low <-which(data < tp[2])
    high <- which(data >= tp[3])
    medium <- which(data >= tp[2] & data <tp[3])
    data[low,] <- 0
    data[medium,] <- 1
    data[high,] <-2}
  a.c = as.data.frame(table(data))
  colnames(a.c) <- c("Value","Freq")
  rownames(a.c) <- c("Level_1", "Level_2", "Level_3")
  #print(a.c)
  spearman.col.score <- cor(DATA, as.numeric(data$DATA),
                            method = "spearman")
  cat(name, "spearman r:", spearman.col.score, "\n")
```

```
  return(data)
}
```

2.9 Plotting Latent Variable Plot

Required parameters: DESIGN, lx(from PLS), ly(from PLS), dimension(default = 1)

Output: LV plot

```
plot.lv <- function(DESIGN, lx, ly, d=1 ){
  color.dot <- as.matrix(DESIGN)
  na.de <- as.matrix(levels(DESIGN))
  num.de <- length(na.de)
  for (i in 1: num.de){
    color.dot[which(color.dot == na.de[i])] <- index[i]
  }
  color.dot <- as.matrix(color.dot)
  rownames(color.dot) <- DESIGN
  latvar <- cbind(lx[,d],ly[,d])
  colnames(latvar) <- c(paste0("Lx ", d),
                        paste0("Ly ", d))

  # compute means
  lv.group <- getMeans(latvar,DESIGN)

  # get bootstrap intervals of groups
  lv.group.boot <- Boot4Mean(latvar, DESIGN)
  colnames(lv.group.boot$BootCube) <- c(paste0("Lx ", d),
                                        paste0("Ly ", d))


  plot.lv <- createFactorMap(latvar,
             col.points = color.dot,
             col.labels = color.dot,
             title = c("Latent Variable Map", d),
                       alpha.points = 0.2
  )
  ind <- c(1:num.de)
  for (i in 1:num.de){
    inde <- match(rownames(color.dot),
                  sort(rownames(lv.group)))
    ind[i] <- which(inde == i)[1]
  }
  grp.ind <- ind
  plot.mean <- createFactorMap(lv.group,
             col.points = color.dot[grp.ind],
```

```
              col.labels = color.dot[grp.ind],
              cex = 4,
              pch = 17,
              alpha.points = 0.8)

  plot.meanCI <- MakeCIEllipses(lv.group.boot$BootCube[,c(1:2),],
                      col = color.dot[grp.ind],
                      names.of.factors = c(paste0("Lx ", d),
                                           paste0("Ly ", d)))

  plot.lv.pls <- plot.lv$zeMap_background +
    plot.meanCI +
    plot.lv$zeMap_dots +
    plot.mean$zeMap_dots +
    plot.mean$zeMap_text

  plot.lv.pls
}
```

2.10 Plotting Partial Factor Scores

Required parameters: DESIGN, fs(factor scores), eigs(eigenvalue), tau(explained variance), dimension(default = 1), partial.fi.array(partial matrix), mm(how many sub data sets)

Output: Partial Factor Scores Map

```
plot.partial.fs <- function(DESIGN, fs,
                            eigs, tau,
                            d, partial.fi.array, mm ){
  # get some colors
  color.dot <- as.matrix(DESIGN)
  na.de <- as.matrix(levels(DESIGN))
  num.de <- length(na.de)
  for (i in 1: num.de){
    color.dot[which(color.dot == na.de[i])] <- index[i]
  }
  color.dot <- as.matrix(color.dot)
  rownames(color.dot) <- DESIGN

  # Factor Map
  factor.graph.out <- createFactorMap(
    X = fs,
    axis = d, axis2 = (d+1),
    col.points = color.dot,
    col.labels = color.dot,
    title = "MFA_Partial_Factor_Map",
```

```
  display.labels = FALSE,
  alpha.points = 0.3,
  alpha.labels = 0.3
)
# Labels of Factor Map
labels4RV <- createxyLabels.gen(d, (d+1),
  lambda = eigs,
  tau = tau,
  axisName = "Dimension "
)
# Groups
fm.tmp <- aggregate(fs, list(DESIGN), mean)
fm <- fm.tmp[,2:ncol(fm.tmp)]
rownames(fm) <- fm.tmp[,1]
ind <- c(1:num.de)
for (i in 1:num.de){
  inde <- match(rownames(color.dot), sort(rownames(fm)))
  ind[i] <- which(inde == i)[1]
}
grp.ind <- ind
rownames(fm[sort(rownames(fm)),]) <- sub("[[:punct:]]","",
                    rownames(fm[sort(rownames(fm)),]))

MapGroup <- PTCA4CATA::createFactorMap(fm[sort(rownames(fm)),],
                          axis1 = d,
                          axis2 = (d+1),
                          constraints = factor.graph.out$constraints,
                          col.points = color.dot[grp.ind],
                          cex = 4,  # size of the dot (bigger)
                          col.labels = color.dot[grp.ind],
                          text.cex = 4,
                          alpha.points = 0.5)

# partial factor
partial.fi.array <- partial.fi.array
MFA.fi.groups.means <- PTCA4CATA::getMeans(
  fs,
  DESIGN)
colnames(MFA.fi.groups.means) <- paste0("Dimension ",
                        1:ncol(MFA.fi.groups.means))
# new array
MFA.partial.fi.groups.means <- array(0,dim=c(nrow(MFA.fi.groups.means),
                                             ncol(MFA.fi.groups.means),
                                             mm))
# for loop to calculate mean
for (i in 1:mm){
  MFA.partial.fi.groups.means[,,i] <- as.matrix(PTCA4CATA::getMeans(
```

```
      partial.fi.array[,,i],
      DESIGN))
  }
  # setting dimnames
  dimnames(MFA.partial.fi.groups.means) <- list(rownames(MFA.fi.groups.means),
                                                colnames(MFA.fi.groups.means),
                                                paste0(LETTERS[1:mm], 1:mm))

  map4PFS <- createPartialFactorScoresMap(
    factorScores = MFA.fi.groups.means,
    partialFactorScores = MFA.partial.fi.groups.means,
    axis1 = d, axis2 = (d+1),
    colors4Items = color.dot,
    colors4Blocks = ,
    size.points = 1.5,
    alpha.points = 0.7,
    alpha.lines = 0.9,
    size.labels = 3,
    names4Partial = dimnames(MFA.partial.fi.groups.means)[[3]],
    font.labels = 'bold')

  partialFS.map.a <- factor.graph.out$zeMap_background +
    factor.graph.out$zeMap_dots +
    factor.graph.out$zeMap_text +
    MapGroup$zeMap_dots +
    MapGroup$zeMap_text +
    map4PFS$mapColByItems + labels4RV
  partialFS.map.b <- factor.graph.out$zeMap_background +
    map4PFS$mapColByBlocks +
    labels4RV +
    MapGroup$zeMap_dots +
    MapGroup$zeMap_text

  partialFS.map.b
}
```

Chapter 3

Data Introduction

Data preprocessing is one of the most important steps for our research. The quality of data preprocessing is directly related with quality of the data analysis. Most of time, the raw data is really messy and disorganized. What I need to is to make the data organized and split it into different data tables for different analysis. For one data table, life is easy because all I need to do is to put variable of interest into a huge matrix. For multiple data table, I need to split it into different data tables based on their basic features (variables measuring similar features, such as friends number and social network size). Most importantly, I need to make a design matrix for the data table, aka **grouping strategy**. Based on different grouping strategies, we can have completely different results in our plot and conclusion. However, it is totally up to you to choose the grouping variables: in my current data set, I can group participants according to their social intelligence, general intelligence, age, sex, gpa, and so on. It is highly recommended to read related article before choose the best group variables for your analysis.

In the rest of this chapter, I will introduce three data sets used in this book, *Collective Action Data set*, *Low Income Sausage* and *Wines*. The main data set is corresponding to PCA, BADA, PLS, MCA, MFA; The second one is for CA and the last one is for DiSTATIS.

3.1 Main data set: Collective Action Data Set

The main data set we are using is from an research article published on PNAS last year. Several part of different data tables consist the data set:

- 1. The first part is experimental data:
  - 1. The game performance for each round
  - 2. The time usage for each round
- 2. Emotional & Social Intelligence test
  - 1. State of reasoning
  - 2. Comprehension
  - 3. Spontaneous

- 3. Academic results
  - 1. SAT&ACT overall
  - 2. SAT&ACT Math
  - 3. SAT&ACT Verbal
  - 4. GPA
- 4. Attitude
  - 1. Global_warming
  - 2. Government
  - 3. Environmental ecology
- 5. Social relationship
  - 1. Close friends number
  - 2. Social network size
- 6. Personality
  - 1. Helpful
  - 2. Trust
  - 3. Hard working
  - 4. Volunteer
  - 5. Take advantage
- 7. Demographoc
  - 1. Age
  - 2. Sex
  - 3. Parent Education
  - 4. Parents' job1
  - 5. Parents' job2
  - 6. Race
  - 7. Income
  - 8. Language
  - 9. Major
  - 10. religious
- 8. Emotional processing ability
  - 1. Eyetest scores
  - 2. Quizzes scores

All these variables are easy to understand except the experiment and emotional processing abilities. Generally, the experiment is a token collection game. Its aim is to collect token as many as possible. The token will automatically generate following a rule. If you and your teammates are following this rule to play the game, you will have higher generate rate of the token during this game; However, if you don't follow it, you will end the game soon with less number of tokens in your packet.

For the emotional processing ability, it used a classic paradigm in psychology: participants are required to watch only eyes parts of many face pictures and make decision on what's the emotions on the faces behind. For the social intelligence part, it is similar but participants are required to read paragraphs and make response, which is called *Short Story Test (SST)*, to measure how much capacity participants can empathize with others.

All these variables have detailed illustration on website. Please see this supplementary information document in PNAS. In my data preprocessing, you will notice that **I only use Negative Condition** as example in this book. The reason why I didn't include Positive and Negative two conditions is that I am running out of time to submit my final project… Definitely I will finish it after the class.

- 1. Negative Condition: R1-R3, HIGH resource, LOW group members; R4-R6, LOW resource, HIGH group members
- 2. Positive Condition: R1-R3, LOW resource, HIGH group members; R4-R6, HIGH resource, LOW group members

In the following chunk, I showed a detailed process of how I did data cleaning and preprocessing in raw data (raw data can be downloaded here [1.3]).

```
# Data preprocessing----
load(url("https://github.com/yilewang/MSA/blob/main/Dataset/CogCollect.RData?raw=true"))

# Data Cleaning: Missing values
rownames(data.tab3) <- 1:360
data.tab3$q2[212] <- 3.08
data.tab3$q2[300] <- 1.7

# Binding all the data sets
all.cog <- cbind(data.frame(data.info_demo), data.frame(data.pca), data.frame(data.tab2[3:8]),
 data.frame(data.tab3[3:19]))

all.cog <- all.cog[order(all.cog[, "ANumber"]), ]
rownames(all.cog) <- c(1:360)

# Give Colnames
colnames(all.cog)[46:59] <- c("age", "GPA", "close_friends", "social_network", "instruction",
 "take_advantage", "helpful", "trust", "volunteer", "global_warming", "hard_work",
 "government", "environment_eco", "parent_edu")

colnames(all.cog)[18:25] <- c("religions", "language", "races", "incomes", "majors",
 "sex", "parent_job_1", "parent_job_2")
```

```r
# create sub data tables
sub.token.collection <- cbind(all.cog[, 26:27], all.cog[, 30:36])

sub.token.collection.time <- cbind(all.cog[, 26:27], all.cog[, 37:42])

sub.empathy <- cbind(all.cog[, 26:27], all.cog[, 43:45], all.cog[, "Spontaneous"])
colnames(sub.empathy)[6] <- "Spontaneous"


sub.emotion <- cbind(all.cog[, 26:27], all.cog[, 28:29])

sub.personality <- cbind(all.cog[, 26:27], all.cog[, 51:54], all.cog[, 56])
colnames(sub.personality)[7] <- "hard_work"

sub.attitude <- cbind(all.cog[, 26:27], all.cog[, 55], all.cog[, 57:58])
colnames(sub.attitude)[3] <- "global_warming"

sub.friends <- cbind(all.cog[, 26:27], all.cog[, 48:49])

sub.demographic <- cbind(all.cog[, 26:27], all.cog[, 46], all.cog[, 18:25], all.cog[,
 "parent_edu"])
colnames(sub.demographic)[12] <- "parent_edu"
colnames(sub.demographic)[3] <- "age"


## ****ACT converts to SAT**** ACT DATA
ACT.math.table <- all.cog[which(all.cog$ACT.MATH.SCORE < 40), ]
ACT.reading.table <- all.cog[which(all.cog$ACT.READING.SCORE < 40), ]
ACT.english.table <- all.cog[which(all.cog$ACT.ENGLISH.SCORE < 40), ]
# ACT_MATH
ACT.math <- cbind(as.matrix(ACT.math.table$ANumber), as.matrix(ACT.math.table$ACT.MATH.SCORE))
# ACT_Verbal
ACT.verbal <- matrix(0, nrow = length(ACT.english.table$ACT.ENGLISH.SCORE))
for (i in 1:length(ACT.english.table$ACT.ENGLISH.SCORE))
{
 ACT.verbal[i] <- sum(ACT.reading.table$ACT.READING.SCORE[i] + ACT.english.table$ACT.ENGLISH.SC
}
ACT.verbal <- cbind(as.matrix(ACT.reading.table$ANumber), ACT.verbal)
# SAT DATA
SAT.math.table <- all.cog[which(all.cog$SAT.MATH.SCORE < 801), ]
SAT.verbal.table <- all.cog[which(all.cog$SAT.VERBAL.SCORE < 801), ]
# SAT_MATH
SAT.math <- cbind(as.matrix(SAT.math.table$ANumber), as.matrix(SAT.math.table$SAT.MATH.SCORE))
# SAT_VERBAL
SAT.verbal <- cbind(as.matrix(SAT.verbal.table$ANumber), as.matrix(SAT.verbal.table$SAT.VERBAL
# Create a calculator to convert ACT to SAT Read data from github
x.math <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/ACT_MATH.csv")
```

```
math_con <- read.csv(text = x.math)
colnames(math_con)[1] <- "ACT_MATH"  ## Avoid UTF-8 error
x.verbal <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/ACT_VERBAL.csv
verbal_con <- read.csv(text = x.verbal)
colnames(verbal_con)[1] <- "ACT_VERBAL"  ## Avoid UTF-8 error
# Create a workspace to store the new MATH and VERBAL info
act.math.con <- ACT.math
act.verbal.con <- ACT.verbal
act.comp.con <- cbind(all.cog[, 26], all.cog[, 7])
colnames(act.comp.con) <- c("ANumber", "Composite")
# Make a calculator to convert ACT scores to SAT scores
calculator.math <- function(score)
{
 index <- which(math_con == score)
 return(math_con[index, 2])
}
calculator.verbal <- function(score)
{
 index <- which(verbal_con == score)
 return(verbal_con[index, 2])
}
# For loop to convert the ACT score to SAT score
for (i in 1:length(ACT.math[, 1]))
{
 act.math.con[i, 2] <- calculator.math(ACT.math[i, 2])
}

for (i in 1:length(ACT.verbal[, 1]))
{
 act.verbal.con[i, 2] <- calculator.verbal(ACT.verbal[i, 2])
}
for (i in 1:length(all.cog[, 1]))
{
 act.comp.con[i, 2] <- calculator.math(all.cog$ACT.COMPOSITE.SCORE[i])
}


# Merge data
MATH <- rbind(act.math.con, SAT.math)
VERBAL <- rbind(act.verbal.con, SAT.verbal)
COMP <- act.comp.con
# Get the RANK
MATH <- MATH[order(MATH[, 1]), ]
MATH <- MATH[-54, ]
VERBAL <- VERBAL[order(VERBAL[, 1]), ]
VERBAL <- VERBAL[-54, ]
COMP <- COMP[order(COMP[, 1]), ]
```

```r
sub.grades <- cbind(all.cog[, 26:27], COMP[, 2], MATH[, 2], VERBAL[, 2], all.cog$GPA)
colnames(sub.grades)[6] <- "GPA"
colnames(sub.grades)[3:5] <- c("SAT.COMP", "SAT.MATH", "SAT.VERBAL")


# the data we can use for our analysis
data.to.use <- cbind(sub.token.collection[, 1:8], sub.token.collection.time[, 3:8],
 sub.friends[, 3:4], sub.grades[, 3:5], sub.empathy[, 3:6], sub.personality[,
  3:7], sub.attitude[, 3:5])
supplement.token <- cbind(all.cog[, 26], all.cog[, 36])
colnames(supplement.token)[1:2] <- c("ANumber", "total_earning")
supplement.token <- as.data.frame(supplement.token)

sub.group <- matrix(0, ncol = 4, nrow = 360)
colnames(sub.group) <- c("gpa", "emo", "group", "col")
rownames(sub.group) <- 1:360
sub.group <- cbind(all.cog[, 26:27], sub.group)

# Grouping variable: GPA
sub.gpa <- cbind(all.cog[, 26:27], all.cog$GPA)
colnames(sub.gpa)[3] <- "general"
quan.gpa <- quantile(as.numeric(all.cog$GPA), seq(0, 1, 1/3))
under <- which(sub.gpa[, 3] <= quan.gpa[2])
between <- which(sub.gpa[, 3] > quan.gpa[2] & sub.gpa[, 3] <= quan.gpa[3])
over <- which(sub.gpa[, 3] > quan.gpa[3])
sub.group[, 3][under] <- 1
sub.group[, 3][between] <- 2
sub.group[, 3][over] <- 3


# Grouping variable: Emo
sub.emo <- cbind(all.cog[, 26:27], all.cog$EyesTestScore)
colnames(sub.emo)[3] <- "general"
quan.emo <- quantile(as.numeric(all.cog$EyesTestScore), seq(0, 1, 1/3))
under <- which(sub.emo[, 3] <= quan.emo[2])
between <- which(sub.emo[, 3] > quan.emo[2] & sub.emo[, 3] <= quan.emo[3])
over <- which(sub.emo[, 3] > quan.emo[3])
sub.group[, 4][under] <- 100
sub.group[, 4][between] <- 200
sub.group[, 4][over] <- 300

# spearman correlation results > 0.90
spearman.gpa <- cor(sub.gpa[, 3], sub.group[, 3], method = "spearman")
spearman.emo <- cor(sub.emo[, 3], sub.group[, 4], method = "spearman")

print(paste("corr gpa:", spearman.gpa))
```

```
[1] "corr gpa: 0.931355248926045"
```

```
print(paste("corr emotional", spearman.emo))
```

```
[1] "corr emotional 0.945673602462476"
```

```
sub.group[, 5] <- sub.group[, 3] + sub.group[, 4]

# for each condition low-median-high
condition1 <- which(sub.group[, 5] == 101)
condition2 <- which(sub.group[, 5] == 102)
condition3 <- which(sub.group[, 5] == 103)
condition4 <- which(sub.group[, 5] == 201)
condition5 <- which(sub.group[, 5] == 202)
condition6 <- which(sub.group[, 5] == 203)
condition7 <- which(sub.group[, 5] == 301)
condition8 <- which(sub.group[, 5] == 302)
condition9 <- which(sub.group[, 5] == 303)

# give names
sub.group[, 5][condition1] <- "LS_LG"
sub.group[, 5][condition2] <- "LS_MG"
sub.group[, 5][condition3] <- "LS_HG"
sub.group[, 5][condition4] <- "MS_LG"
sub.group[, 5][condition5] <- "MS_MG"
sub.group[, 5][condition6] <- "MS_HG"
sub.group[, 5][condition7] <- "HS_LG"
sub.group[, 5][condition8] <- "HS_MG"
sub.group[, 5][condition9] <- "HS_HG"

# choose 4 conditions
sub.conditions.1 <- sub.group[which(sub.group[, 5] == "LS_LG"), ]
sub.conditions.2 <- sub.group[which(sub.group[, 5] == "HS_LG"), ]
sub.conditions.3 <- sub.group[which(sub.group[, 5] == "HS_HG"), ]
sub.conditions.4 <- sub.group[which(sub.group[, 5] == "LS_HG"), ]

# put them into a group
sub.conditions.1$group <- as.factor(sub.conditions.1$group)
sub.conditions.2$group <- as.factor(sub.conditions.2$group)
sub.conditions.3$group <- as.factor(sub.conditions.3$group)
sub.conditions.4$group <- as.factor(sub.conditions.4$group)

# make sub condition and organize them
sub.condition <- rbind(sub.conditions.1, sub.conditions.2, sub.conditions.3, sub.conditions.4)
data.to.use.sup <- cbind(data.to.use, supplement.token$total_earning)
colnames(data.to.use.sup)[32] <- "total_earning"
final.table <- merge(sub.condition, data.to.use.sup, by = "ANumber")
```

```
final.table <- select(final.table, -c("SessionType.y"))
names(final.table)[names(final.table) == "SessionType.x"] <- "SessionType"

### _____________________________________### different conditions, the two
### condition, postive and negative we can use for analysis
exp.neg <- final.table[which(final.table$SessionType == "Low_High" | final.table$SessionType ==
 "8.4"), ]
exp.pos <- final.table[which(final.table$SessionType == "High_Low" | final.table$SessionType ==
 "4.8"), ]

## color

index <- prettyGraphsColorSelection(n.colors = 4, starting.color = sample((1:300),
 4))

# manually setting color design
m.color.design <- as.matrix(colnames(exp.neg[8:36]))

m.color.design[1:3] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[4:6] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[7:9] <- m.color.design[1:3]
m.color.design[10:12] <- m.color.design[4:6]
m.color.design[13:14] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[15:17] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[18:21] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[22:26] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))
m.color.design[27:29] <- prettyGraphsColorSelection(starting.color = sample(1:170,
 1))



### _____________________________________###

## **** ending of the data preprocessing ****
```

It is worth to mention that the chunk above is the data preprocessing for quantitative data.

We used similar strategy for qualitative data preprocessing. The essential part of qualitative data coding is that I need to convert all the quantitative data into categorical format (1,2,3) by using `quantile` function. I already compile a new function for the bin coding, please see Function Chapter[2.8] in this book.

```
bins.exp.neg <- as.data.frame(exp.neg)
for(i in 7:26){
  bins.exp.neg[,i] <- bins_helper(bins.exp.neg[,i],
                                  colnames(bins.exp.neg)[i])
}
```

```
## R1 spearman r: 0.9425675
## R2 spearman r: 0.9433265
## R3 spearman r: 0.9408037
## R4 spearman r: 0.9436019
## R5 spearman r: 0.9418503
## R6 spearman r: 0.9436669
## R1.TimeLeft spearman r: 0.9446858
## R2.TimeLeft spearman r: 0.9502324
## R3.TimeLeft spearman r: 0.9486652
## R4.TimeLeft spearman r: 0.9481581
## R5.TimeLeft spearman r: 0.945816
## R6.TimeLeft spearman r: 0.9473766
## close_friends spearman r: 0.9432373
## social_network spearman r: 0.942117
## SAT.COMP spearman r: 0.9438146
## SAT.MATH spearman r: 0.9404738
## SAT.VERBAL spearman r: 0.9433591
## Overall spearman r: 0.9451715
## Comprehension spearman r: 0.9781535
## State.Reasoning spearman r: 0.9233701
```

Let's check the histograms

```
hist.afterbin <- multi.hist(bins.exp.neg[,7:length(bins.exp.neg[1,])])
```

3.2 Low Income Sausage

From this sausage data, we can know that there are five products and 27 emotional words. 20 Judges are required to evaluate these products by these words. The question is: How do you feel when you taste this product? Emotions were presented in a list, and consumers were able to choose as many emotions as they want.

```
# import data
low.income.sausage <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/6Low
table.sausage <- read.csv(text = low.income.sausage)
colnames(table.sausage)[1] <- "Product"
kable(head(table.sausage[, 1:5]), align = "c")
```

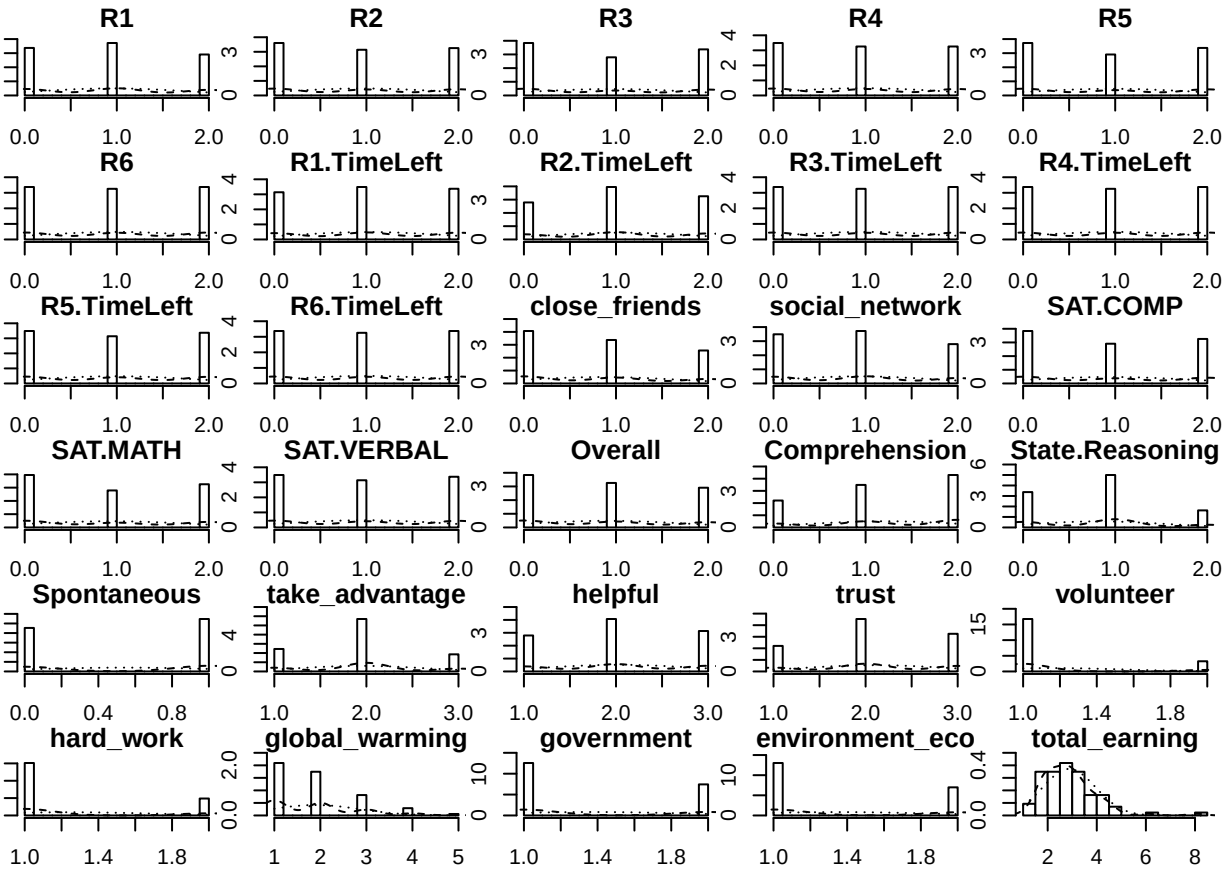


Figure 3.1: Histogram of Categorical Variables

Product	Happy	Pleasantly.surprised	Unpleasantly.surprised	Salivating
Alpino	27	21	11	19
Bafar	28	20	6	19
Chimex	26	20	7	12
Capistrano	32	16	6	15
Duby	33	17	9	12

3.3 Wines

There are several wines from two places: South Africa and France. The `wines` table includes all the rating from Judges; The `wines.name` is table with list of emotional words in English and French; The `wines.flavors` table is some emotional words which are used to describe these wines.

```
# import data
wines <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/Data_sorting_tri_
wines.names <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/Dictionary.
wines.flavors <- getURL("https://raw.githubusercontent.com/yilewang/MSA/main/Dataset/Voc.csv")

table.wines <- read.csv(text = wines)
table.wines.names <- read.csv(text = wines.names)
table.wines.flavors <- read.csv(text = wines.flavors)

### Avoid UTF-8 error
colnames(table.wines)[1] <- "X"
colnames(table.wines.names)[1] <- "French"
colnames(table.wines.flavors)[1] <- "X"


kable(head(table.wines[, 1:5]), align = "c")
```

X	J1	J2	J3	J4
FCAR	4	2	4	1
SRUD	3	4	3	1
FBAU	5	4	2	2
FROC	4	1	4	1
SFED	5	4	2	1
SREY	1	1	4	3

```
kable(head(table.wines.names), align = "c")
```

French	English
acide	acidic
peu acide	low acidity
agressif	aggressive
agrumes	citrus
agréable	pleasant
alcool	alcohol

```
kable(head(table.wines.flavors[, 1:5]), align = "c")
```

X	acide	peu.acide	agressif	agrumes
FCAR	5	2	0	2
SRUD	4	3	0	2
FBAU	2	5	0	0
FROC	7	1	0	0
SFED	6	2	1	2
SREY	7	2	1	1

Chapter 4

Principal Component Analysis

4.1 Introduction of PCA

In the first statistical method, I will start from **Principal Component Analysis** (PCA).PCA is a common analysis I used for data exploration and dimensionality reduction. I aim to use PCA to **extract important information** , and **find the similarity** from sets of variables I have. Since the origin of PCA can be traced to concepts of Correlation and Linear Analysis, I can see large overlap between PCA and other linear methods. However, the uniqueness of the PCA relies on its calculation process: **its goal is to convert the original data table (information) into new orthogonal variables.** These new orthogonal variables will have same numbers with the original data table but generally the first few variables will explain most of the total variance for the whole data table, which I called them *principal components* in our analysis. Generating these new *principal components* will be beneficial for us to know the what's dominant features in our data set and get rid of these noisy ones. Let's me give an example: In questionnaire study, sometimes I will ask questions with high similarity:

1. *Do you like pet?*
2. *Do you enjoying playing with your pet in your spare time?*

If I am a dog person, I will definitely answer "yes" at the first question and "My pleasure" at the second one. Reversely, If I am allergic to dog's hair and was bitten by a dog during my childhood, I may answer "No" and "Absolutely No" at the two questions. I bet 99% people will have consistent answers for the two questions.

Except questionnaire study, there are many other place I might need to reduce the similarity and extract the most important information from our data set. In neuroimaging study, sometimes I will need to find out which brain regions will have similar activation; In real life research, after collecting hundreds of participants' data, I want to know if male/female, or young/aged will have significant separation on some biological characteristics. PCA will be a good fit to reduce these kinds of redundancy. By looking into the `Scree Plot`[2.1], I can know how many variables explain much more variance than others. By looking into the `Loading`[2.6], I can know the relationship among different column variables. By looking into the `Factor Scores`[2.4] and `Contribution Barplot`[2.7], I can know which component are more closely associated with the separation of different groups. I will introduce these concepts in details in the following content.

Overall, PCA is a powerful statistical tool which is designed to help us find out the connections between variables and observation, and connection among multiple variables in single numeric data table (In the following chapter, I will mention more about the multiple data tables).

4.2 Computing

The first step I need to do is to run PCA analysis. In this part, I will use some trick to make my computational life easier. All important matrix will be stored as a short character for convenience.

- fs: factor scores
- eigs: eigenvalue
- tau: represents how many varience explained by the factor
- p.vals: the p value for the scree plot, which can tell us that what's the significant components in our results.
- boot.ratios: it is bootstrap ratio matrix helps us to generate bootstrap barplot in the end.
- cj: contribution ratio for the column variables
- fj: factor scores of column variables

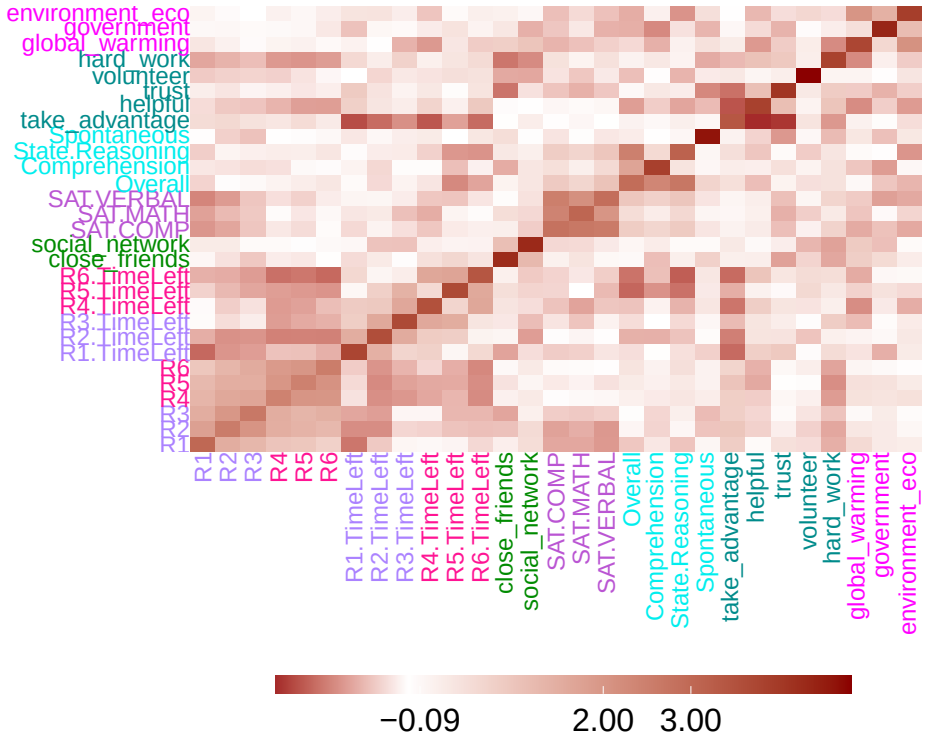
```
res.PCA <- epPCA(DATA = exp.neg[7:35], center = TRUE,
                 scale = 'SS1',
                 DESIGN = exp.neg$group, graphs = FALSE)
res.PCA.inference <- epPCA.inference.battery(DATA = exp.neg[7:35],
                                             center = TRUE,
                                              scale = 'SS1',
                                              DESIGN = exp.neg$group,
                                             graphs = FALSE)
fs <- res.PCA$ExPosition.Data$fi
eigs <- res.PCA$ExPosition.Data$eigs
tau <- res.PCA$ExPosition.Data$t
p.eigs <- res.PCA.inference$Inference.Data$components$p.vals
eigs.permu <- res.PCA.inference$Inference.Data$components$eigs.perm
boot.ratios <- res.PCA.inference$Inference.Data$fj.boots$tests$boot.ratios
cj <- res.PCA$ExPosition.Data$cj
fj <- res.PCA$ExPosition.Data$fj
```

4.3 Heatmap

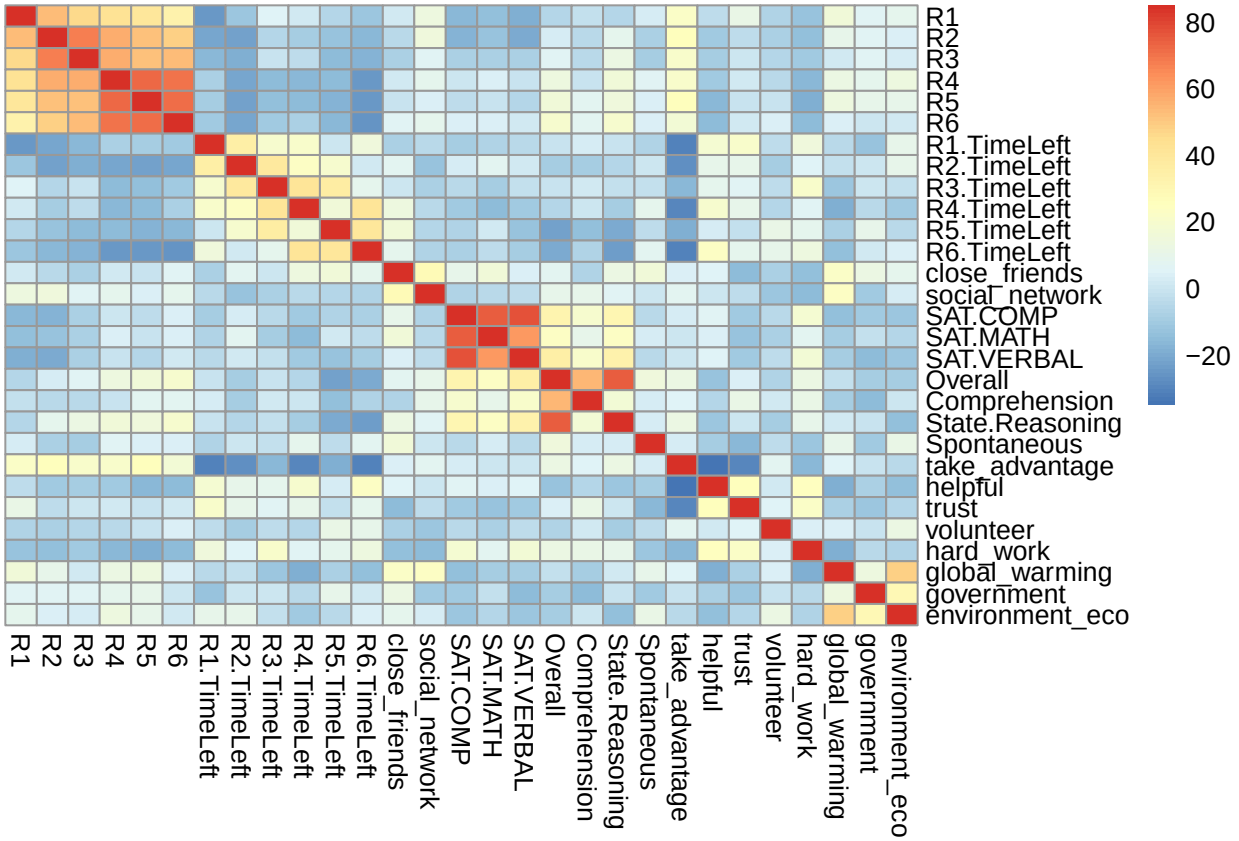
In the heatmap part, the input data is X^TX, which represent the co-variance Matrix or the correlation matrix in PCA. From the heatmap, I can know that some variables are closely correlated with each other, such as game performance, time usage, personality trait and attitude toward global environment. The heatmap can give us a hint about the following PCA results.

```
# plot
pca.xtx <- as.matrix(t(scale(exp.neg[,7:35],
```

```
                              center = TRUE,
                              scale = TRUE)) %*% scale(exp.neg[,7:35],
                                                       center = TRUE,
                                                       scale = TRUE))
plot.heatmap(DATA = pca.xtx,
             xcol = m.color.design,
             ycol = m.color.design,
             textsize = 3)
```



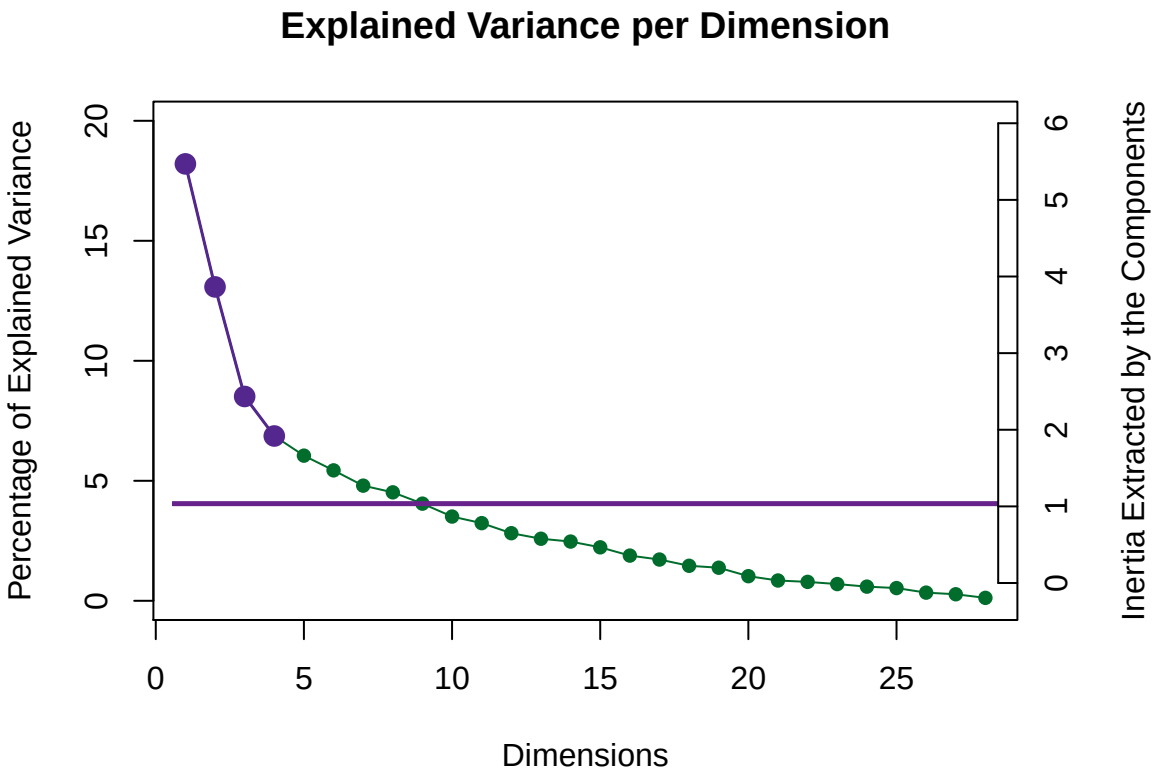

```
pheatmap(mat=pca.xtx, cluster_rows = FALSE,
         cluster_cols = FALSE )
```



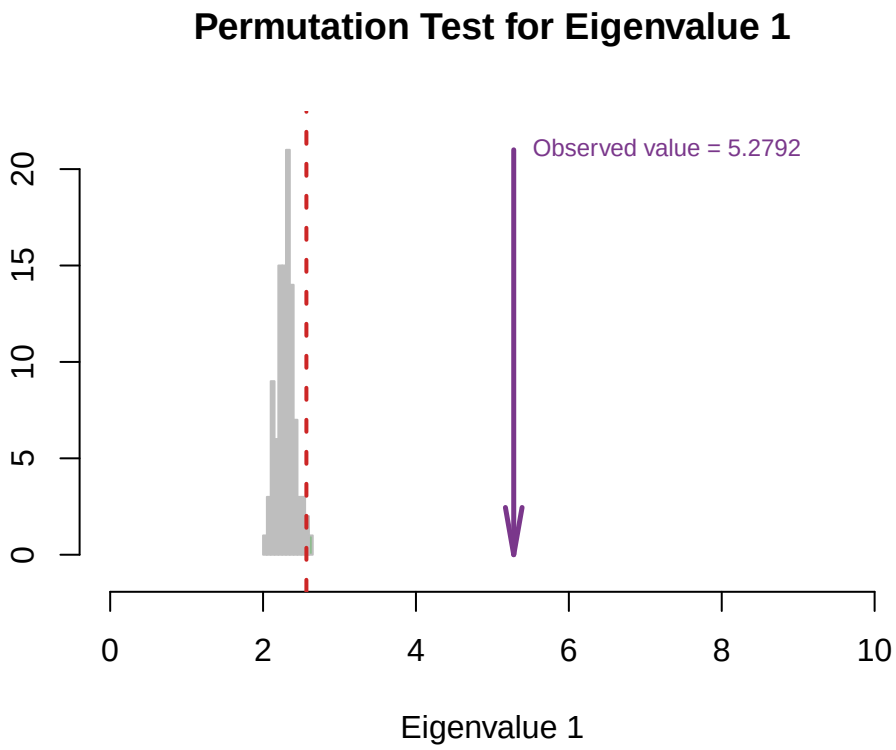
4.4 Scree Plot

For the scree plot and permutation test, I can know how many components are statistically significant in our study, and whether the results is stable. Permutation test is designed for a *NS* test, to test the null hypothesis. By repeatedly sampling from data (putting it back, compared to bootstrap ratio), it will generate a new frequency distribution about the overall probability of outcome I observed. By comparing the value in the distribution range, I can know how confident I are to reject the null hypothesis and how much probability I have to make the typeI error. From the results below, I conclude that

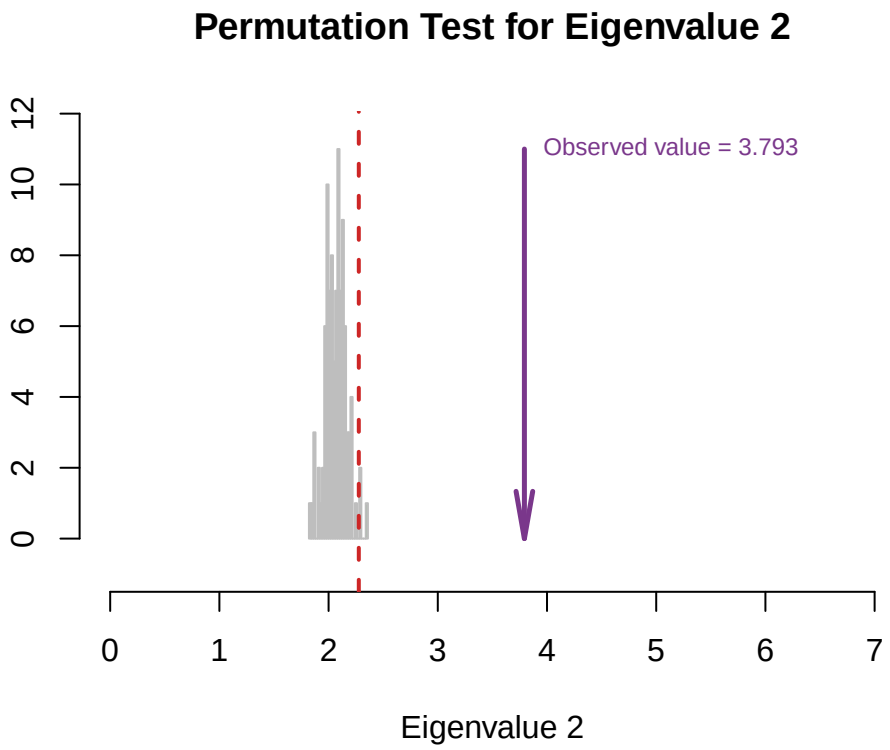
```
plot.scree(eigs, p.eigs)
```




```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs, para1=10)
```



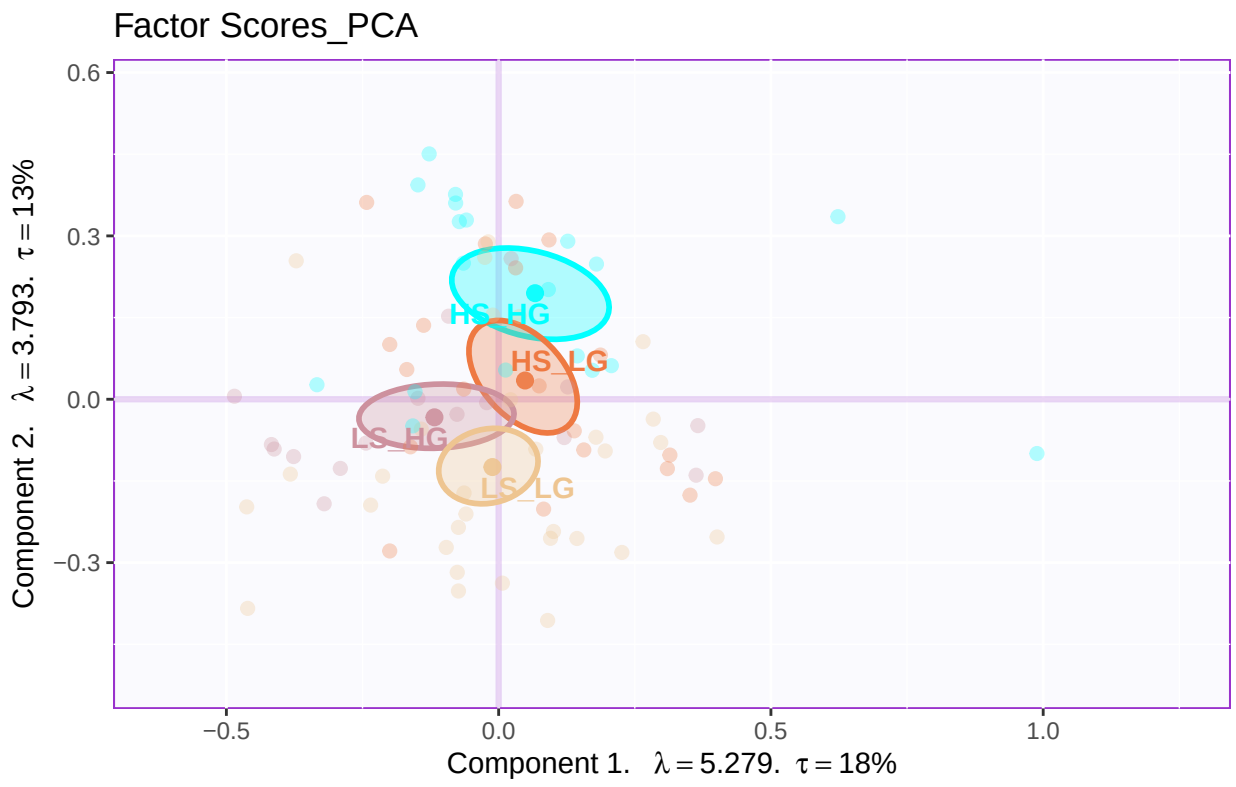

```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs, para1 = 7, Dim = 2)
```



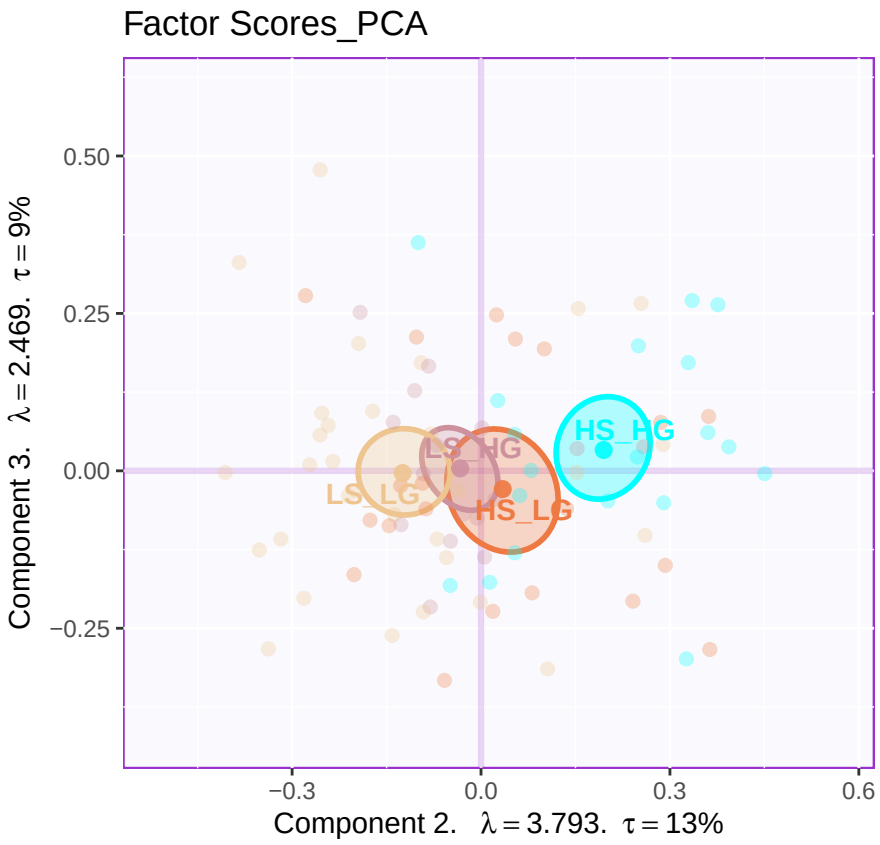
4.5 Factor Scores

The factor scores is the most important part in our PCA analysis. As you can see, with the help of the function, things become such easy to have our factor scores plot. In the two plot below, I can know that they are separated with each other in great extend. The dimension 2 perfectly distinguishes **Hig** social intelligence and **High** general intelligence participants with **Low** social intelligence and **Low** general intelligence ones. The interesting thing is that it looks like a gradient distribution: HIGH_HIGH at the top, LOW_LOW at the bottom, and HIGH_LOW in the middle. From the factor scores plot, I can also know that high social intelligence is closer to double high participants. However, the Dimension 1 and Dimension 3 are not much informative for us from the plots.

```
plot.fs(exp.neg$group,
        fs=fs,
        eigs=eigs,
        tau=tau,
        d = 1,
        mode="CI",
        method = "PCA")
```



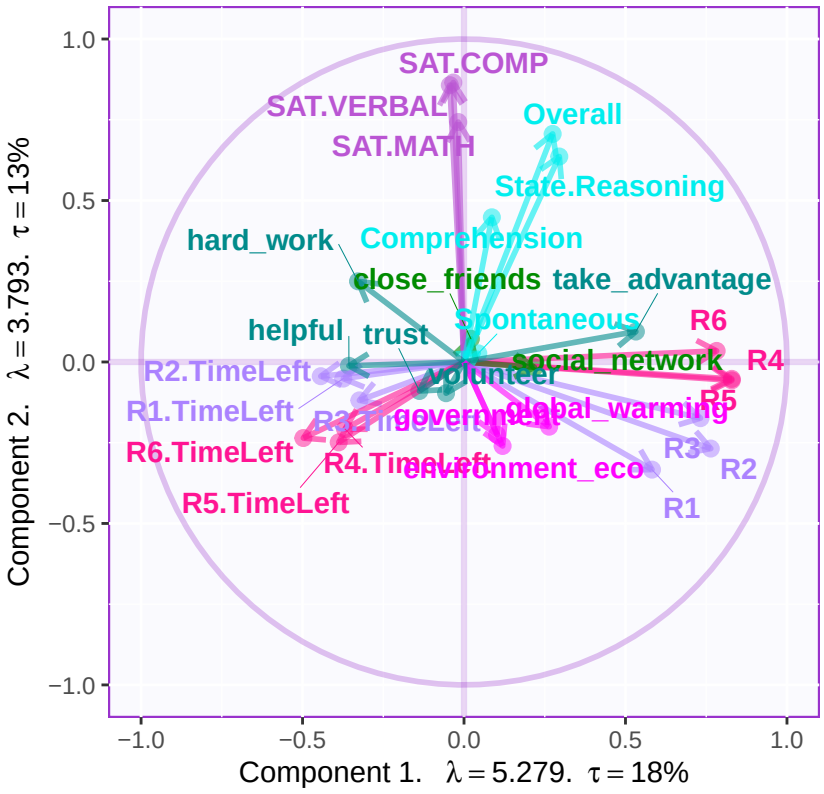

```
plot.fs(exp.neg$group,
        fs=fs,
        eigs=eigs,
        tau=tau,
        d = 2,
        mode="CI",
        method = "PCA")
```



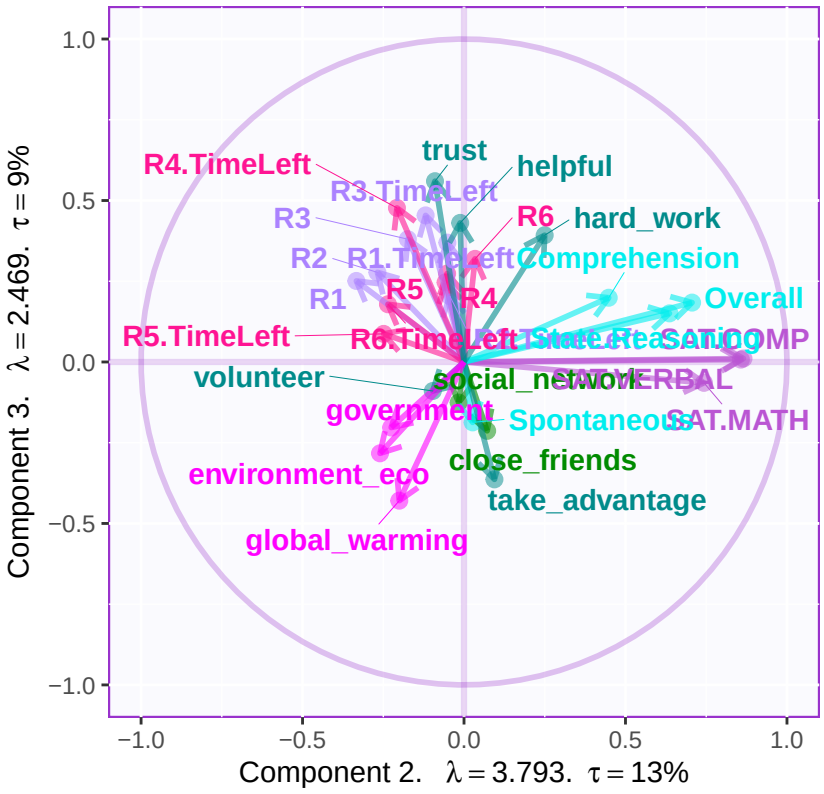
4.6 Loading

From the loading plot, I can see that they are some determined factors associated more with dimension 2: The SAT results and empathy are related with dimension 2, which represents that the two aspects are important for the separation of the groups.

```
plot.loading(exp.neg[,7:35],
             col=m.color.design,
             fs=fs,
             eigs=eigs,
             tau=tau, d=1)
```



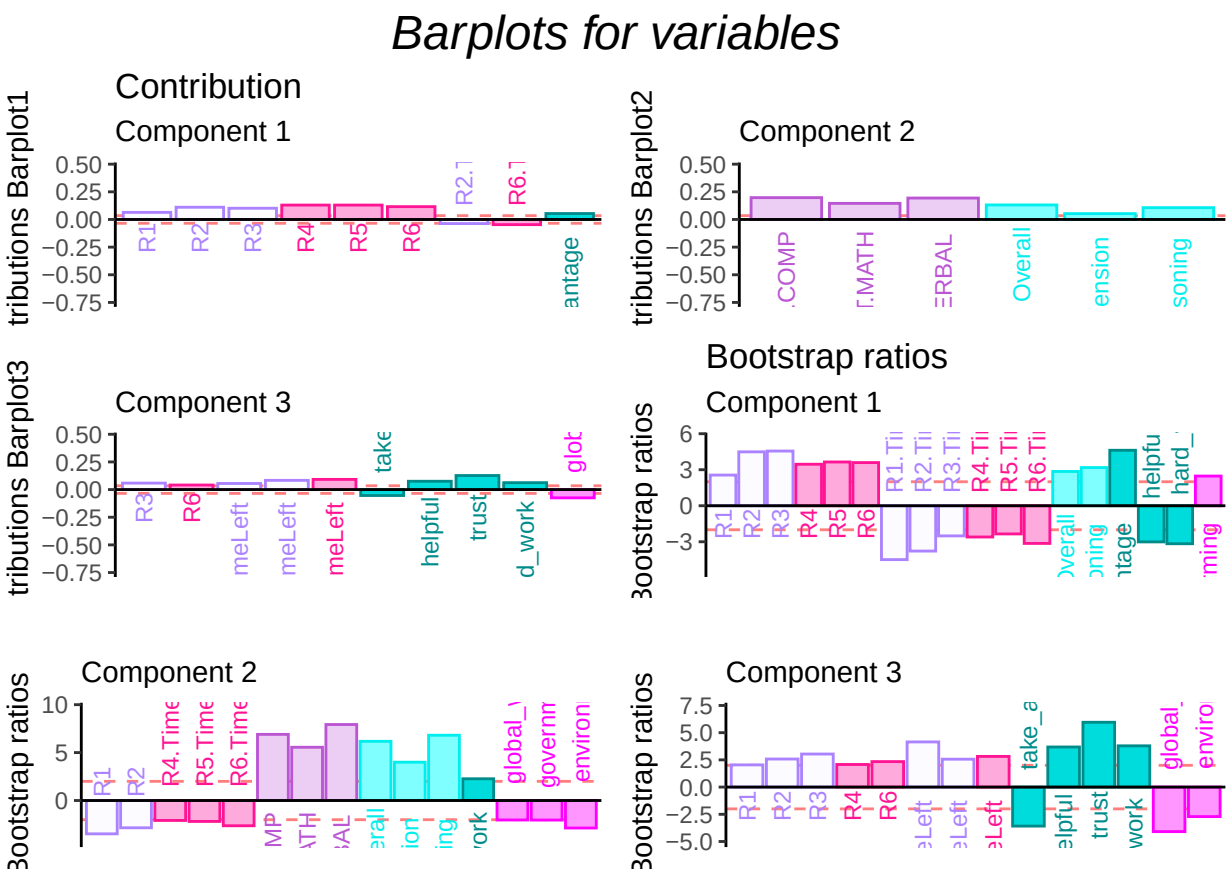

```
plot.loading(exp.neg[,7:35],
             col=m.color.design,
             fs=fs,
             eigs=eigs,
             tau=tau, d=2)
```



4.7 Contriution and Bootstrap Ratio Barplots

From the contribution and bootstrap ratio barplots, I can know that the first component is closely related with participants' game performance. SAT results and empathy related ability are significantly contributing to component2, which aligns with the conclusion of the published article.

```
plot.cb(cj=cj,
        fj=fj,
        col = m.color.design,
        boot.ratios = boot.ratios,
        signifOnly = TRUE,
        fig = 3,
        horizontal = TRUE,
        colrow = "col")
```



4.8 Conclusion

It is easy to observe that they are connection between the game performance results and some personality or attitude variables. However, it is worth to mention that currently the separation doesn't represent that I have already had the evidence the collective action will be affected by social intelligence and general intelligence. Please remember our group variable is **GPA** and **Emotion Processing Ability**, the two variables are naturally correlated with the previous two aspects. I need more evidence on other variables to prove that I are right.

Chapter 5

Correspondence Analysis

5.1 Introduction of CA

What's Correspondence Analysis (CA)? If you want me to answer this question shortly, CA is a special version of PCA. Originally, what CA does is to analyze contingency table (frequency distribution table). If you still remember the metaphor that doing research is to observe texture inside the tree (data), CA is the saw (method) exclusive to categorical data (tree). In CA, rows and columns will be treated equivalently and new orthogonal components will be generated for each item based on the contingency table.

Why PCA is not a good idea to analyze distribution data? PCA is sensitive mainly to the `number` instead of a `relative frequencies`, which implies us that when we want to investigate the relationship among this kind of variable, PCA will give us an inaccurate results. In order to reveal hidden information from data *distribution* and *proportion*, preprocessing and transformation are necessary before our analysis. Generally what we will do is to compute the average of rows and columns, then calculate their expect number (masses). After that, we try to look at the residuals between expected proportions and real numbers. By using the singular vector decomposition (SVD), we can have left and right singular vectors correspond to row and columns of the table. Based on the results, we can plot row factor scores and column factor scores from the first two dimension. Basically it is what CA do in our analysis.

In the rest of this chapter, we will focus on the interpretation of our results in low income sausage example [3.2]

5.2 Computation

First of all, similar with PCA and all other analysis, a preprocessing step is necessary for our results. In CA's analysis, I am not using our main data set, instead, I am using a commercial food data table including various kinds of sausage but for low income population. although I don't know why it is labeled as "low income".

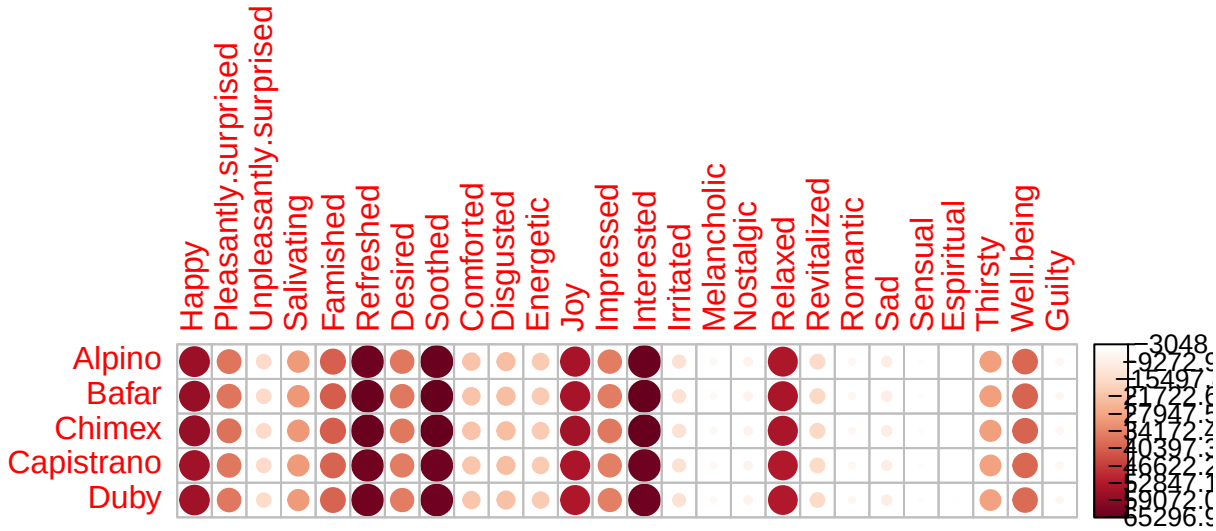
There are several matrices important for plotting our heatmap. Since it is a frequency data table (people chose words that best describe their feeling on each product), chi2-test is necessary. From the results of chi2-test and heatmap, we can know that specific emotional words are highly related with some products, such as "comforted feeling" for Capistraino

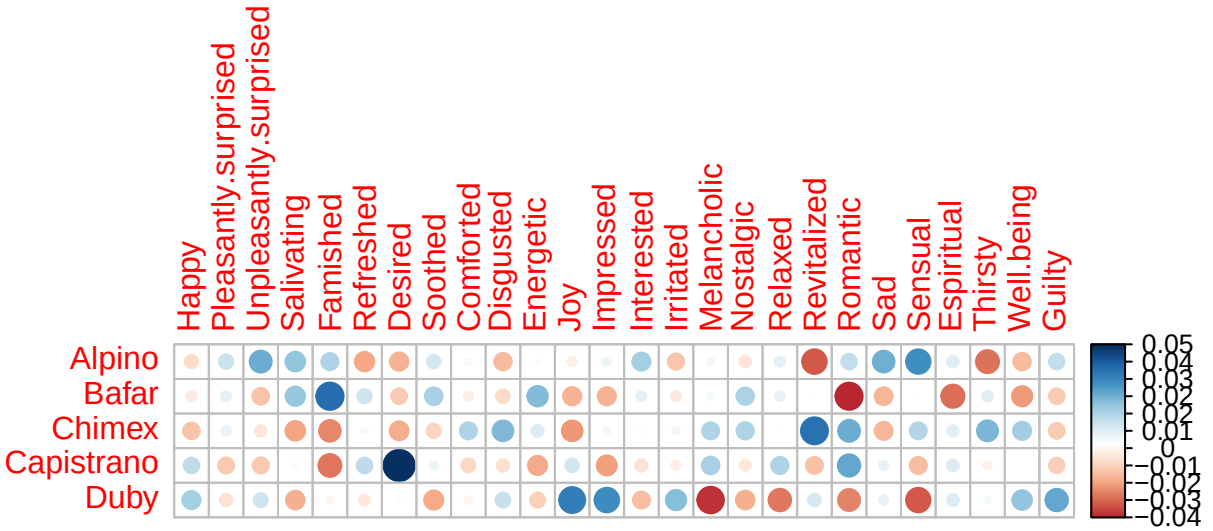
```r
# Data preprocessing
rownames(table.sausage) <- table.sausage$Product
table.sausage.ca <- subset(table.sausage,
                           select = -c(Product))

# get chi2
chi2.sausage <- chisq.test(table.sausage.ca)
inertia.cells <- chi2.sausage$residuals / sqrt(sum(table.sausage.ca))
Z <- table.sausage.ca / sum(table.sausage.ca) # observed
r <- as.matrix(rowSums(table.sausage.ca)) # expected row
c <- as.matrix(colSums(table.sausage.ca)) # expected column

# SVD computation
test.inertia.cells <- as.matrix(diag(as.vector(r^(-1/2)))) %*%
  as.matrix(Z - r%*%t(c)) %*% as.matrix(diag(as.vector(c^(-1/2))))
rownames(test.inertia.cells) <- rownames(table.sausage.ca)
colnames(test.inertia.cells) <- colnames(table.sausage.ca)
```

5.3 Heatmap

Also, it is worth to mention that the subtraction of rcT from Z gave us a intrinsic results about which emtional words will be associated with all the products.From the inertia heatmap, we can know that some consumers/judges will have more difference on some product, such as Desired on Capistrano. The heatmap gave us an overall information about some statistical attribution of the data table.

```r
# heatmap of Z - r%*%t(c)
Z.heatmap <- corrplot(as.matrix((Z - r%*%t(c))),
                      is.cor = FALSE)
# heatmap of inertia cells
inertia.heatmap <- corrplot(as.matrix((inertia.cells)),
                            is.cor = FALSE)
```


Happy
Pleasantly.surprised
Unpleasantly.surprised
Salivating
Famished
Refreshed
Desired
Soothed
Comforted
Disgusted
Energetic
Joy
Impressed
Interested
Irritated
Melancholic
Nostalgic
Relaxed
Revitalized
Romantic
Sad
Sensual
Espiritual
Thirsty
Well.being
Guilty
Alpino
Bafar
Chimex
Capistrano
Duby
3048



5.4 Computation

And then, as usual, we need to do some computation to have our CA results. Due to the switch on and off at Symmetric option, we can have totally different two factor scores map in our study: The Asymmetric one and Symmetric one. For better understanding, the symmetric one means that we treated column and rows in same manner; The asymmetric method means that sometimes the row and column variables have special connection (such as dependent and independent variables). Asymmetric method can help us better understanding the distance between each observation or each group in factor scores.

```
res.CA.sym <- epCA(t(table.sausage.ca),
                   symmetric = TRUE,
                   graphs = FALSE)
res.CA.sym.inf <- epCA.inference.battery(t(table.sausage.ca),
                                         symmetric = TRUE,
                                         graphs = FALSE)
res.CA.sym.inf.IJ <- epCA.inference.battery(table.sausage.ca,
                                            symmetric = TRUE,
                                            graphs = FALSE)
res.CA.asym <- epCA(t(table.sausage.ca),
                    symmetric = FALSE,
```

```
                    graphs = FALSE)
res.CA.asym.inf <- epCA.inference.battery(t(table.sausage.ca),
                                          symmetric = FALSE,
                                          graphs = FALSE)
eigs <- res.CA.sym$ExPosition.Data$eigs
eigs.permu <- res.CA.sym.inf$Inference.Data$components$eigs.perm
tau <- res.CA.sym$ExPosition.Data$t
p.val <- res.CA.sym.inf$Inference.Data$components$p.vals
fs <- res.CA.sym$ExPosition.Data$fi # sym and assym are same
fj.s <- res.CA.sym$ExPosition.Data$fj
fj.a <- res.CA.asym$ExPosition.Data$fj
cj <- res.CA.sym.inf$Fixed.Data$ExPosition.Data$cj
ci <- res.CA.sym$ExPosition.Data$ci
boot.ratios.fj <- res.CA.sym.inf$Inference.Data$fj.boots$tests$boot.ratios
boot.ratios.fi <- res.CA.sym.inf.IJ$Inference.Data$fj.boots$tests$boot.ratios
```

5.5 Scree Plot

For the Scree plot and permutation, I have already introduced the concept before, so it is needless to repeat them again [4.4]. From the results, I can know that our CA results may be located at the 5% position in the distribution.

```
plot.ca.scree <- plot.scree(eigs, p.val)
```

```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs,
                 Dim = 1,
                 para1 = 0.06)
```

```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs,
                 Dim = 2,
                 para1 = 0.06)
```

5.6 Symmetric & Asymmetric Plots

This chunk contains symmetric plotting. We can see that I manually classified these emotional words myself. It is obvious that component 1 mainly represents the *positive* words and component 2 is *negative*.

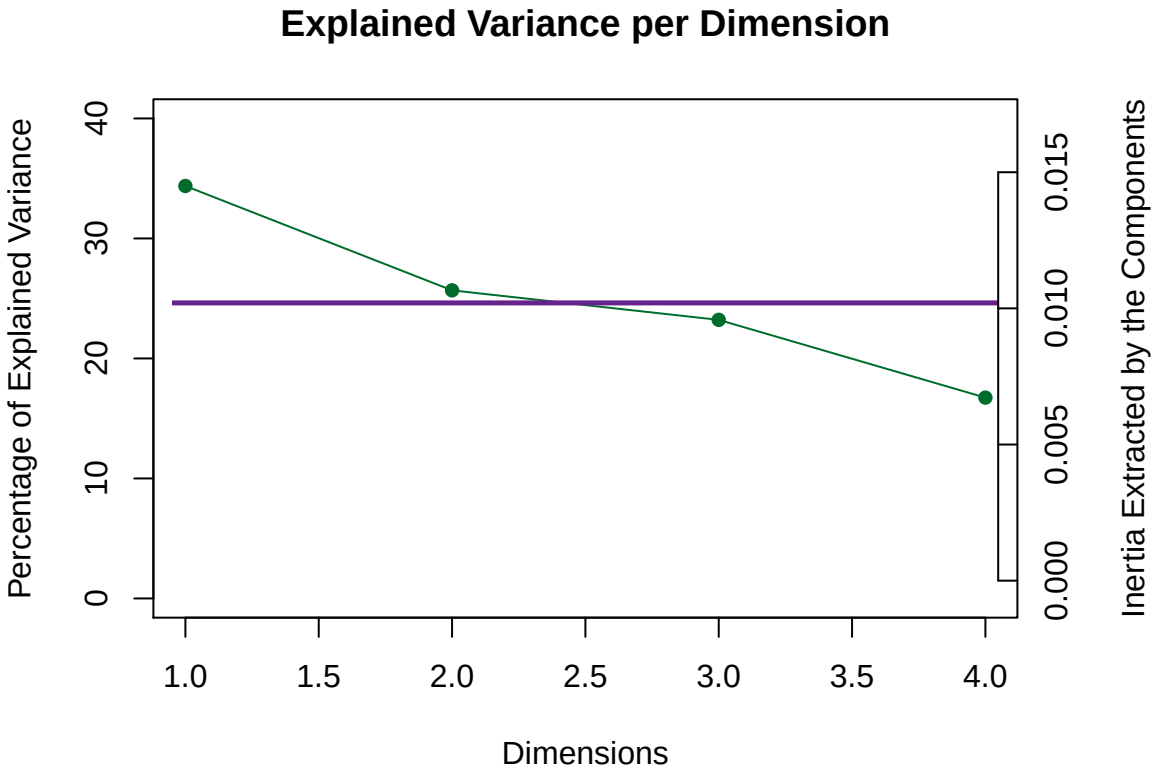


Figure 5.1: CA Scree Plot

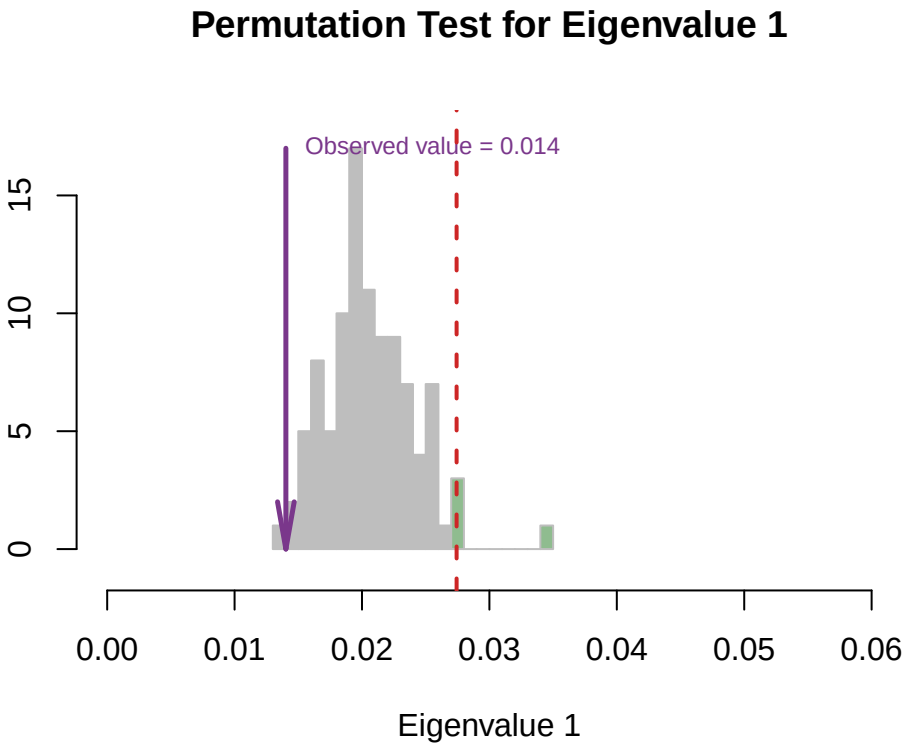


Figure 5.2: CA Scree Plot

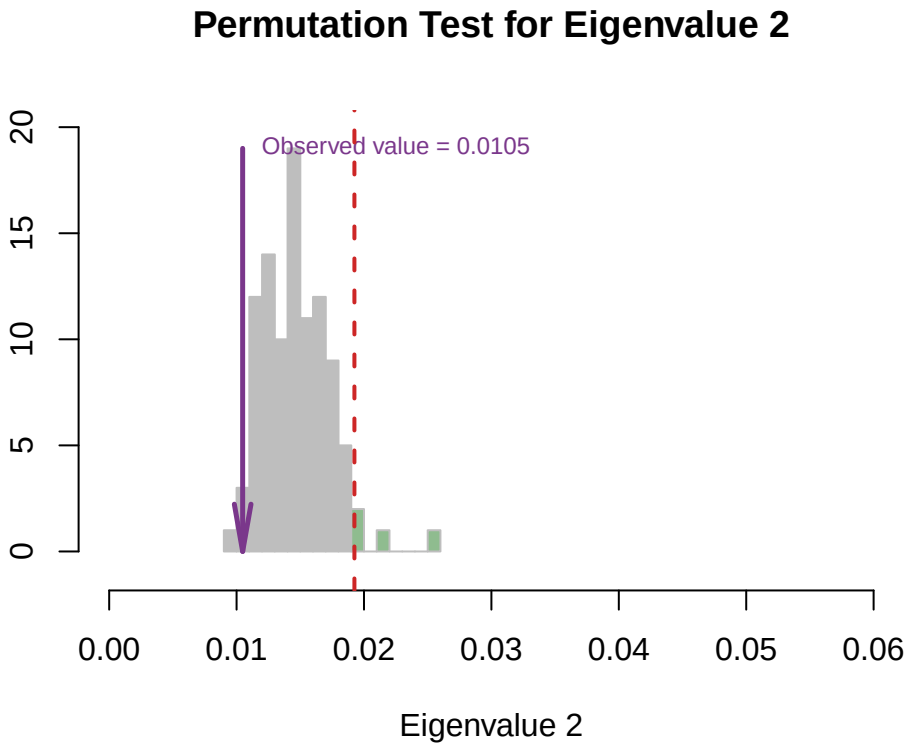


Figure 5.3: CA Scree Plot

```
# get some color, positive =1 and negative =2
color.ca <- as.matrix(t(table.sausage.ca)[,1])
rownames(color.ca) <- rownames(t(table.sausage.ca))
color.ca <- c(1,1,2,1,1,1,1,1,1,2,1,1,1,1,2,2,2,1,1,1,2,1,1,2,1,2)
color.ca[which(color.ca == 1)] <- index[1]
color.ca[which(color.ca == 2)] <- index[2]

# dimension 1
d = 1

# generate Symmetric Map
CA.symMap  <- createFactorMapIJ(fs,
                                fj.s,
                                col.points.i = color.ca,
                                col.labels.i = color.ca)

fj.labels <- createxyLabels.gen(d,(d+1),
                            lambda = eigs,
                            tau = round(tau),
                            axisName = "Component "
                            )

Map.fs.sym <- CA.symMap$baseMap +
  fj.labels +
  CA.symMap$I_points +
  CA.symMap$I_labels
Map.fj.sym <- CA.symMap$baseMap +
  fj.labels +
  CA.symMap$J_points +
  CA.symMap$J_labels

Map.fs.sym
Map.fj.sym
```

From the Asymmetric simplex plot, we can have a much closer look at the distance between products and emotional words. Chimex and Capistrano may give us a romantic feeling and Bafar is more sensual.

```
CA.asymMap  <- createFactorMapIJ(fs,
                                fj.a,
                                col.points.i = color.ca,
                                col.labels.i = color.ca,
                                cex.i = 3,
                                cex.j = 3,
                                text.cex.i = 5,
                                text.cex.j = 5)
```

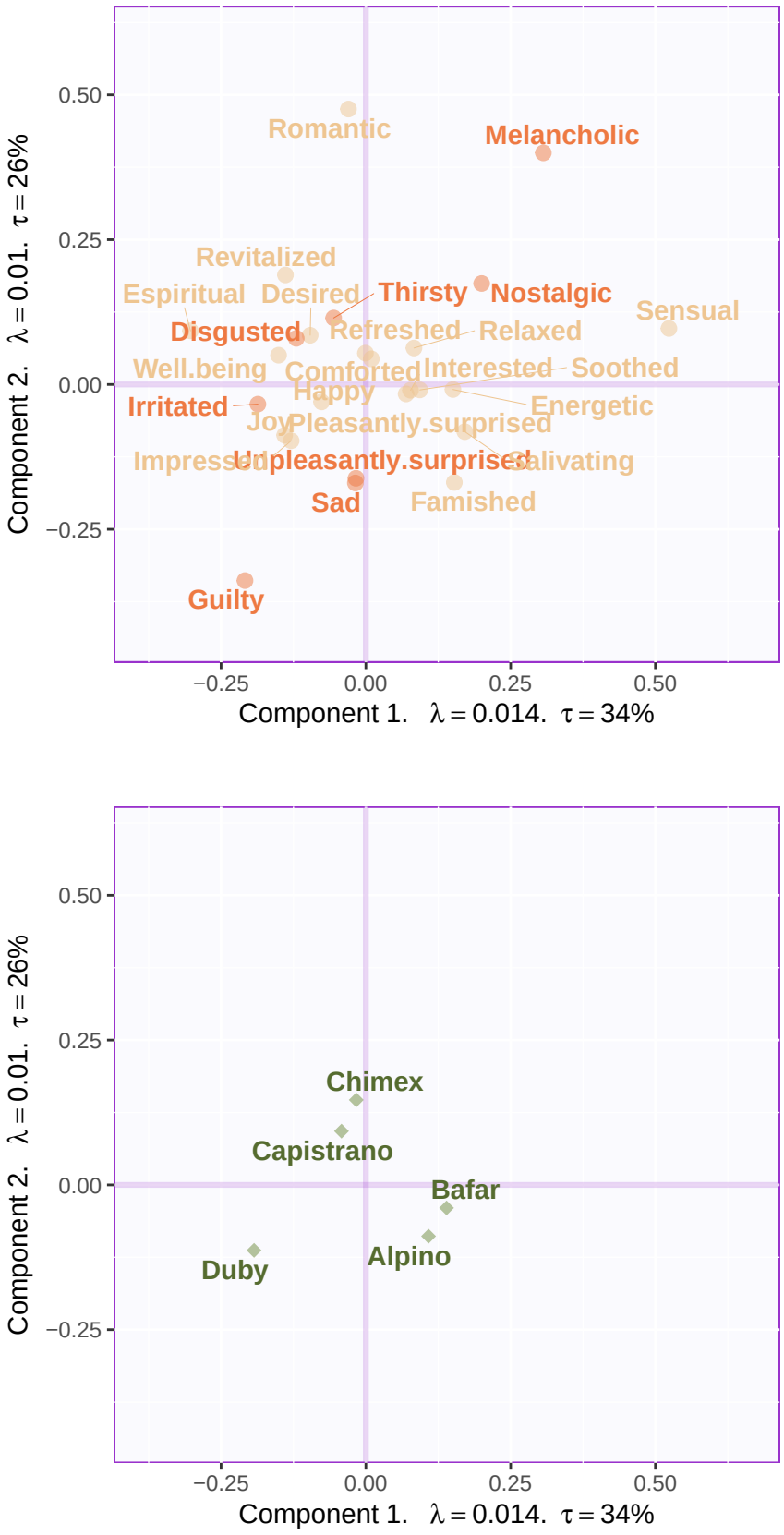
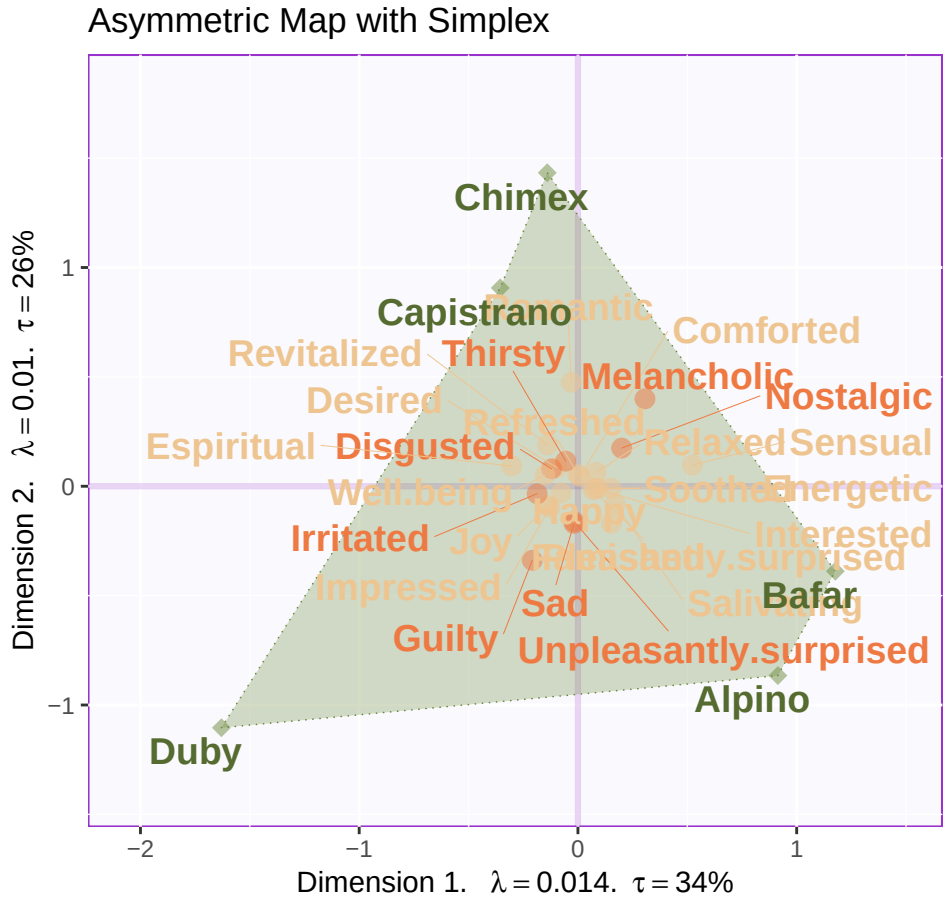


Figure 5.4: CA Symmetric Plot

```
zeSimplex <- PTCA4CATA::ggdrawPolygon(fj.a,
                                      color = 'darkolivegreen4',
                                      size = .3,
                                      fill =  'darkolivegreen4',
                                      alpha = .3,
                                      linetype = 3
                                      )

labels4CA <- createxyLabels(resCA = res.CA.asym)

Map.Asym <- CA.asymMap$baseMap +
  zeSimplex +
  labels4CA +
  CA.asymMap$I_points +
  CA.asymMap$J_points +
  CA.asymMap$I_labels +
  CA.asymMap$J_labels +
  ggtitle("Asymmetric Map with Simplex")

Map.Asym
```



5.7 Contribution and Bootstrap Ratio Barplot

The contribution and boostrap ratio will give us more precise information about each variable's importance in my results. It is interesting that in the barplots, I found that there are some positive words have different direction on component contribution. The romantic and melancholic has different valence but they are more closer to each other; Joy and sensual are much different than I expect for describing the sausage. As far as I am concerned, these emotional words are still too abstract to precise describe food. Thus, it could be the reason why the postive and negative valence are mixed together in our results.

```
# plot contribution barplots
plot.cb(cj=cj,
        fj=fj.s,
        boot.ratios = boot.ratios.fj,
        fig=3,
        horizontal = TRUE,
        signifOnly = FALSE,
        colrow = "row")
plot.cb(cj=ci,
```

```
        fj=fs,
        boot.ratios = boot.ratios.fi,
        fig=3,
        horizontal = TRUE,
        colrow = "row",
        signifOnly = FALSE,
        col = color.ca)
```

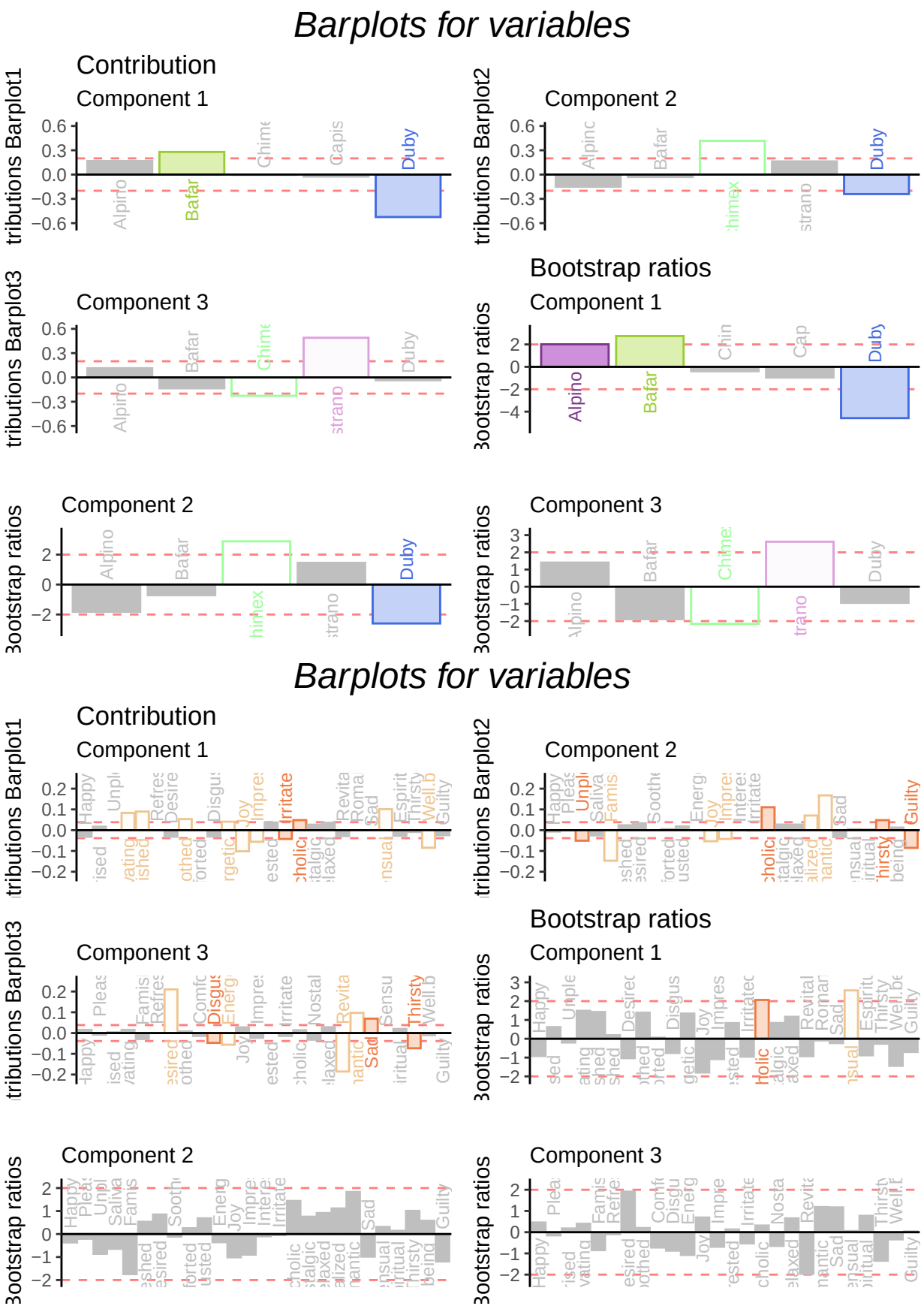


Figure 5.5: CA Contribution Barplots

Chapter 6

Discriminant Correspondence Analysis

6.1 Introdcution of DiCA

As we can know from the name, DiCA is an extensional branch of CA. The goal of DiCA is always categorize observation into different clusters. Sometimes we will make comparision between BADA and DiCA. An easy way to distinguish the two methods is that BADA is used with quantitative variables and DiCA is used with qualitative variable.

In this example, we will used quantitative version of our main data set to do the DiCA analysis. For more info about the main data set, you can check **data intro** part in this book 3.1.

In the data preprocessing stage, we have a difficulty here: The global_warming variable cannot be included into our inference so I have to drop it in my subsequent analysis. The main reason could be that the distribution of global_warming are too skewed.

```
# import data
# data set: bins.data.to.use.sup

# design: exp.neg \ exp.pos


# Cannot analyze this variables, so I have to drop it
DiCA.raw.data <- select(bins.exp.neg, c(-"global_warming"))

res.DiCA <- tepDICA(DATA=DiCA.raw.data[,7:34],
                    DESIGN = exp.neg$group, make_data_nominal = TRUE,
                    symmetric = TRUE ,
                    graphs = FALSE)
res.DiCA.inf <- tepDICA.inference.battery(DATA= DiCA.raw.data[,7:34],
                                          DESIGN = exp.neg$group,
                                          make_data_nominal = TRUE,
                                          symmetric = TRUE ,
```

```
                                          graphs = FALSE,
                                          test.iters = 100)
```

```
## [1] "It is estimated that your iterations will take 0.05 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take not
## ============================================================================
```

As usual, I will prepare all the matrices and parameters for plotting usage.

```
fs <- res.DiCA$TExPosition.Data$fi
fj <- res.DiCA$TExPosition.Data$fj
eigs <- res.DiCA$TExPosition.Data$eigs
tau <- res.DiCA$TExPosition.Data$t
cj <- res.DiCA$TExPosition.Data$cj
p.vals <- res.DiCA.inf$Inference.Data$components$p.vals
eigs.permu <- res.DiCA.inf$Inference.Data$components$eigs.perm
boot.ratios <- res.DiCA.inf$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
fii <- res.DiCA$TExPosition.Data$fii
```

6.2 Histgram of Binning Variables

I will use bins variable in this example instead of the continuous data table. If you are interested in the details about how I bin these variables, please see the functions `Bins_Helper`[2.8]

6.3 Heatmap

In the heatmap, I used a trick to get means and scale them in rows. As I can know from the results, The High_High group (high social intelligence and high general intelligence) will be positively correlated to their SAT scores, and also the empathy ability. It is expected because I am using college GPA and emotional processing ability as the grouping variables. It is not surprised that they are correlated in pairs (GPA with SAT; empathy with emotional processing).

Don't forget about the collective action, which is the main characters in this story: It is interesting that Low social intelligence but High general intelligence group has negative correlation with all the R-R6 game performance. It may imply something in our results.

```
# as.matrix((t(DESIGN %*% X)))

raw.dica <- scale(exp.neg[7:35],
                  scale = TRUE,
                  center = TRUE)
tmp <- aggregate(raw.dica, list(exp.neg$group),
                 mean)
mean.dica.raw <- tmp[,2:ncol(tmp)]
```

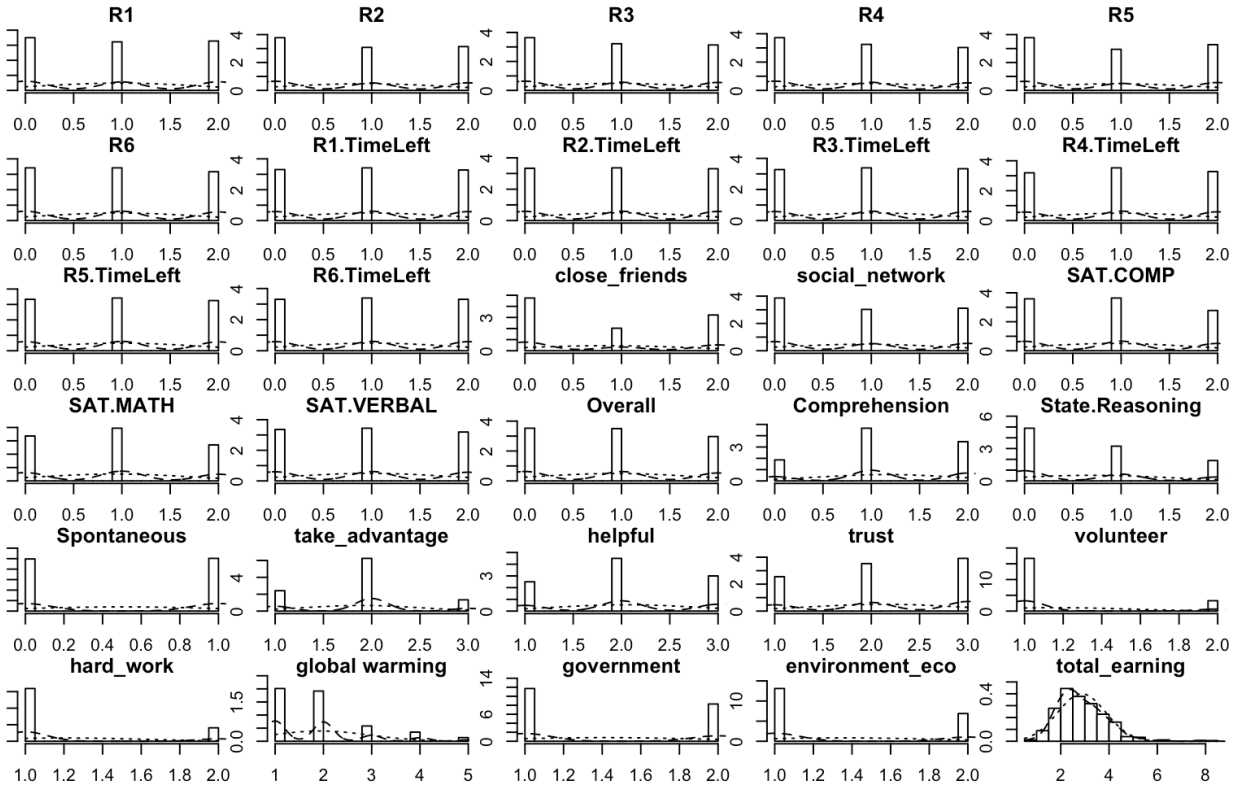
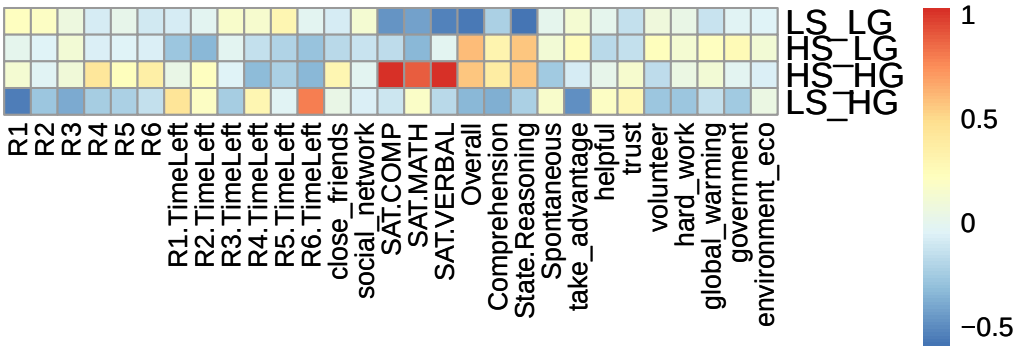


Figure 6.1: Multiple Histgram of Binning Data

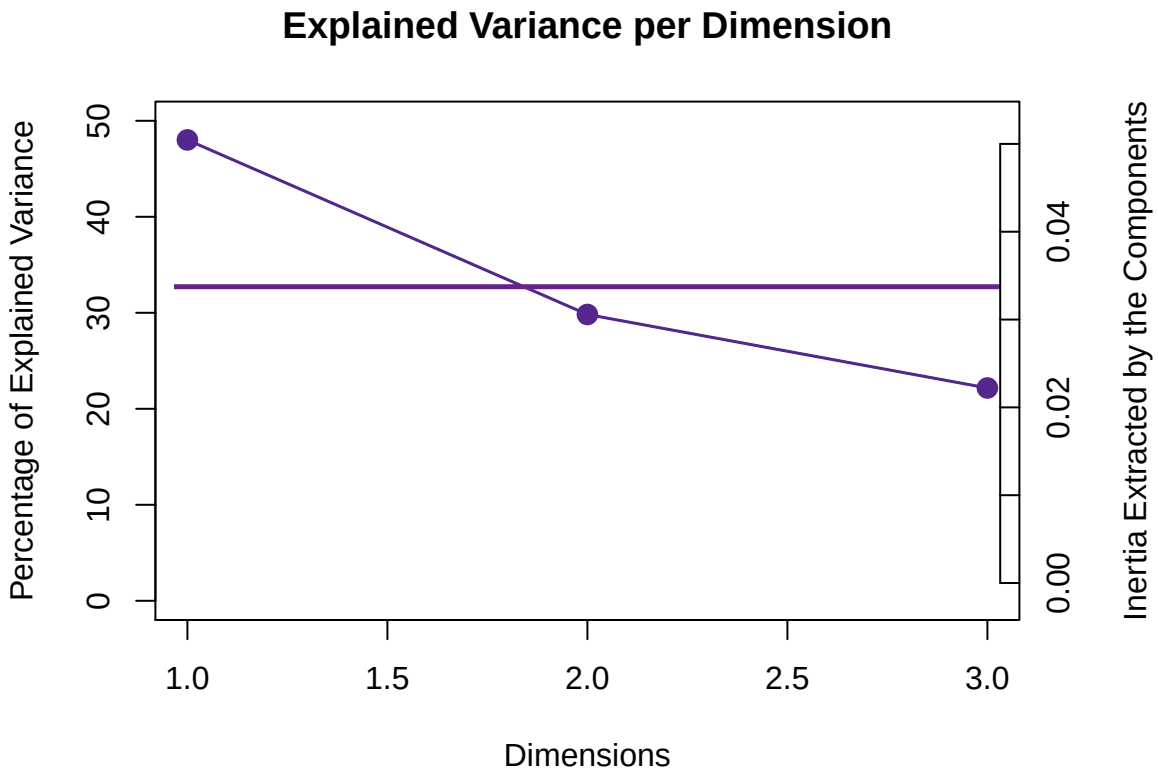
```
rownames(mean.dica.raw) <- tmp[,1]

dica.hm <- pheatmap(mat=mean.dica.raw,
                    fontsize_row =13,
                    fontsize =10,
                    cellheight = 10,
                    cellwidth = 10,
                    angle_col = 90,
                    cluster_rows = FALSE,
                    cluster_cols = FALSE)
dica.hm
```



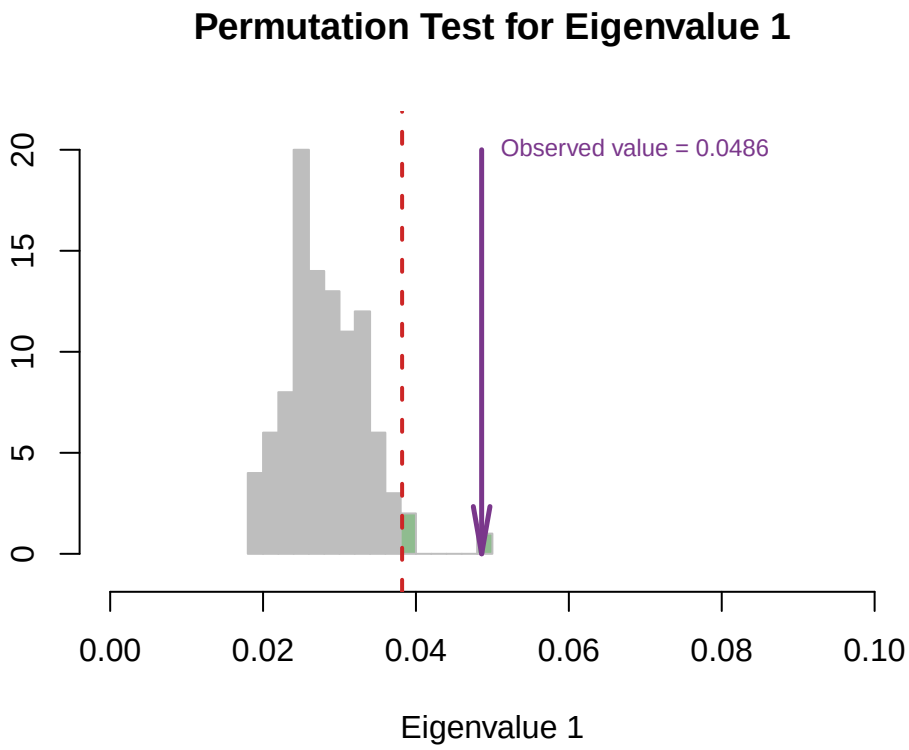
6.4 Scree Plot

From the scree plot, as usual, it will tell me how much variance explained by each component. From the permutation test, I can know I can reject the null hypothesis ??.

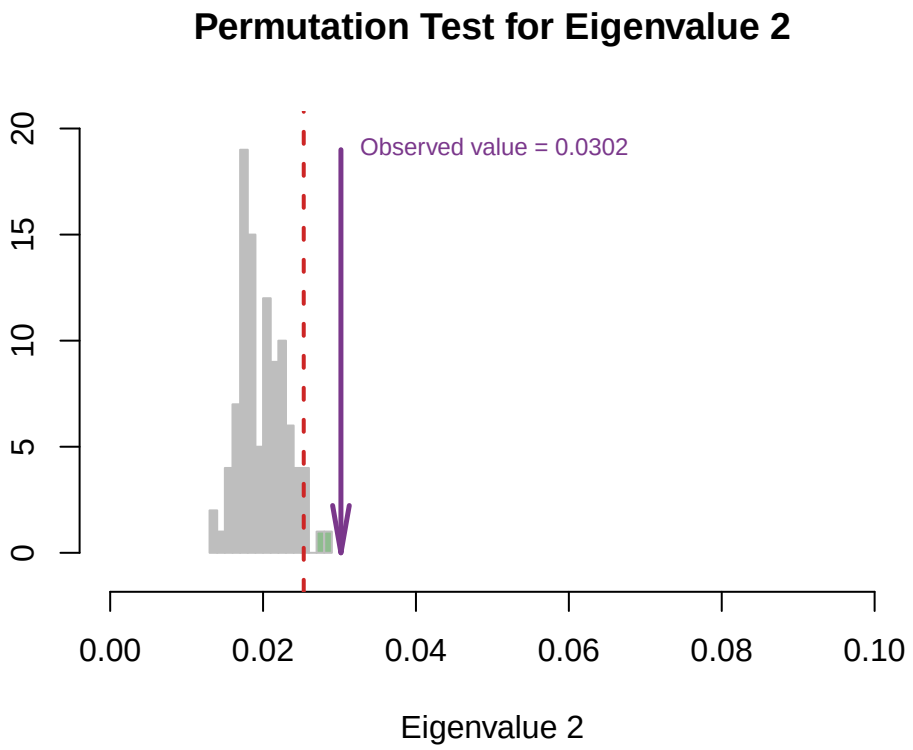
```
plot.scree(eigs = eigs, p.vals = p.vals)
```




```
plot.permutation(eigs.perm = eigs.permu,
                 eigs,
                 para1 = 0.1,
                 Dim = 1)
```




```
plot.permutation(eigs.perm = eigs.permu,
                 eigs,
                 para1 = 0.1,
                 Dim = 2)
```

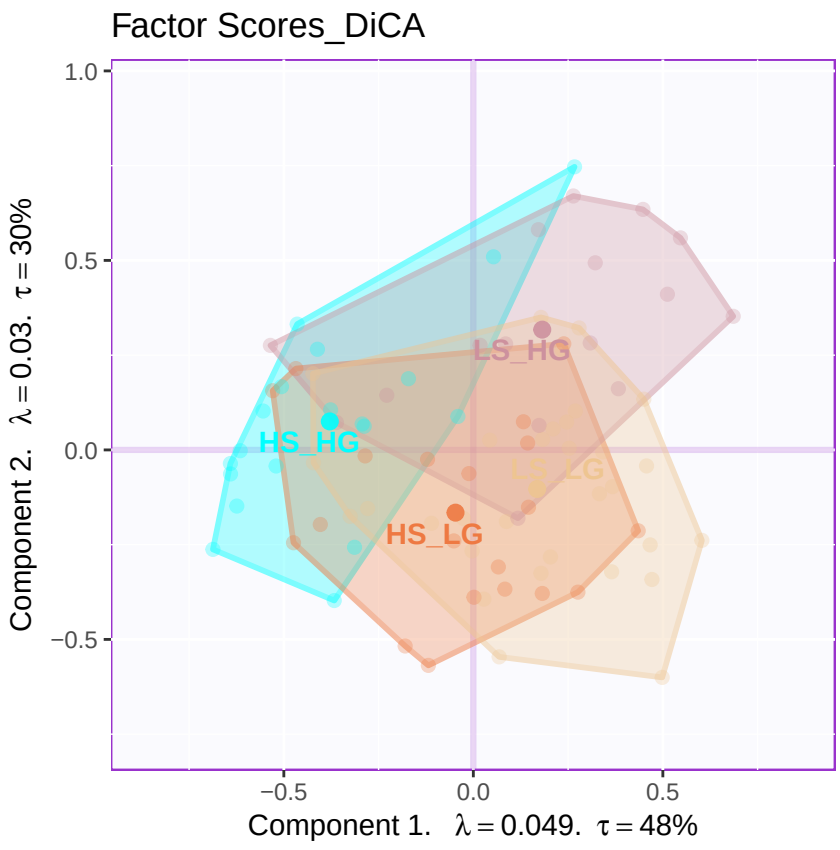


6.5 Factor Scores

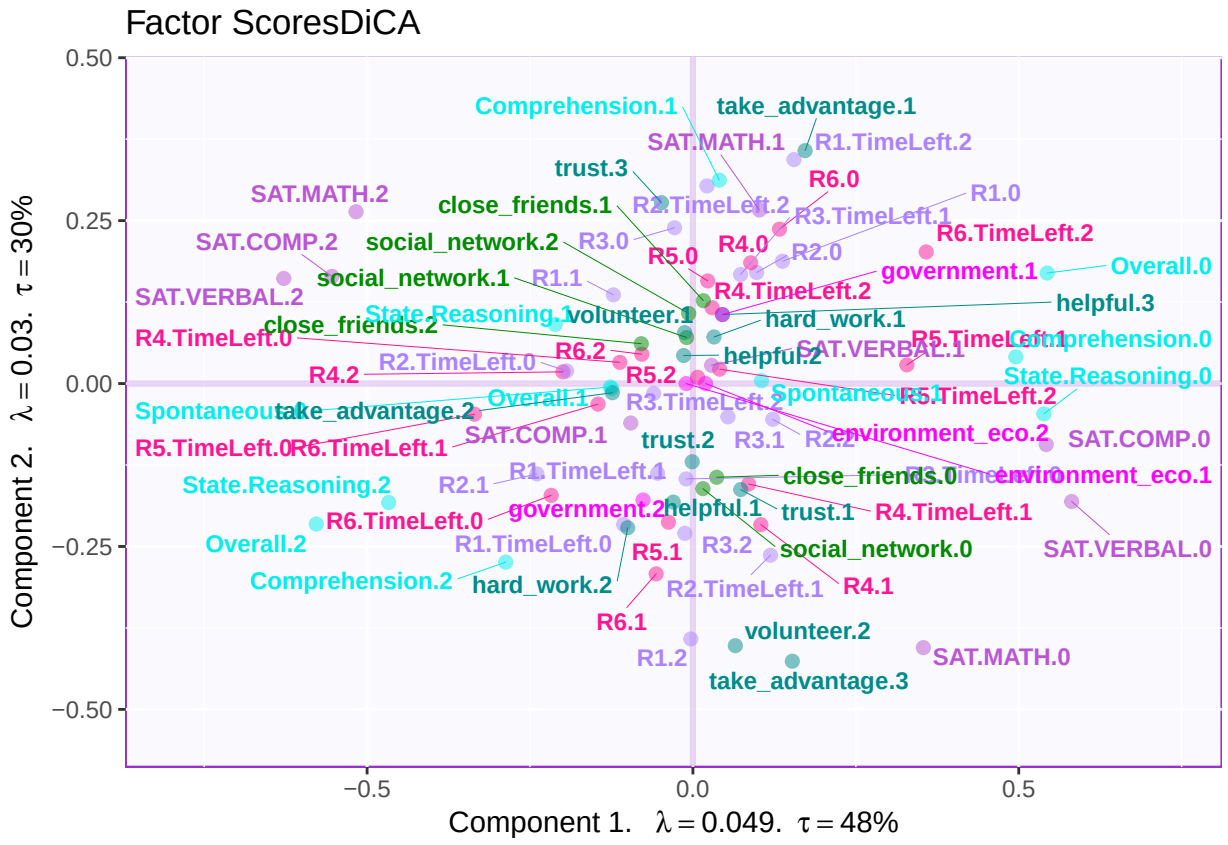
In DiCA's factor scores results, it is quite different with traditional PCA results. In PCA's factor score, generally I will have row factor scores for each observation and column factor scores for each variables. However, since I used bin coding in DiCA, I will have several levels for each variables (0-low, 1-medium, 2-high). In the row factor scores plot, I chose `hull` mode over `CI` (confidence interval) to better illustrate the overlap among groups. For the column factor scores, as it is shown below, it is quite messy if we don't select important variables. However, we can still see that the low levels and high levels are distributed in opposite directions. The line graph illustrated the linear relationship in levels for each important variable more clearly.

```
# get some color

# special version of dica
m.color.design.dica <- as.matrix(m.color.design[1:(nrow(m.color.design)-1)])

col4Levels <- data4PCCAR::coloringLevels(
  rownames(fj), m.color.design.dica)
col4Labels <- col4Levels$color4Levels
```

```
# important variables

varCtr <- data4PCCAR::ctr4Variables(cj)
rownames(m.color.design.dica) <- rownames(varCtr)
var12 <- data4PCCAR::getImportantCtr(ctr = varCtr,
                                     eig = eigs)
importantVar <- var12$importantCtr.1or2
col4ImportantVar <- m.color.design.dica
col4NS <-'gray90'
col4ImportantVar[!importantVar] <- col4NS
col4Levels.imp <- data4PCCAR::coloringLevels(rownames(fj),
                  col4ImportantVar)

# plot

plot.fs(exp.neg$group,
        fs=fii,
        eigs=eigs,
        tau=tau,
        d=1,
        mode = "hull",
        method = "DiCA")
```

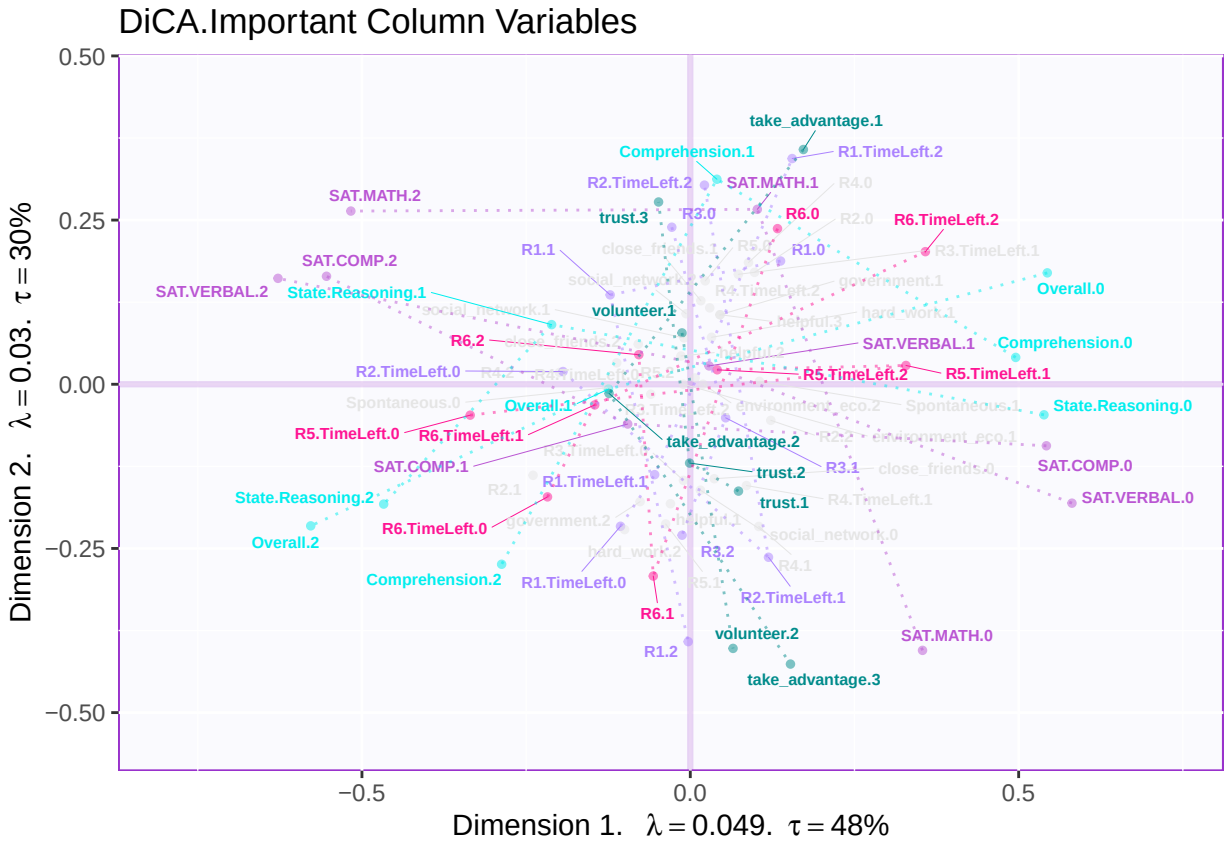


```
plot.cfs(fj=fj,
         eigs=eigs,
         tau=tau,
         d=1,
         col = col4Labels,
         method = "DiCA")
```




```
# plot important line plot
labels4MCA <- createxyLabels.gen(x_axis = 1,
                                 y_axis = 2,
                                 lambda = eigs,
                                 tau = round(tau))

fj.imp <- createFactorMap(X = fj,
                      axis1 = 1,
                      axis2 = 2,
                      title = "DiCA.Important Column Variables",
                      col.points = col4Levels.imp$color4Levels,
                      cex = 1,
                      col.labels = col4Levels.imp$color4Levels,
                      text.cex = 2,
                      force = 2)
```

```
fj.base <- fj.imp$zeMap + labels4MCA

zeNames <- getVarNames(rownames(fj))
importantsLabels <- zeNames$stripedNames %in%
  zeNames$variableNames[importantVar]
fj.imp <- fj[importantsLabels,]
lines4J.imp <- addLines4MCA(fj.imp,
                col4Var = col4Levels$color4Variables[which(importantVar)],
                size = .5,
                linetype = 3,
                alpha = .5)
fj.dica <- fj.base + lines4J.imp
fj.dica
```



6.6 Confusion Matrix

As a statistical method designed to do discriminative jobs, DiCA could also provide us confusion matrix info. From the tables below, we know that the predictive accuracy for fixed effect is good but the generalized ability is poor. The fixed confusion accuracy can reach **72%** but the random confusion matrix can only be 42%.

Table 6.1: Fixed Confustion Matrix

	LS_LG.actual	HS_LG.actual	LS_HG.actual	HS_HG.actual
LS_LG.predicted	21	3	2	1
HS_LG.predicted	4	14	0	0
LS_HG.predicted	2	1	11	2
HS_HG.predicted	4	2	3	16

```
fixed.confusion <- res.DiCA.inf$Inference.Data$loo.data$fixed.confuse
random.confusion <- res.DiCA.inf$Inference.Data$loo.data$loo.confuse
fixed.acc <- res.DiCA.inf$Inference.Data$loo.data$fixed.acc
random.acc <- res.DiCA.inf$Inference.Data$loo.data$loo.acc

# rename
rownames(fixed.confusion) <- sub("[[:punct:]]","",
                                 rownames(fixed.confusion))
rownames(random.confusion) <- sub("[[:punct:]]","",
                                  rownames(random.confusion))
colnames(fixed.confusion)<- sub("[[:punct:]]","",
                                colnames(fixed.confusion))
colnames(random.confusion)<- sub("[[:punct:]]","",
                                 colnames(random.confusion))
rownames(fixed.confusion) <- paste0(rownames(fixed.confusion),
                                    ".predicted")
colnames(fixed.confusion) <- paste0(colnames(fixed.confusion),
                                    ".actual")


# print table and accurarcy
kable(fixed.confusion,
      caption = "Fixed Confustion Matrix")
```

```
cat("fixed accurarcy: ",
    fixed.acc)
```

```
## fixed accurarcy:  0.7209302
```

```
kable(random.confusion,
      caption = "Random Confusion Matrix")
```

```
cat("random accurarcy: ",
    random.acc)
```

```
## random accurarcy:  0.4186047
```

Table 6.2: Random Confusion Matrix

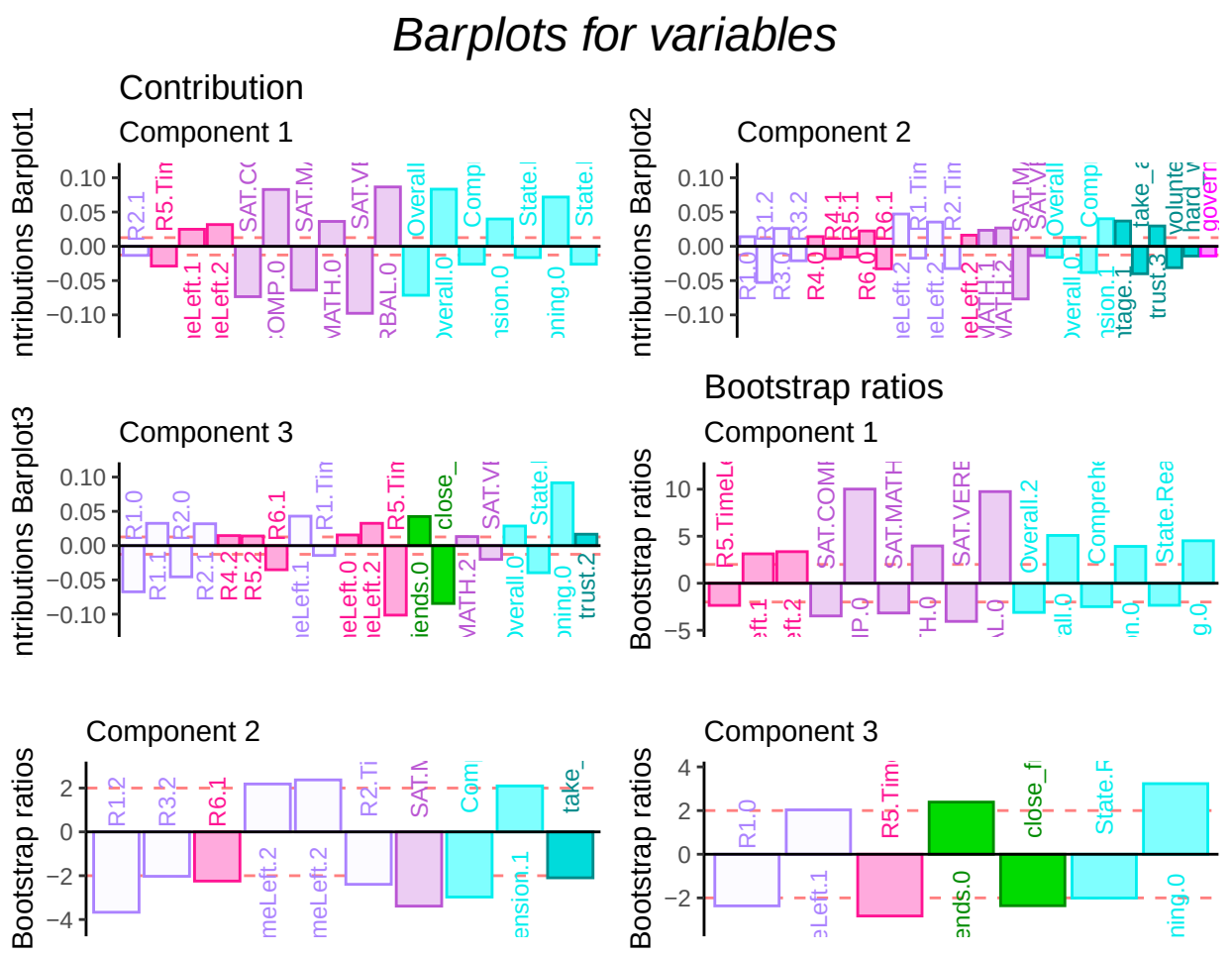
	LS_LG.actual	HS_LG.actual	LS_HG.actual	HS_HG.actual
LS_LG.predicted	16	10	5	2
HS_LG.predicted	4	2	1	3
LS_HG.predicted	7	2	7	3
HS_HG.predicted	4	6	3	11

6.7 Contribution and Bootstrap Ratio Barplots

For the contribution barplot, I focused on the first three components and the important variables instead of all the variables. From the component2, we can know that the high performance of first half game (R1.high and R3.high) is related with high Math ability, low comprehension and high take_advantage. It is reasonable because at the beginning stage, everyone is not much familiar with how the game played (although there are practice session) and everyone in the team is stranger. The situation is better through game going (R4.high and R5.high have same direction with trust and low take_advantage).

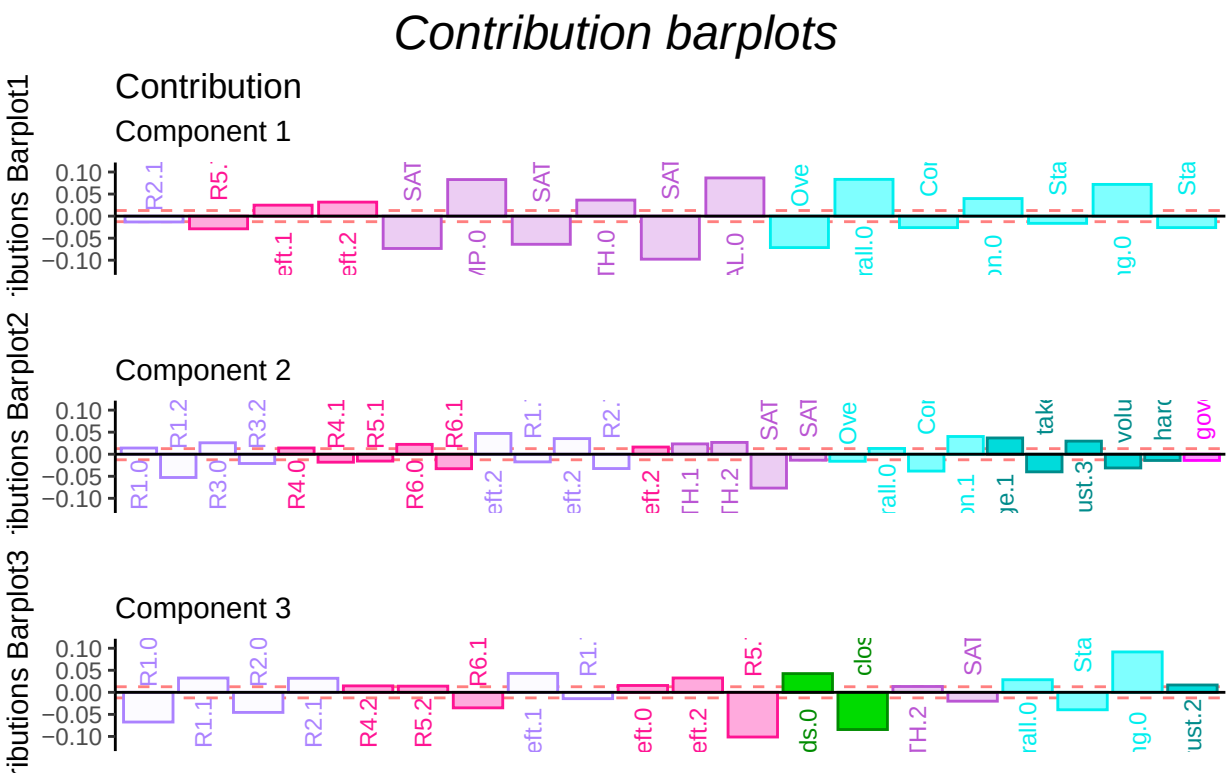
In conclusion, from the DiCA results, the relationship between each level of collective game playing and personality are revealed. However, since the data set is a quantitative data instead of qualitative data, I may expect more on BADA's result in the following chapter.

```
# Computational Convenience: using my function:
plot.cb(cj,
        fj,
        col=col4Labels,
        boot.ratios,
        signifOnly = TRUE,
        fig = 3,
        horizontal = TRUE,
        colrow = "row",
        fontsize = 3)
```



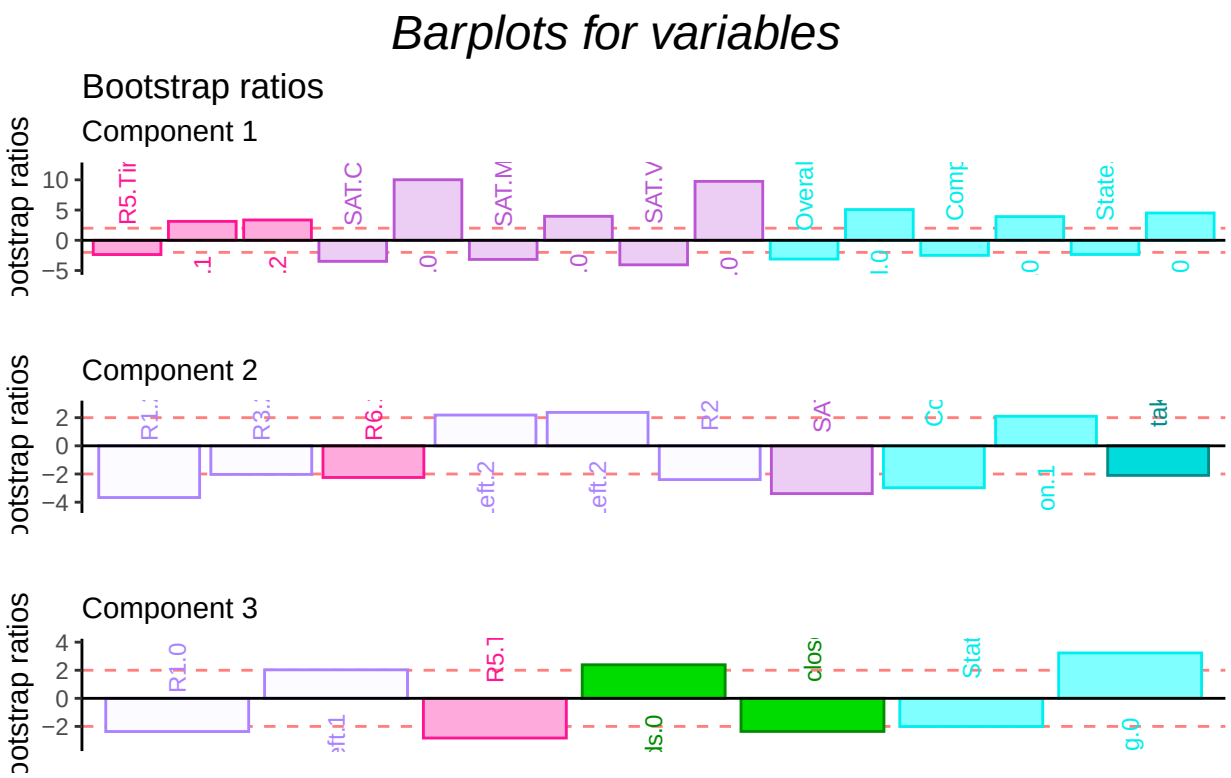

```
signed.ctrJ <- cj * sign(fj)
laDim = 1
ctrJ.1 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                  threshold = 1 / NROW(signed.ctrJ),
                  font.size = 3,
                  signifOnly = TRUE,
                  horizontal = TRUE,
                  color4bar = col4Labels,
                  main = 'Variable Contributions (Signed)',
                  ylab = paste0('Contributions Barplot',
                                laDim),
                  ylim = c(1.2*min(signed.ctrJ),
                           1.2*max(signed.ctrJ))
  ) + ggtitle("Contribution",subtitle = paste0('Component ',
                                               laDim))

### plot contributions for component 2
laDim =2
ctrJ.2 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                  threshold = 1 / NROW(signed.ctrJ),
```

```
                font.size = 3,
                color4bar = col4Labels,
                signifOnly =TRUE,
                horizontal = TRUE,
                main = 'Variable Contributions (Signed)',
                ylab = paste0('Contributions Barplot',
                              laDim),
                ylim = c(1.2*min(signed.ctrJ),
                         1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))
laDim =3
ctrJ.3 <- PrettyBarPlot2(signed.ctrJ[,laDim],
               threshold = 1 / NROW(signed.ctrJ),
               font.size = 3,
               color4bar = col4Labels, # we need hex code
               signifOnly = TRUE,
               horizontal = TRUE,
               main = 'Variable Contributions (Signed)',
               ylab = paste0('Contributions Barplot', laDim),
               ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))


gridExtra::grid.arrange(as.grob(ctrJ.1),
                as.grob(ctrJ.2),
                as.grob(ctrJ.3),
                ncol=1,
                top = textGrob("Contribution barplots",
                gp=gpar(fontsize=18,font=3)))
```




```
### Bootstrap of columns vectors
BR <- boot.ratios
laDim = 1

# Plot the bootstrap ratios for Dimension 1
ba001.BR1 <- PrettyBarPlot2(BR[,laDim],
                    threshold = 2,
                    font.size = 3,
                    signifOnly = TRUE,
                    horizontal = TRUE,
                    color4bar = col4Labels, # we need hex code
                    ylab = 'Bootstrap ratios',
                    ylim = c(1.2*min(BR[,laDim]),
                             1.2*max(BR[,laDim]))
) + ggtitle("Bootstrap ratios", subtitle = paste0('Component ',
                                                  laDim))

# Plot the bootstrap ratios for Dimension 2
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR[,laDim],
                  threshold = 2,
                  font.size = 3,
                  signifOnly = TRUE,
                  horizontal = TRUE,
                  color4bar =  col4Labels, # we need hex code
```

```
                ylab = 'Bootstrap ratios',
                ylim = c(1.2*min(BR[,laDim]),
                         1.2*max(BR[,laDim]))
) + ggtitle("",subtitle = paste0('Component ', laDim))

laDim = 3
ba003.BR3 <- PrettyBarPlot2(BR[,laDim],
                threshold = 2,
                font.size = 3,
                signifOnly = TRUE,
                horizontal = TRUE,
                color4bar =  col4Labels, # we need hex code
                ylab = 'Bootstrap ratios',
                ylim = c(1.2*min(BR[,laDim]),
                         1.2*max(BR[,laDim]))
) + ggtitle("",subtitle = paste0('Component ', laDim))


contr <- grid.arrange(
  as.grob(ba001.BR1),
  as.grob(ba002.BR2),
  as.grob(ba003.BR3),
  ncol = 1,
  top = textGrob("Barplots for variables",
                 gp = gpar(fontsize = 18,
                           font = 3))
)
```


Barplots for variables
Bootstrap ratios
Component 1
Component 2
Component 3

Chapter 7

Barycentric Discriminant Analysis

7.1 Introduction of BADA

The full name of BADA is Barycentric Discriminant Analysis. Basically, it is a discriminative analysis technique similar with DiCA. The main purpose of BADA is to maximize the discrepancy between groups and make best prediction based on the distance on factor scores. The unique advantage of BADA is that BADA performs well when the number of observation is smaller but the variables are big - the big data situation. However, in this chapter, we will only use the traditional side of the BADA - make prediction on high/low social intelligence and high/low general intelligence groups in our main data set. For more information about the data and experiment, please check data intro part[3.1]

7.2 Computation

The computational process is similar with all the analysis before. As usual, I will store all important matrices into work environment for plotting.

```
res.BADA <- tepBADA(DATA = exp.neg[7:35],
                    scale = 'SS1',
                    center = TRUE,
                    DESIGN = exp.neg$group,
                    graphs = FALSE)
res.BADA.inf <- tepBADA.inference.battery(DATA = exp.neg[7:35],
                   scale = 'SS1',
                   center = TRUE,
                   DESIGN = exp.neg$group,
                   graphs = FALSE)
```

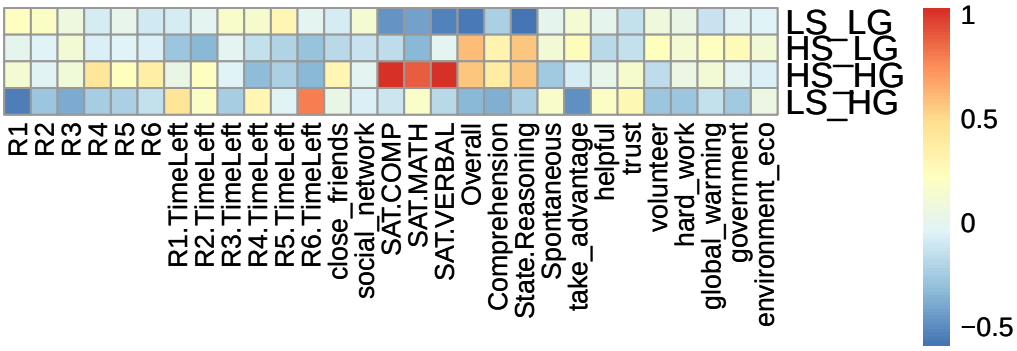
```
## [1] "It is estimated that your iterations will take 0.05 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take not
## ================================================================================
```

```
eigs <- res.BADA$TExPosition.Data$eigs
tau <- res.BADA$TExPosition.Data$t
fm <- res.BADA$TExPosition.Data$fi
fj <- res.BADA$TExPosition.Data$fj
cj <- res.BADA$TExPosition.Data$cj
eigs.permu <- res.BADA.inf$Inference.Data$components$eigs.perm
p.vals <- res.BADA.inf$Inference.Data$components$p.vals
fii <- res.BADA$TExPosition.Data$fii
boot.ratios <- res.BADA.inf$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
```

7.3 Heatmap

The heatmap is same with the DiCA one. I calculated the group mean for each variables and plot it with 'pheatmap'.

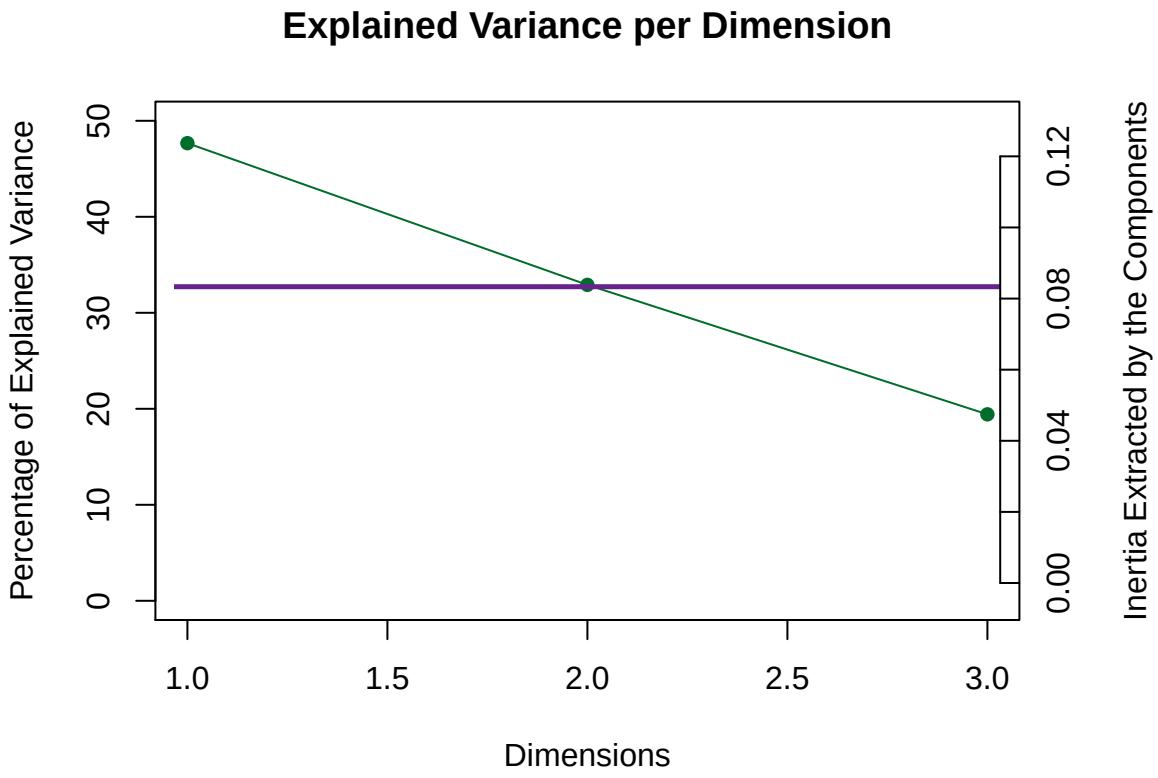
```
# getmeans
raw.bada <- scale(exp.neg[7:35],
                  scale = TRUE,
                  center = TRUE)
tmp <- aggregate(raw.bada, list(exp.neg$group), mean)
mean.bada.raw <- tmp[,2:ncol(tmp)]
rownames(mean.bada.raw) <- tmp[,1]

bada.hm <- pheatmap(mat=mean.bada.raw,
                    fontsize_row =13,
                    fontsize =10,
                    cellheight = 10,
                    cellwidth = 10,
                    angle_col = 90,
                    cluster_rows = FALSE,
                    cluster_cols = FALSE)
bada.hm
```



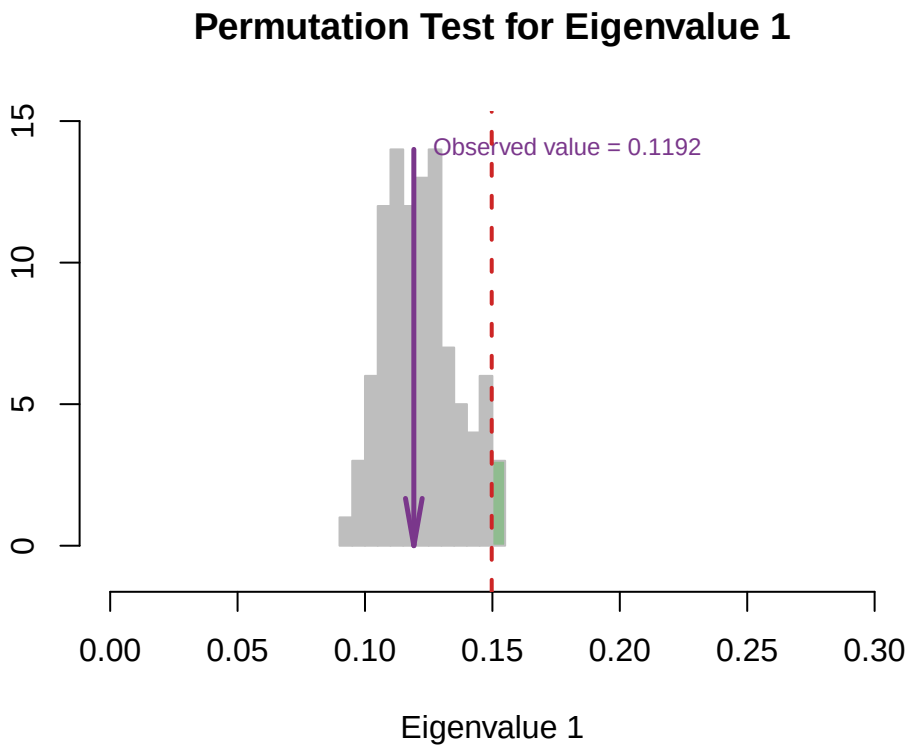
7.4 Scree

The scree plot indicated that the first component explained nearly half of the variance and the second one explained nearly 30%. The permutation test results are insignificant, which means that the experimental observation may not as stable as we expect.

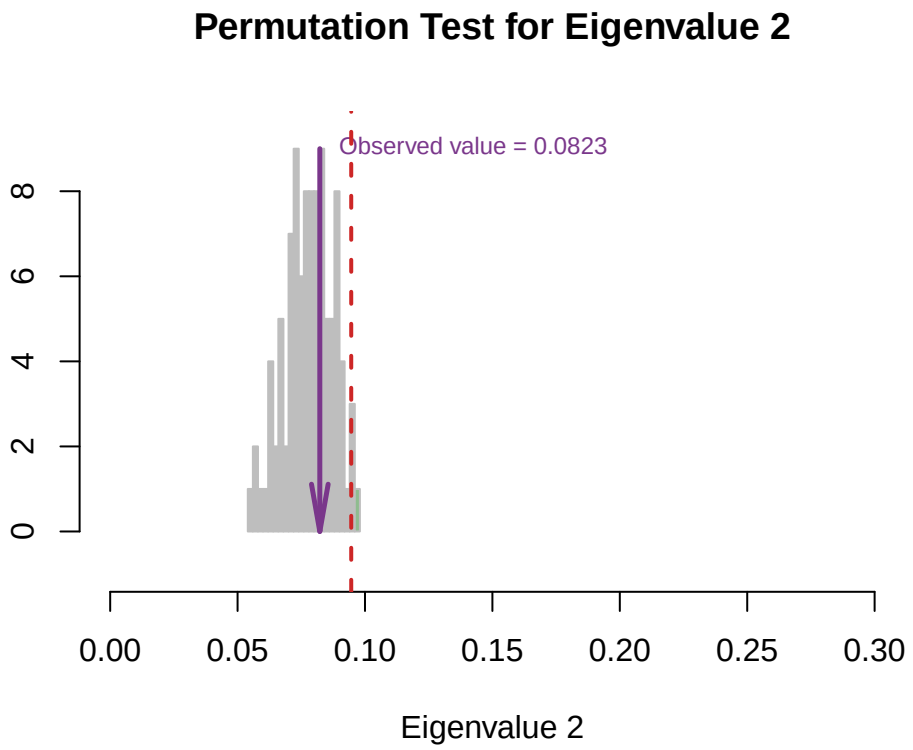
```
plot.scree(eigs, p.vals = p.vals)
```




```
plot.permutation(eigs.permu,
                 eigs,
                 para1 =0.3,
                 Dim = 1)
```



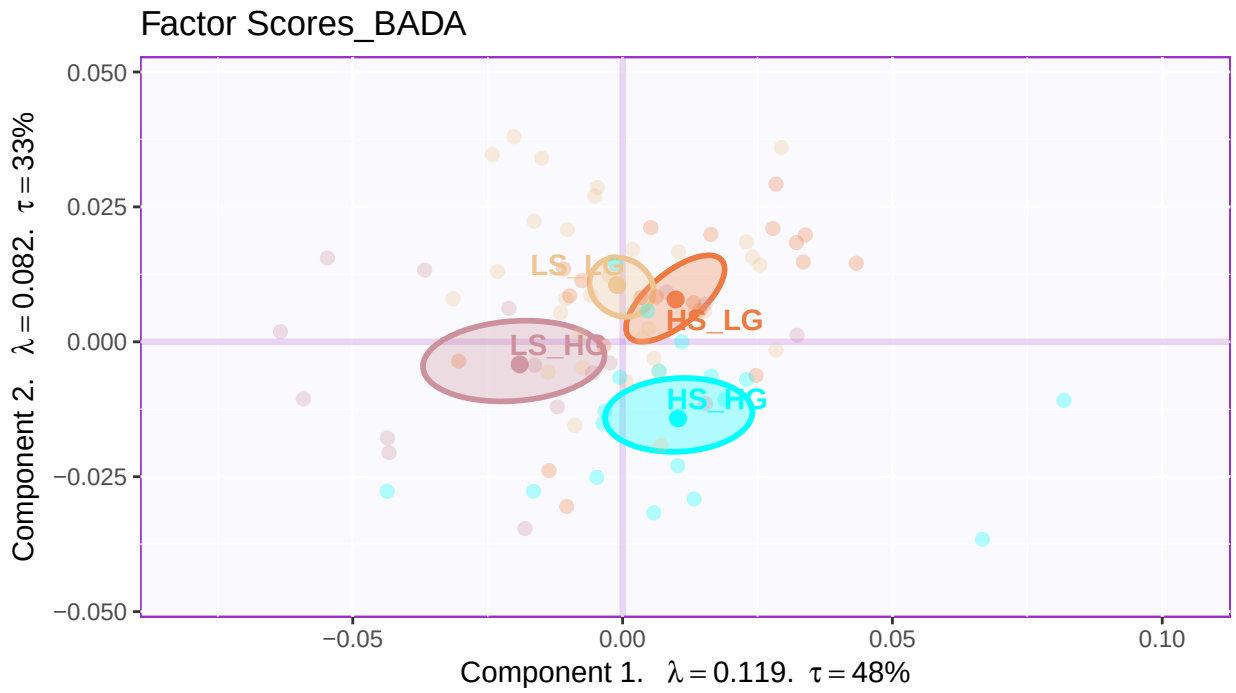

```
plot.permutation(eigs.permu,
                 eigs,
                 para1=0.3,
                 Dim = 2)
```



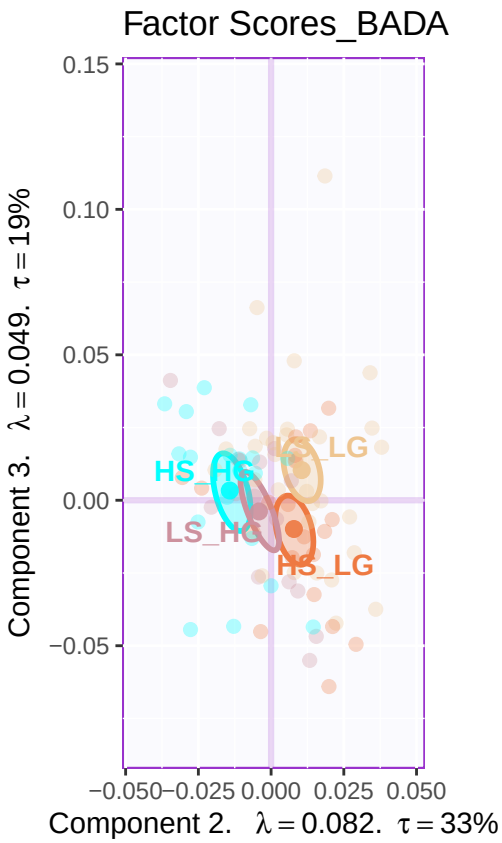
7.5 Factor Scores

Similar with PCA, the row factor scores gave back to me a clear separation among groups in first and second dimensions. However, the results from dimension 2 and 3 may not be very informative. As I can know that the first dimension is the most important contributor for group separation.

```
plot.fs(DESIGN = exp.neg$group,
        fs=fii,
        eigs=eigs,
        tau=tau,
        d = 1,
        mode="CI",
        method = "BADA")
```



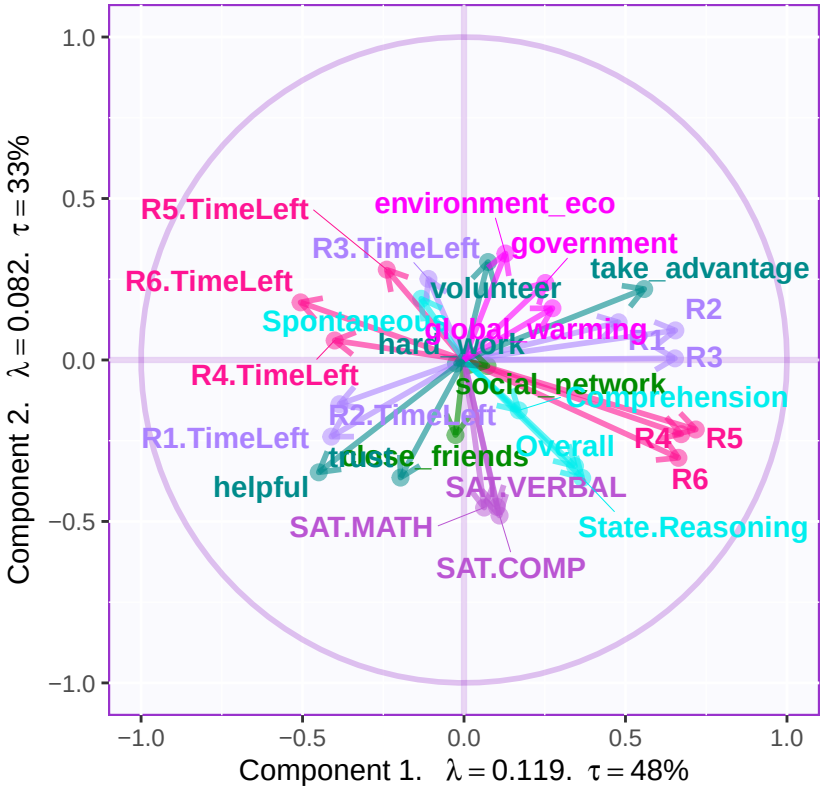

```
plot.fs(DESIGN = exp.neg$group,
        fs=fii,
        eigs=eigs,
        tau=tau,
        d = 2,
        mode="CI",
        method = "BADA")
```



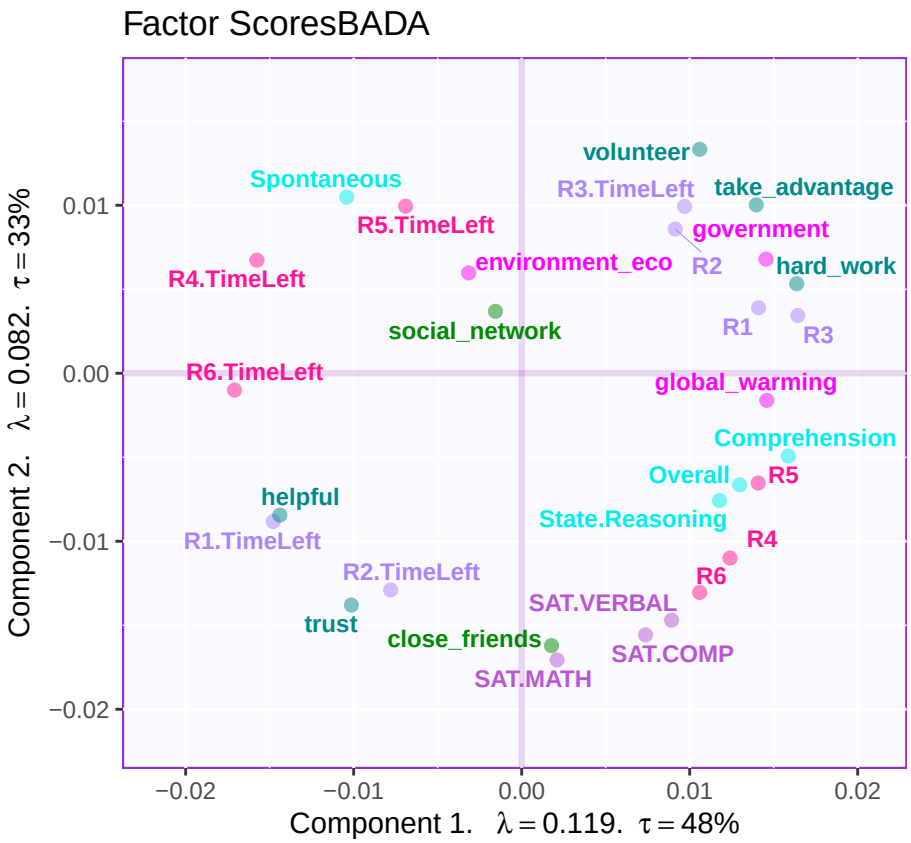
7.6 Loading

The loading part is interesting. From the column factor scores, we know that the first and second half of the collective game are separated by dimension2, which is fantastic. The dimension1 separates `volunteer`, `take_advantage` and `hard_working` with `helpful` and `trust`. The dominant variables in dimension 2 is SAT results. Not only the collective game performance, the time usage of first and second half of the game is separated by dimension2.

```
plot.loading(data=raw.bada,
             col=m.color.design,
             fs=fii,
             eigs=eigs,
             tau=tau,
             d=1)
```




```
plot.cfs(fj=fj,
         eigs=eigs,
         tau=tau,
         col=m.color.design,
         method = "BADA")
```



7.7 Confusion Matrix

The confusion matrix of BADA provide me the accuracy of its prediction on each group. Sadly, the results are not as good as DiCA, which is unexpected. The fixed confusion accuracy is **53%** and he random confusion accuracy is **45%**

```
# computation
fixed.confusion <- as.matrix(res.BADA.inf$Inference.Data$loo.data$fixed.confuse)
random.confusion <- as.matrix(res.BADA.inf$Inference.Data$loo.data$loo.confuse)
fixed.acc <- res.BADA.inf$Inference.Data$loo.data$fixed.acc
random.acc <- res.BADA.inf$Inference.Data$loo.data$loo.acc

# rename
rownames(fixed.confusion) <- sub("[[:punct:]]","",rownames(fixed.confusion))
rownames(random.confusion) <- sub("[[:punct:]]","",rownames(random.confusion))
colnames(fixed.confusion)<- sub("[[:punct:]]","",colnames(fixed.confusion))
colnames(random.confusion)<- sub("[[:punct:]]","",colnames(random.confusion))
rownames(fixed.confusion) <- paste0(rownames(fixed.confusion), ".predicted")
colnames(fixed.confusion) <- paste0(colnames(fixed.confusion), ".actual")

# print table and accurarcy
kable(fixed.confusion, caption = "Fixed Confustion Matrix")
```

Table 7.1: Fixed Confustion Matrix

	LS_LG.actual	HS_LG.actual	LS_HG.actual	HS_HG.actual
LS_LG.predicted	22	6	6	6
HS_LG.predicted	8	11	3	4
LS_HG.predicted	0	1	6	2
HS_HG.predicted	1	2	1	7

Table 7.2: Random Confusion Matrix

	LS_LG.actual	HS_LG.actual	LS_HG.actual	HS_HG.actual
LS_LG.predicted	21	8	8	8
HS_LG.predicted	8	9	3	5
LS_HG.predicted	0	1	5	2
HS_HG.predicted	2	2	0	4

```
cat("fixed accurarcy: ", fixed.acc)
```

```
## fixed accurarcy:  0.5348837
```

```
kable(random.confusion, caption = "Random Confusion Matrix")
```

```
cat("random accurarcy: ", random.acc)
```

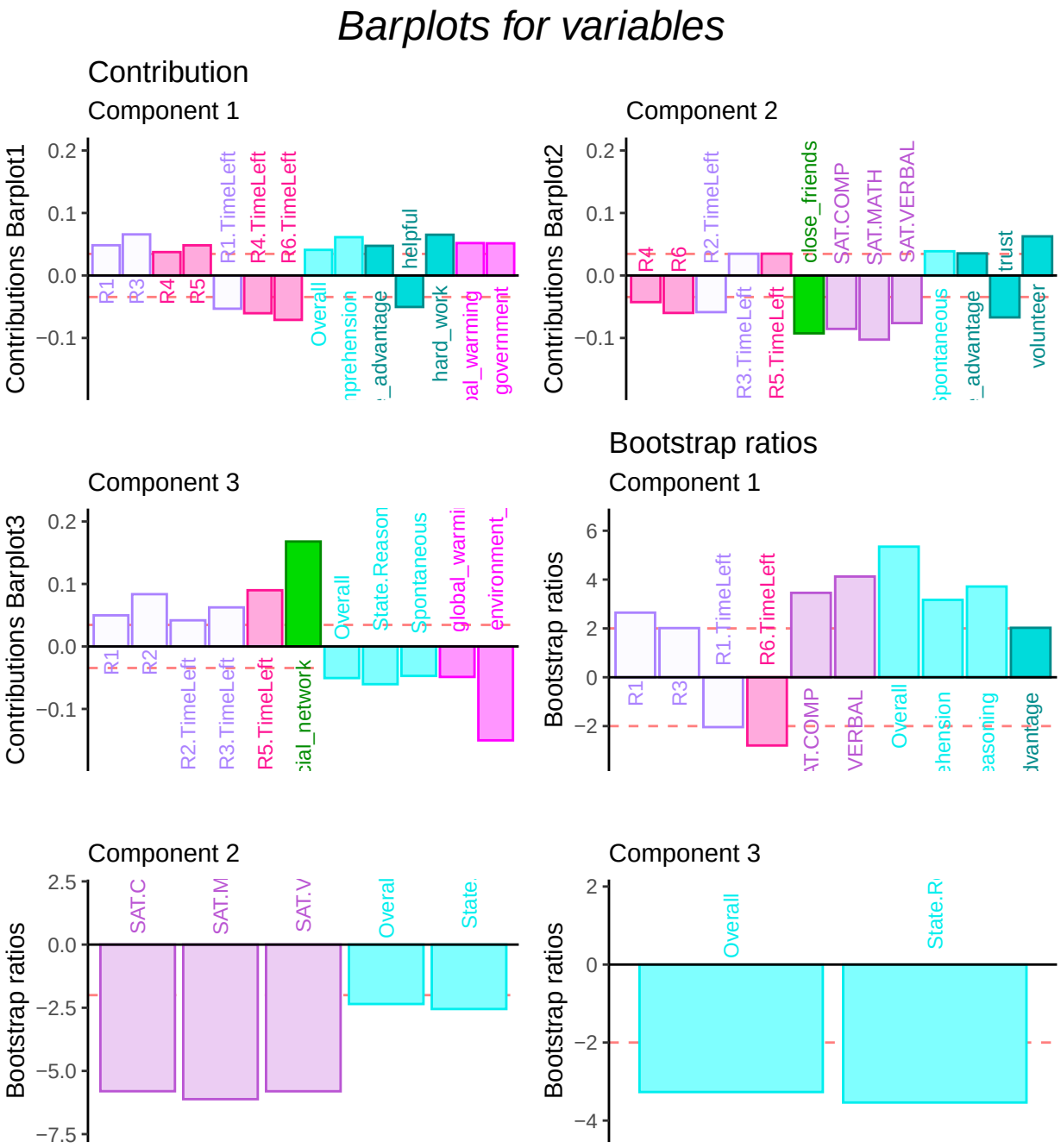
```
## random accurarcy:  0.4534884
```

7.8 Contribution and Bootstrap Ratio Barplot

The results of contribution barplots indicated that the collective game performance is negatively associated with the time usage of the game. The less usage of the time they used, the better teamwork they had during the experiment. The previous and current results all point out that the time usage is in same contribution direction with `helpful` and `trust`. From the dimension 1, we also found out that the attitude toward global environment is positively correlated with collective game performance. From the dimension 2, I can conclude that the second half of the game may more rely on team work since the `close_friend` and `trust` variables explained the most variance.

From BADA's results, it looks like that I can say, the first half of the game (for the negative condition, they will have ample resource at the first half of the game but limited resource at the second half of the game), participants are more willing to working independently. However, in the second half of the game, they are more likely to have collective behaviors and teamwork. The results is consistent with research article.

```
plot.cb(cj=cj,
        fj=fj,
        col = m.color.design,
        boot.ratios=boot.ratios,
        signifOnly = TRUE,
        fig = 3,
        horizontal = TRUE,
        colrow = "row",
        fontsize = 3)
```



Chapter 8

Multiple Corresponding Analysis

8.1 Introduction of MCA

MCA is an updated version of CA. When I have multiple nominal variable in my data set, it is the perfect time to use MCA. When I am using the MCA on my main data set, the most important thing is to convert my quantitative data table into the nominal ones. I used a handmade function called `bins_helper` to help me finish this job before hand.

8.2 Histgram of Binning Variables

Annot: The total earning is supplementary variable, so I didn't bin it.

8.3 Computation

Same as usual

```
# computation
res.MCA <- epMCA(DATA = bins.exp.neg[7:35],
                 DESIGN = exp.neg$group,
                 graphs = FALSE)
res.MCA.inf <- epMCA.inference.battery(DATA = bins.exp.neg[7:35],
                                       DESIGN = exp.neg$group,
                                       graphs = FALSE)
```

```
## [1] "It is estimated that your iterations will take 0.03 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take not
## ================================================================================
```

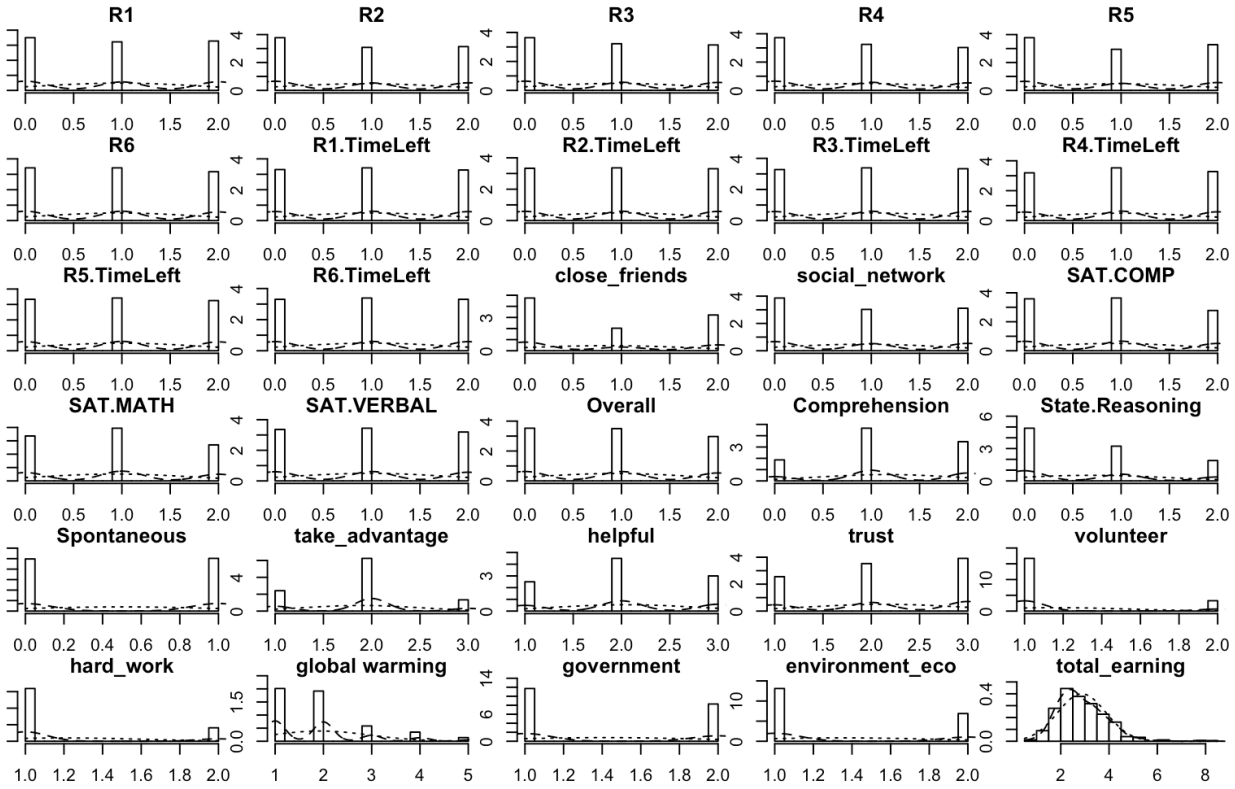


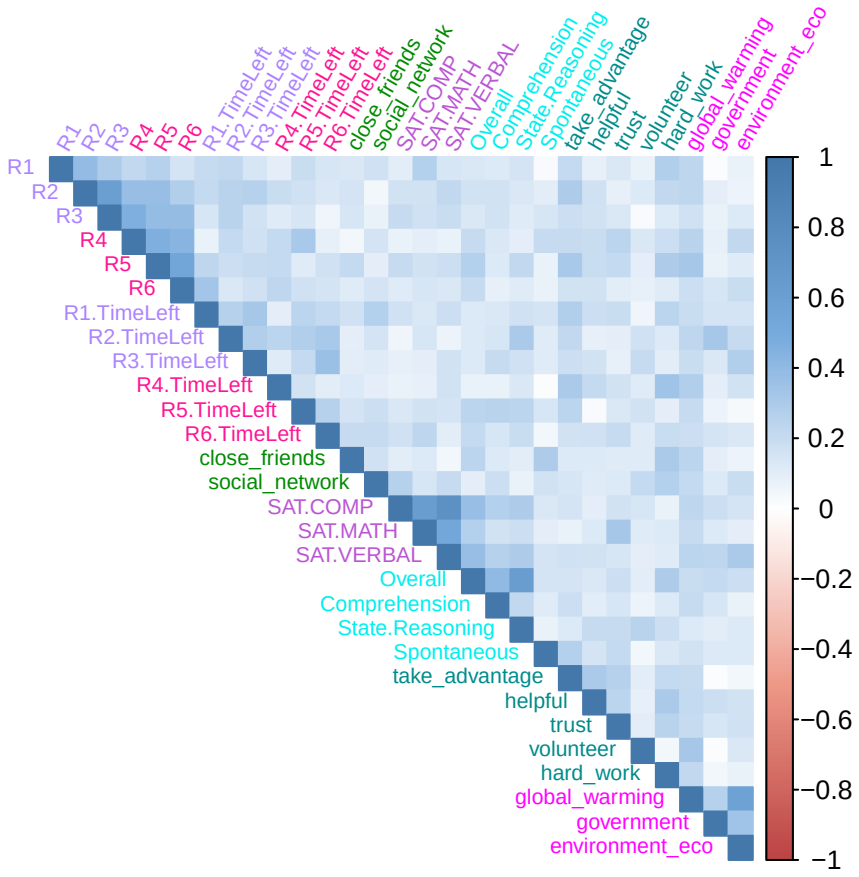
Figure 8.1: Multiple Histgram of Binning Data

```
# contribution J
cj <- res.MCA$ExPosition.Data$cj
fj <- res.MCA$ExPosition.Data$fj
fs <- res.MCA$ExPosition.Data$fi

boot.ratios <- res.MCA.inf$Inference.Data$fj.boots$tests$boot.ratios
corrMatBurt.list <- phi2Mat4BurtTable(bins.exp.neg[7:35])
eigs <- res.MCA$ExPosition.Data$eigs
tau <- res.MCA$ExPosition.Data$t
p.eigs <- res.MCA.inf$Inference.Data$components$p.vals
eigs.permu <- res.MCA.inf$Inference.Data$components$eigs.perm

# get some color
col4Levels <- data4PCCAR::coloringLevels(
  rownames(fj), m.color.design)
col4Labels <- col4Levels$color4Levels
```

8.4 Heatmap

The correlation heatmap can tell me how close these variables to each other. The clustering trends are observed in collective game, social&general intelligence and attitudes.

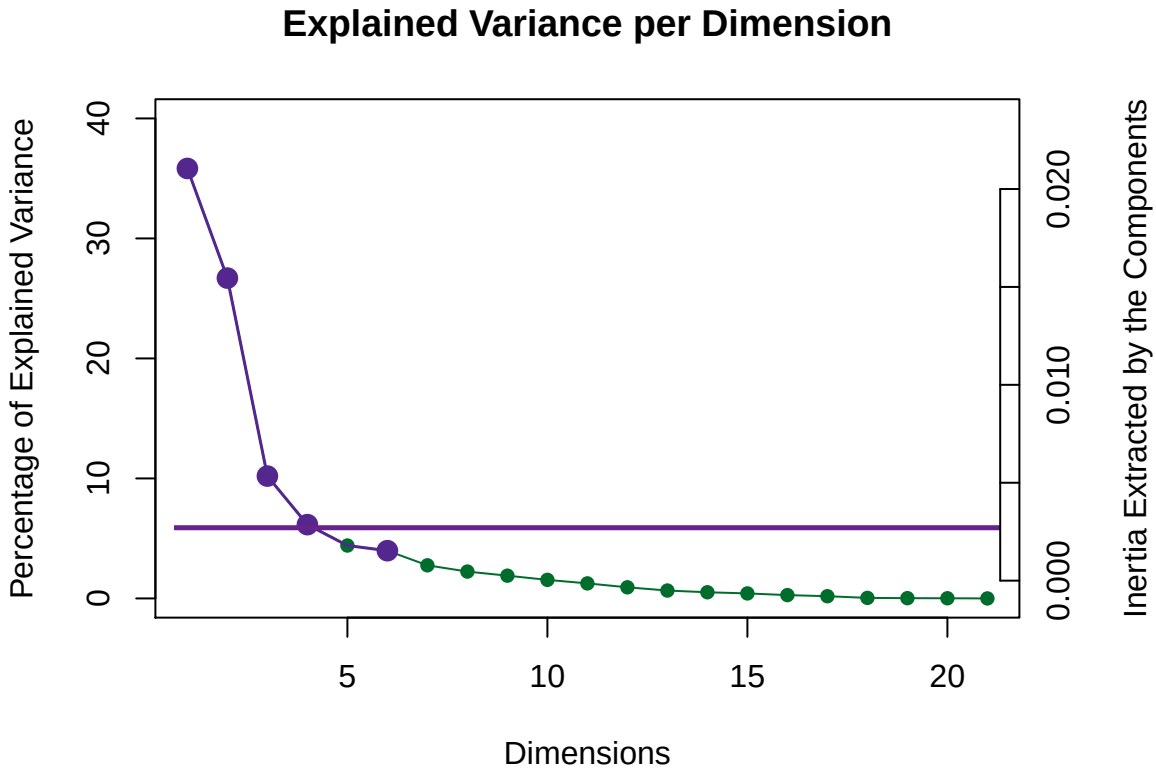
```
# plot color
col <-colorRampPalette(c("#BB4444",
                         "#EE9988",
                         "#FFFFFF",
                         "#77AADD",
                         "#4477AA"))

# plot
corr4MCA.r <- corrplot::corrplot(
  as.matrix(corrMatBurt.list$phi2.mat^(1/2)),
  method="color",
  col=col(200),
  type="upper",
  #addCoef.col = "black",
  tl.col = m.color.design,
  tl.cex = .7,
  tl.srt = 60,#Text label color and rotation
  #number.cex = .5,
  #order = 'FPC',
  diag = TRUE # needed to have the color of variables correct
  )
```



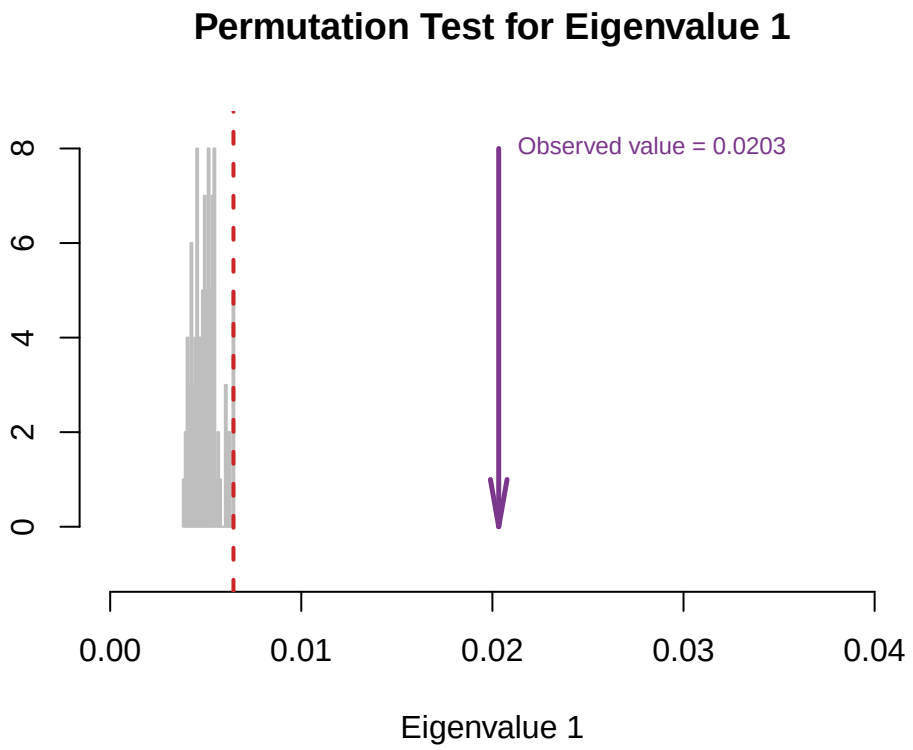
8.5 Scree Plot

From the scree plot and permutation test results, we can know that there are several significant components and I can confidently reject null hypothesis.

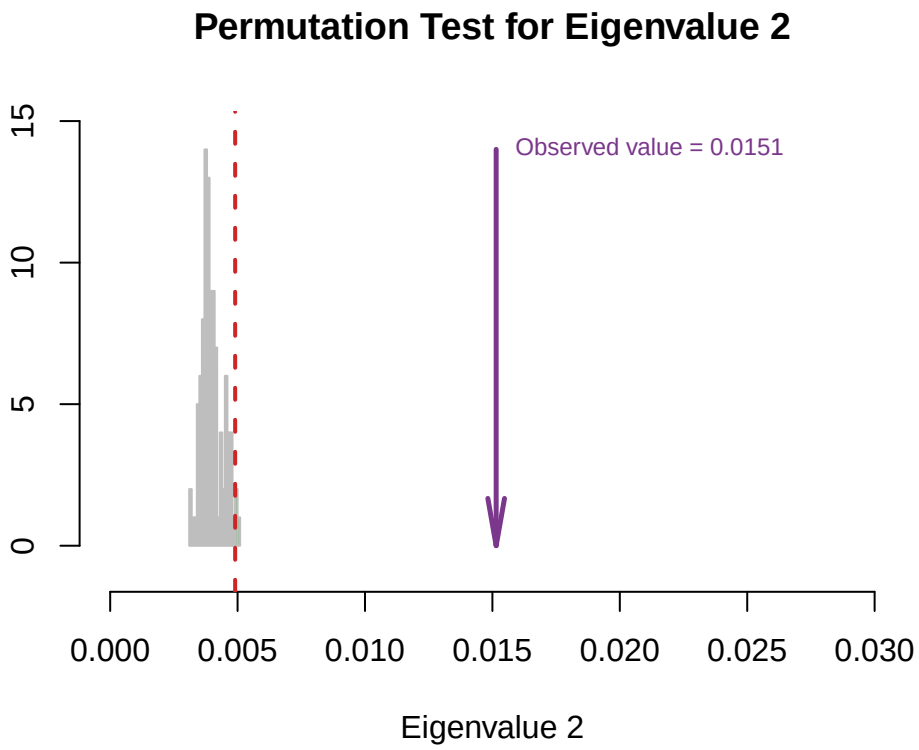
```
plot.scree(eigs, p.eigs)
```




```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs,
                 Dim =1,
                 para1 = 0.04)
```



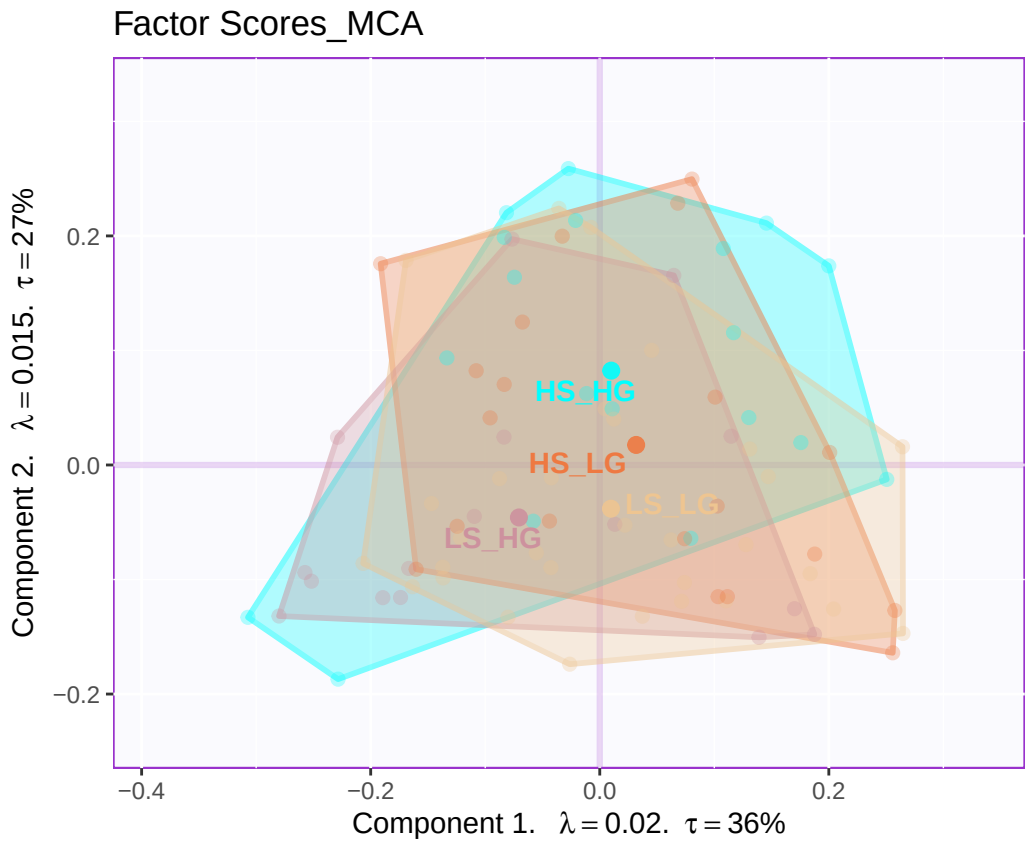

```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs,
                 Dim =2,
                 para1 = 0.03)
```



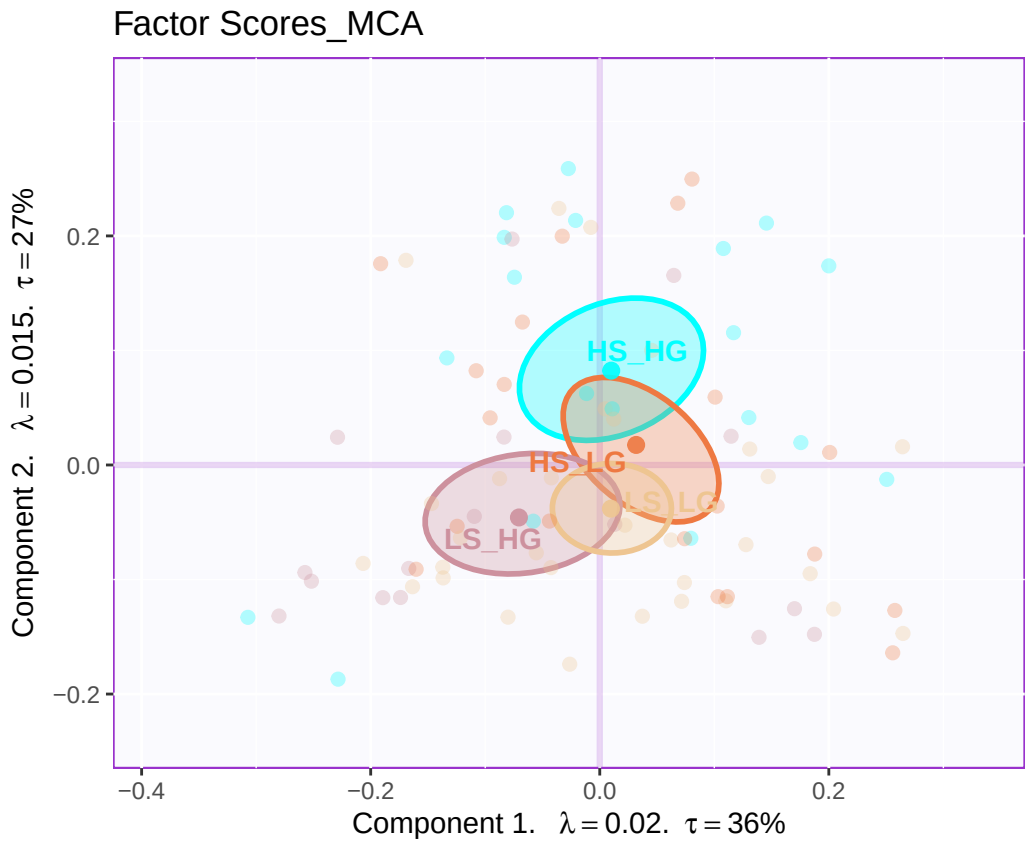
8.6 Row Factor Scores

From the row factor scores, we can know that there are huge overlap existing among groups in hull. Situation is better when we calcualted the CI. However, it is still not as good as PCA or BADA's results.

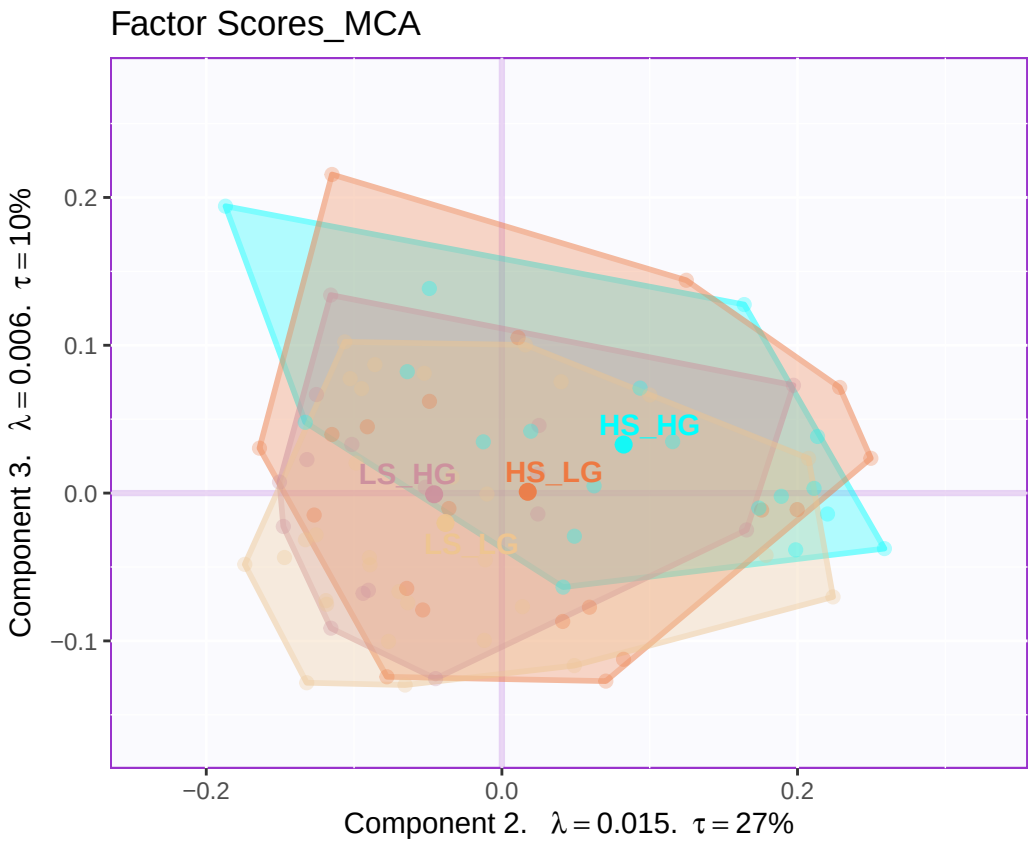
```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=1,
        mode="hull",
        method = "MCA")
```



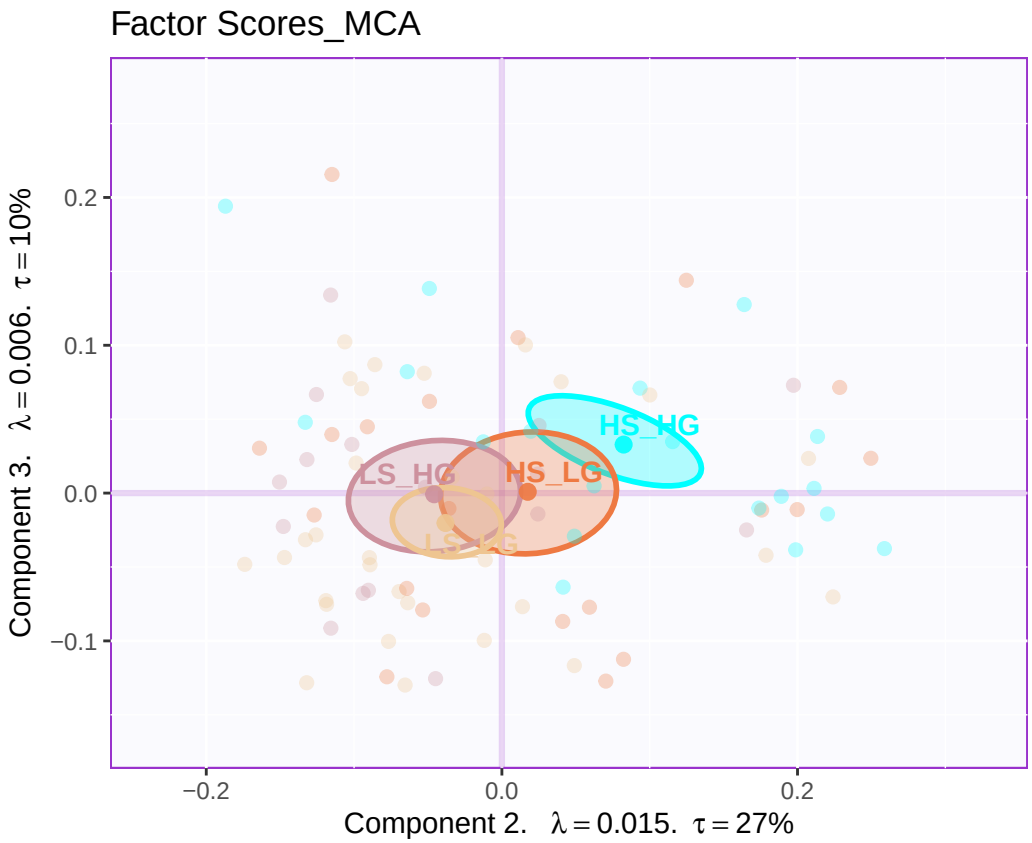

```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=1,
        mode="CI",
        method = "MCA")
```



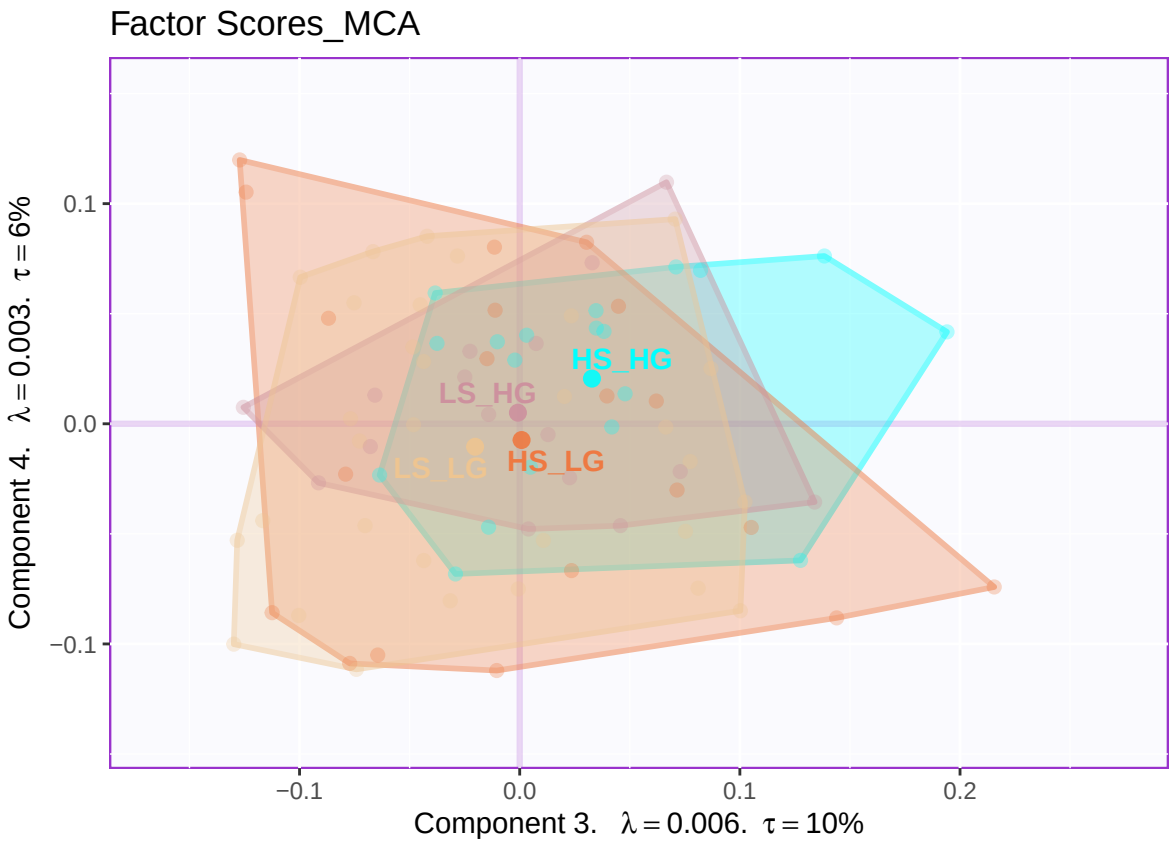

```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=2,
        mode="hull",
        method = "MCA")
```



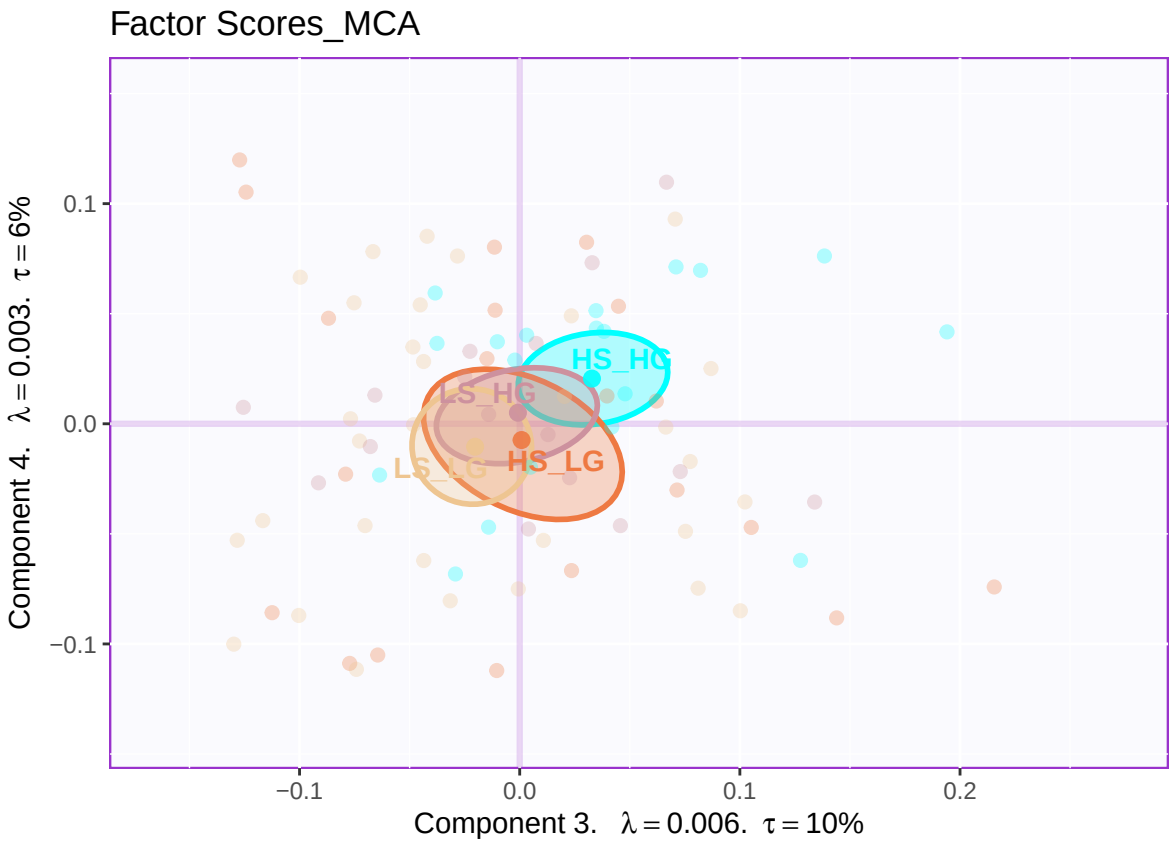

```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=2,
        mode="CI",
        method = "MCA")
```




```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=3,
        mode="hull",
        method = "MCA")
```

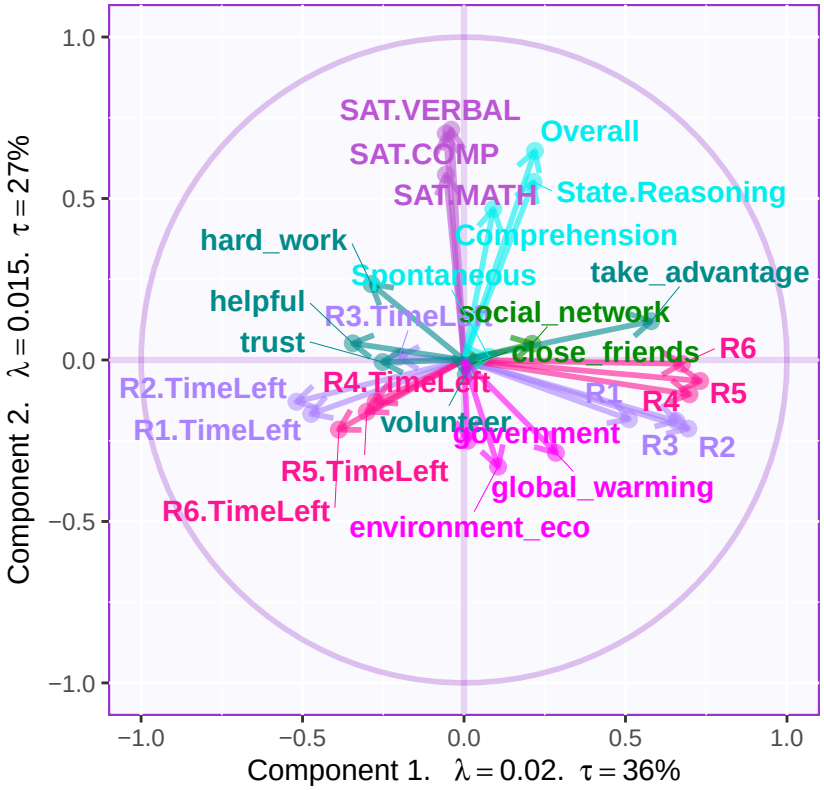



```
plot.fs(exp.neg$group,
        fs,
        eigs,
        tau,
        d=3,
        mode="CI",
        method = "MCA")
```

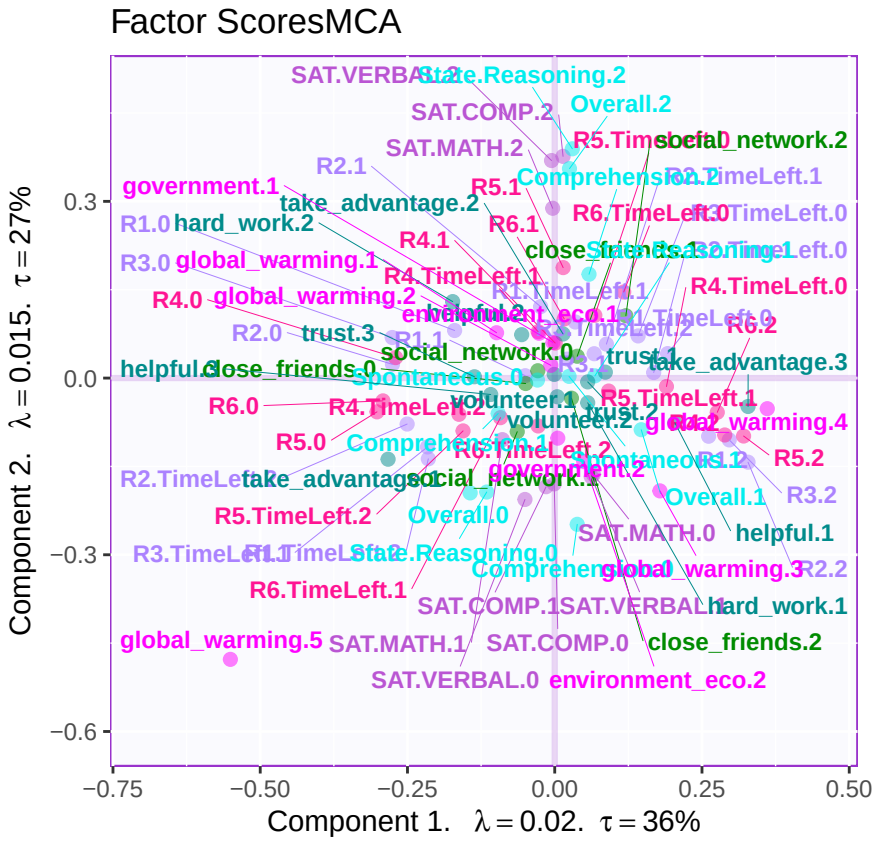


8.7 Important Variables Line Plots

```
# loading

plot.loading(exp.neg[7:35],
             col=m.color.design,
             fs, eigs, tau, d=1)
```

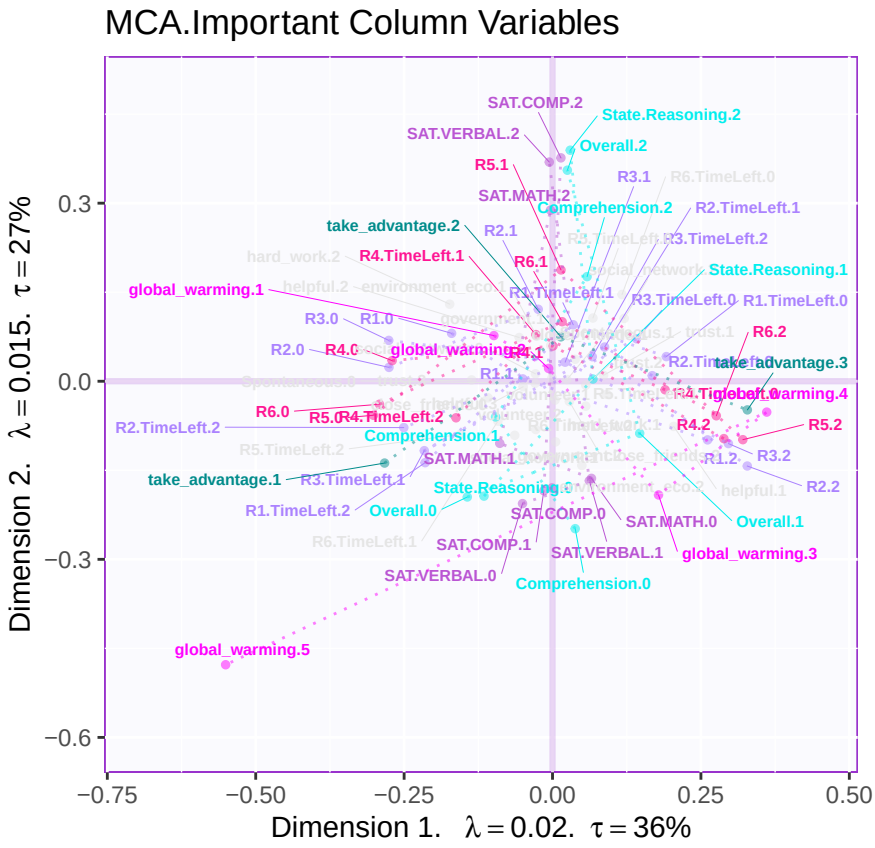



```
# basic column factor scores
plot.cfs(fj, eigs, tau, d=1, col=col4Labels,
         method = "MCA", colrow = "row")
```




```
# important variables

varCtr <- data4PCCAR::ctr4Variables(cj)
rownames(m.color.design) <- rownames(varCtr)
var12 <- data4PCCAR::getImportantCtr(ctr = varCtr,
                                     eig = eigs)
importantVar <- var12$importantCtr.1or2
col4ImportantVar <- m.color.design
col4NS <-'gray90'
col4ImportantVar[!importantVar] <- col4NS
col4Levels.imp <- data4PCCAR::coloringLevels(rownames(fj),
                                             col4ImportantVar)

# plot important line plot
labels4MCA <- createxyLabels.gen(x_axis = 1, y_axis = 2,
                                 lambda = eigs,
                                 tau = round(tau))

fj.imp <- createFactorMap(X = fj,
                          axis1 = 1,
                          axis2 = 2,
                          title = "MCA.Important Column Variables",
```

```
                          col.points = col4Levels.imp$color4Levels,
                          cex = 1,
                          col.labels = col4Levels.imp$color4Levels,
                          text.cex = 2,
                          force = 2)
fj.base <- fj.imp$zeMap + labels4MCA

zeNames <- getVarNames(rownames(fj))
importantsLabels <- zeNames$stripedNames %in%
  zeNames$variableNames[importantVar]
fj.imp <- fj[importantsLabels,]
lines4J.imp <- addLines4MCA(fj.imp,
                            col4Var = col4Levels$color4Variables[which(importantVar)],
                            size = .5,
                            linetype = 3,
                            alpha = .5)
fj.mca <- fj.base + lines4J.imp
fj.mca
```



8.8 Contribution and Bootstrap Ratio Barplots

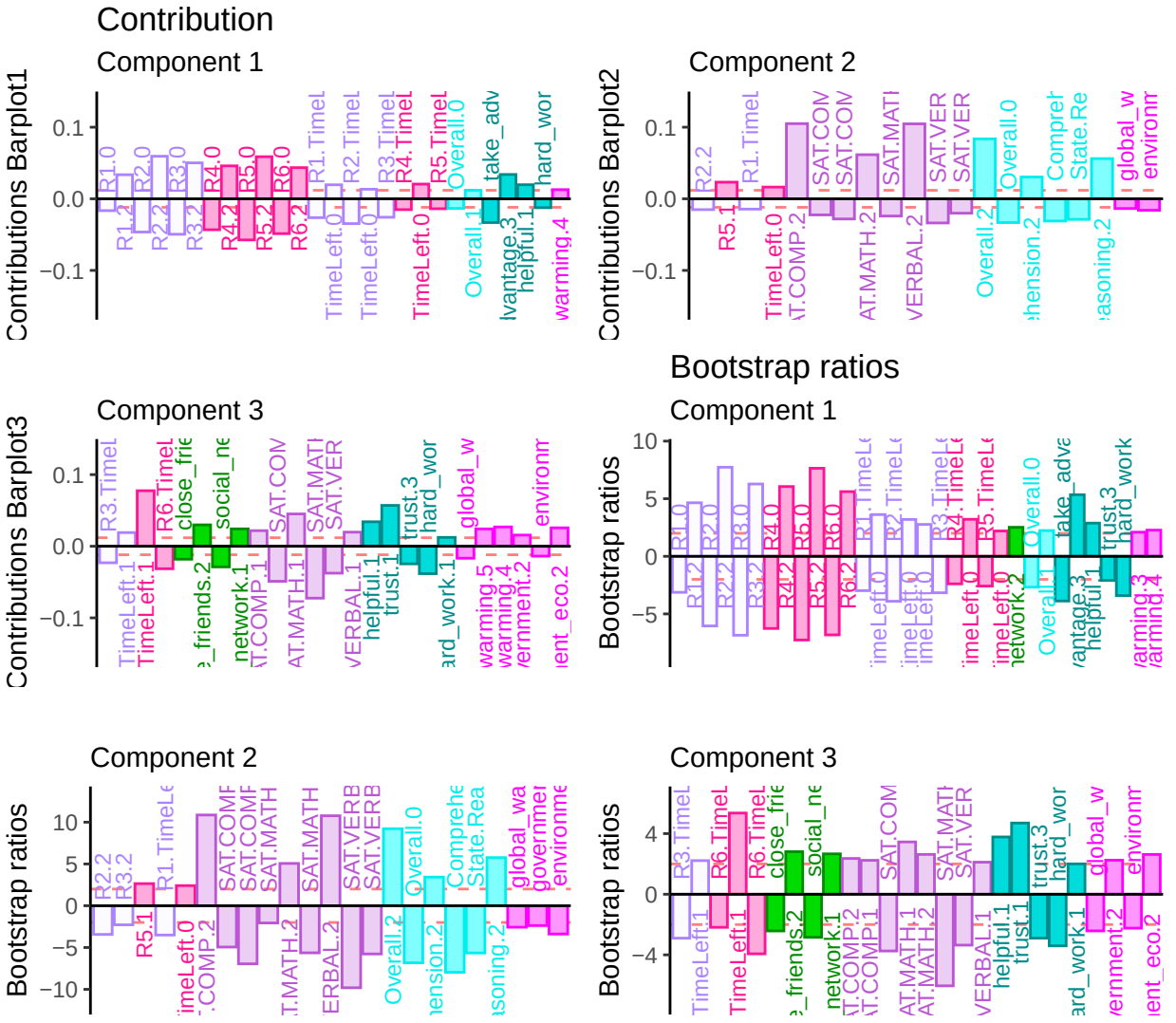
The contribution barplots are consistent with our previous results that the dimension 1 is dedicated to collective game bahavors.

In the dimension 1, all high level of R1-R3 are corresponding to high level of R4-R6, vice versa. The low level helpful and high-level of taking advantage is related with high level of the game playing. The game playing results are also associated with high level of global attitude.

Dimension 2 is all about the SAT and empathy.

Dimension 3 is showing that high level of helpful and trust are in the same direction with high level of attitude.

```
plot.cb(cj=cj,
        fj=fj,
        boot.ratios = boot.ratios,
        fig=3,
        horizontal = TRUE,
        signifOnly = TRUE,
        colrow = "row",
        col = col4Labels,
        fontsize = 3)
```


Barplots for variables
Contribution
Component 1
Component 2
Component 3
Contributions Barplot1
Contributions Barplot2
Contributions Barplot3
Bootstrap ratios
Component 1
Component 2
Component 3
Bootstrap ratios
Bootstrap ratios
Bootstrap ratios

Chapter 9

Partial Least Square

9.1 Introduction of PLS-C

The full name of the PLS-C is **Partial Least Square Correlation**, which is a technique to process two different kinds of data tables. Personally I have special interest in this method because it could help me find out the bridge between brain response and behaviors. In one sentence, PLS-C is helping me to extract commonalities between two data sets. During the analysis, PLS will generate latent variables and salience for data sets respectively. The latent variable is as same as the factor scores and the salience is the loading in previous MSA method.

PLS has several different modalities. When its goal is to predict one data table the other one, it is called partial least square regression; PLS also has a version for qualitative data, which is called PLS-CA. Today in our example, we will focus on PLS-C, to investigate the commonality between two cognitive data table.

9.2 Computation

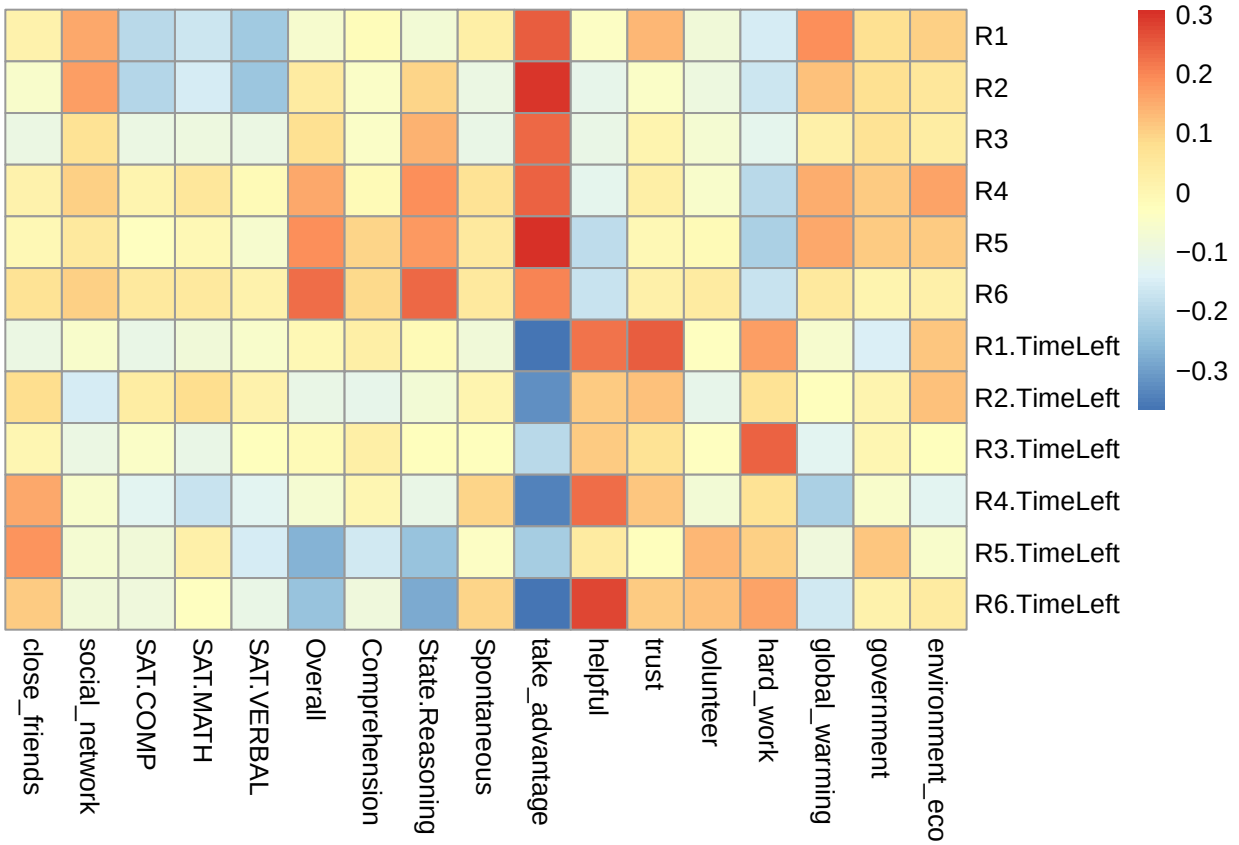
```
pls.data1 <- exp.neg[7:18]
pls.data2 <- exp.neg[19:35]

res.PLS <- tepPLS(DATA1 = pls.data1,
                  scale1 = "SS1",
                  center1 = TRUE,
                  DATA2 = pls.data2,
                  scale2 = "SS1",
                  center2 = TRUE,
                  DESIGN = exp.neg$group,
                  graphs = FALSE)


eigs <- res.PLS$TExPosition.Data$eigs
tau <- res.PLS$TExPosition.Data$t
```

```
fs <- res.PLS$TExPosition.Data$fi
cj <- res.PLS$TExPosition.Data$cj
ci <- res.PLS$TExPosition.Data$ci
fj <-res.PLS$TExPosition.Data$fj
lx <- res.PLS$TExPosition.Data$lx
ly <- res.PLS$TExPosition.Data$ly
p.pls <- res.PLS$TExPosition.Data$pdq$p
q.pls <- res.PLS$TExPosition.Data$pdq$q
```

9.3 Heatmap

From the heatmap, I found out that R1-R6 is significantly negative correlated with time usage R1-R6, which means that the higher game performance the participants have, the low time they used.

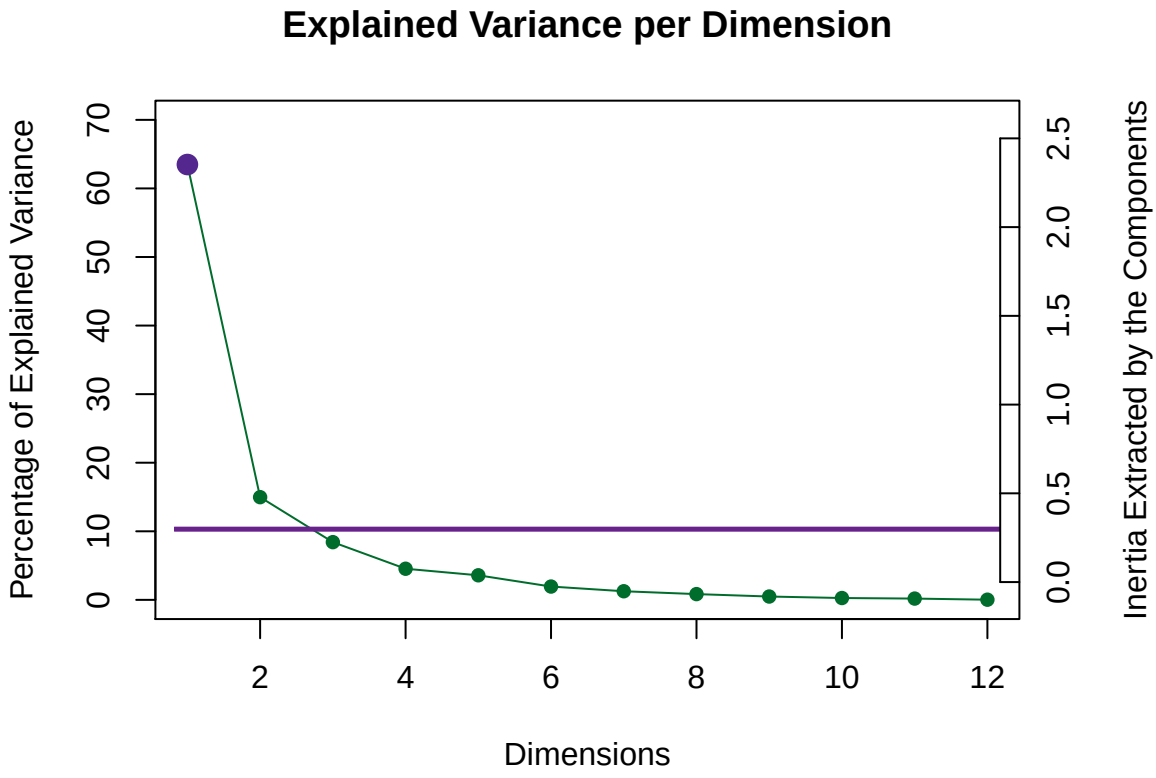
```
hm.pls <- pheatmap(mat=cor(pls.data1, pls.data2),
                   cluster_rows = FALSE,
                   cluster_cols = FALSE)
```



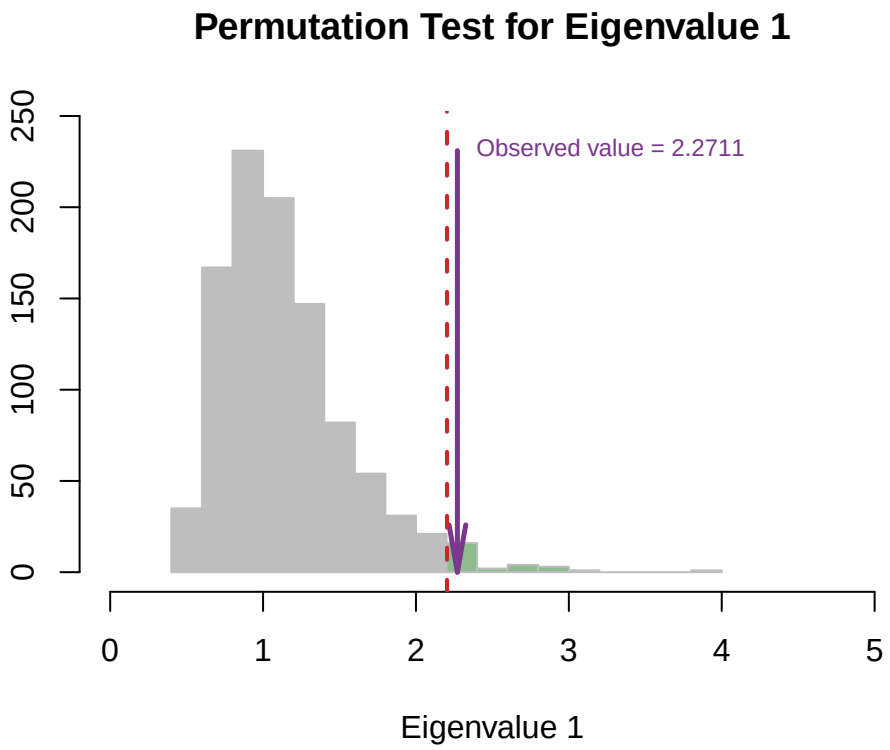
9.4 Scree Plot

The permutation test indicated that the first dimension is more stable than second one.

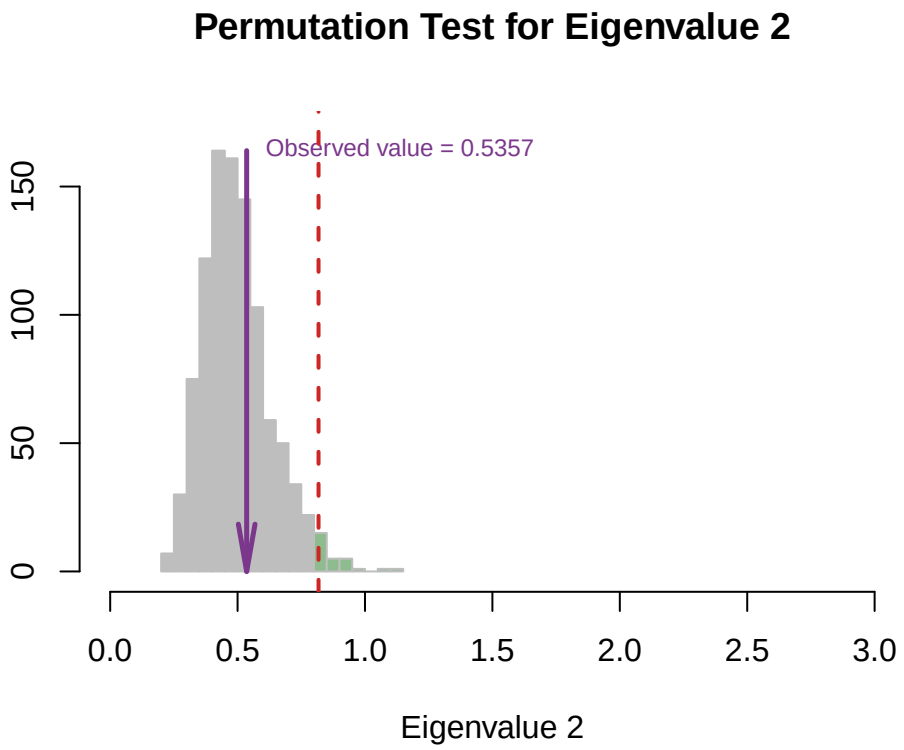
```
nIter = 1000
resPerm4PLSC <- perm4PLSC(pls.data1, pls.data2, nIter = nIter)

eigs.permu <- resPerm4PLSC$permEigenvalues

plot.scree(eigs, resPerm4PLSC$pEigenvalues)
```




```
plot.permutation(eigs.perm = eigs.permu, eigs = eigs)
```



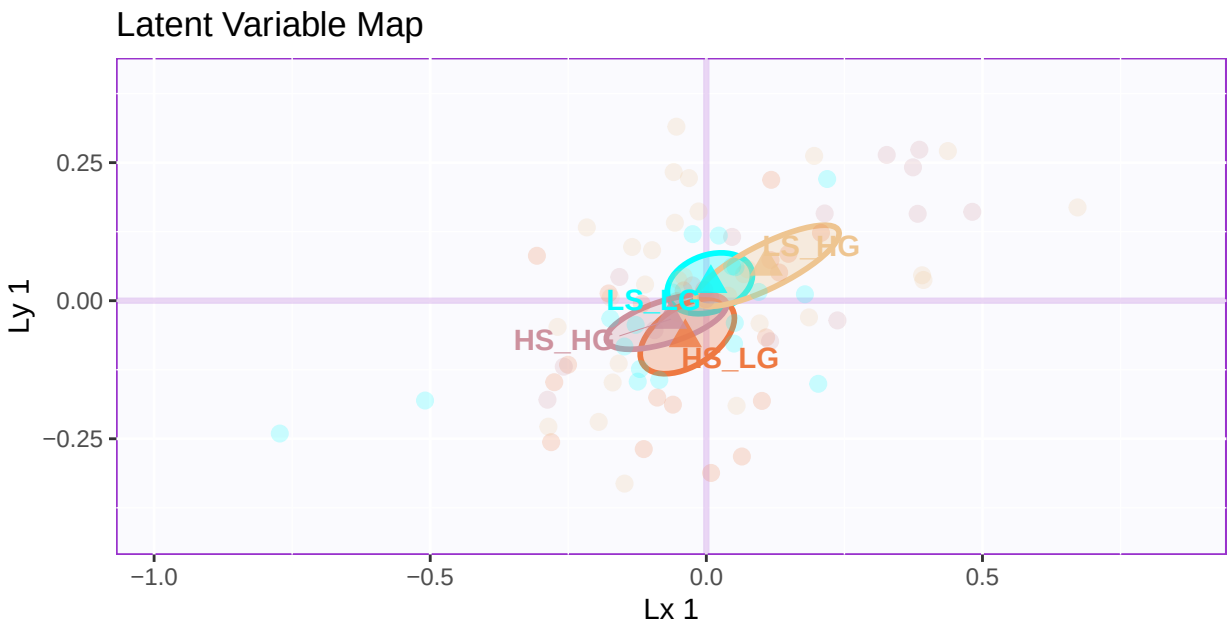

```
plot.permutation(eigs.perm = eigs.permu,
                 eigs = eigs,Dim = 2,para1 = 3)
```



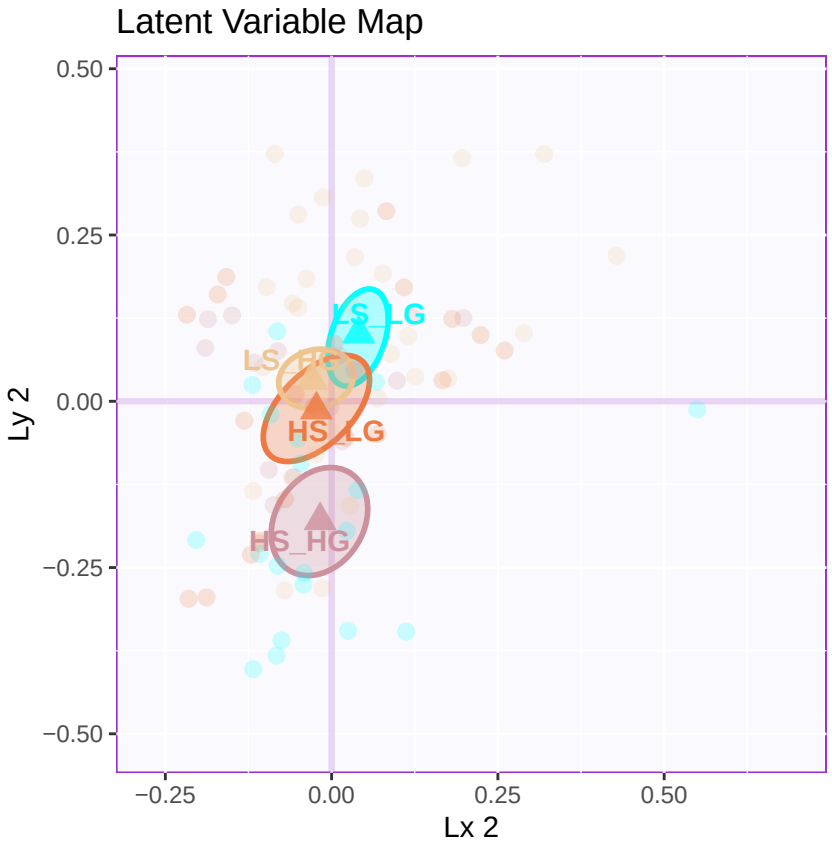
9.5 Latent Variables

However, from the latent variable plot, I found out that the lv 2 gave me the best separation among groups.

```
plot.lv(exp.neg$group, lx, ly, d=1)
```




```
plot.lv(exp.neg$group, lx, ly, d=2)
```



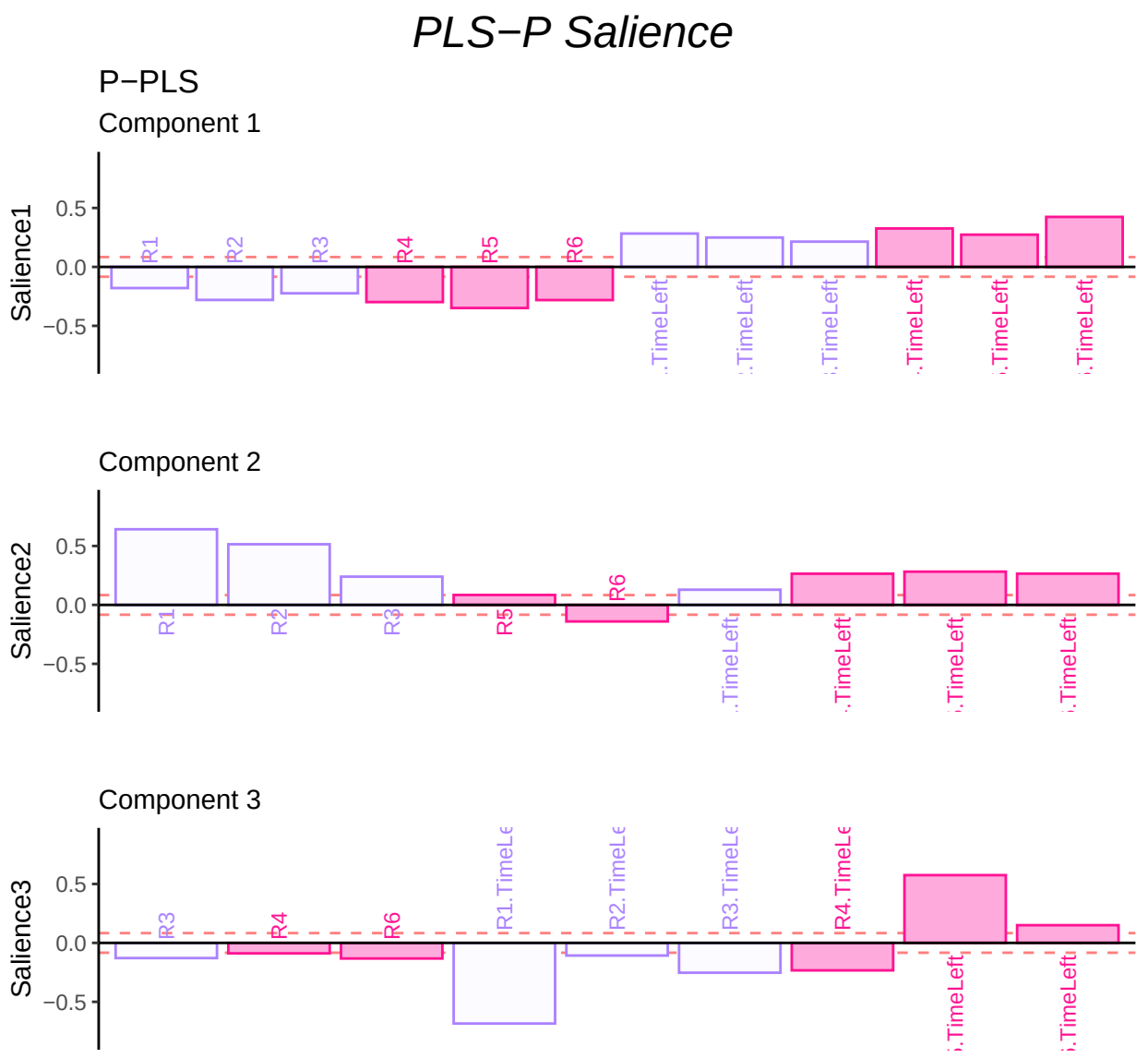
9.6 Salience

The salience results gave back to me similar information with previous analysis.

- In the dimension 1:
  - data1: contrast of collective behavor and time usage
  - data2: helpful, trust vs attitude, take_advantage
- In the dimension 2:
  - data1: the first half of the game vs R6
  - data2: SAT & empathy vs helpful, trust and attitude

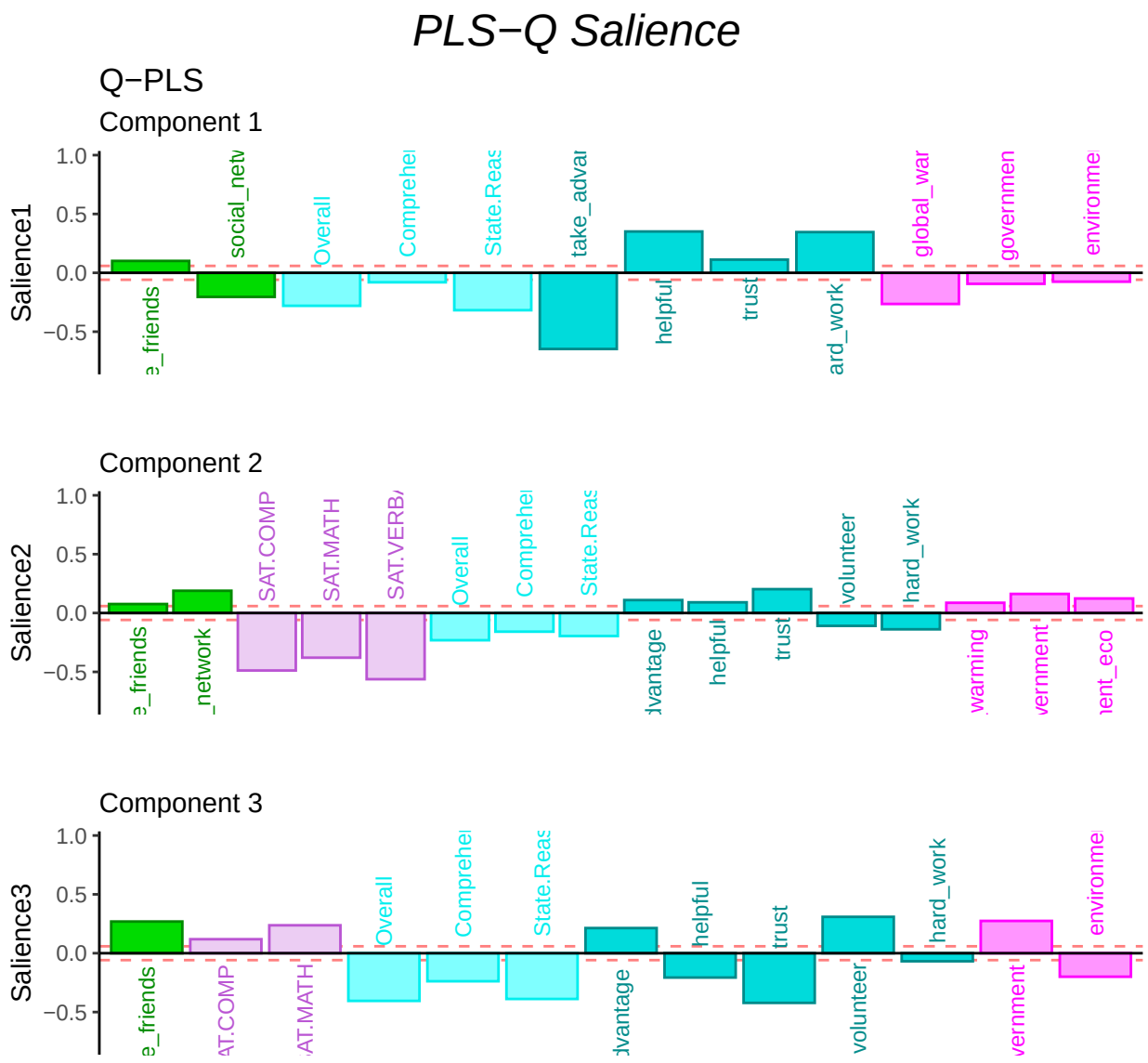
```
# PLS-P

laDim = 1
ctrJ.1 <- PrettyBarPlot2(p.pls[,laDim],
                         threshold = 1 / NROW(p.pls),
                         font.size = 3,
                         signifOnly = TRUE,
```

```
                            horizontal = TRUE,
                            color4bar = m.color.design[1:12],
                            main = 'Salience',
                            ylab = paste0('Salience', laDim),
                            ylim = c(1.2*min(p.pls), 1.2*max(p.pls))
  ) + ggtitle("P-PLS",subtitle = paste0('Component ', laDim))

### plot contributions for component 2
laDim =2
ctrJ.2 <- PrettyBarPlot2(p.pls[,laDim],
                         threshold = 1 / NROW(p.pls),
                         font.size = 3,
                         color4bar = m.color.design[1:12],
                         signifOnly = TRUE,
                         horizontal = TRUE,
                         main = 'Salience',
                         ylab = paste0('Salience', laDim),
                         ylim = c(1.2*min(p.pls), 1.2*max(p.pls))
)+ ggtitle("",subtitle = paste0('Component ', laDim))
laDim =3
ctrJ.3 <- PrettyBarPlot2(p.pls[,laDim],
                         threshold = 1 / NROW(p.pls),
                         font.size = 3,
                         color4bar = m.color.design[1:12],
                         signifOnly = TRUE,
                         horizontal = TRUE,
                         main = 'Salience',
                         ylab = paste0('Salience', laDim),
                         ylim = c(1.2*min(p.pls), 1.2*max(p.pls))
)+ ggtitle("",subtitle = paste0('Component ', laDim))


gridExtra::grid.arrange(as.grob(ctrJ.1), as.grob(ctrJ.2),as.grob(ctrJ.3),
    ncol=1, top = textGrob("PLS-P Salience",gp=gpar(fontsize=18,font=3)))
```




```
## PLS-Q

laDim = 1
ctrJ.1 <- PrettyBarPlot2(q.pls[,laDim],
                         threshold = 1 / NROW(q.pls),
                         font.size = 3,
                         signifOnly = TRUE,
                         horizontal = TRUE,
                         color4bar = m.color.design[13:29],
                         main = 'Salience',
                         ylab = paste0('Salience', laDim),
                         ylim = c(1.2*min(q.pls), 1.2*max(q.pls))
  ) + ggtitle("Q-PLS",subtitle = paste0('Component ', laDim))
```

```
### plot contributions for component 2
laDim =2
ctrJ.2 <- PrettyBarPlot2(q.pls[,laDim],
                         threshold = 1 / NROW(q.pls),
                         font.size = 3,
                         color4bar = m.color.design[13:29],
                         signifOnly = TRUE,
                         horizontal = TRUE,
                         main = 'Salience',
                         ylab = paste0('Salience', laDim),
                         ylim = c(1.2*min(q.pls), 1.2*max(q.pls))
)+ ggtitle("",subtitle = paste0('Component ', laDim))
laDim =3
ctrJ.3 <- PrettyBarPlot2(q.pls[,laDim],
                         threshold = 1 / NROW(q.pls),
                         font.size = 3,
                         color4bar = m.color.design[13:29],
                         signifOnly = TRUE,
                         horizontal = TRUE,
                         main = 'Salience',
                         ylab = paste0('Salience', laDim),
                         ylim = c(1.2*min(q.pls), 1.2*max(q.pls))
)+ ggtitle("",subtitle = paste0('Component ', laDim))


gridExtra::grid.arrange(as.grob(ctrJ.1), as.grob(ctrJ.2),as.grob(ctrJ.3),
    ncol=1, top = textGrob("PLS-Q Salience",gp=gpar(fontsize=18,font=3)))
```

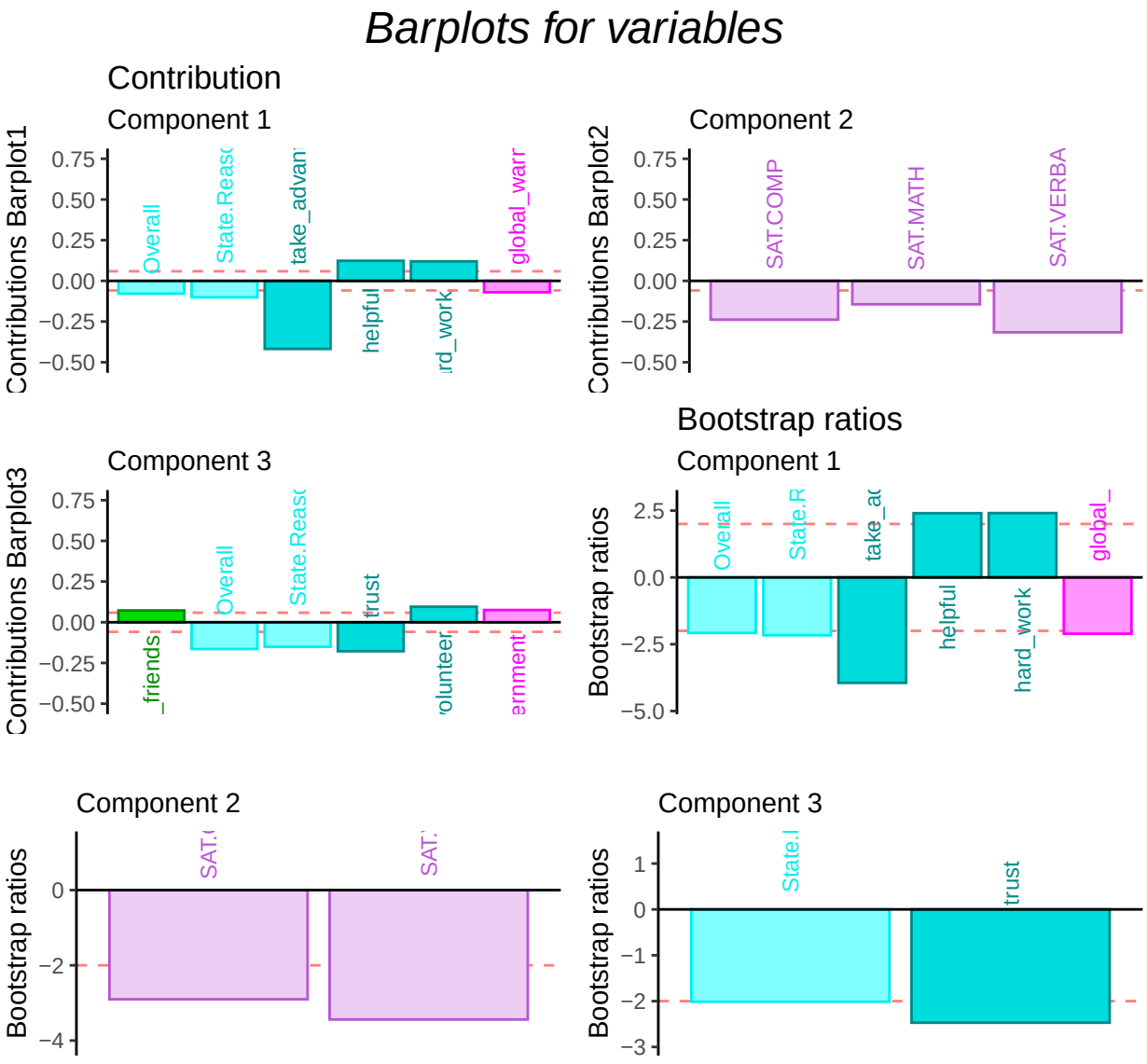


9.7 Bootstrap Ratio Barplot

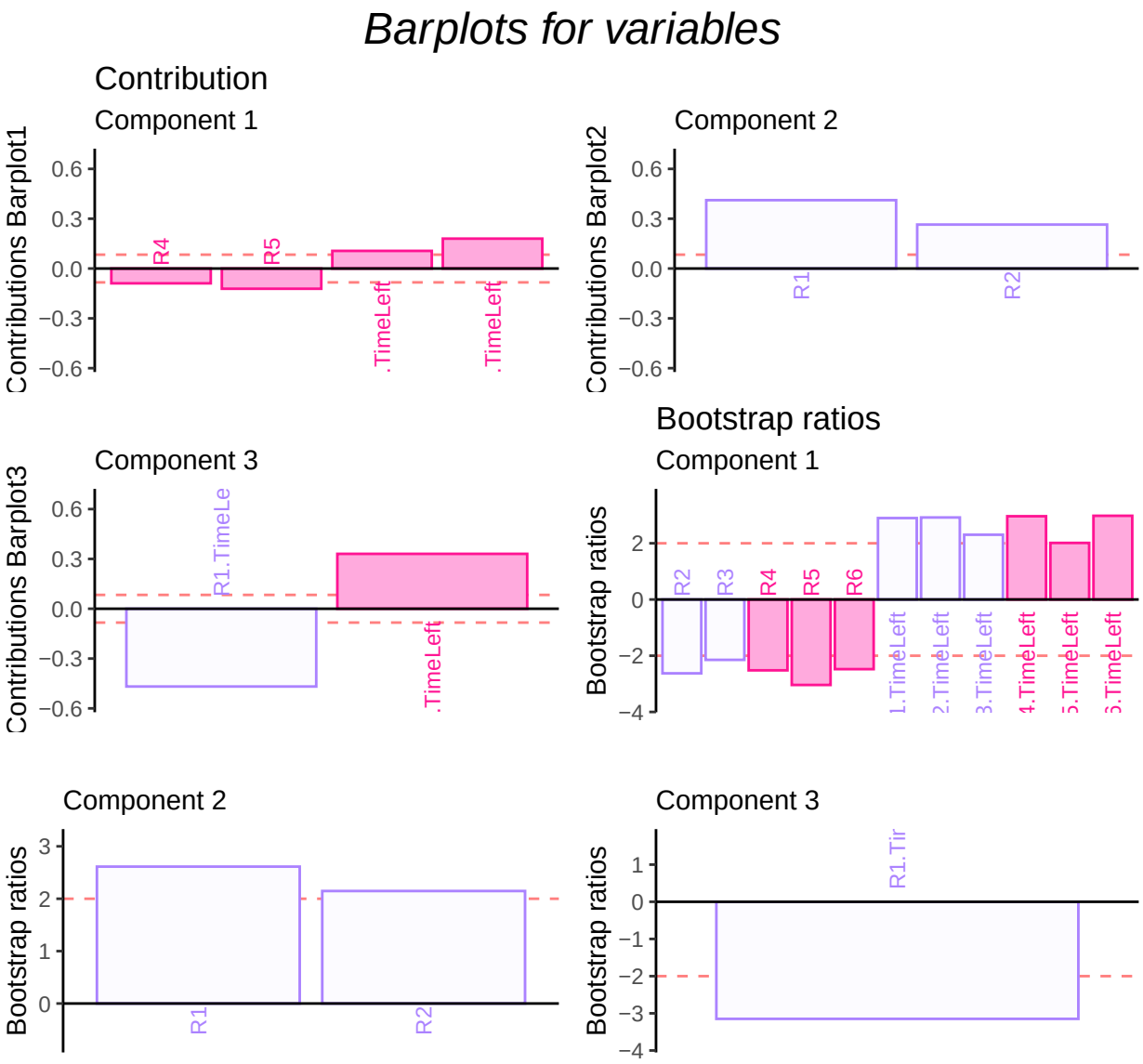
The bootstrap ratio gave me back the similar info on the two data sets.

```
res.Boot4PLSC <- Boot4PLSC(pls.data1, # First Data matrix
                           pls.data2, # Second Data matrix
                           nIter = 1000, # How many iterations
                           Fi = fs,
                           Fj = fj,
                           nf2keep = 3,
                           critical.value = 2,
                           # To be implemented later
                           # has no effect currently
```

```
                          alphaLevel = .05)

plot.cb(cj=cj,
        fj=fj,
        col = m.color.design[13:29],
        boot.ratios = res.Boot4PLSC$bootRatios.j,
        fig = 3,
        colrow = "row",
        horizontal = TRUE,
        fontsize = 3,
        signifOnly = TRUE)
```



```
plot.cb(ci,
        fs,
        col = m.color.design[1:12],
        boot.ratios = res.Boot4PLSC$bootRatios.i,
        fig = 3, colrow = "row",
        horizontal = TRUE,
        fontsize = 3,
        signifOnly = TRUE)
```



Chapter 10

DiSTATIS

10.1 Introduction of DiSTATIS

DiSTATIS is a statistical method designed for analyzing multiple data table for qualitative data from multidimensional scaling. DiSTATIS is capable for analyzing multiple sub data tables. It is different from MCA because MCA is only processing one data table with different nominal variables. However, DiSTATIS is processing multiple data tables at the same time. An much easier way to distinguish them is that DiSTATIS is corresponding to MFA for nominal data. Since it is a distance analysis method, the core concept is to know the similarity in the data pattern.

In this chapter, I am using a different data set called `wines` data set to conduct DiSTATIS. Please see more detailed info at data intro part[3.3]

10.2 Computation

Since DiSTATIS cares about the distance, I need to create a distance cube for the analysis here, which is different than usual analysis.

```
# table.wines, table.wines.names, table.wines.flavors

# have the design
place.design <- as.matrix(table.wines$X)
rownames(place.design) <- table.wines$X
for(i in 1:nrow(place.design)){
  place.design[i,] <- str_sub(rownames(place.design)[i], 1, 1)
}
rownames(table.wines) <- table.wines$X
table.wines.data <- table.wines[-1]
rownames(table.wines.flavors) <- table.wines.flavors$X
table.wines.flavors.data <- table.wines.flavors[-1]
colnames(table.wines.flavors.data) <- table.wines.names[,2]

place.design.col <- place.design
```

```
place.design.col[which(place.design.col == "F")] <- index[1]
place.design.col[which(place.design.col == "S")] <- index[3]



# creating distance cube
DistanceCube <- DistanceFromSort(table.wines.data)

# run distance cube
res.Distatis <- distatis(DistanceCube)

# get the factors from the Cmat analysis

G <- res.Distatis$res4Cmat$G
C <- res.Distatis$res4Cmat$C
eigs <- res.Distatis$res4Cmat$eigValues
tau <- res.Distatis$res4Cmat$tau
cj <- res.Distatis$res4Splus$ctr
eigs.compromise <- res.Distatis$res4Splus$eigValues
# get partial factor array
BootF <- BootFactorScores(res.Distatis$res4Splus$PartialF)
```

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

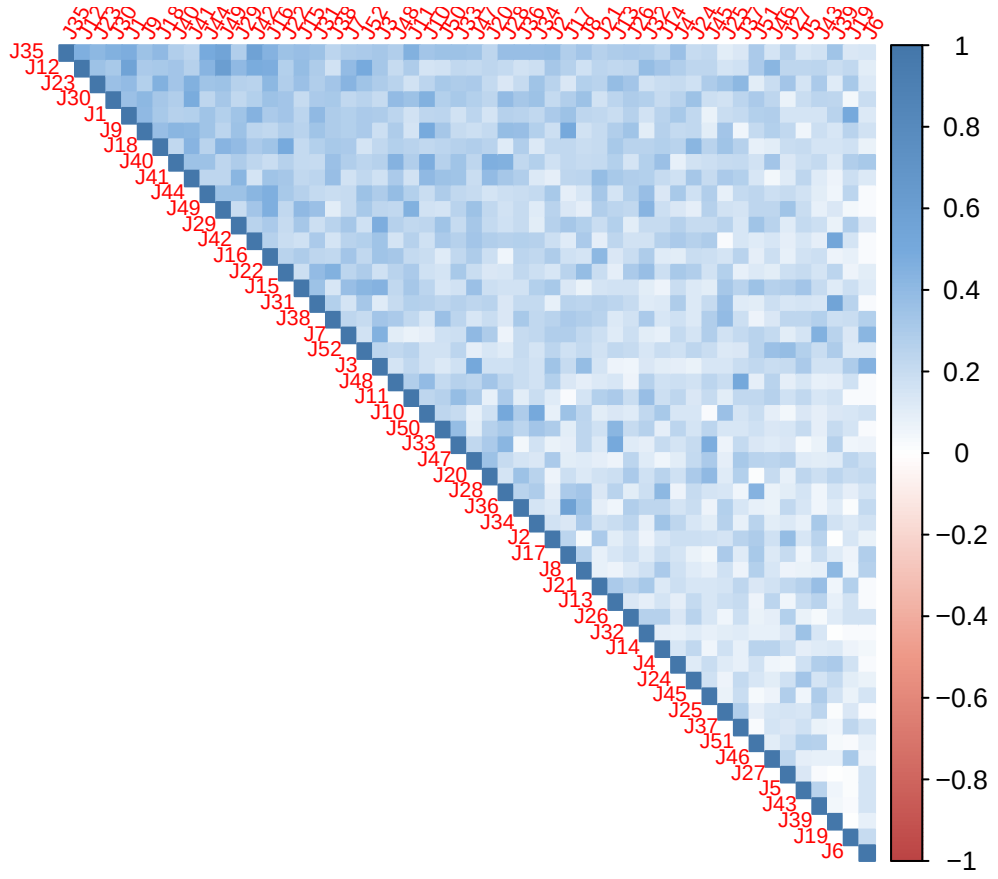
```
F <- res.Distatis$res4Splus$F
pf.j <- res.Distatis$res4Splus$PartialF
alpha <- res.Distatis$res4Cmat$alpha
```

10.3 HeatMap

I used C, the correlation matrix from the analysis to plot the heatmap. By using the 'first principal component', I am able to know what's the inner pattern in these judges and make clustering based on their response.

```
# plot color
col <-colorRampPalette(c("#BB4444",
                         "#EE9988",
                         "#FFFFFF",
                         "#77AADD",
                         "#4477AA"))

# plot
corr4MCA.r <- corrplot::corrplot(
  C,
```

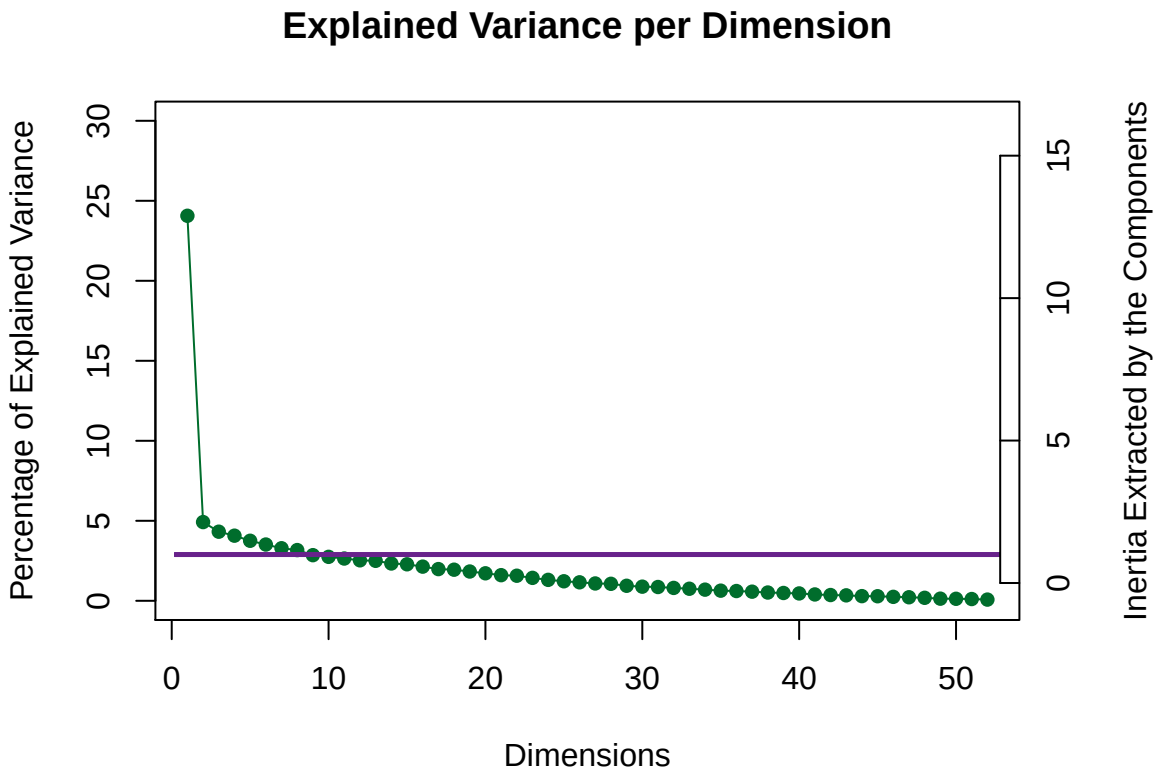
```
  method="color",
  col=col(200),
  type="upper",
  #addCoef.col = "black",# Add coefficient of correlation
  tl.cex = .7,
  tl.srt = 60,#Text label color and rotation
  order = "FPC",
  #number.cex = 0.5,
  diag = TRUE # needed to have the color of variables correct
  )
```



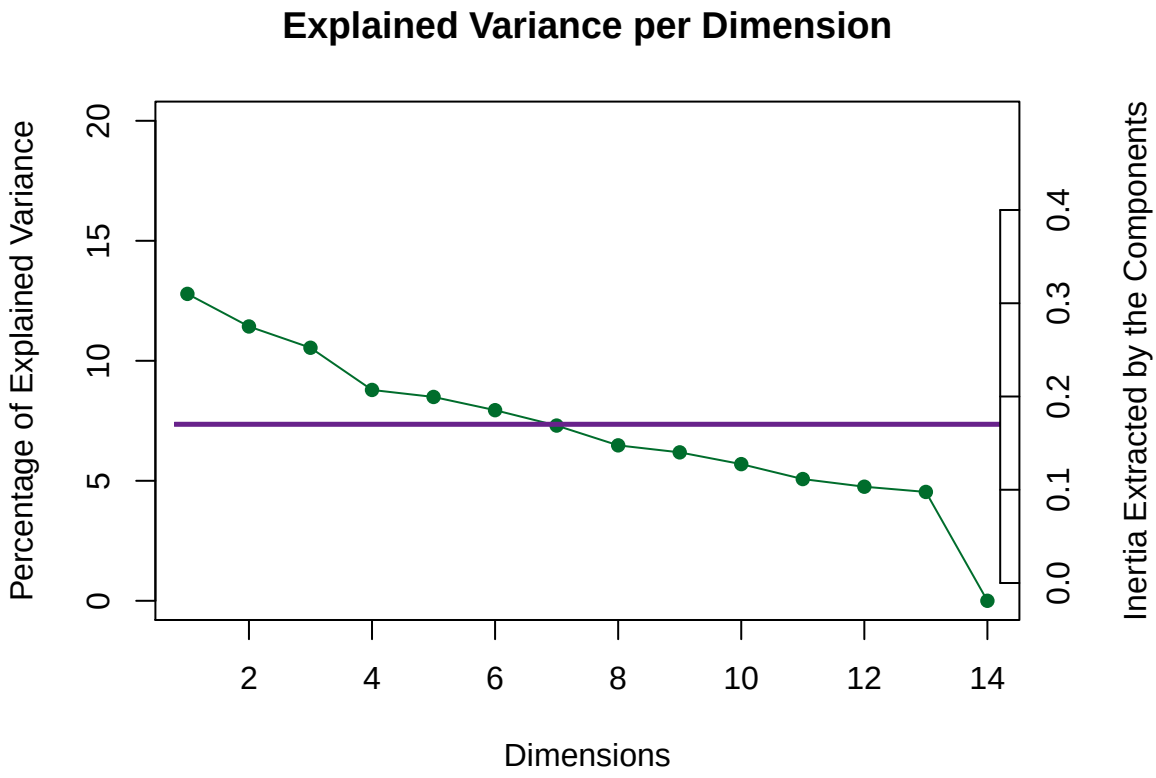
10.4 Scree Plot

I have tow scree plots here: 1 is for the global one (all judges), one is for the compromise data (wines).

```
my.scree <- PlotScree(ev = eigs,
  plotKaiser = TRUE,
  title = "Explained Variance per Dimension")
```




```
my.scree <- PlotScree(ev = eigs.compromise,
  plotKaiser = TRUE,
  title = "Explained Variance per Dimension")
```

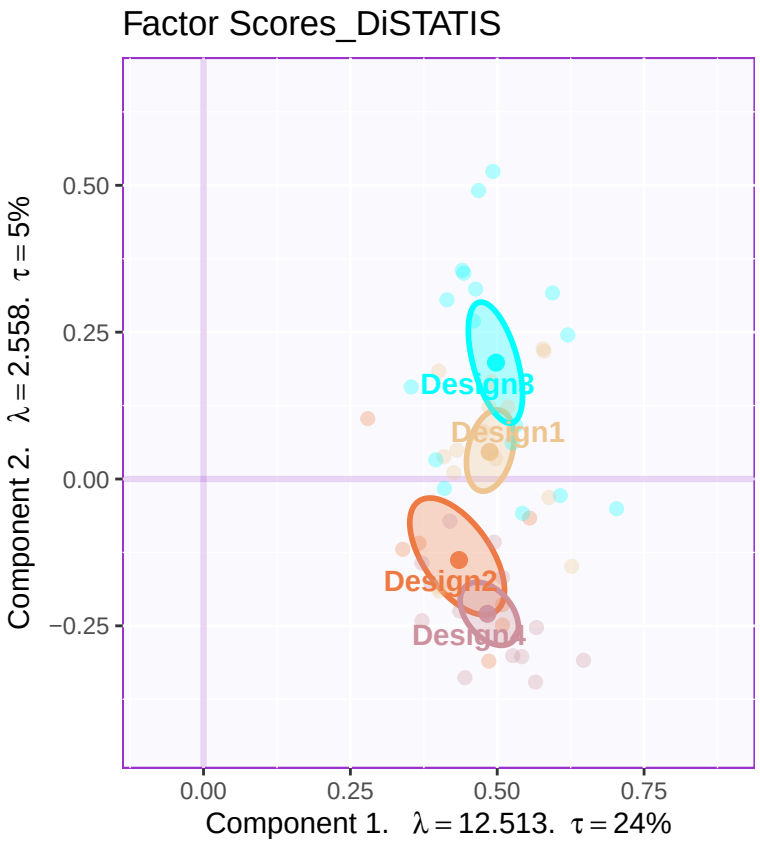


10.5 Global Factor Scores

Since I don't have any pre-group before, based on the factor scores, I used kmeans to manually cluster 4 groups for labels. From the global factor scores, it serves well.

```
# cluster analysis

colnames(G) <- paste0("Dimension ", 1:ncol(G))

fit <- kmeans(G[,1:5],4)

Judge.design <- as.matrix(fit$cluster)
Judge.design[which(Judge.design == 1)] <- "Design1"
Judge.design[which(Judge.design == 2)] <- "Design2"
Judge.design[which(Judge.design == 3)] <- "Design3"
Judge.design[which(Judge.design == 4)] <- "Design4"
colnames(Judge.design) <- "group"

J.D <- as.matrix(Judge.design)
# get some color
nominal.Judges <- makeNominalData(as.data.frame(J.D))
```

```
color4Judges.list <- prettyGraphs::createColorVectorsByDesign(nominal.Judges)

plot.fs(as.factor(J.D), G[,1:5],
        eigs, tau, method = "DiSTATIS")
```



10.6 Partial Factor Scores

The first partial factor scores is to compute a partial fi array.

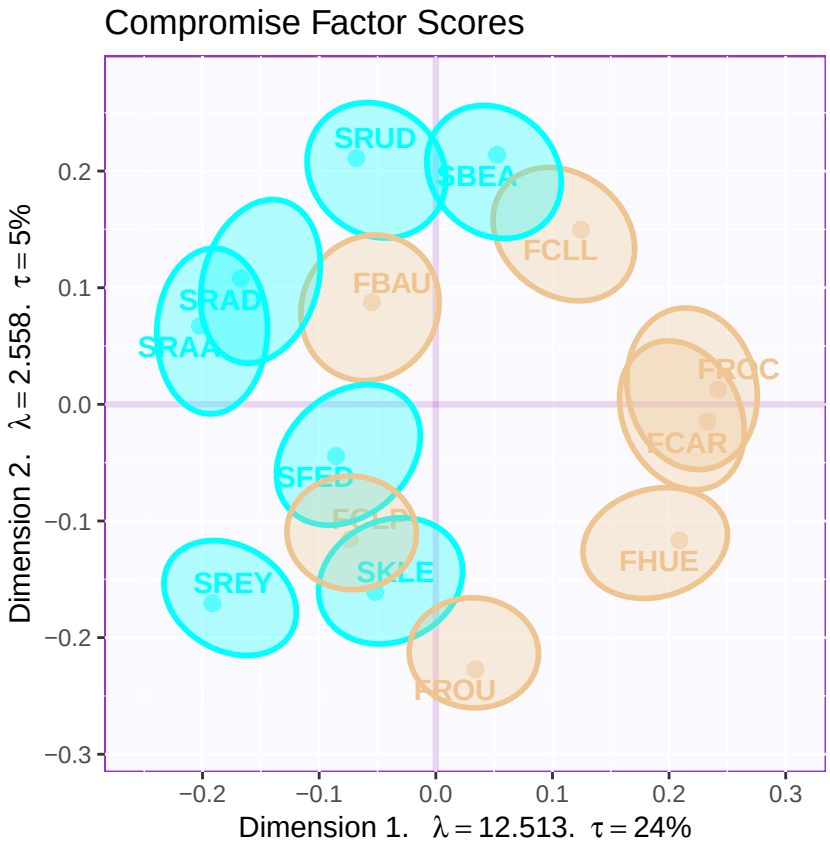
```
# create the groups of Judges
code4Groups <- unique(J.D)
nK <- length(code4Groups)
# initialize F_K and alpha_k
F_k <- array(0, dim = c(dim(pf.j)[[1]],
                        dim(pf.j)[[2]],nK))

dimnames(F_k) <- list(dimnames(pf.j)[[1]],
dimnames(pf.j)[[2]], code4Groups)
alpha_k <- rep(0, nK)
names(alpha_k) <- code4Groups
```

```
Fa_j <- pf.j
# A horrible loop
for (j in 1:dim(pf.j)[[3]]){ Fa_j[,,j] <- pf.j[,,j] * alpha[j]}

# Another horrible loop
for (k in 1:nK){
lindex <- J.D == code4Groups[k]
alpha_k[k] <- sum(alpha[lindex])
F_k[,,k] <- (1/alpha_k[k])*apply(Fa_j[,,lindex],c(1,2),sum)
}
```

From the partial factor scores plot, I can compare the 4 groups differently with their own judging response. It is interesting that design 2 group unstable evaluation on French wines and design 3 & 4 groups have huge inner discrepancy on South Africa's wines.

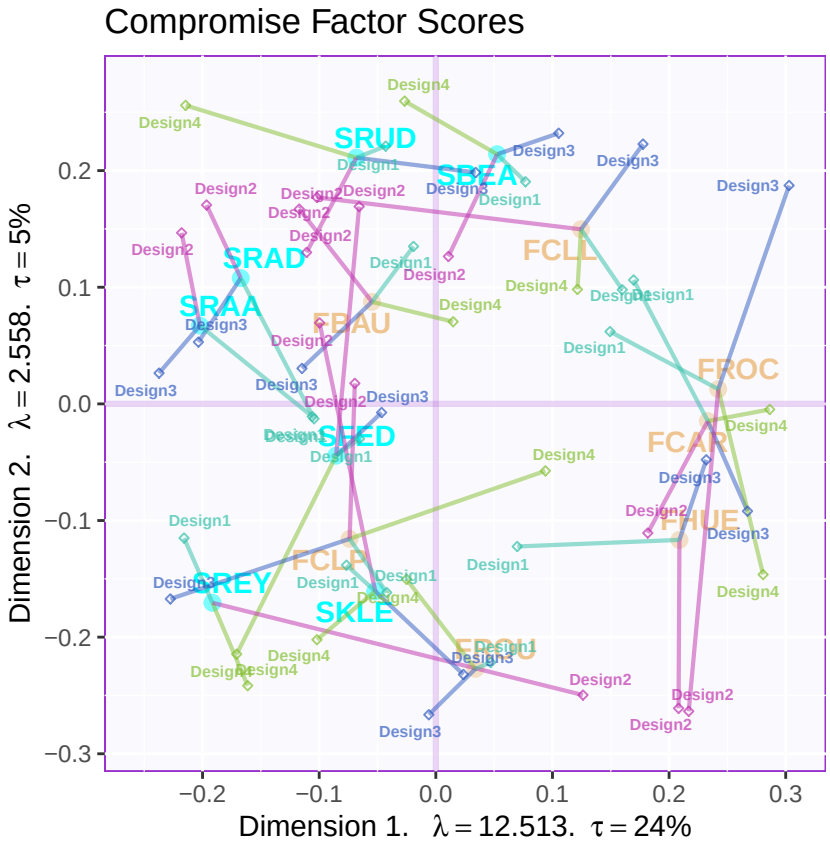
```
# 5.2 a compromise plot

# To get graphs with axes 1 and 2:
h_axis = 1
v_axis = 2
# To get graphs with say 2 and 3

### base map
gg.compromise.graph.out <- createFactorMap(F,
              axis1 = h_axis,
              axis2 = v_axis,
              title = "Compromise Factor Scores",
              col.points = place.design.col,
              col.labels = place.design.col)

# labels
label4S <- createxyLabels.gen(x_axis= h_axis,
                              y_axis = v_axis,
                              lambda= eigs,
                              tau = tau,
                              axisName = "Dimension ")


#### a bootstrap confidence interval plot
gg.boot.graph.out.elli <- MakeCIEllipses(
                        data = BootF[,c(h_axis,v_axis),],
                        names.of.factors =
                        c(paste0('Factor ',h_axis),
                          paste0('Factor ',v_axis)),
                        col = place.design.col)

# Add ellipses to compromise graph
```

```
gg.map.elli <- gg.compromise.graph.out$zeMap +
  gg.boot.graph.out.elli +
  label4S #
gg.map.elli
```




```
### get the partial map

map4PFS <- createPartialFactorScoresMap(
                factorScores = F,
                partialFactorScores = F_k,
                axis1 = 1, axis2 = 2,
                colors4Items = as.vector(place.design.col),
                colors4Blocks = unique(color4Judges.list$oc),
                names4Partial = dimnames(F_k)[[3]],
                font.labels = 'bold')

d1.partialFS.map.byProducts <- gg.compromise.graph.out$zeMap +
  map4PFS$mapColByItems +
  label4S

d2.partialFS.map.byCategories <- gg.compromise.graph.out$zeMap +
map4PFS$mapColByBlocks +
```

```
  label4S

d2.partialFS.map.byCategories
```

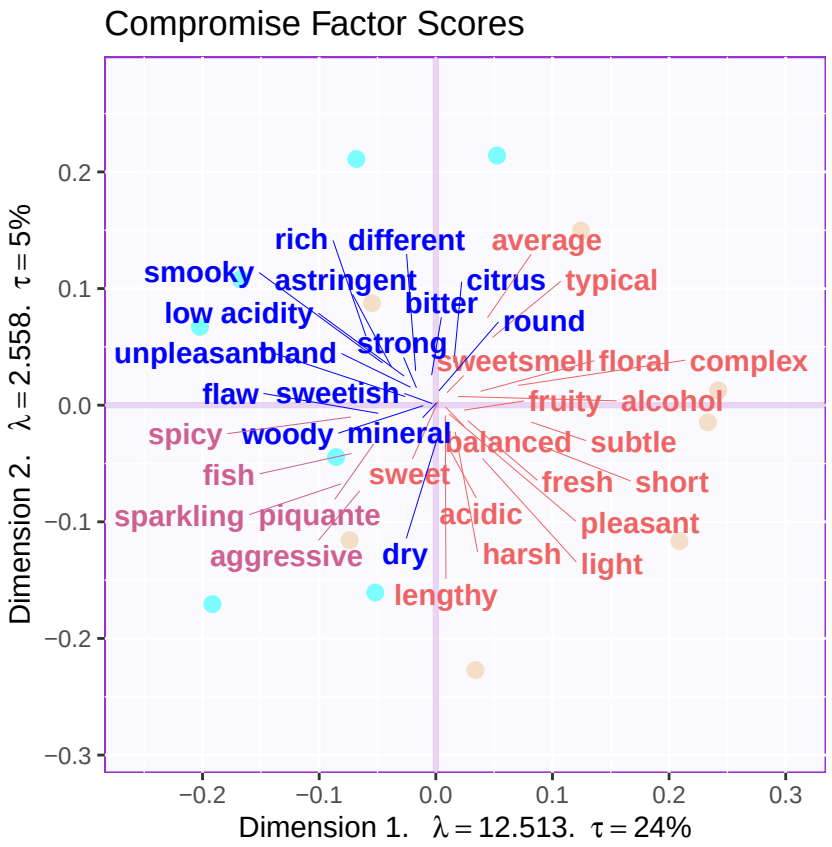


10.7 Vocabulary graphs

By using Kmeans again, I tried to cluster all these flavor tag in the vocabulary data table. Right dot for the French and left dot for the South Africa, I found that French wines are described as more balanced and typical flavors; However, South Africa is considered as rich and strong taste, even sometimes it is not a good thing. Some other vocabulares, spicy, fish, aggressive are clustered to anther group. Since there are still some strong flavors, they are more closer to the South Africa wines.

```
# 5.5. Vocabulary
# 5.5.2 CA-like Barycentric (same Inertia as products)

F4Voc <- DistatisR::projectVoc(table.wines.flavors.data, F)

set.seed(44)
Voc.clusters <- kmeans(F4Voc$Fvoca.bary, 3)
```

```
Voc.color <- Voc.clusters$cluster
Voc.color[which(Voc.color == 1)] <- prettyGraphsColorSelection(starting.color = sample(1:170,1)
Voc.color[which(Voc.color == 2)] <- prettyGraphsColorSelection(starting.color = sample(1:170,1)
Voc.color[which(Voc.color == 3)] <- prettyGraphsColorSelection(starting.color = sample(1:170,1)



gg.voc.bary <- createFactorMap(F4Voc$Fvoca.bary,
title = 'Vocabulary',
col.points = Voc.color,
col.labels = Voc.color,
display.points = FALSE,
constraints = gg.compromise.graph.out$constraints)
#
gg.voc.bary.gr <- gg.voc.bary$zeMap + label4S
#print(e1.gg.voc.bary.gr)
gg.voc.bary.dots.gr <- gg.compromise.graph.out$zeMap_background +
  gg.compromise.graph.out$zeMap_dots +
  #gg.compromise.graph.out$zeMap_text +
  gg.voc.bary$zeMap_text + label4S
gg.voc.bary.dots.gr
```

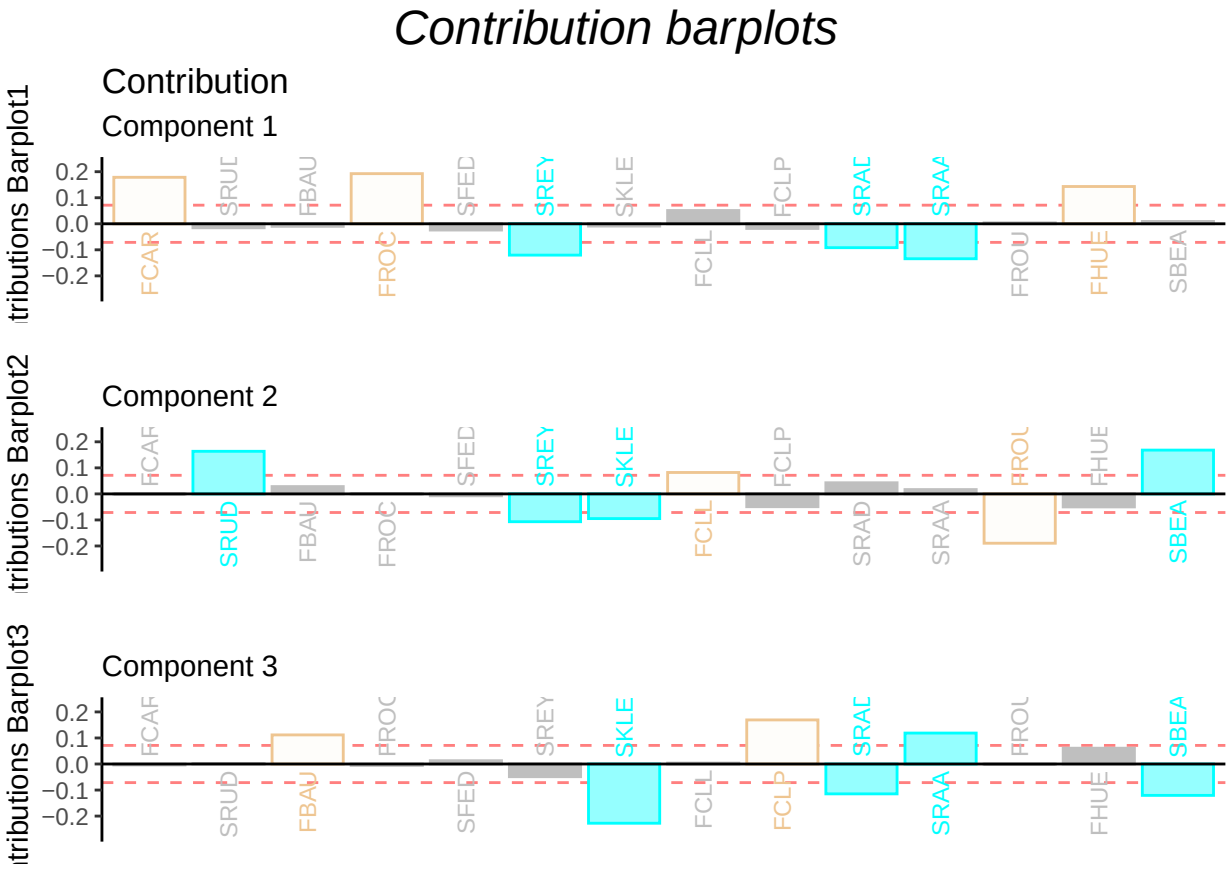


10.8 Contribution Barplots

The dimension 1 can perfectly distinguish the two wines from French and South Africa. There are also some ambiguous ones such as SRAA, FROU hard to distinguish from the data.

```
  signed.ctrJ <- cj * sign(F)
  laDim = 1
  ctrJ.1 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                threshold = 1 / NROW(signed.ctrJ),
                font.size = 3,
                signifOnly = FALSE,
                horizontal = TRUE,
                color4bar = place.design.col,
                main = 'Variable Contributions (Signed)',
                ylab = paste0('Contributions Barplot',laDim),
                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
    ) + ggtitle("Contribution",subtitle = paste0('Component ', laDim))

  ### plot contributions for component 2
  laDim =2
  ctrJ.2 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                threshold = 1 / NROW(signed.ctrJ),
                font.size = 3,
                color4bar = place.design.col,
                signifOnly = FALSE,
                horizontal = TRUE,
                main = 'Variable Contributions (Signed)',
                ylab = paste0('Contributions Barplot', laDim),
                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
  )+ ggtitle("",subtitle = paste0('Component ', laDim))
  laDim =3
  ctrJ.3 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                threshold = 1 / NROW(signed.ctrJ),
                font.size = 3,
                color4bar = place.design.col,
                signifOnly = FALSE,
                horizontal = TRUE,
                main = 'Variable Contributions (Signed)',
                ylab = paste0('Contributions Barplot', laDim),
                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
  )+ ggtitle("",subtitle = paste0('Component ', laDim))


gridExtra::grid.arrange(as.grob(ctrJ.1),
                        as.grob(ctrJ.2),
                        as.grob(ctrJ.3),
                        ncol=1,
```

```
                                top = textGrob("Contribution barplots",gp=gpar(fontsize=18,font=3)))
```



Chapter 11

Multiple Factor Analysis

11.1 Introduction of MFA

The Multiple Factor Analysis (MFA) is another external version of the PCA, aka PCA for multiple data set. The principle of the MFA is to do the two steps: * 1. do the PCA and normalize each data table by the first singular value (alpha); * 2. all these normalized data will be compiled into a new grand table for analysis. The advantage of the MFA is that the size of the data table will not take up larger variance in grand data table (it has been normalized). Besides, MFA can provide more details about how each sub data tables contribute/associate with my observation. Compared to other MSA method, MFA has more flexibility on data tables, which is suitable for exploration study. In this chapter, I will use 7 sub data table to conduct the MFA to investigate the realtionship between collective behaviors and social & general intelligence in college students.

11.2 Computation

It's worth to mention that the seven sub data tables' names are listed in the data intro part for your quick reference[3.1].

```
# import data
raw.mfa <- exp.neg[7:35]
raw.col.design <- as.matrix(raw.mfa[1,])
raw.col.design[1:6] <- "A1"
raw.col.design[7:12] <- "B1"
raw.col.design[13:14] <- "C1"
raw.col.design[15:17] <- "D1"
raw.col.design[18:21] <- "E1"
raw.col.design[22:26] <- "F1"
raw.col.design[27:29] <- "G1"
colnames(raw.mfa) <- paste0(raw.col.design[1:29],".",
                            colnames(raw.mfa)[1:29])
```

```
res.MFA <- mpMFA(raw.mfa,
                 column.design = raw.col.design,
                 DESIGN = exp.neg$group, graphs = FALSE)
```

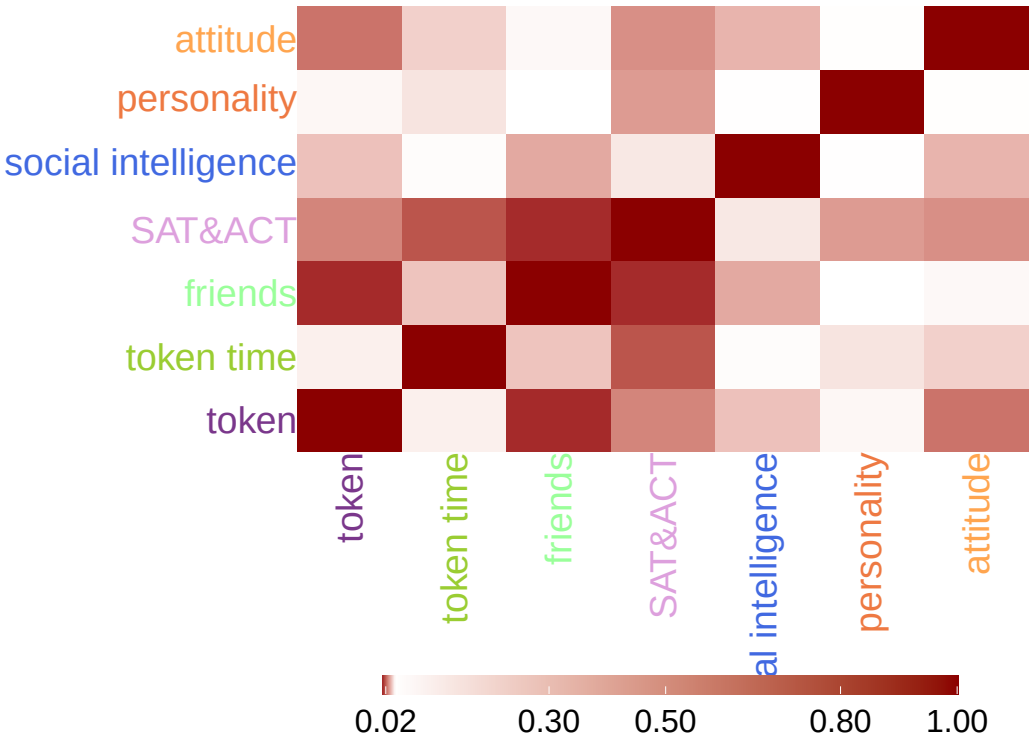
```
## [1] "Preprocessed the Rows of the data matrix using:  None"
## [1] "Preprocessed the Columns of the data matrix using:  Center_1Norm"
## [1] "Preprocessed the Tables of the data matrix using:  MFA_Normalization"
## [1] "Preprocessing Completed"
## [1] "Optimizing using:  None"
## [1] "Processing Complete"
```

```
rv.mfa <- res.MFA$mexPosition.Data$InnerProduct$RVMatrix
eigs <- res.MFA$mexPosition.Data$Table$eigs
fs <- res.MFA$mexPosition.Data$Table$fi
fj <- res.MFA$mexPosition.Data$Table$Q
cj <- res.MFA$mexPosition.Data$Table$cj
partial.fi.array <- res.MFA$mexPosition.Data$Table$partial.fi.array
tau <- res.MFA$mexPosition.Data$Table$t
```

11.3 Heatmap

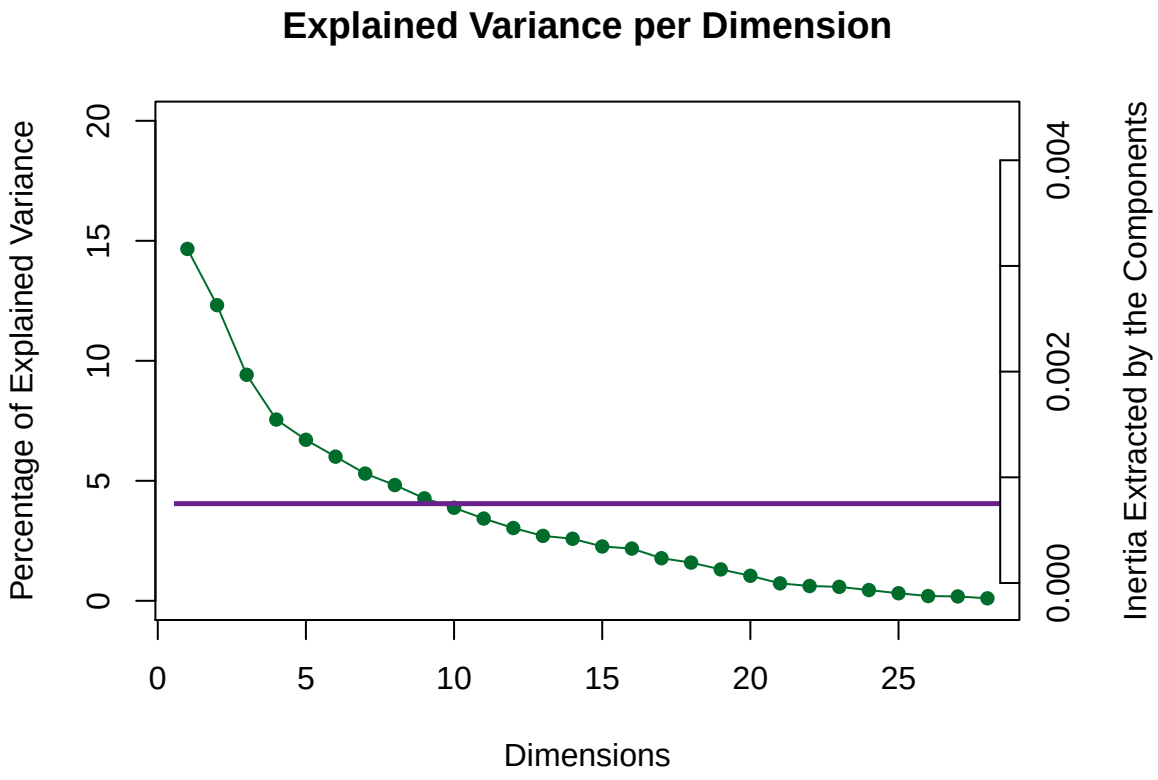
The RV heatmap will tell me the covariance among these seven data tables. From the heatmap, it is obvious that there are some sub tables have high covariance between each other, `friends` with `token`, `SAT` with `friends`.

```
# give names
names.subtable <- c("token",
                    "token time",
                    "friends",
                    "SAT&ACT",
                    "social intelligence",
                    "personality",
                    "attitude")
rownames(rv.mfa) <- names.subtable
colnames(rv.mfa) <- names.subtable

plot.heatmap(DATA = rv.mfa, scale = FALSE)
```



11.4 Scree Plot

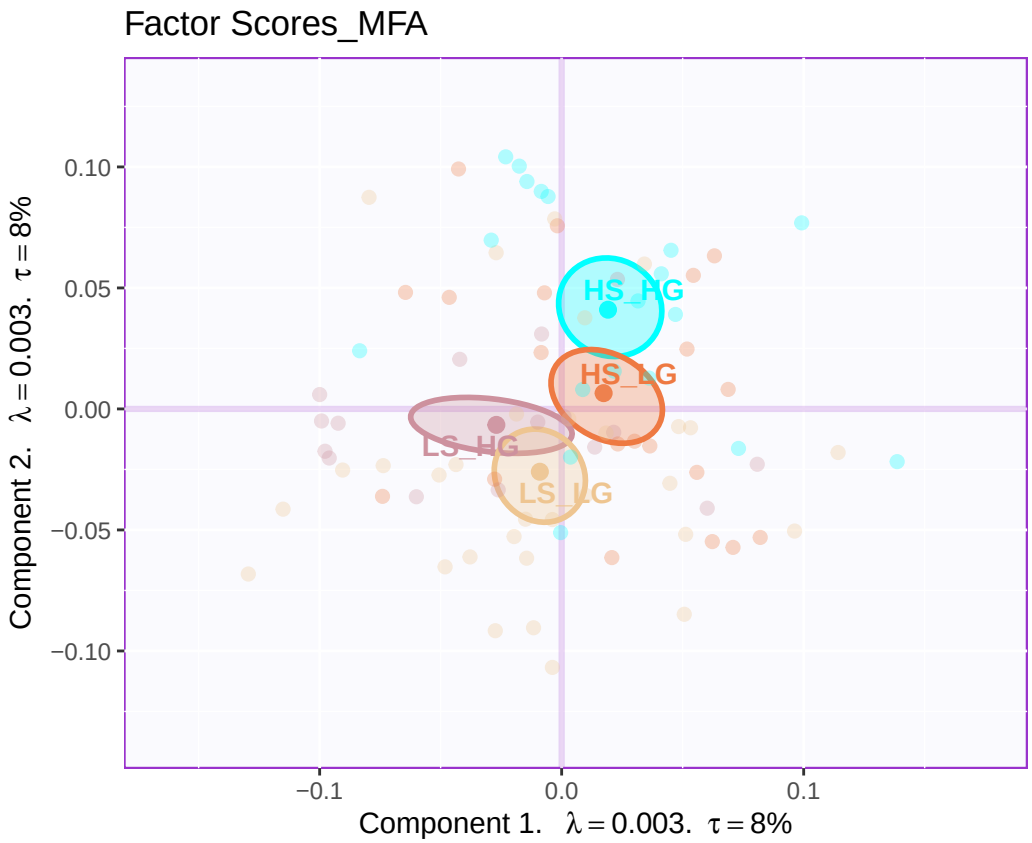
```
my.scree <- PlotScree(ev = eigs,
  plotKaiser = TRUE,
  title = "Explained Variance per Dimension")
```



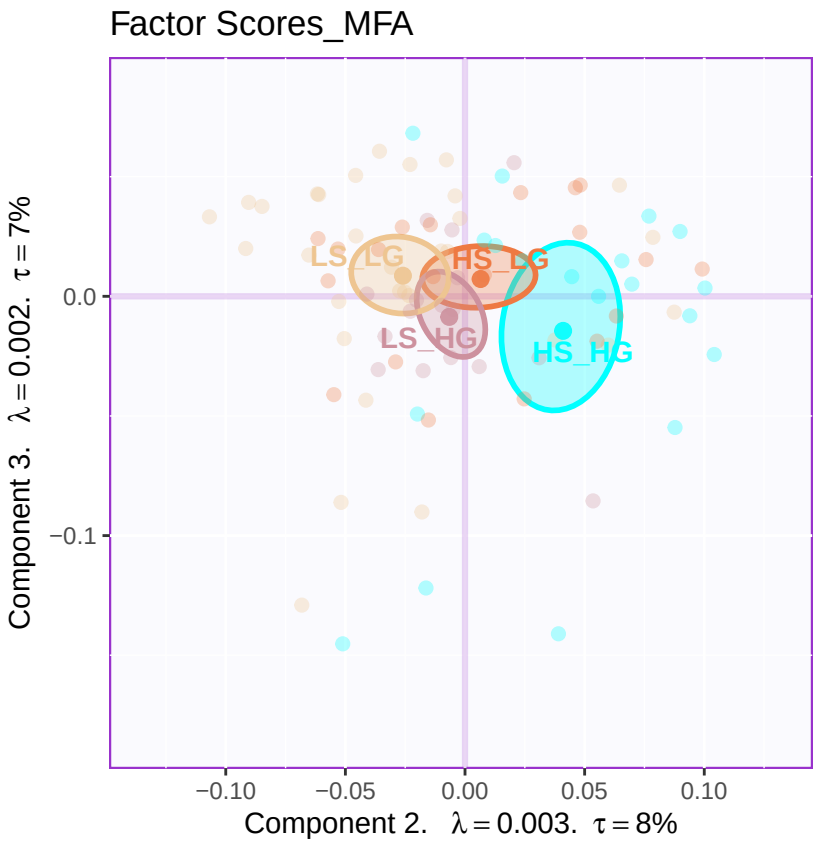
11.5 Global Factor Scores

The plot of global factor scores has shown similar results with PCA before. The only difference is that the dimension 3 looks more meaningful than PCA.

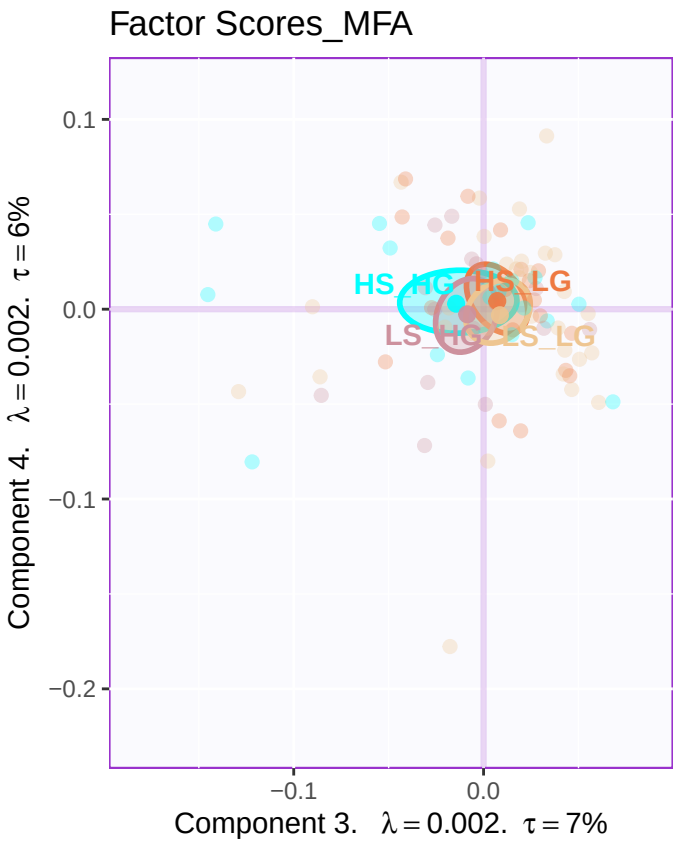
```
plot.fs(DESIGN = exp.neg$group,
        fs=fs, eigs=eigs,
        tau=tau, d = 1, mode="CI", method = "MFA")
```




```
plot.fs(DESIGN = exp.neg$group,
        fs=fs, eigs=eigs,
        tau=tau, d = 2, mode="CI", method = "MFA")
```



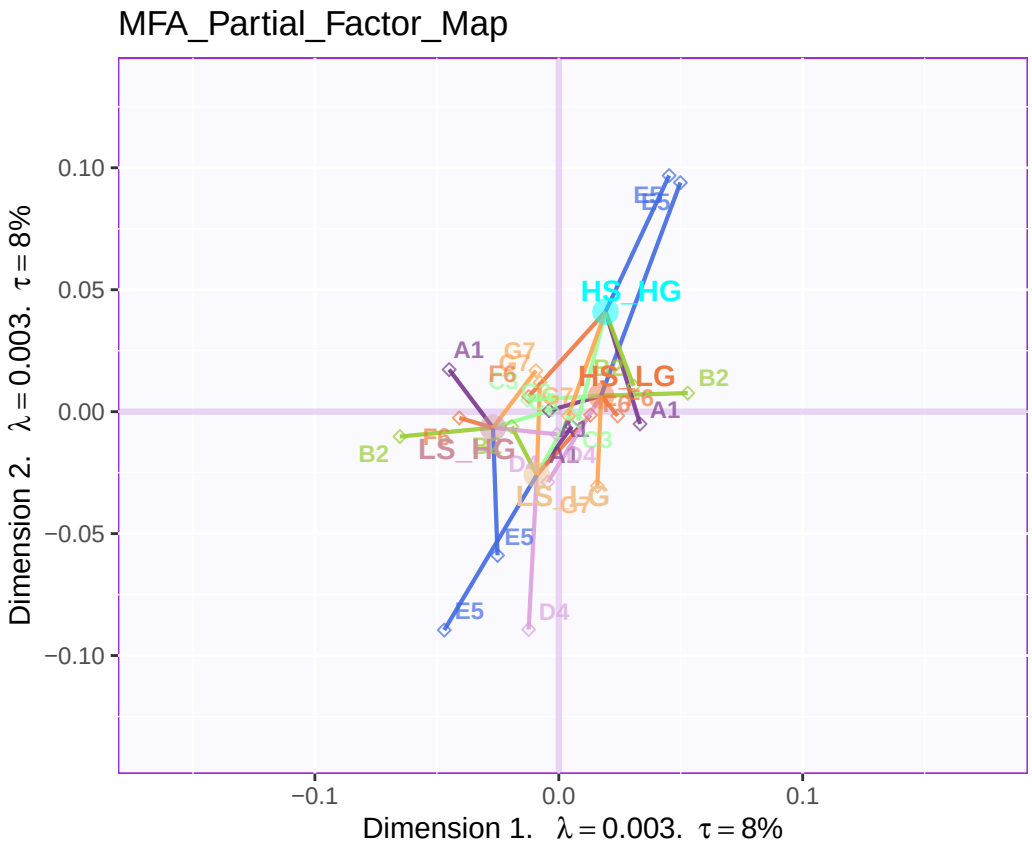

```
plot.fs(DESIGN = exp.neg$group,
        fs=fs, eigs=eigs,
        tau=tau, d = 3, mode="CI", method = "MFA")
```



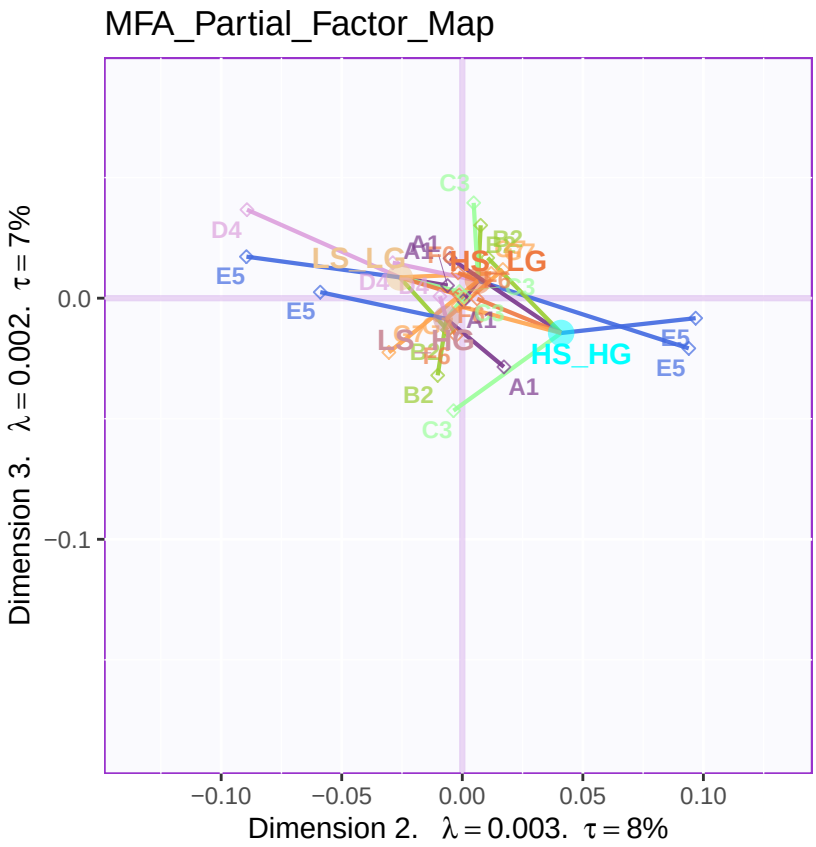
11.6 Partial Factor Scores

From the Partial Factor Scores, the most significant sub data table is D4 and E5, which represent **Social Intelligence, empathy** and **General Intelligence, SAT**. Also, A1, the collective game performance contributes much to the separation too. Compared to other results, since all the four groups are clustering at the center so it is hard to tell the specific difference. However, what I know is that most of these sub data tables will have different direction on HIGH_HIGH group and LOW_LOW group.

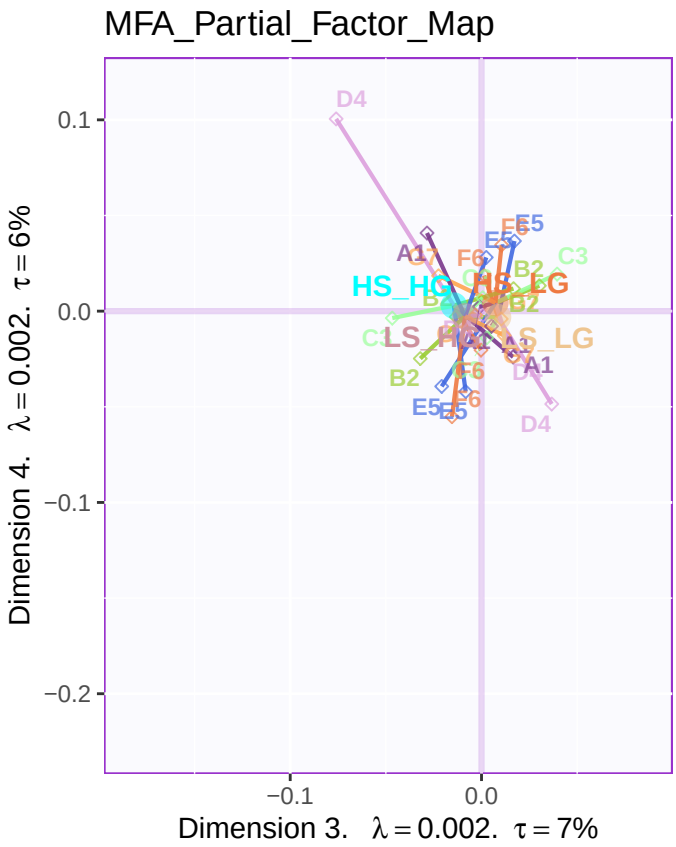
```
plot.partial.fs(DESIGN = exp.neg$group,
                fs = fs,
                eigs = eigs,
                tau = tau,
                d=1,
                partial.fi.array=partial.fi.array,
                mm = 7)
```




```
plot.partial.fs(DESIGN = exp.neg$group,
                fs = fs,
                eigs = eigs,
                tau = tau,
                d=2,
                partial.fi.array=partial.fi.array,
                mm = 7)
```




```
plot.partial.fs(DESIGN = exp.neg$group,
                fs = fs,
                eigs = eigs,
                tau = tau,
                d=3,
                partial.fi.array=partial.fi.array,
                mm = 7)
```



11.7 Contribution Barplots

The results from contribution barplots are quite similar with before.

In dimension 1, the time usage is negatively associated with collective behavior, and helpful is in same line with time usage

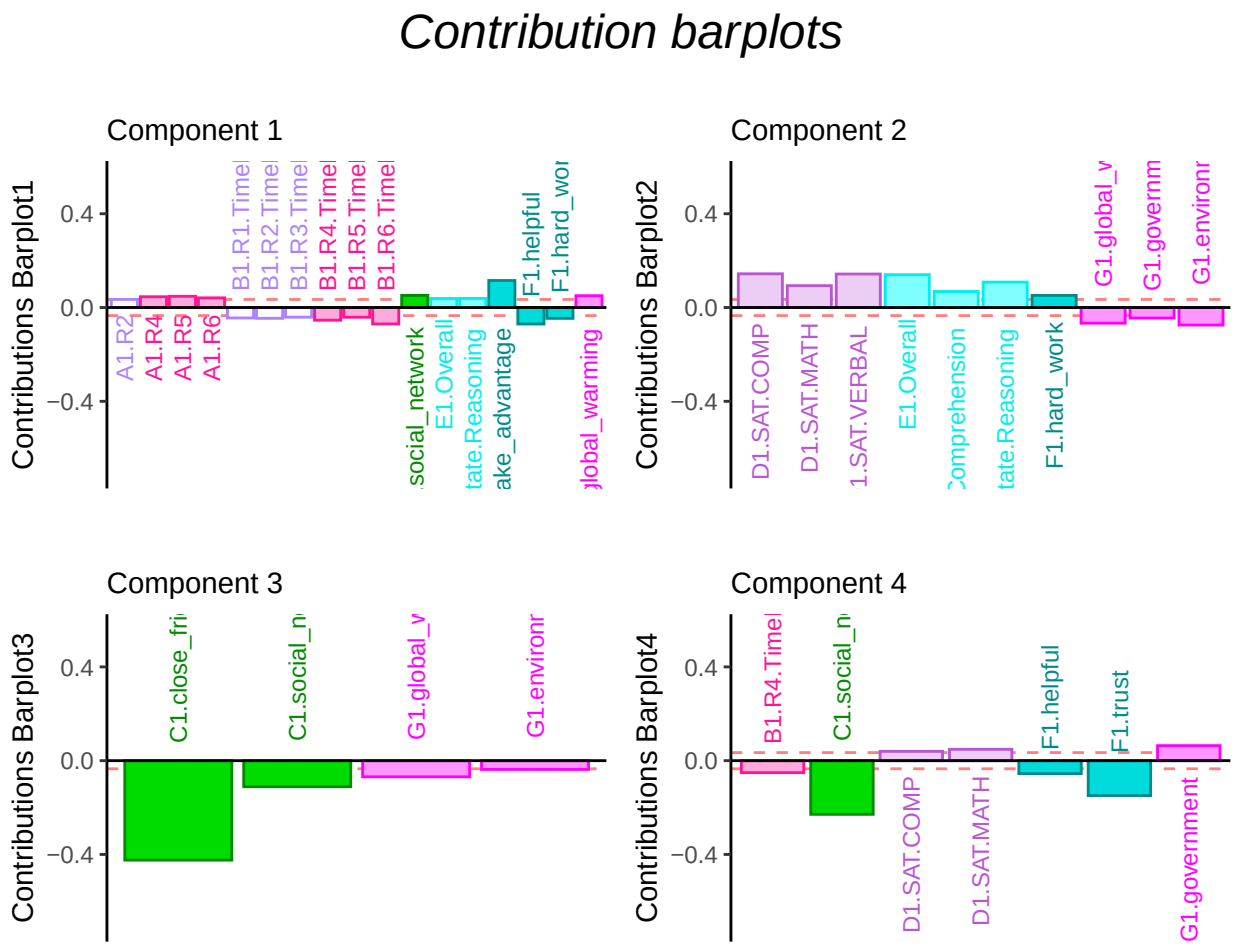
In dimension 2, attitude is in opposite direction with empathy ability

In dimension 3, friends is positive related with attitude.

```
# Contribution Plot
### plot contributions for component 1
col <- m.color.design
signed.ctrJ <- cj * sign(fj)
laDim = 1
ctrJ.1 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                    threshold = 1 / NROW(signed.ctrJ),
                    font.size = 3,
                    signifOnly = TRUE,
                    horizontal = TRUE,
                    color4bar = col, # we need hex code
                    main = 'Variable Contributions (Signed)',
```

```
                    ylab = paste0('Contributions Barplot', laDim),
                    ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
  ) + ggtitle("",subtitle = paste0('Component ', laDim))

### plot contributions for component 2
laDim =2
ctrJ.2 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                    threshold = 1 / NROW(signed.ctrJ),
                    font.size = 3,
                    color4bar = col, # we need hex code
                    signifOnly = TRUE,
                    horizontal = TRUE,
                    main = 'Variable Contributions (Signed)',
                    ylab = paste0('Contributions Barplot', laDim),
                    ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))
laDim =3
ctrJ.3 <- PrettyBarPlot2(signed.ctrJ[,laDim],
                    threshold = 1 / NROW(signed.ctrJ),
                    font.size = 3,
                    color4bar = col, # we need hex code
                    signifOnly = TRUE,
                    horizontal = TRUE,
                    main = 'Variable Contributions (Signed)',
                    ylab = paste0('Contributions Barplot', laDim),
                    ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))

laDim =4
ctrJ.4 <- PrettyBarPlot2(signed.ctrJ[,laDim],
            threshold = 1 / NROW(signed.ctrJ),
            font.size = 3,
            color4bar = col,
            signifOnly = TRUE,
            horizontal = TRUE,
            main = 'Variable Contributions (Signed)',
            ylab = paste0('Contributions Barplot', laDim),
            ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)+ ggtitle("",subtitle = paste0('Component ', laDim))



gridExtra::grid.arrange(as.grob(ctrJ.1),
                        as.grob(ctrJ.2),
                        as.grob(ctrJ.3),
                        as.grob(ctrJ.4),
  ncol=2, top = textGrob("Contribution barplots",
```

```
gp=gpar(fontsize=18,font=3)))
```



Chapter 12

The End

Finally it is the last chapter of this book.

12.1 Final Conclusion

At the end of this research story, I just want to give some take home messages to this book's readers and future myself:

The research question we asked at the beginning:

1. Will people with various levels of social intelligence and general intelligence behave differently in this collective game?
2. How will people behave when they are in negative condition?

Going through all these MSA methods, my final answers of the two questions are

- 1. When global environment situation is challenging and complex (negative), higher social and general intelligence are necessary for groups to effectively learn and respond as a team to changing circumstances to achieve sustainability goals;
- 2. People in negative situations will have motivation from help and trust to do the collective behavior.

In my previous analysis, which includes positive situation for comparison, I also found out that people in positive condition will have different motivation, like the self-attitude, to do collective behavior. It is interesting and meaningful because I can know what's is the optimized cognitive strategy when facing resource sufficiency and resource shortage. After submitting my final project for RM3, I will keep undating this book and include positive condition for comparison.

12.2 Appreciation

Thanks for reading this book. Also, I want to express the highest level of appreciation to Dr. Abid, Yu-chi, Luke and Brandon's generous support and assistance during the class. As I said in the

preface, it is the most useful and challenging class I have ever had as student. I am glad I survive and have this book now. Before this semester, I knew nothing about R programming but now I believe I am capable to do more developing job and apply it to my own research projects. I also want to appreciate myself for working hard and balancing time for PhD application, moving, research projects and writing this book. It is really a tough period for me but I am glad I survive.

I know it is not a perfect book now but I am sure that it will get better in future. At last, hope everyone enjoys this book!