Multivariate Analysis using R

Singh Richa

2018-11-27

Contents

1 Introduction

The general purpose of all statistical methods – univariate, bivariate, or multivariate is to summarize, reduce, and interpret raw data. Essentially, multivariate analysis is a tool to find patterns and relationships between several variables simultaneously. It lets us predict the effect a change in one variable will have on other variables.This gives multivariate analysis a decisive advantage over other forms of analysis.

The cookbook is all about using different techniques using multivariate Analysis. We will use different types of Multivariate techniques on the IBM-NOAttrition-Hypothetical Dataset and see how different techniques respond to the dataset and which method is best for our dataset i.e exploratory data analysis is done to find out the research answers from the data that are intended. Also, in genral we will conclude which technique is used in what all scenerios.

2 Dataset : IBM-HR-Employee-NoAttrition

There are four measurement scales (or types of data): nominal, ordinal, interval and ratio. These are simply ways to categorize different types of variables.

Nominal

"Nominal" scales could simply be called "labels." They are also known as Qualitative Variables. e.g - What is you Gender? Male and Female are the two labels.

Ordinal

With ordinal scales, it is the order of the values is what's important and significant, but the differences between each one is not really known.Ordinal scales are typically measures of non-numeric concepts like satisfaction, happiness, discomfort, etc.eg- How do you feel today? 1,2,3,4,5 where 1 is very happy and 5 is being unhappy

Qunatitative

Quantitative Variable. Variables that have are measured on a numeric or quantitative scale. Ordinal, interval and ratio scales are quantitative. A country's population, a person's shoe size, or a car's speed are all quantitative variables.

Ratio

Ratios tell us about the order, they tell us the exact value between units and they also have an absolute zero–which allows for a wide range of both descriptive and inferential statistics to be applied.

DataSet Description

The dataset consists of 1233 observations of HR-IBM Employees who didn't leave the office and 32 variables describing them.

Quantitative Variables : Sub,Age,Monthly Income,Daily Rate,Hourly Rate, MonthlyRate, DistanceFrome-Home,PercentSalaryHike,TotalWorkingYears, TrainingTimesLastYear,WorkLifeBalance,WorkLifeBalance,YearsAtCompany,Y YearsWithCurrManager

Ordinal Variables : PerformanceRating, Education, JobInvolvement,Joblevel,StockOptionLevel,EnvironmentSatisfaction, JobSatisfaction, RelationshipSatisfaction

Qualitative variable : Attrition,BusinessTravel,EducationField,Gender,JobRole,MaritalStatus,OverTime

Note: Education: 1 'Below College' 2 'College' 3 'Bachelor' 4 'Master' 5 'Doctor' EnvironmentSatisfaction: 1 'Low' 2 'Medium' 3 'High' 4 'Very High' JobInvolvement: 1 'Low' 2 'Medium' 3 'High' 4 'Very High' JobSatisfaction: 1 'Low' 2 'Medium' 3 'High' 4 'Very High' PerformanceRating: 1 'Low' 2 'Good' 3 'Excellent' 4 'Outstanding' RelationshipSatisfaction: 1 'Low' 2 'Medium' 3 'High' 4 'Very High' WorkLifeBalance: 1 'Bad' 2 'Good' 3 'Better' 4 'Best'

Research Question

How do all 1233 HR-IBM Employees differ on the variables like Gender type vs Department ? Job level vs MonthlyIncome ? etc.Explore important questions such as 'show me a breakdown of distance from home by job role' or 'compare average monthly income by education'. This is a fictional data set created by IBM data scientists.

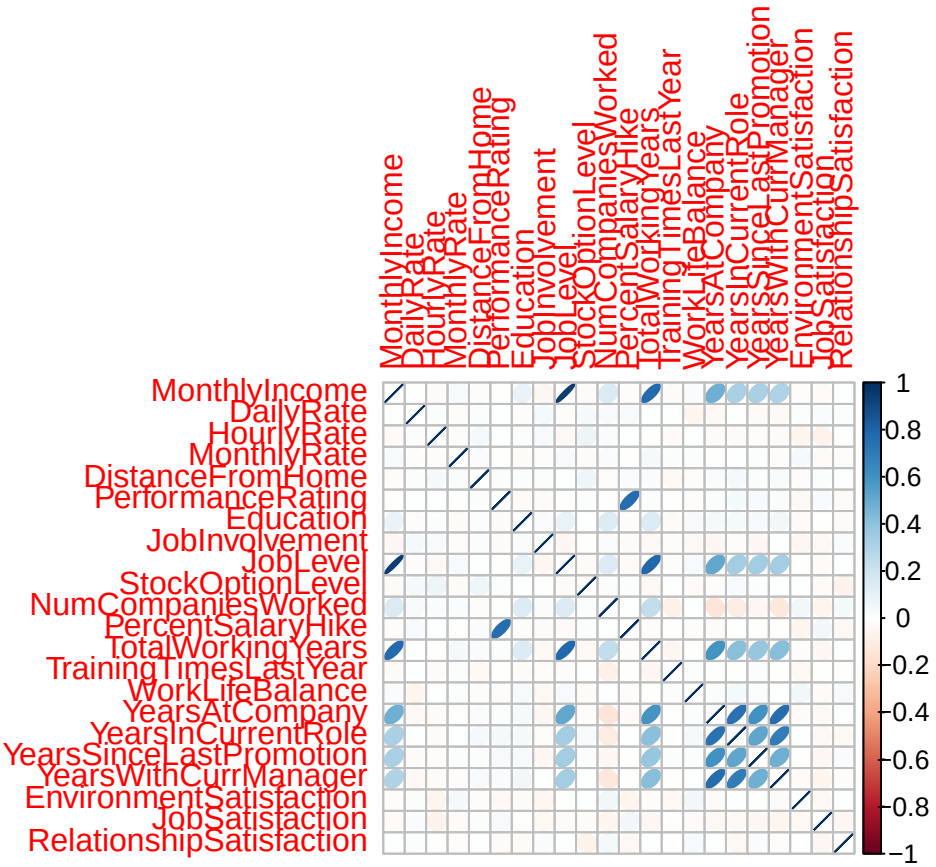
Correlation Plot

In statistics, the correlation coefficient r measures the strength and direction of a linear relationship between two variables on a scatterplot. The value of r is always between +1 and –1.The measure calculates the strength of the relationship between the relative movements of the two variables.

```
datar <- my_data1[,2:23]
datae <- my_data[,5]
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
cor.my_data <- cor(datar)
corrplot(cor.my_data, method = "ellipse")
```




```
options(knitr.duplicate.label = 'allow')
```

3 PCA

Principal Component Analysis is the analysis of data to identify patterns and finding patterns to reduce the dimensions of the dataset with minimal loss of information. Here, our desired outcome of the principal component analysis is to project a feature space (our dataset consisting of n d-dimensional samples) onto a smaller subspace that represents our data well.PCA gives one map for the rows (called factor scores), and one map for the columns (called loadings).These 2 maps are related, because they both are described by the same components. However, these 2 maps project different kinds of information onto the components, and so they are *interpreted differently*. Factor scores are the coordinates of the row observations. They are interpreted by the distances between them, and their distance from the origin. Loadings describe the column variables. Loadings are interpreted by the angle between them, and their distance from the origin.

`The distance from the origin is important in both maps, because squared distance from the mean is`

We will use the InPosition library(epPCA package) to do PCA: Because each variable is measured on different units, I choose to center and scale the columns. The rows are color-coded by the DESIGN variable, state.division. * `center = TRUE`: substracts the mean from each column * `scale = TRUE`: after centering (or not), scales each column (see the help for different scaling options) * `DESIGN`: colors the observations (rows)

```
library(InPosition)
resPCA <- epPCA(DATA = datar,
                scale = 'SS1', # Make to use 'SS1' rather than TRUE
                DESIGN = datae,
                graphs =  FALSE # TRUE first pass only
```

```
)
testVari    <- data4PCCAR::epVari(resPCA)
resPCA.inf <- InPosition::epPCA.inference.battery(DATA = datar,
                                                  scale = 'SS1', # Make sure to use 'SS1' rather than T
                                                  DESIGN = datae,
                                                  graphs =  FALSE # TRUE first pass only
)
```

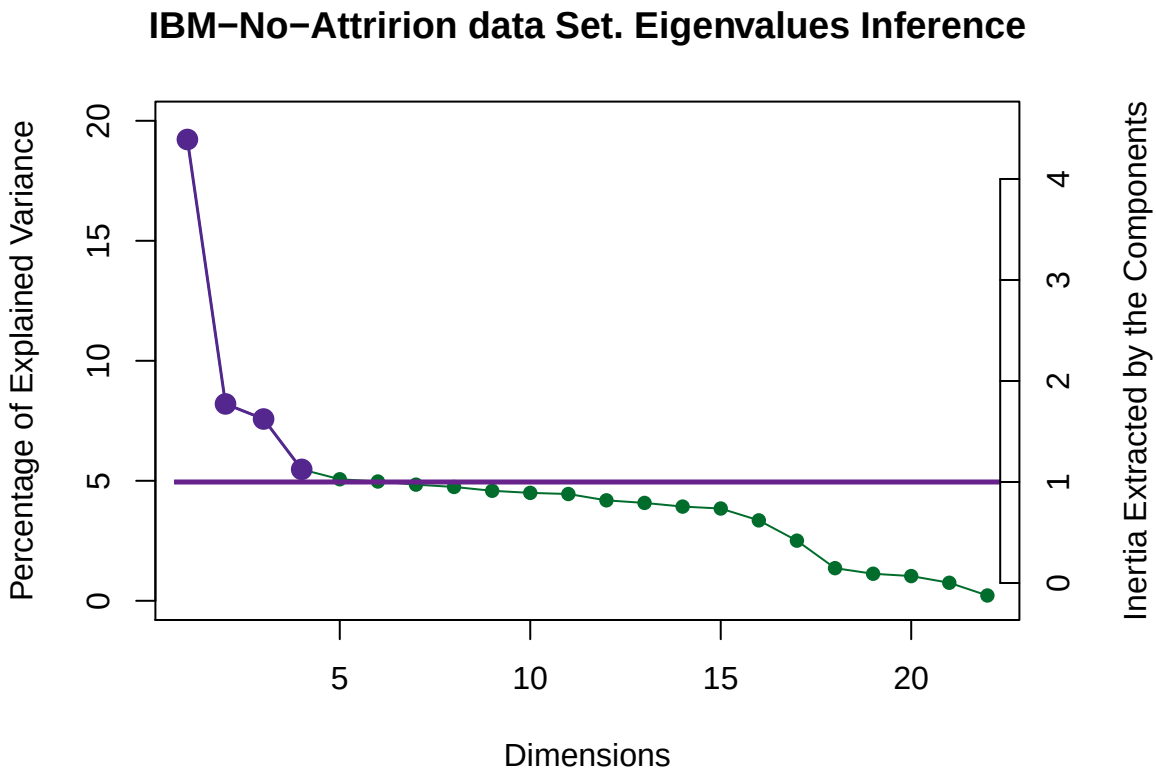
```
## [1] "It is estimated that your iterations will take 0.05 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note of the
## ============================================================================
```

3.1 Scree Plot

A Scree Plot is a simple line segment plot that shows the fraction of total variance in the data as explained or represented by each PC.(`In the PCA literature, the plot is called a 'Scree' Plot because it often looks like a 'scree' slope, where rocks have fallen down and accumulated on the side of a mountain.`)The scree plot shows the eigenvalues, the amount of information on each component. The number of components (the dimensionality of the factor space) is min(nrow(DATA), ncol(DATA)) minus 1. Here, min(1233,31)-1 give 30 components. The scree plot is used to determine how many of the components should be interpreted.

- `plot` draws the line that connects all data points by `type = "l"`
- The first `points` function draws round purple dots.
- The second `points` function draws black circles around the dots (just to make it prettier).

```
PlotScree(ev = resPCA$ExPosition.Data$eigs,
          p.ev =  resPCA.inf$Inference.Data$components$p.vals,
          title = 'IBM-No-Attririon data Set. Eigenvalues Inference',
          plotKaiser = TRUE
)
```



The top 4 compnents can be analyzed.

3.2 Factor Scores

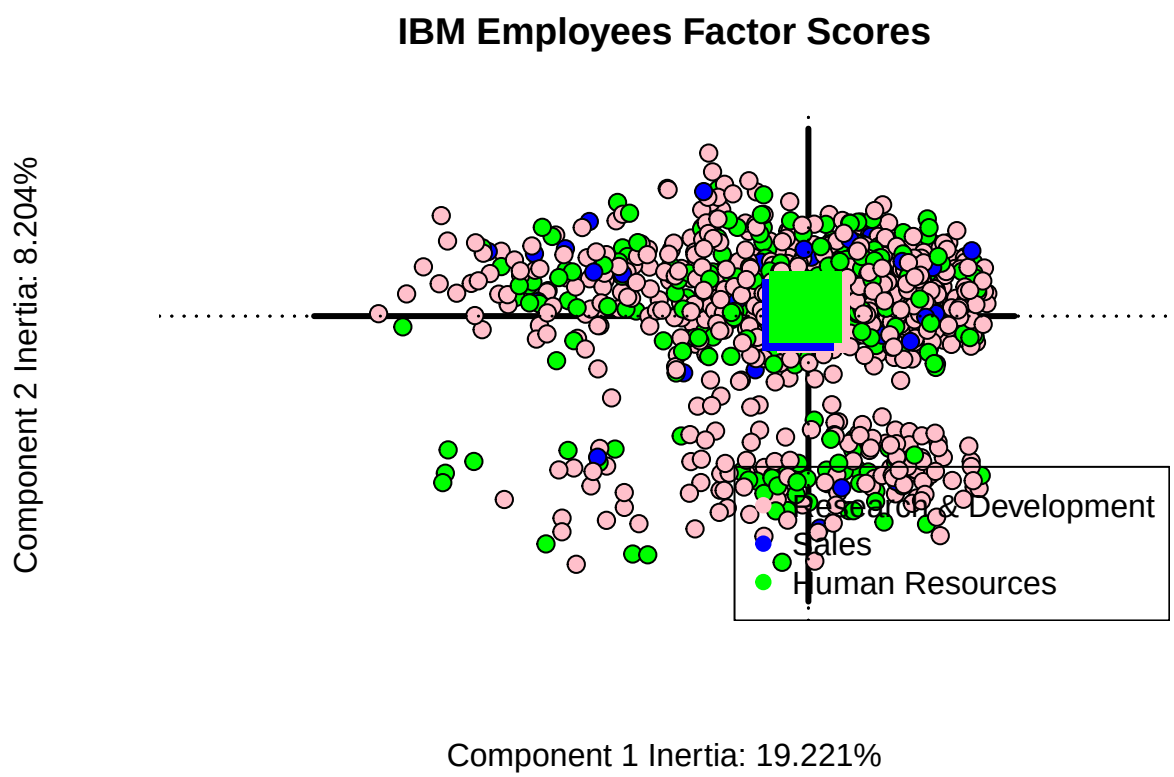
Department Variable

Component 1 Vs Component 2 Factor scores are the coordinates of the 1233 rows of the employees on the components. The distances between them show which all employees are the most similar ones. Factor scores (employees) can be color-coded to help interpret the components.

- `prettyPlot` helps plot the factor scores. In order to print the result in an Rmd, `dev.new` needs to be `FALSE`.

```
#FACTOR SCORES
OriginAsDesignMat <- makeNominalData(as.matrix(my_data$Department))
Fi2plot <- resPCA$ExPosition.Data$fi
fi_all.sum <- t(Fi2plot) %*% OriginAsDesignMat
fi_all.count <- colSums(OriginAsDesignMat)
fi_all.mean <- t(fi_all.sum)/fi_all.count

#pretty plot between component1 and 2
name_the_plot <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                            dev.new=FALSE,
                            main = "IBM Employees Factor Scores",
                            x_axis = 1, y_axis = 2,
                            contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                            display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN$rows$Region$color_
```

```
                            display_names = FALSE,
                            xlab = paste0("Component 1 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
                            ylab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
legend(x="bottomright", pch = 19, legend = DESIGN$rows$Region$labels, col=DESIGN$rows$Region$color_grou
prettyPlot(fi_all.mean,
           col = DESIGN$rows$Region$color_groups,
           display_names = FALSE,
           pch = 15,
           cex = 5.0,
           dev.new = FALSE,
           new.plot = FALSE)
```

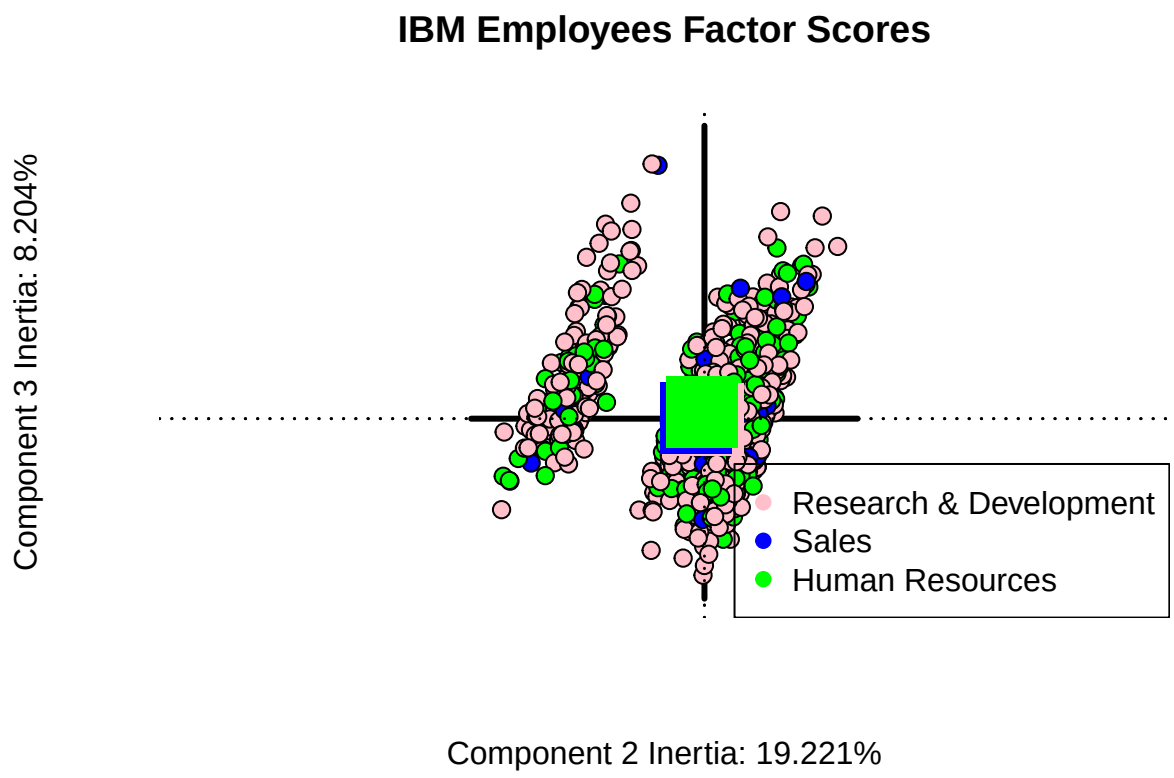


- Component 1: May be Sales vs Human Resources
- Component 2: No significant inference could be observed

Component 2 Vs Component 3

```
name_the_plot <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                            dev.new=FALSE,
                            main = "IBM Employees Factor Scores",
                            x_axis = 2, y_axis = 3,
                            contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                            display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN$rows$Region$color_
                            display_names = FALSE,
                            xlab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
```

```
                            ylab = paste0("Component 3 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
legend(x="bottomright", pch = 19, legend = DESIGN$rows$Region$labels, col=DESIGN$rows$Region$color_grou
prettyPlot(fi_all.mean,
           col = DESIGN$rows$Region$color_groups,
           display_names = FALSE,
           pch = 15,
           cex = 5.0,
           dev.new = FALSE,
           new.plot = FALSE)
```

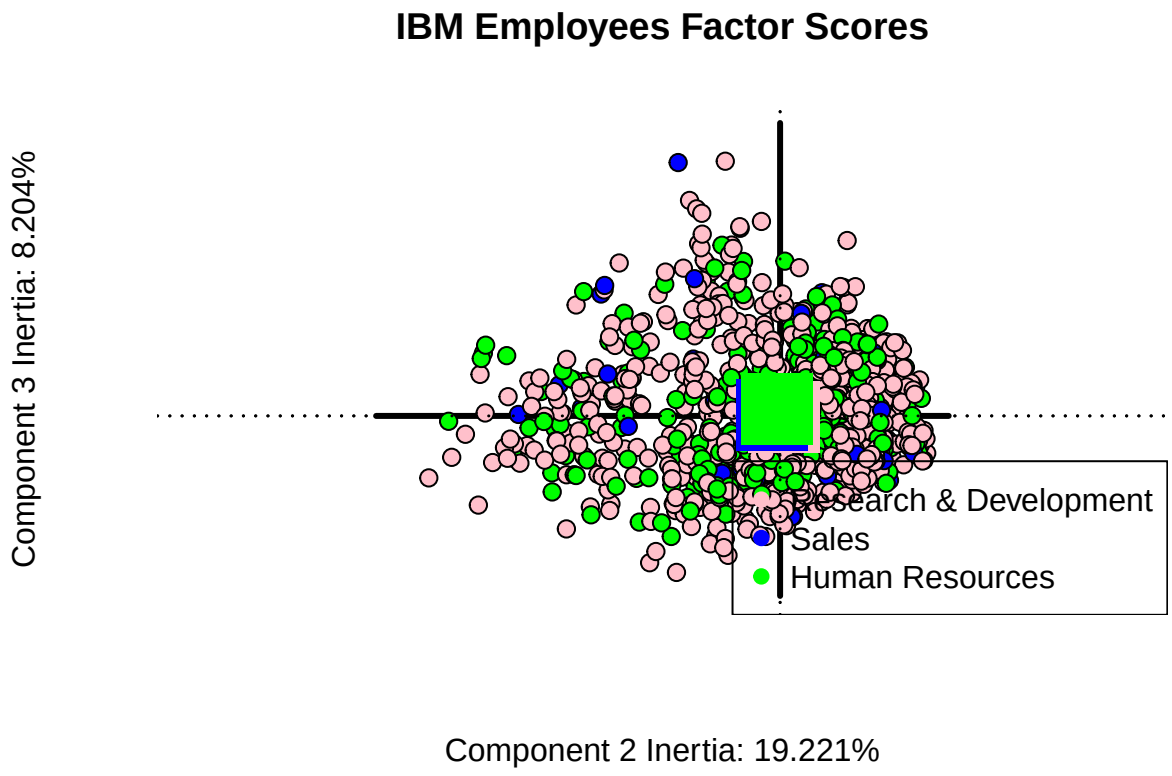


- Component 2: Definitely Sales vs Human Resources
- Component 3: Maybe divides Sales & Human Rescources VS Research and Development

Component 1 Vs Component 3

```
name_the_plot <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                            dev.new=FALSE,
                            main = "IBM Employees Factor Scores",
                            x_axis = 1, y_axis = 3,
                            contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                            display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN$rows$Region$color_
                            display_names = FALSE,
                            xlab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
                            ylab = paste0("Component 3 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
```

```
legend(x="bottomright", pch = 19, legend = DESIGN$rows$Region$labels, col=DESIGN$rows$Region$color_grou
prettyPlot(fi_all.mean,
           col = DESIGN$rows$Region$color_groups,
           display_names = FALSE,
           pch = 15,
           cex = 5.0,
           dev.new = FALSE,
           new.plot = FALSE)
```



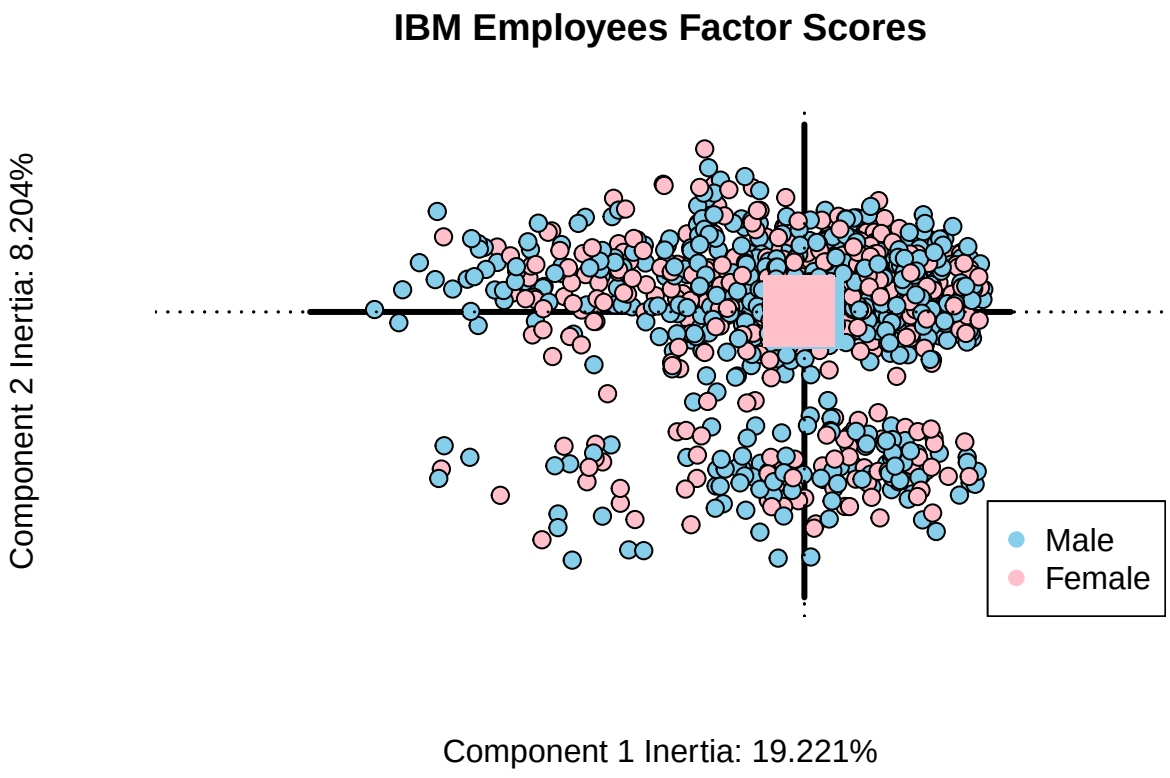
No significant observation can be concluded both from Component 1 and Component 3

Gender Variable

Component 1 Vs Component 2

```
name_the_plot1 <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                             dev.new=FALSE,
                             main = "IBM Employees Factor Scores",
                             x_axis = 1, y_axis = 2,
                             contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                             display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN1$rows$Region$color
                             display_names = FALSE,
                             xlab = paste0("Component 1 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
                             ylab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
legend(x="bottomright", pch = 19, legend = DESIGN1$rows$Region$labels, col=DESIGN1$rows$Region$color_gr
prettyPlot(fi_all.mean1,
           col = DESIGN1$rows$Region$color_groups,
```

```
            display_names = FALSE,
            pch = 15,
            cex = 5.0,
            dev.new = FALSE,
            new.plot = FALSE)
```

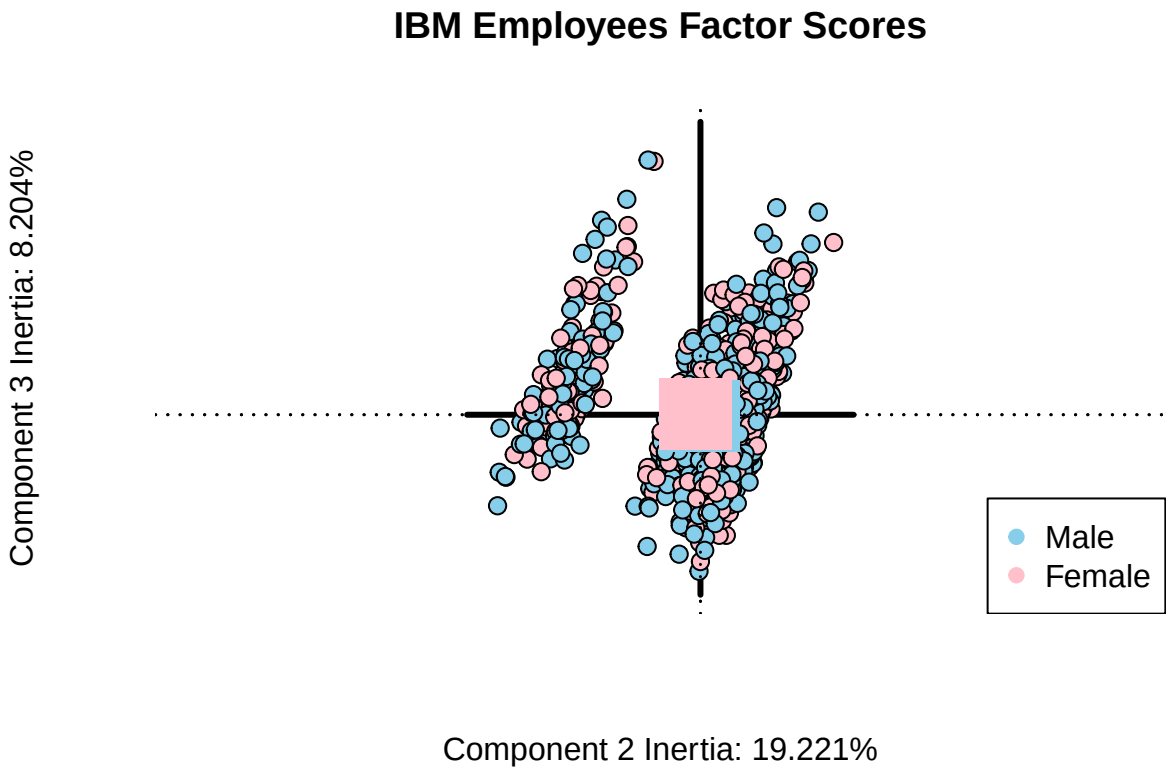


- Both the means lie almost on the origin so can't make any significant inference for both the Components.

Component 2 Vs Component 3

```
name_the_plot2 <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                             dev.new=FALSE,
                             main = "IBM Employees Factor Scores",
                             x_axis = 2, y_axis = 3,
                             contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                             display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN1$rows$Region$color
                             display_names = FALSE,
                             xlab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
                             ylab = paste0("Component 3 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
legend(x="bottomright", pch = 19, legend = DESIGN1$rows$Region$labels, col=DESIGN1$rows$Region$color_gr
prettyPlot(fi_all.mean1,
            col = DESIGN1$rows$Region$color_groups,
            display_names = FALSE,
            pch = 15,
            cex = 5.0,
            dev.new = FALSE,
```

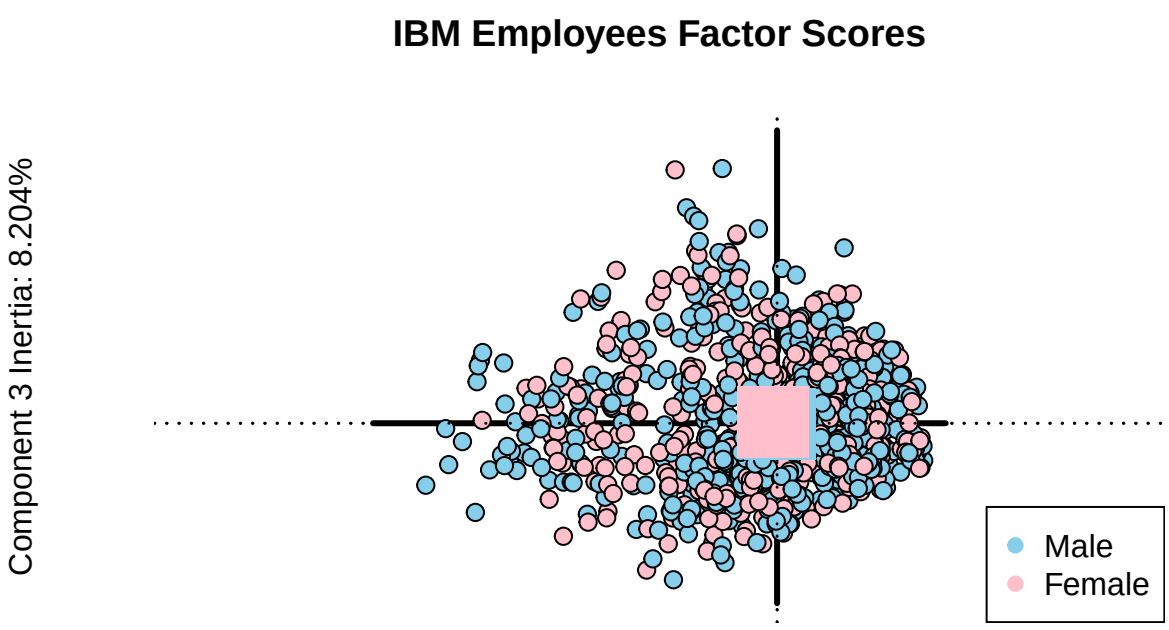
```
            new.plot = FALSE)
```



- Component 2: Maybe we can say that Component 2 divides Female vs Male
- Component 3: No significant inference

Component 1 Vs Component 3

```
name_the_plot3 <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fi,
                             dev.new=FALSE,
                             main = "IBM Employees Factor Scores",
                             x_axis = 1, y_axis = 3,
                             contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$ci,
                             display_points = TRUE, pch = 21, cex = 1.2, col = DESIGN1$rows$Region$color
                             display_names = FALSE,
                             xlab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[1],3)
                             ylab = paste0("Component 3 Inertia: ", round(resPCA$ExPosition.Data$t[2],3)
)
legend(x="bottomright", pch = 19, legend = DESIGN1$rows$Region$labels, col=DESIGN1$rows$Region$color_gr
prettyPlot(fi_all.mean1,
           col = DESIGN1$rows$Region$color_groups,
           display_names = FALSE,
           pch = 15,
           cex = 5.0,
           dev.new = FALSE,
           new.plot = FALSE)
```



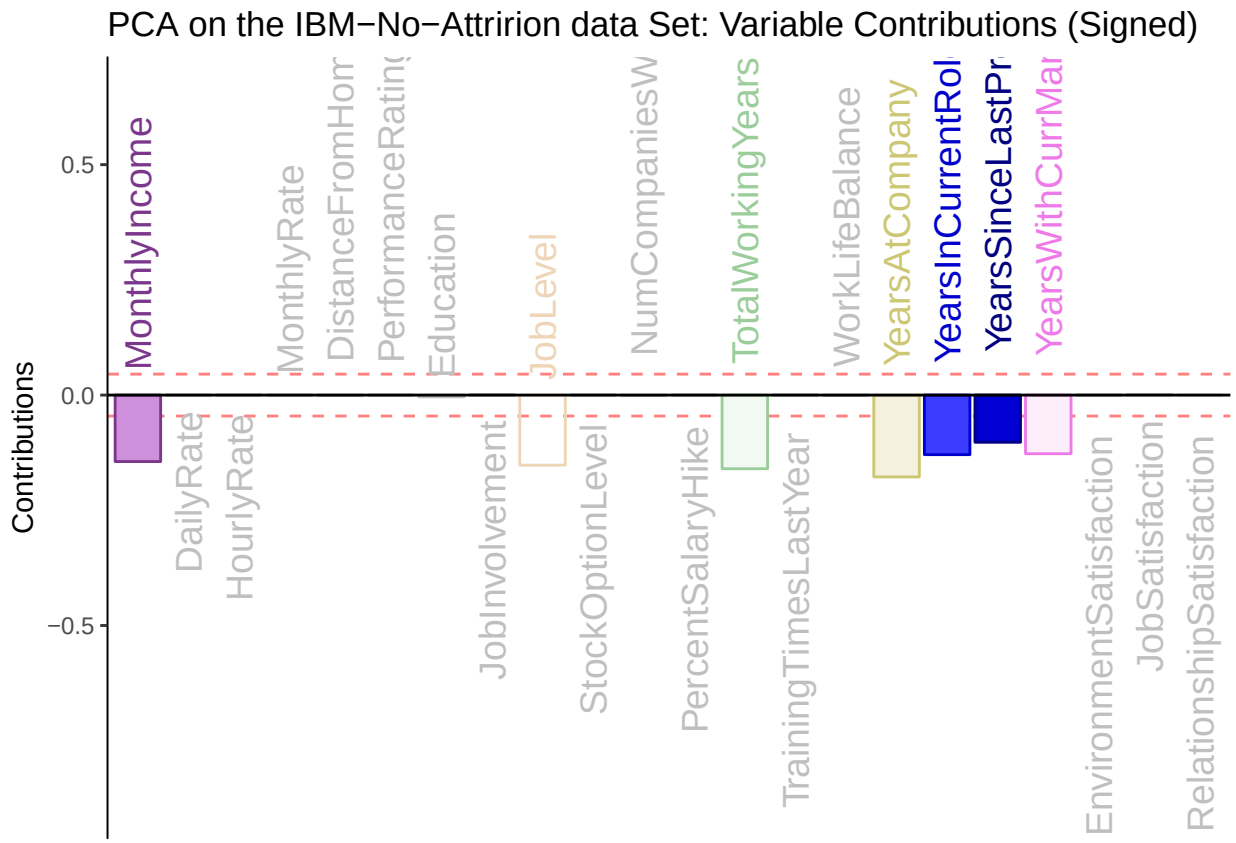
Loadings

```
col4J.ibm <- prettyGraphsColorSelection(NCOL(datar))
baseMap.j <- PTCA4CATA::createFactorMap(resPCA$ExPosition.Data$fj,col.points = col4J.ibm,
                                        col.labels = col4J.ibm ,
                                        alpha.points =  .3)
# arrows
zeArrows <- addArrows(resPCA$ExPosition.Data$fj , col = col4J.ibm)
# A graph for the J-set
b000.aggMap.j <- baseMap.j$zeMap_background + # background layer
  baseMap.j$zeMap_dots + baseMap.j$zeMap_text +  # dots & labels
  label4Map + zeArrows
# We print this Map with the following code
dev.new()
print(b000.aggMap.j)
```

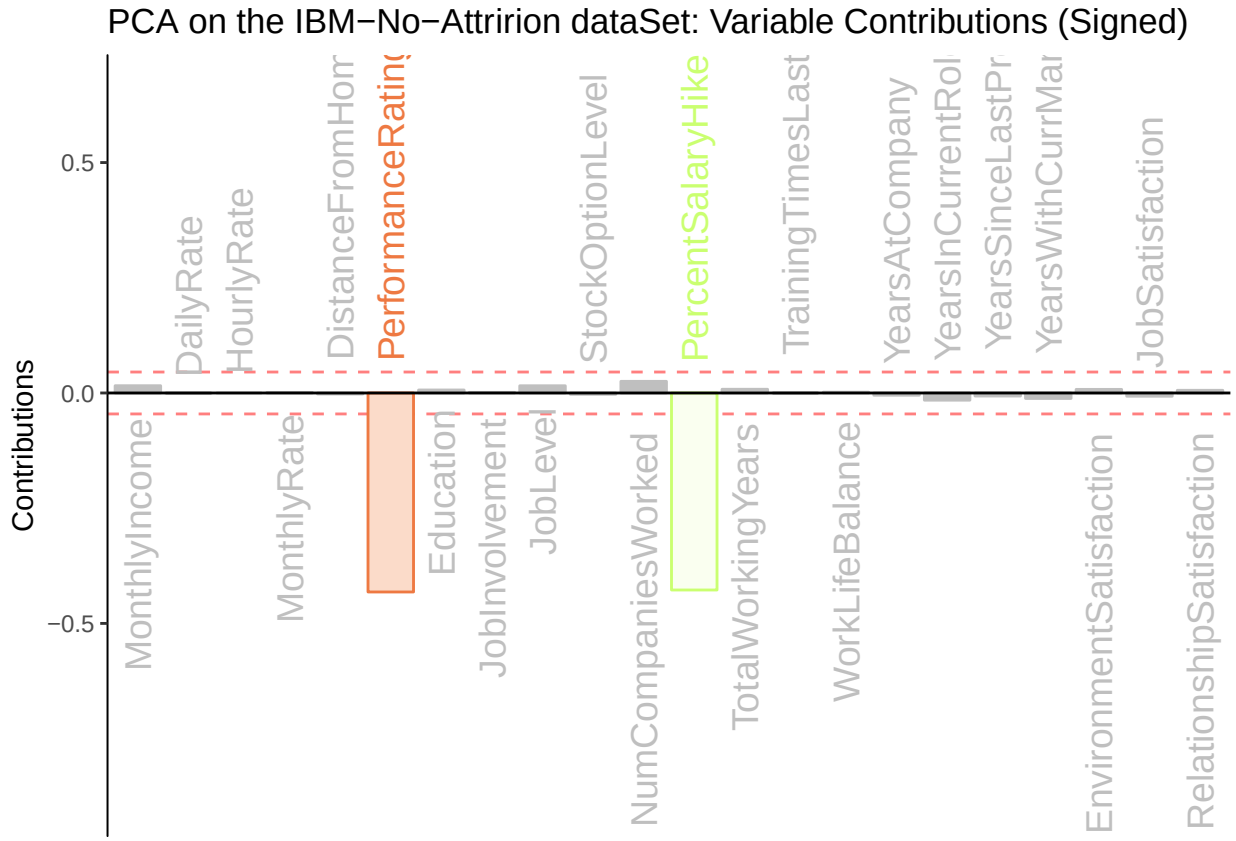
3.3 Contribution

```
signed.ctrJ <- resPCA$ExPosition.Data$cj * sign(resPCA$ExPosition.Data$fj)
b003.ctrJ.s.1 <- PrettyBarPlot2(signed.ctrJ[,1],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'PCA on the IBM-No-Attririon data Set: Variable Contributions (S
                                ylab = 'Contributions',
```

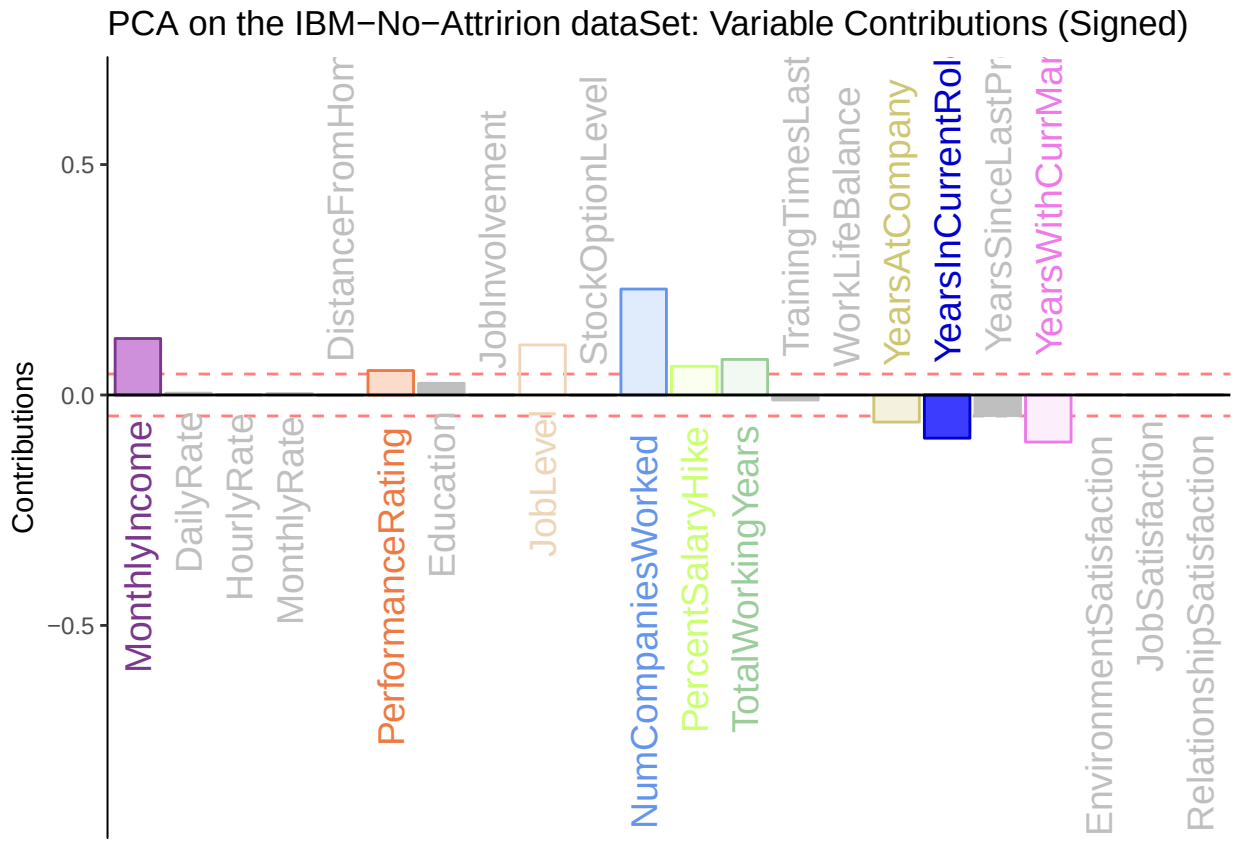
```
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b003.ctrJ.s.1)
```




```
b004.ctrJ.s.2 <- PrettyBarPlot2(signed.ctrJ[,2],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'PCA on the IBM-No-Attririon dataSet: Variable Contributions (Sig
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b004.ctrJ.s.2)
```



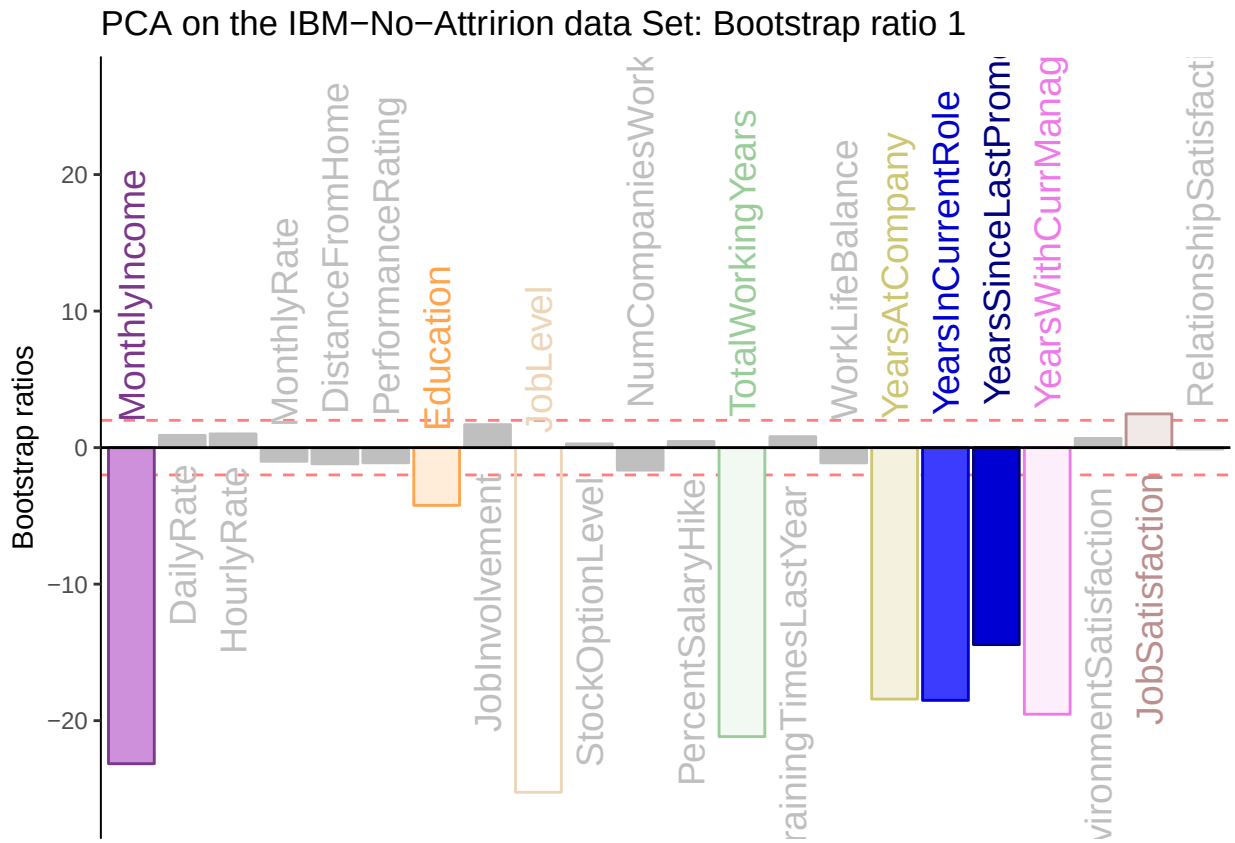

```
b004.ctrJ.s.3 <- PrettyBarPlot2(signed.ctrJ[,3],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'PCA on the IBM-No-Attririon dataSet: Variable Contributions (Si
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b004.ctrJ.s.3)
```



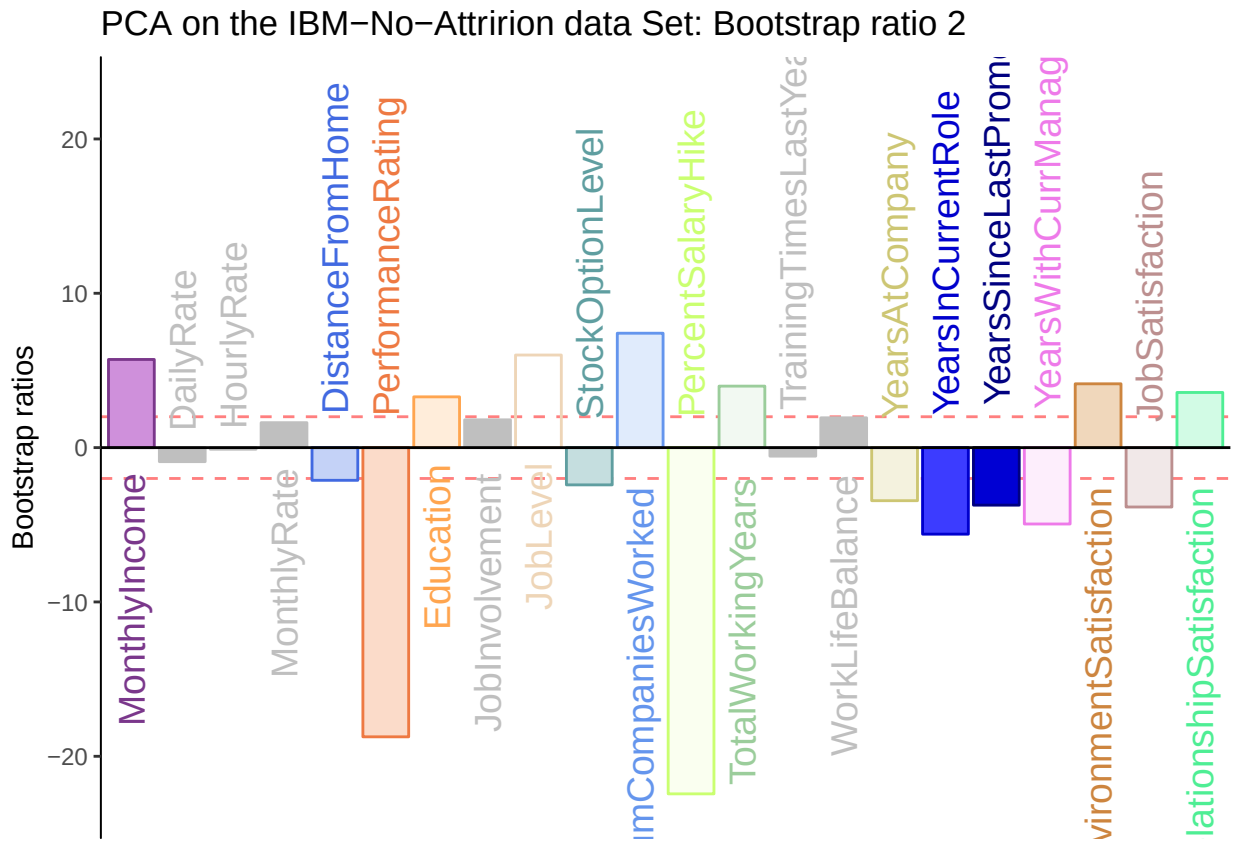
From the contribution graph it is clear that on the first component Monthly Income, JobLevel , TotalWorkingYears , YearsAtCompany , YearsInCurrentRole , YearsSinceLastPromotion is evident . For the second component Performance Rating and PerformanceSalaryHike are major contributors. For the third component MonthlyIncome, PerformanceRating, NumCompaniesWorked , YearsINCurrentRole are major contributors.

3.4 Bootstrap Ratios

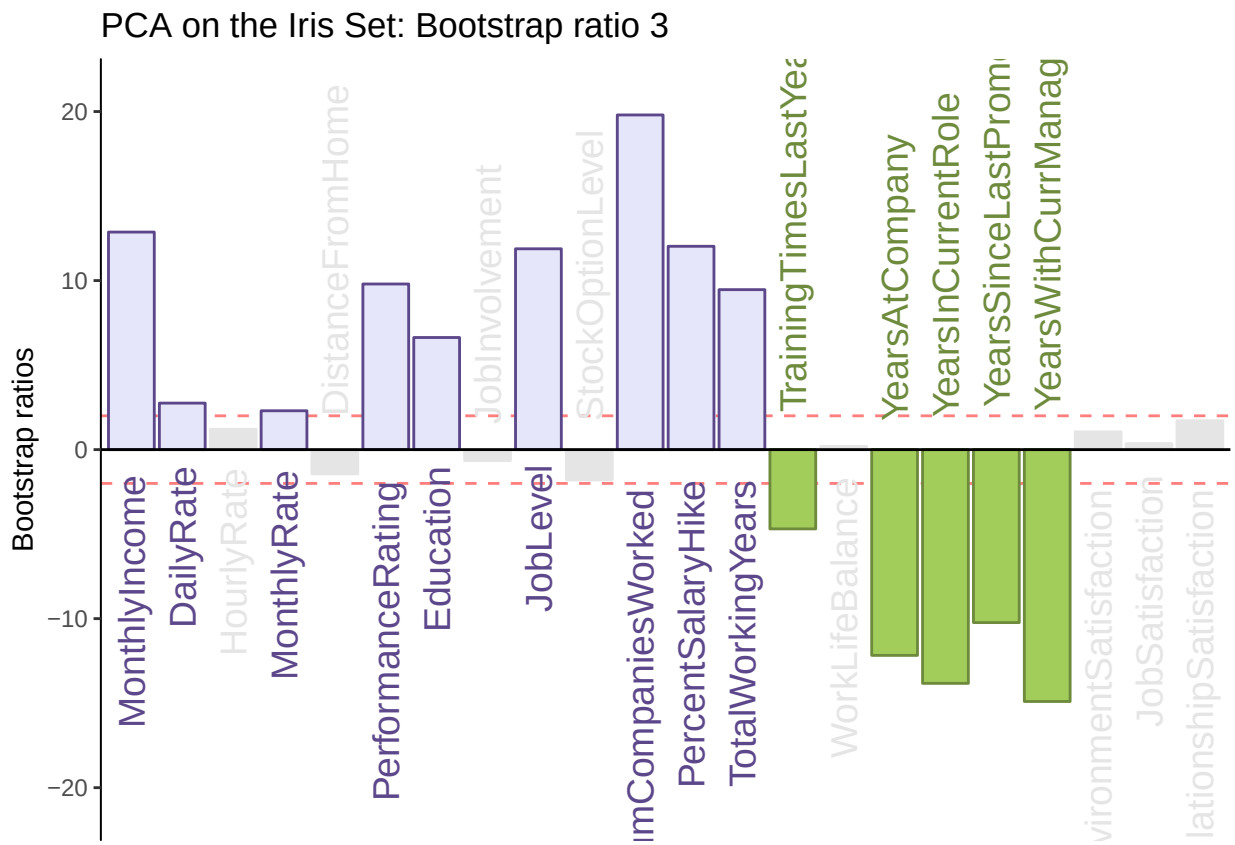
```
BR <- resPCA.inf$Inference.Data$fj.boots$tests$boot.ratios
laDim = 1
ba001.BR1 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                            color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0( 'PCA on the IBM-No-Attririon data Set: Bootstrap ratio ',laDi
                            ylab = 'Bootstrap ratios'
                            #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR1)
```




```
#
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                            color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0(
                              'PCA on the IBM-No-Attririon data Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
)
print(ba002.BR2)
```



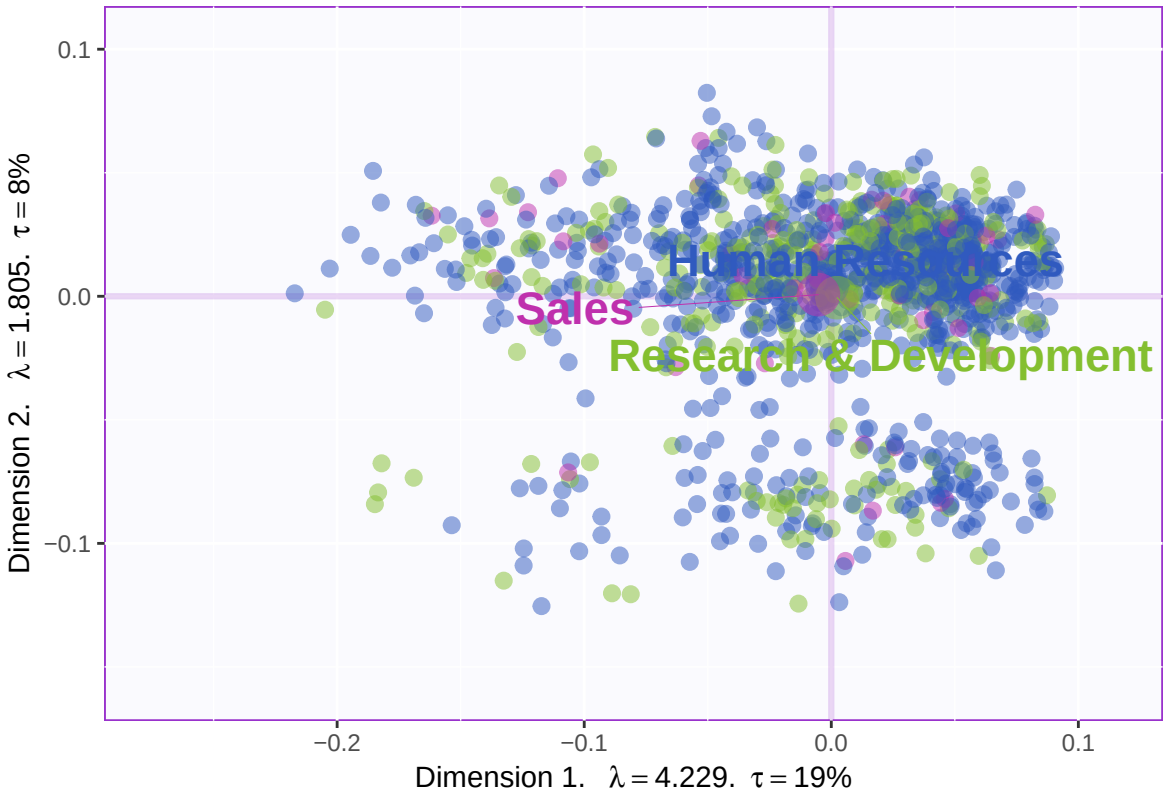

```
laDim = 3
ba002.BR3 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                            main = paste0(
                              'PCA on the Iris Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
)
print(ba002.BR3)
```



C.I of the mean

```
# Bootstrap for CI:
BootCube.Gr <- PTCA4CATA::Boot4Mean(resPCA$ExPosition.Data$fi,
                                    design = datae,
                                    niter = 100,
                                    suppressProgressBar = TRUE)
#_____________________________________________________________________
# Bootstrap ratios ----
bootRatios.Gr <- boot.ratio.test(BootCube.Gr$BootCube)
#**********************************************************************
# eigenvalues: MonteCarlo Approach ----
#
random.eigen <- data4PCCAR::monteCarlo.eigen(X = datar, nIter = 100)
#
# eigenvalues: Bootstrap approach
#
bootstrap.eigen <- data4PCCAR::boot.eigen(datar, nIter = 100)
# Mean Map
#  create the map for the means
#  get the means by groups

dataMeans <- PTCA4CATA::getMeans(resPCA$ExPosition.Data$fi, datae)
# a vector of color for the means
col4data <- resPCA$Plotting.Data$fi.col
col4Means <- unique(col4data)
# the map
```

```
MapGroup <- PTCA4CATA::createFactorMap(dataMeans,
                                       # use the constraint from the main map
                                       constraints = my_data1.Imap$constraints,
                                       col.points = col4Means,
                                       cex = 7,   # size of the dot (bigger)
                                       col.labels = col4Means,
                                       text.cex = 6)
# The map with observations and group means
a003.Map.I.withMeans <- a002.Map.I +
  MapGroup$zeMap_dots + MapGroup$zeMap_text
print(a003.Map.I.withMeans)
```

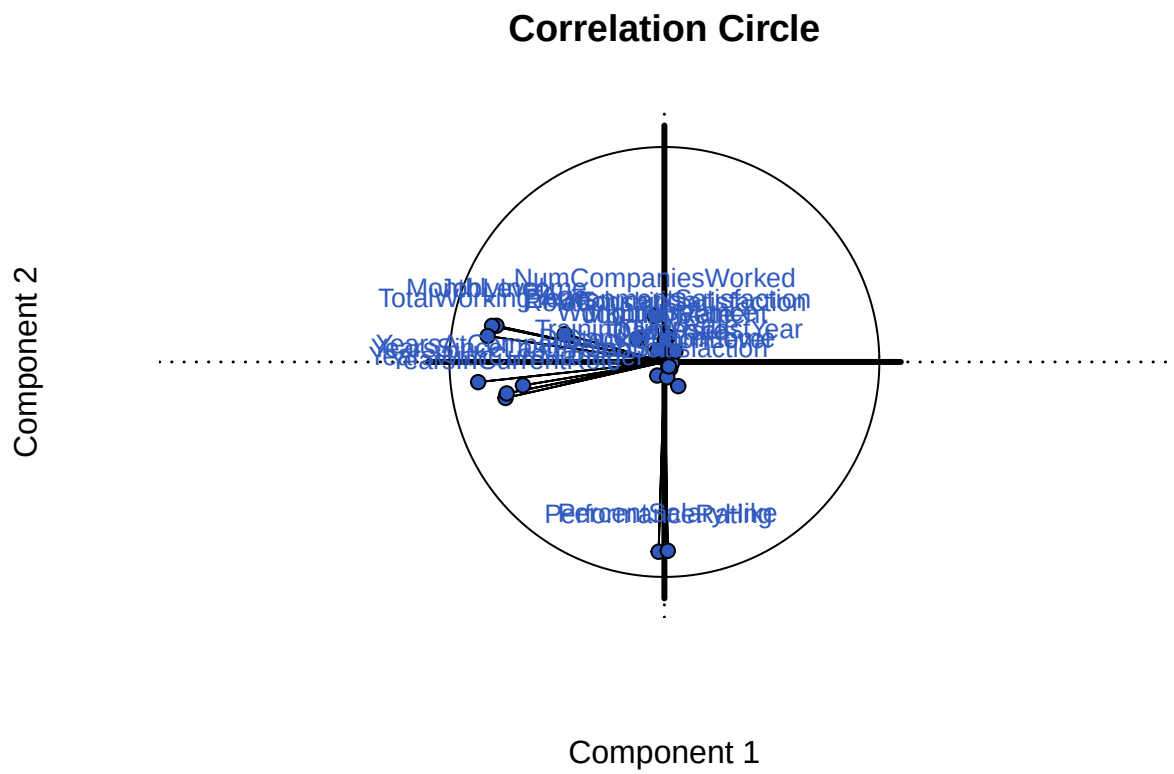



```
#_____________________________________________________________________
# Create the ellipses
# Bootstrapped CI ----
#_____________________________________________________________________
# Create Confidence Interval Plots
# use function MakeCIEllipses from package PTCA4CATA
GraphElli <- PTCA4CATA::MakeCIEllipses(BootCube.Gr$BootCube[,1:2,],
                                       names.of.factors = c("Dimension 1","Dimension 2"),
                                       col = col4Means,
                                       p.level = .95
)
#_____________________________________________________________________
# create the I-map with Observations, means and confidence intervals
#
```

```
a004.Map.I.withCI <-  a002.Map.I + MapGroup$zeMap_text +  GraphElli
#_____________________________________________________________________
# plot it!
dev.new()
print(a004.Map.I.withCI)
```

Correlation Circle

```
correlationCircle <- correlationPlotter(data_matrix = my_data1 , factor_scores =resPCA$ExPosition.Data$
```



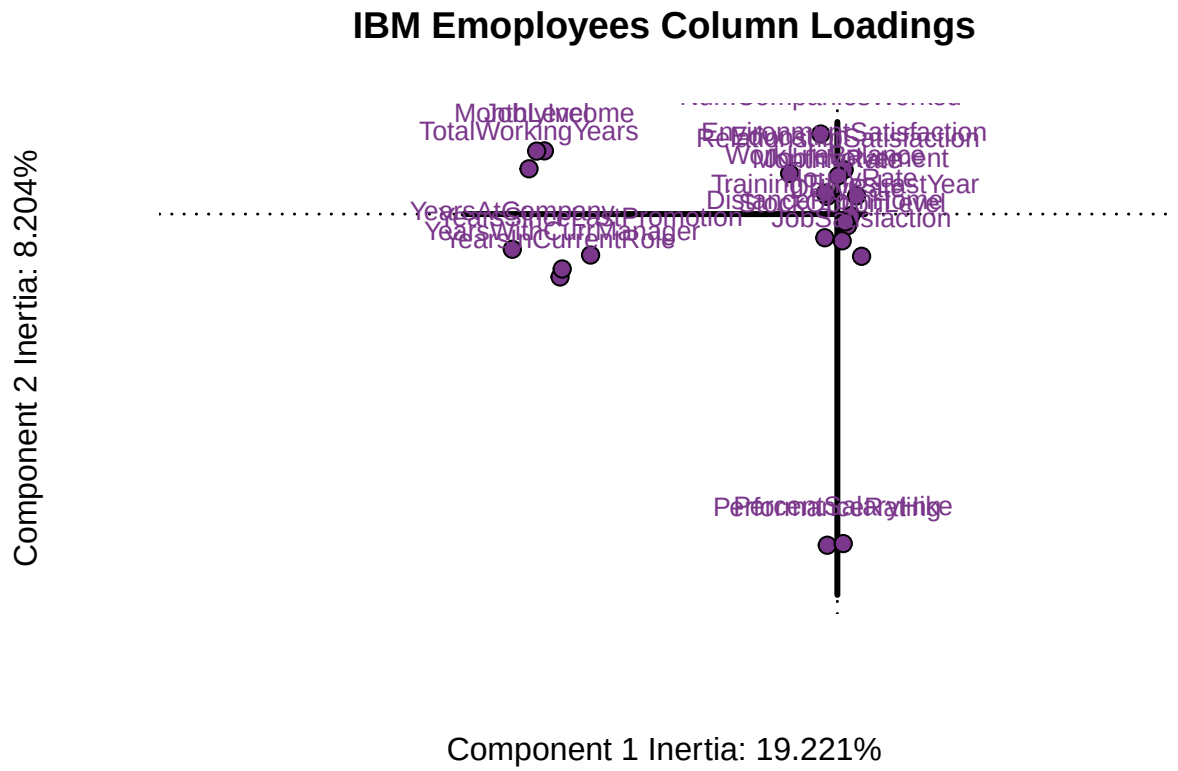
3.5 Loadings

Component 1 Vs Component 2

Loadings describe the similarity (angular distance) between the variables. Loadings show how the input variables relate to each other. Loadings also show which variables are important for (which components load on) a certain component.

```
name_another_plot <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fj,
                                dev.new=FALSE,
                                main = "IBM Emoployees Column Loadings",
                                x_axis = 1, y_axis = 2,
                                contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$cj,
                                display_points = TRUE, pch = 21, cex = 1.2, col = resPCA$Plotting.Data$
                                display_names = TRUE,
                                xlab = paste0("Component 1 Inertia: ", round(resPCA$ExPosition.Data$t[1
```

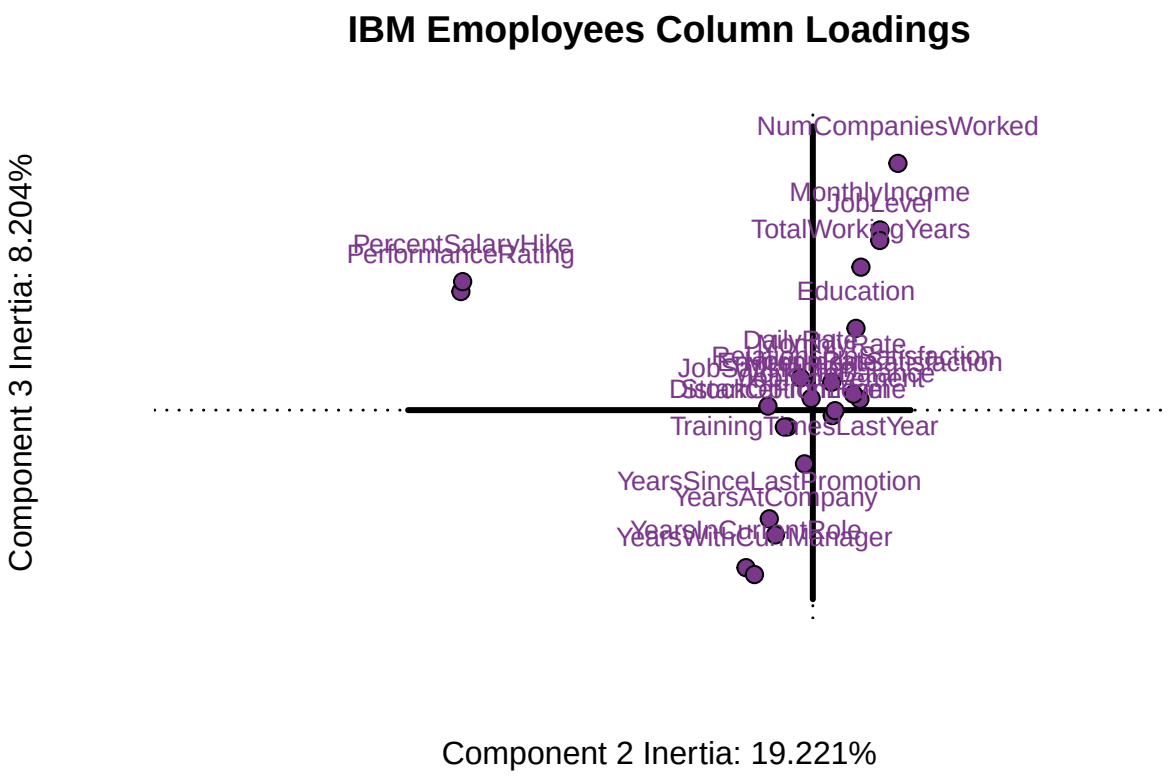
```
                                ylab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[2]
)
```



- Component 1: YearsatCompany & JobLevel & MonthlyIncome
- Component 2: Performance Rating VS NumCompaniesWorked

Component 2 Vs Component 3

```
name_another_plot <- prettyPlot(data_matrix = resPCA$ExPosition.Data$fj,
                                dev.new=FALSE,
                                main = "IBM Emoployees Column Loadings",
                                x_axis = 2, y_axis = 3,
                                contributionCircles = FALSE, contributions = resPCA$ExPosition.Data$cj,
                                display_points = TRUE, pch = 21, cex = 1.2, col = resPCA$Plotting.Data$
                                display_names = TRUE,
                                xlab = paste0("Component 2 Inertia: ", round(resPCA$ExPosition.Data$t[1]
                                ylab = paste0("Component 3 Inertia: ", round(resPCA$ExPosition.Data$t[2]
)
```



- Component 2: YearsatCompany VS NumCompaniesWorked
- Component 3: Performance Rating & PercentSalaryHike

3.6 Summary

When we interpret the factor scores and loadings together, the PCA revealed:

- Component 1: Usually the Research and Development people have more work-experience & Monthly Income in comparison to other department type
- Component 2: Human Resources people have high Performance Rating
- Component 3: Males have worked in more number of Companies

```
options(knitr.duplicate.label = 'allow')
```

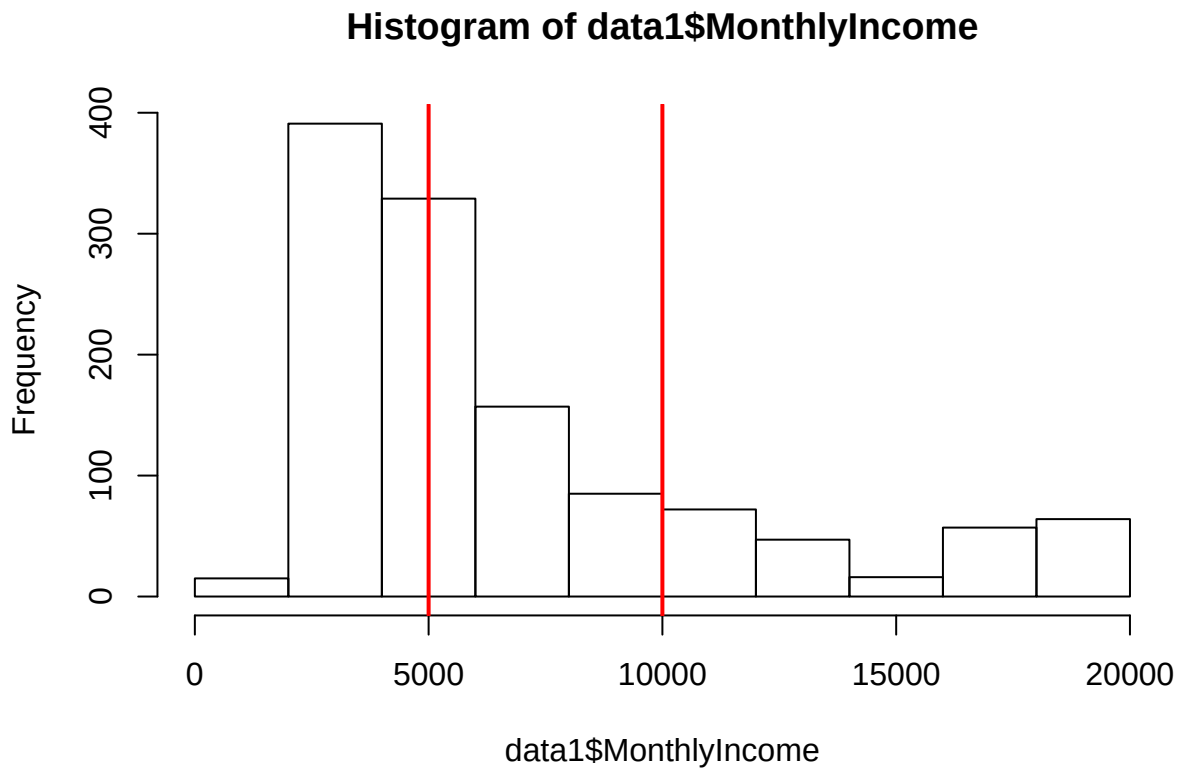
4 Multiple Correspondance Analysis

Multiple correspondence analysis (MCA) is an extension of correspondence analysis (CA) which allows one to analyze the pattern of relationships of several categorical dependent variables. MCA is used to analyze a set of observations described by a set of nominal variables. Each nominal variable comprises several levels, and each of these levels is coded as a binary variable. MCA can also accommodate quantitative variables by recoding them as bins.Because MCA has been (re)discovered many times, equivalent methods are known under several different names such as optimal scaling, optimal or appropriate scoring, dual scaling, homogeneity analysis, scalogram analysis, and quantification method. Technically MCA is obtained by

using a standard correspondence analysis on an indicator matrix (a matrix whose entries are 0 or 1). The percentages of explained variance need to be corrected, and the correspondence analysis interpretation of inter-point distances needs to be adapted.

4.1 Binning the quantitative data

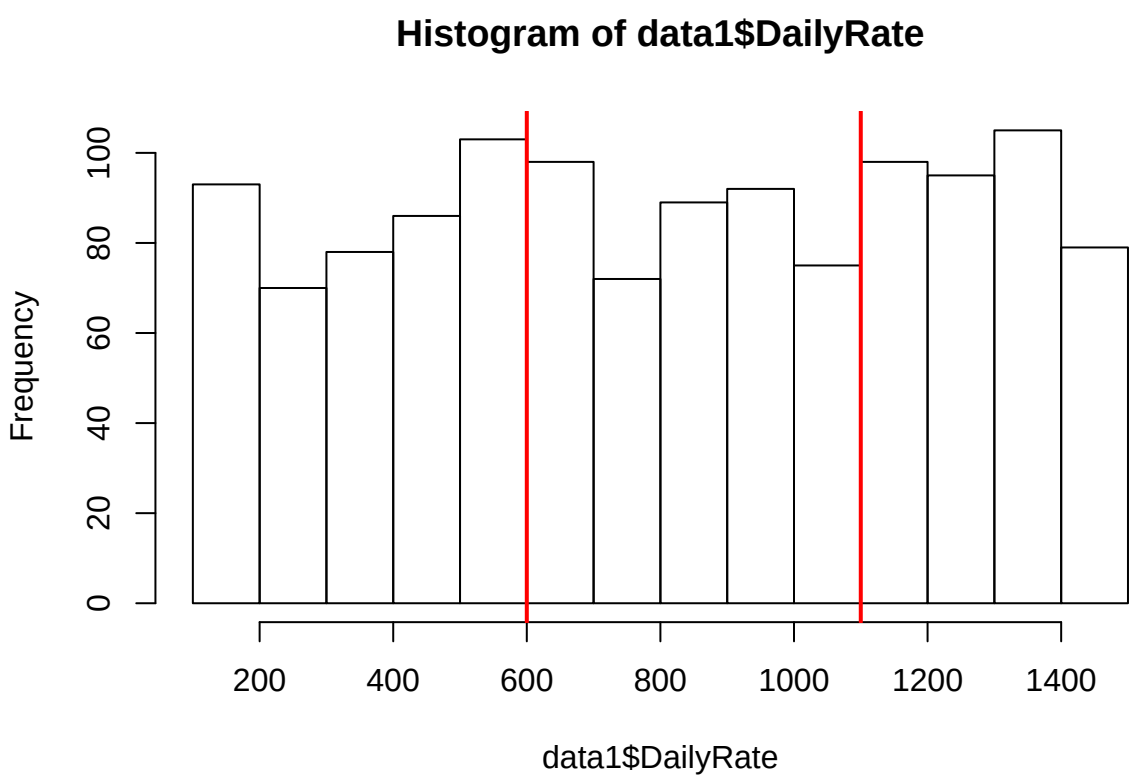
```
hist(data1$MonthlyIncome)
abline(v = c(5000,10000), col = "red", lwd =2)
```




```
data1$MonthlyIncome <- cut(data1$MonthlyIncome,breaks = c(min(data1$MonthlyIncome)-1,5000 ,10000, max(da
cor(as.numeric(data1$MonthlyIncome),my_data$MonthlyIncome,method = "spearman")
```

```
## [1] 0.9229362
```

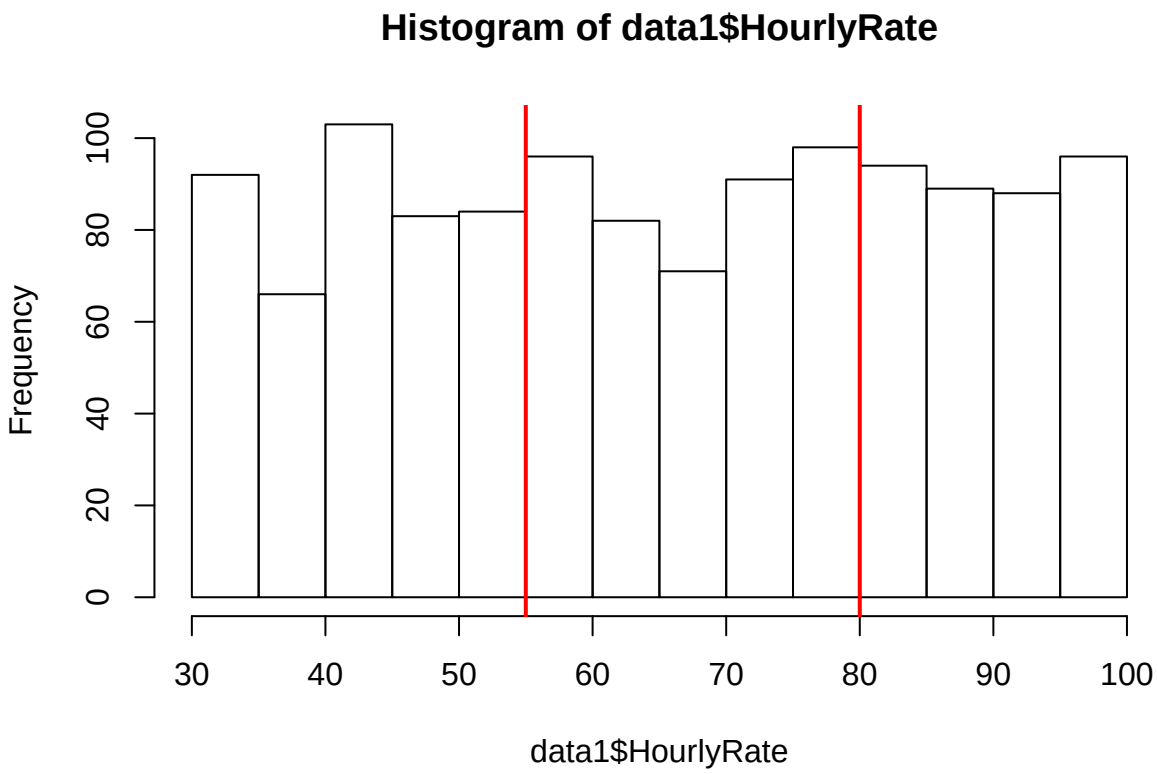
```
hist(data1$DailyRate)
abline(v = c(600,1100), col = "red", lwd =2)
```




```
data1$DailyRate <- cut(data1$DailyRate,breaks = c(min(data1$DailyRate)-1,600 ,1100, max(data1$DailyRate)
cor(as.numeric(data1$DailyRate),my_data$DailyRate,method = "spearman")
```

```
## [1] 0.9422108
```

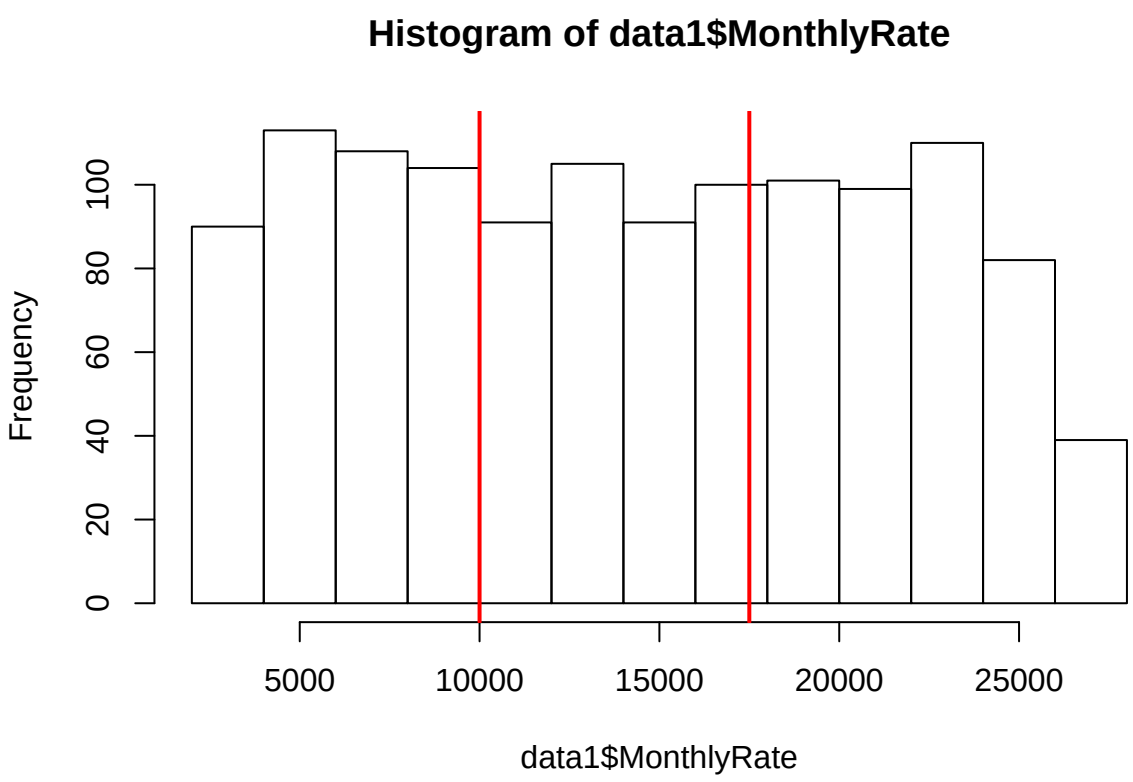
```
hist(data1$HourlyRate)
abline(v = c(55,80), col = "red", lwd =2)
```




```
data1$HourlyRate <- cut(data1$HourlyRate,breaks = c(min(data1$HourlyRate)-1,55 ,80, max(data1$HourlyRate
cor(as.numeric(data1$HourlyRate),my_data$HourlyRate,method = "spearman")
```

```
## [1] 0.9419043
```

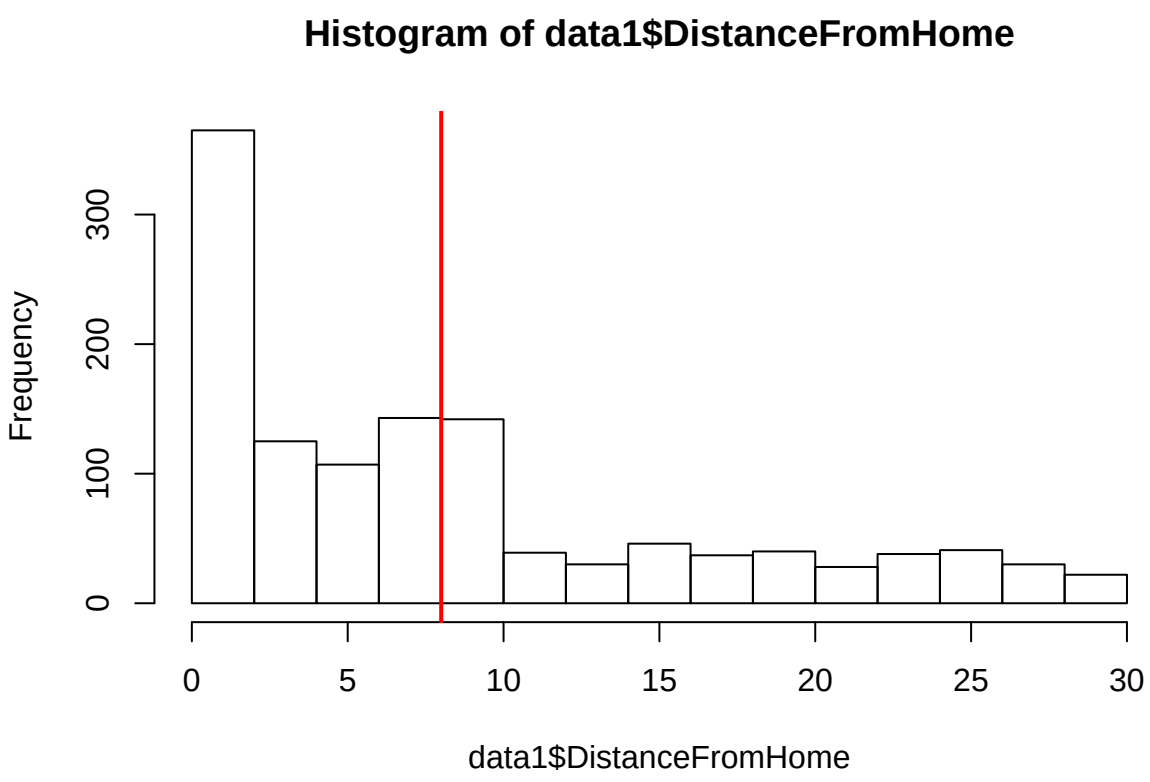
```
hist(data1$MonthlyRate)
abline(v = c(10000,17500), col = "red", lwd =2)
```




```
data1$MonthlyRate <- cut(data1$MonthlyRate,breaks = c(min(data1$MonthlyRate)-1,10000 ,17500, max(data1$
cor(as.numeric(data1$MonthlyRate),my_data$MonthlyRate,method = "spearman")
```

```
## [1] 0.9412661
```

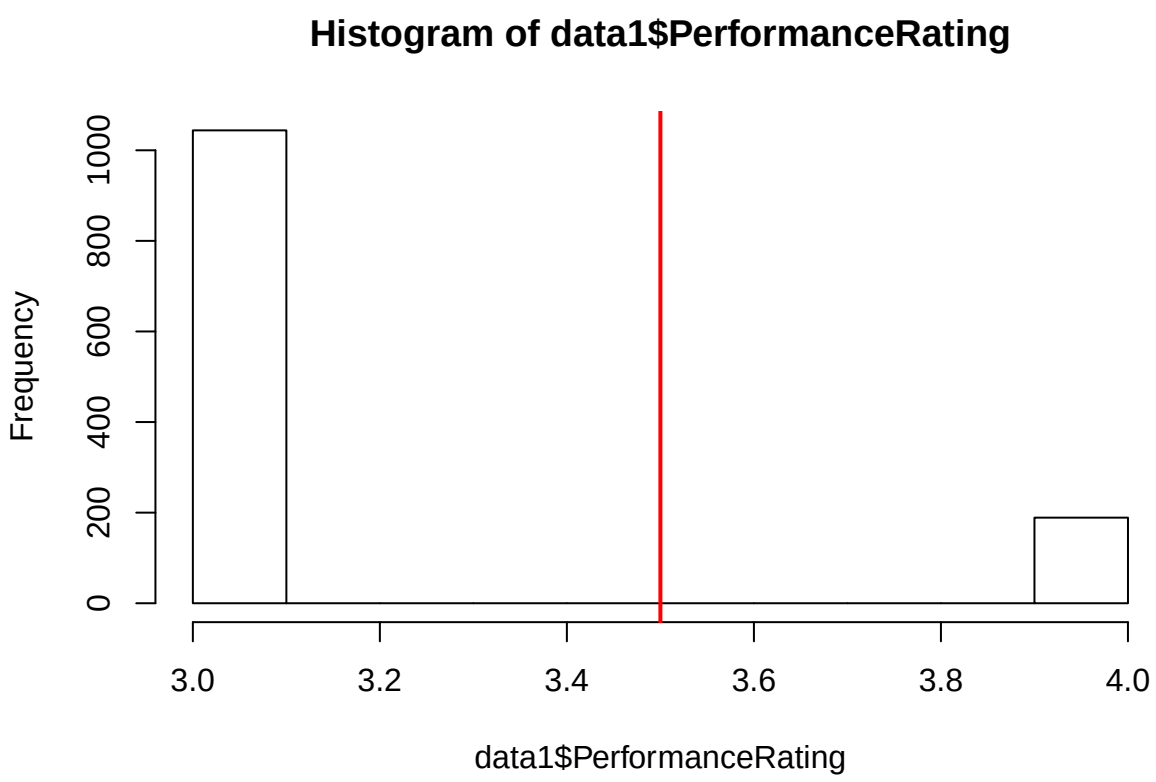
```
hist(data1$DistanceFromHome)
abline(v = c(8), col = "red", lwd =2)
```



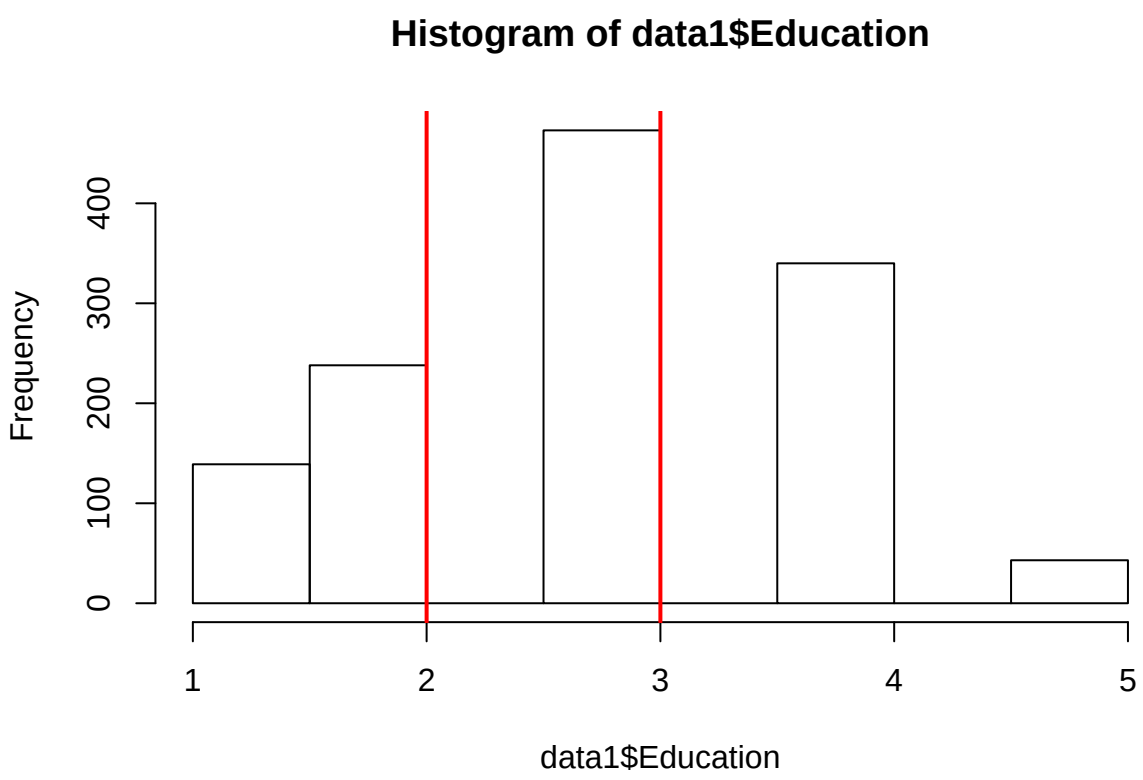

```
data1$DistanceFromHome <- cut(data1$DistanceFromHome,breaks = c(min(data1$DistanceFromHome)-1,6 , max(da
cor(as.numeric(data1$DistanceFromHome),my_data$DistanceFromHome,method = "spearman")
```

```
## [1] 0.8689751
```

```
hist(data1$PerformanceRating)
abline(v = c(3.5), col = "red", lwd =2)
```



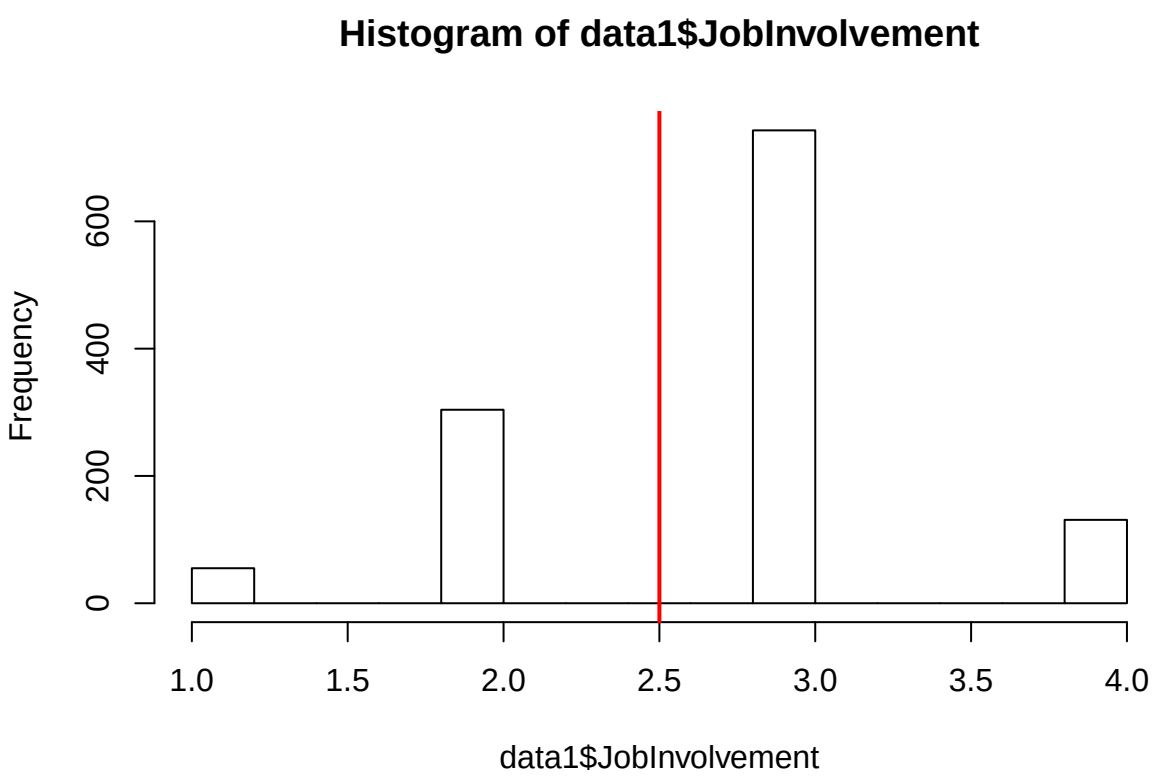

```
hist(data1$Education)
abline(v = c(2,3), col = "red", lwd =2)
```




```
data1$Education <- cut(data1$Education,breaks = c(min(data1$Education)-1, 2,3 , max(data1$Education)+1)
cor(as.numeric(data1$Education),my_data$Education,method = "spearman")
```

```
## [1] 0.9840498
```

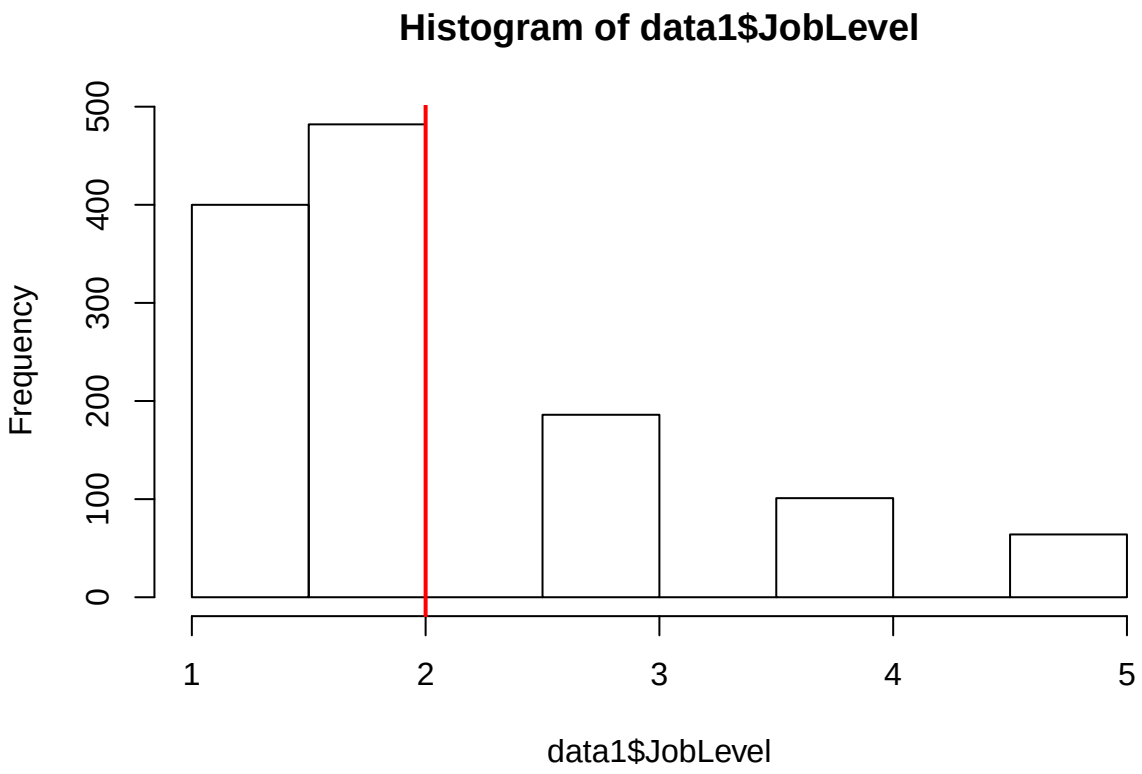
```
hist(data1$JobInvolvement)
abline(v = c(2.5), col = "red", lwd =2)
```




```
data1$JobInvolvement <- cut(data1$JobInvolvement,breaks = c(min(data1$JobInvolvement)-1,2.5, max(data1$
cor(as.numeric(data1$JobInvolvement),my_data$JobInvolvement,method = "spearman")
```

```
## [1] 0.8996954
```

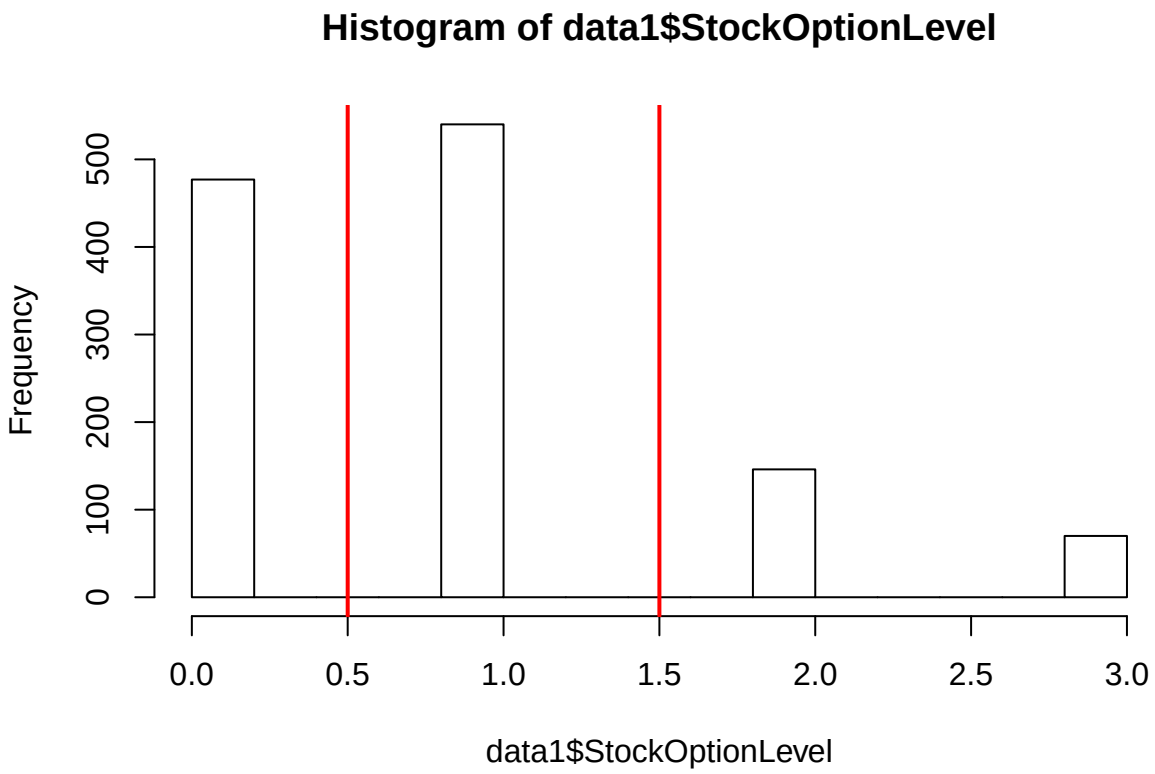
```
hist(data1$JobLevel)
abline(v = c(2), col = "red", lwd =2)
```




```
data1$JobLevel <- cut(data1$JobLevel,breaks = c(min(data1$JobLevel)-1,2, max(data1$JobLevel)+1),labels =
cor(as.numeric(data1$JobLevel),my_data$JobLevel,method = "spearman")
```

```
## [1] 0.8229676
```

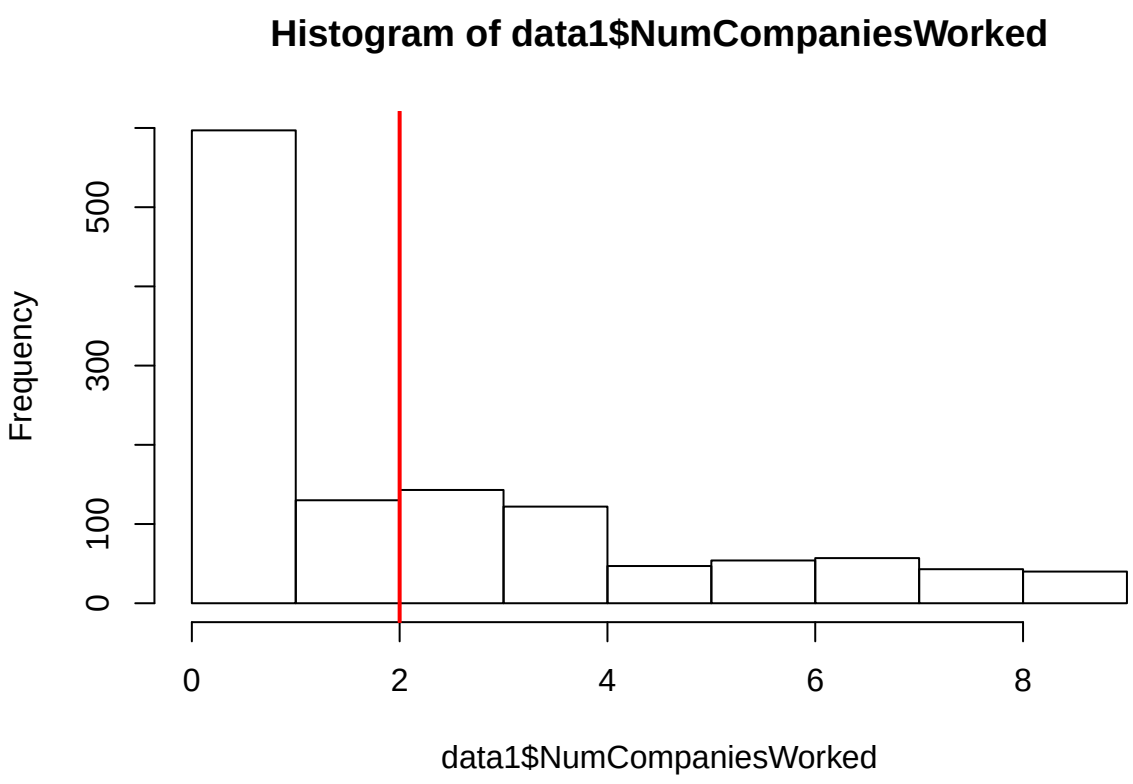
```
hist(data1$StockOptionLevel)
abline(v = c(0.5,1.5), col = "red", lwd =2)
```




```
data1$StockOptionLevel <- cut(data1$StockOptionLevel,breaks=c(min(data1$StockOptionLevel)-1,0.5,1.5,max
cor(as.numeric(data1$StockOptionLevel),my_data$StockOptionLevel,method = "spearman")
```

```
## [1] 0.9979348
```

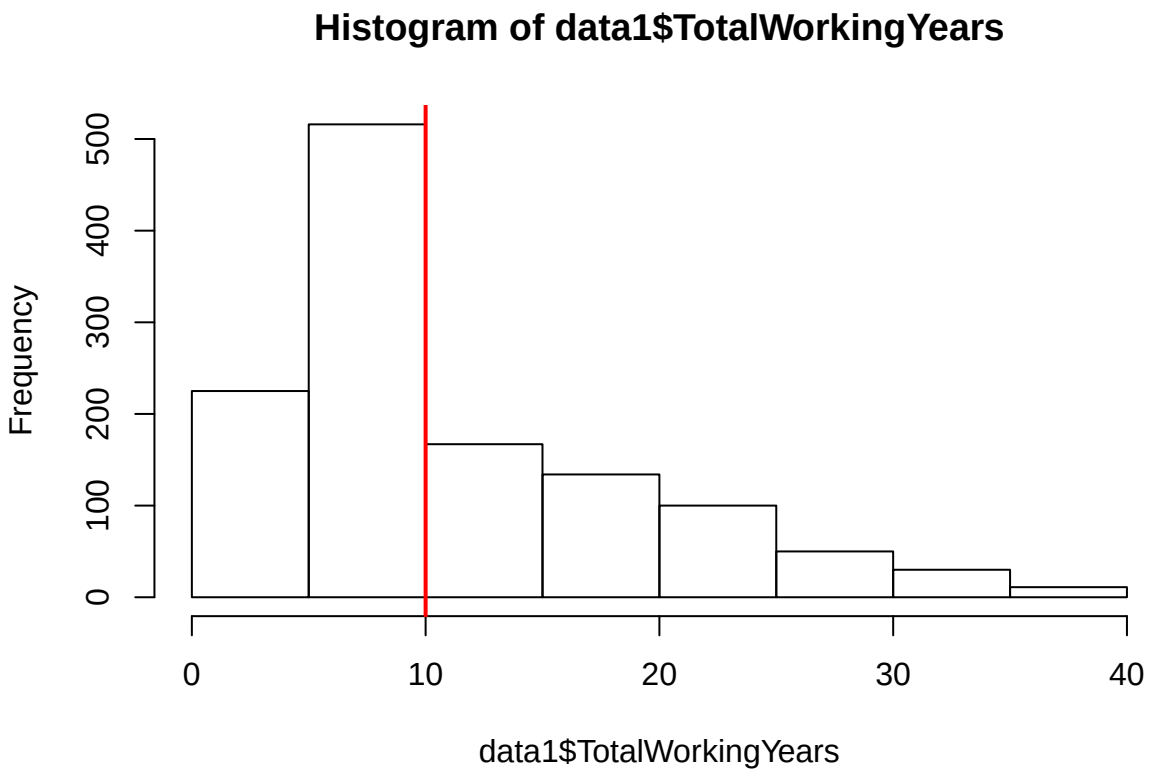
```
hist(data1$NumCompaniesWorked)
abline(v = c(2), col = "red", lwd =2)
```




```
data1$NumCompaniesWorked <- cut(data1$NumCompaniesWorked,breaks = c(min(data1$NumCompaniesWorked)-1, 2,
cor(as.numeric(data1$NumCompaniesWorked),my_data$NumCompaniesWorked,method = "spearman")
```

```
## [1] 0.8728502
```

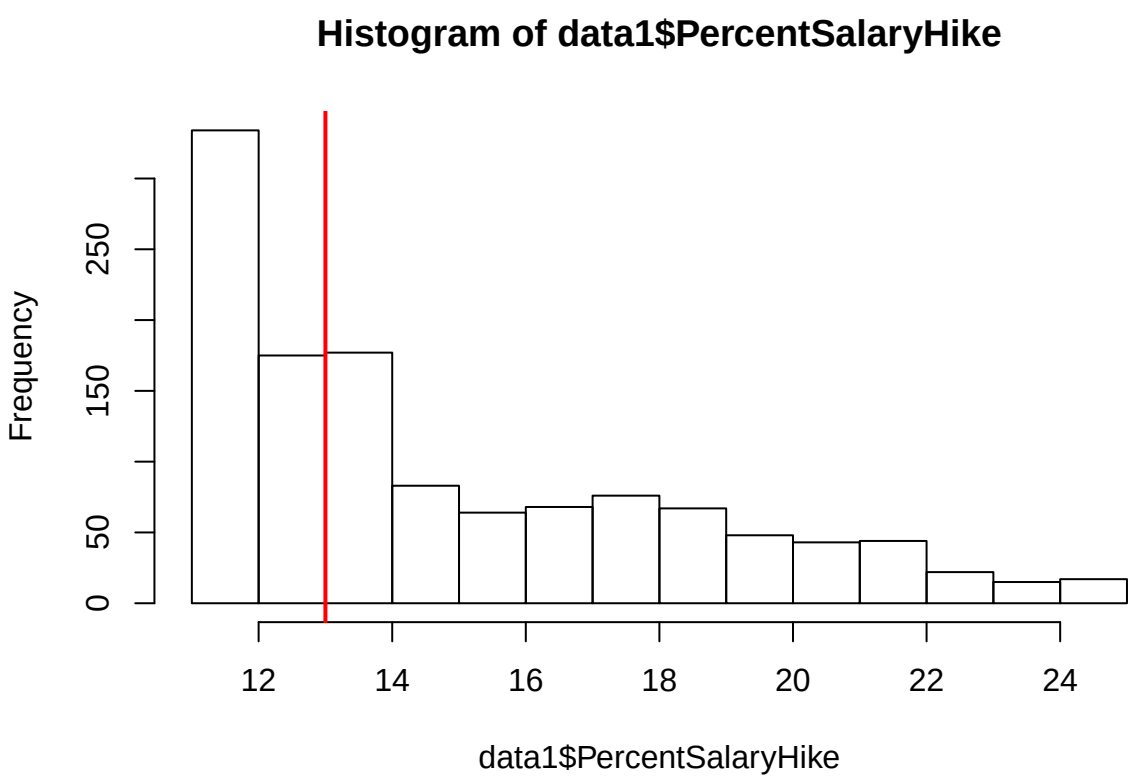
```
hist(data1$TotalWorkingYears)
abline(v = c(10), col = "red", lwd =2)
```




```
data1$TotalWorkingYears <- cut(data1$TotalWorkingYears,breaks = c(min(data1$TotalWorkingYears)-1, 10, ma
cor(as.numeric(data1$TotalWorkingYears),my_data$TotalWorkingYears,method = "spearman")
```

```
## [1] 0.8502664
```

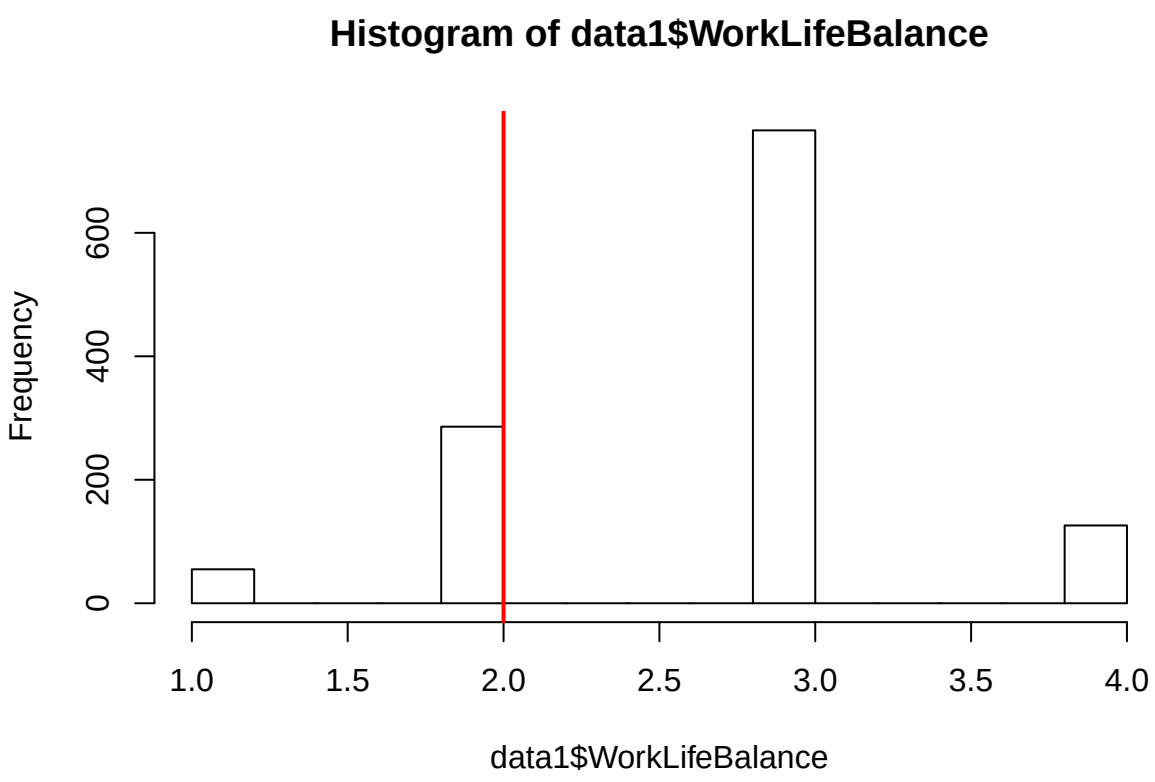
```
hist(data1$PercentSalaryHike)
abline(v = c(13), col = "red", lwd =2)
```




```
data1$PercentSalaryHike <- cut(data1$PercentSalaryHike,breaks = c(min(data1$PercentSalaryHike)-1, 13, ma
cor(as.numeric(data1$PercentSalaryHike),my_data$PercentSalaryHike,method = "spearman")
```

```
## [1] 0.8579005
```

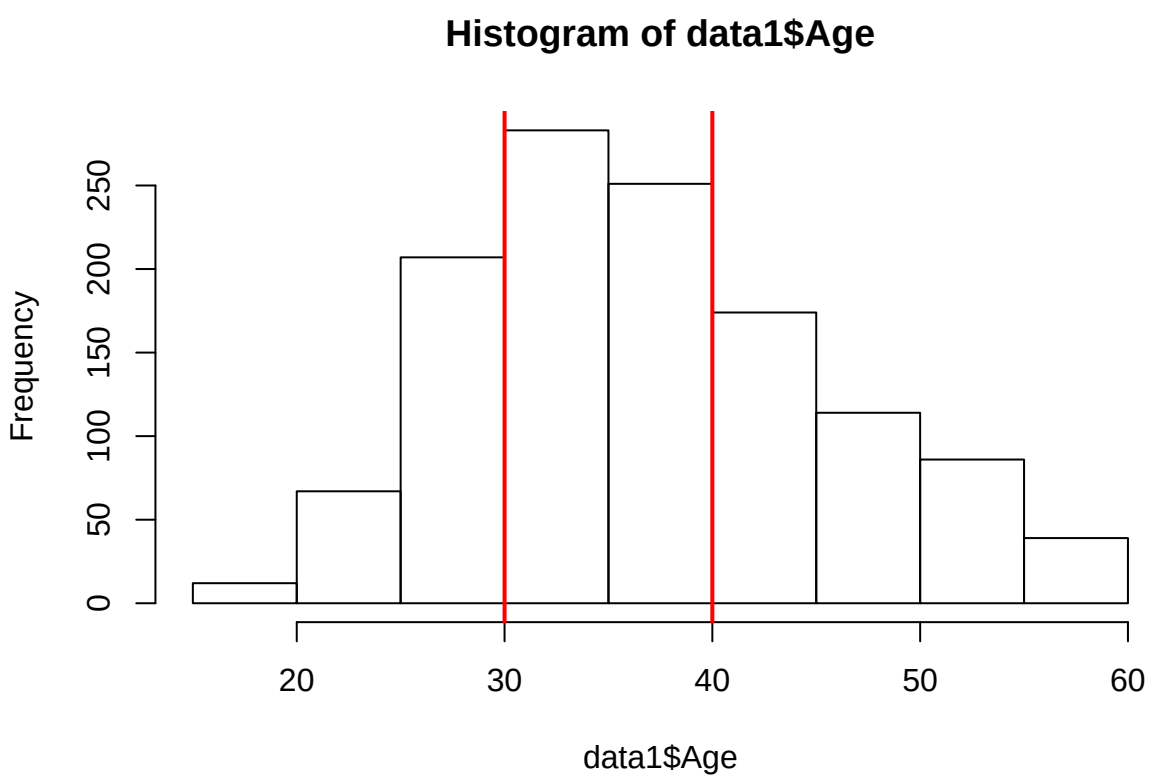
```
hist(data1$WorkLifeBalance)
abline(v = c(2), col = "red", lwd =2)
```




```
data1$WorkLifeBalance <- cut(data1$WorkLifeBalance,breaks = c(min(data1$WorkLifeBalance)-1, 2, max(data
cor(as.numeric(data1$PercentSalaryHike),my_data$PercentSalaryHike,method = "spearman")
```

```
## [1] 0.8579005
```

```
hist(data1$Age)
abline(v = c(30,40), col = "red", lwd =2)
```




```
data1$Age <- cut(data1$Age,breaks = c(min(data1$Age)-1, 30,40, max(data1$Age)+1),labels = c(1,2,3))
cor(as.numeric(data1$Age),my_data$Age,method = "spearman")
```

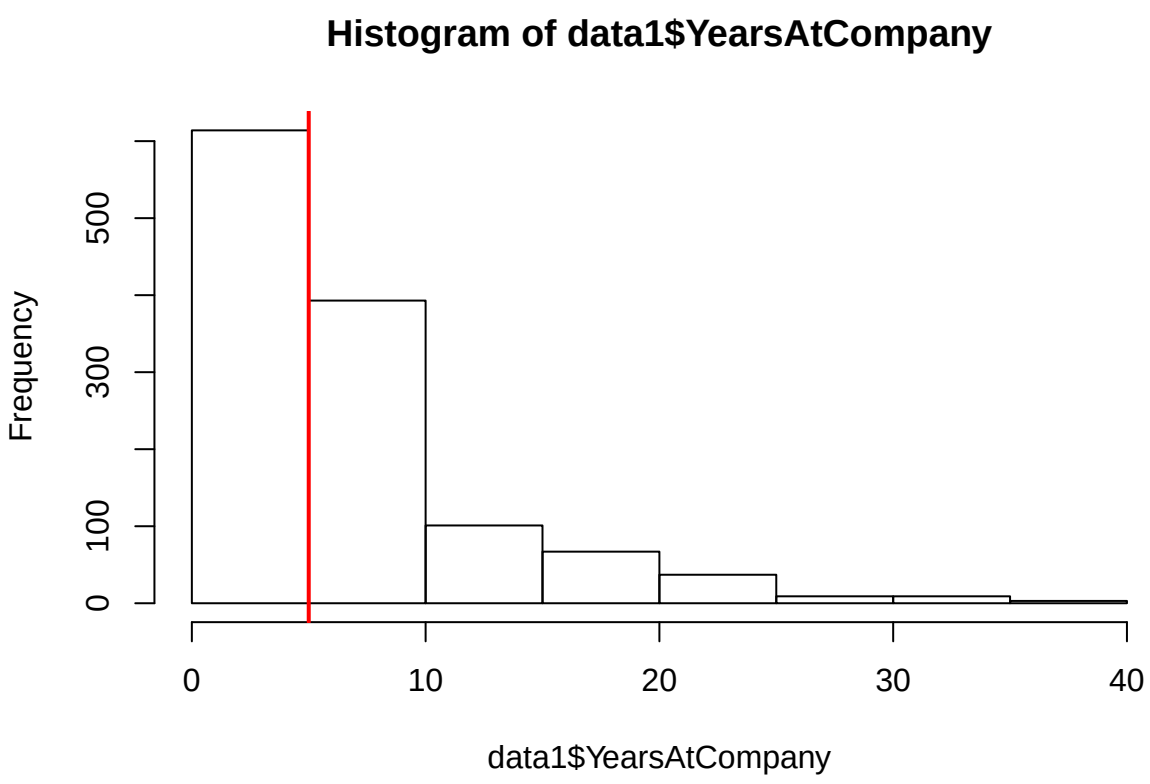
```
## [1] 0.9326494
```

```
hist(data1$TrainingTimesLastYear)
abline(v = c(2), col = "red", lwd =2)
```


```
data1$TrainingTimesLastYear <- cut(data1$TrainingTimesLastYear,breaks = c(min(data1$TrainingTimesLastYea
cor(as.numeric(data1$TrainingTimesLastYear),my_data$TrainingTimesLastYear,method = "spearman")
```

```
## [1] 0.9024219
```

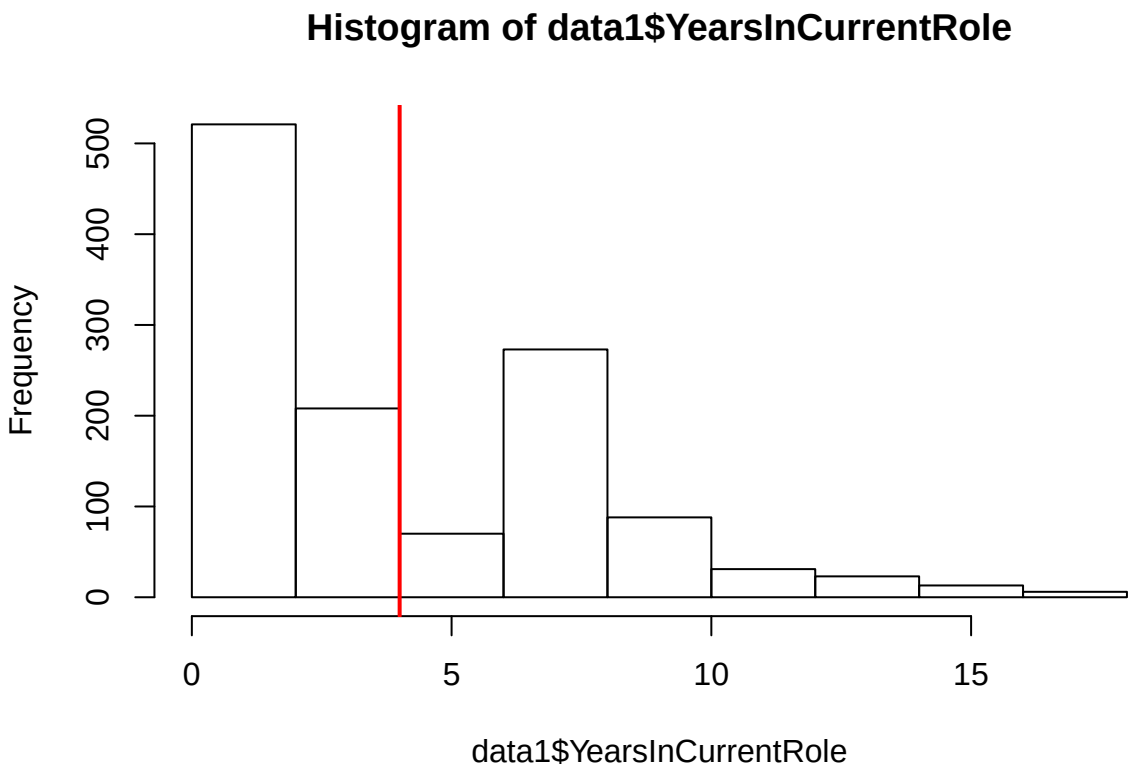
```
hist(data1$YearsAtCompany)
abline(v = c(5), col = "red", lwd =2)
```




```
data1$YearsAtCompany <- cut(data1$YearsAtCompany,breaks = c(min(data1$YearsAtCompany)-1, 5, max(data1$Y
cor(as.numeric(data1$YearsAtCompany),my_data$YearsAtCompany,method = "spearman")
```

```
## [1] 0.8689195
```

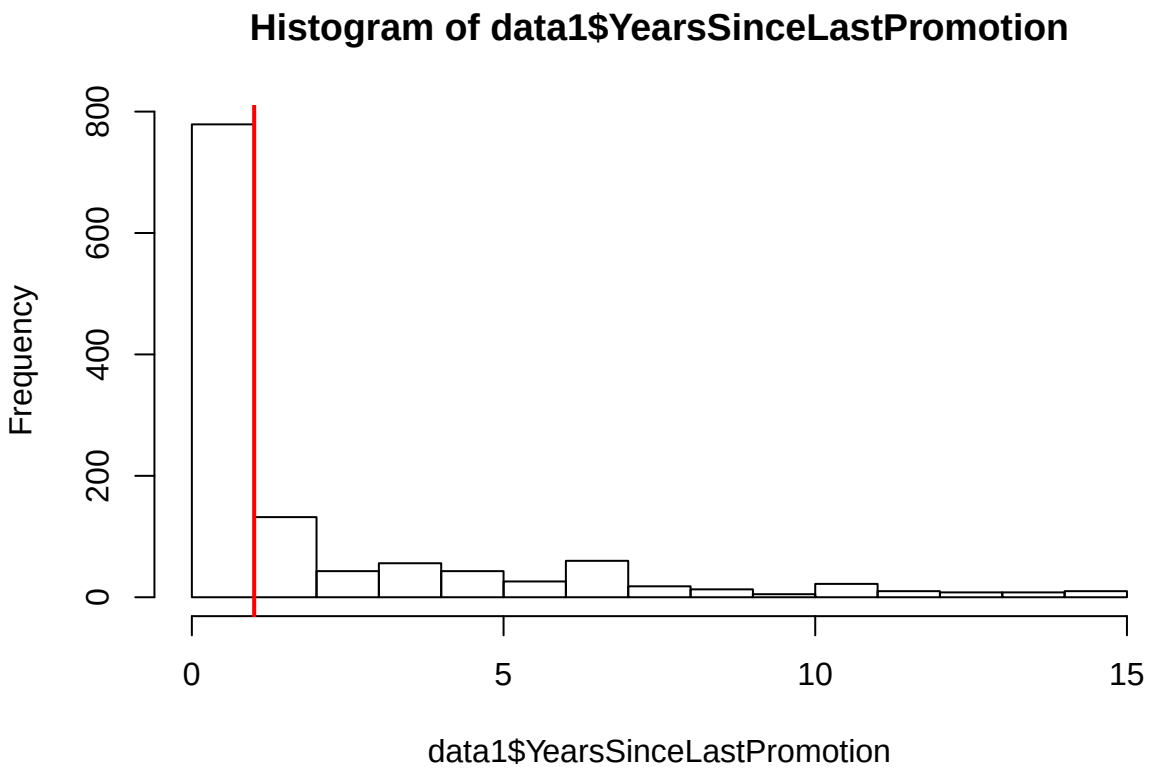
```
hist(data1$YearsInCurrentRole)
abline(v = c(4), col = "red", lwd =2)
```




```
data1$YearsInCurrentRole <- cut(data1$YearsInCurrentRole,breaks = c(min(data1$YearsInCurrentRole)-1, 4,
cor(as.numeric(data1$YearsInCurrentRole),my_data$YearsInCurrentRole,method = "spearman")
```

```
## [1] 0.8615273
```

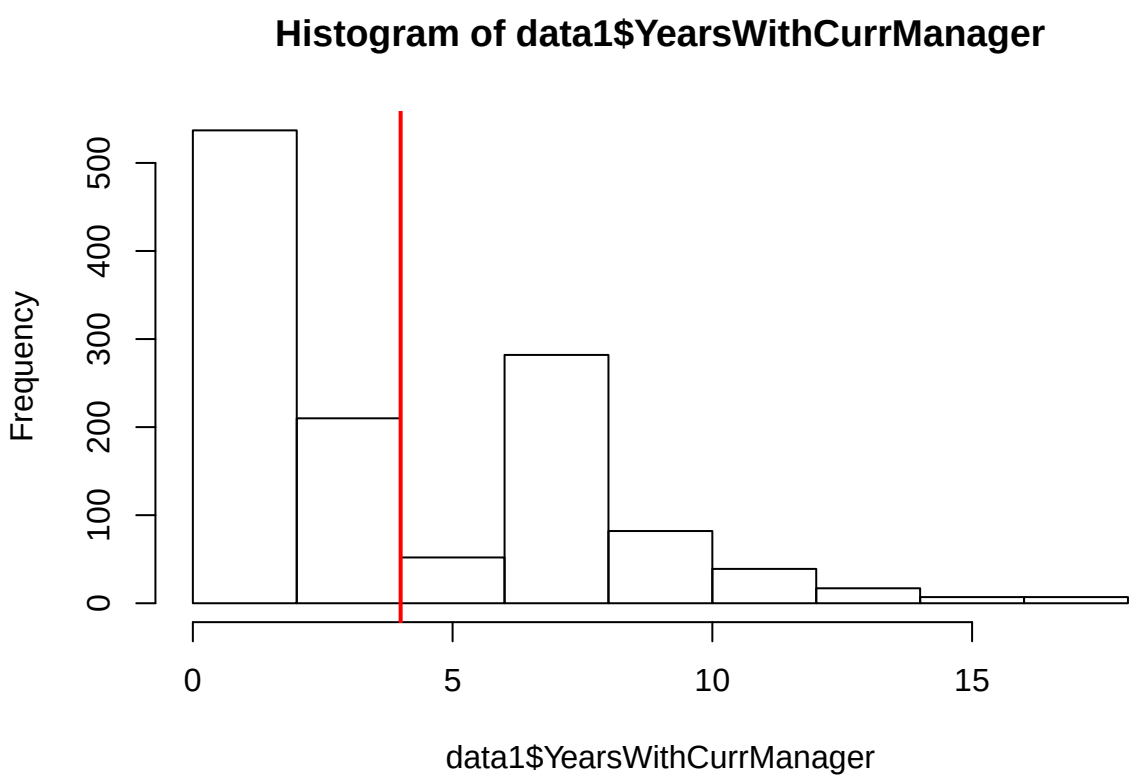
```
hist(data1$YearsSinceLastPromotion)
abline(v = c(1), col = "red", lwd =2)
```




```
data1$YearsSinceLastPromotion <- cut(data1$YearsSinceLastPromotion,breaks = c(min(data1$YearsSinceLastPr
cor(as.numeric(data1$YearsSinceLastPromotion),my_data$YearsSinceLastPromotion,method = "spearman")
```

```
## [1] 0.8676085
```

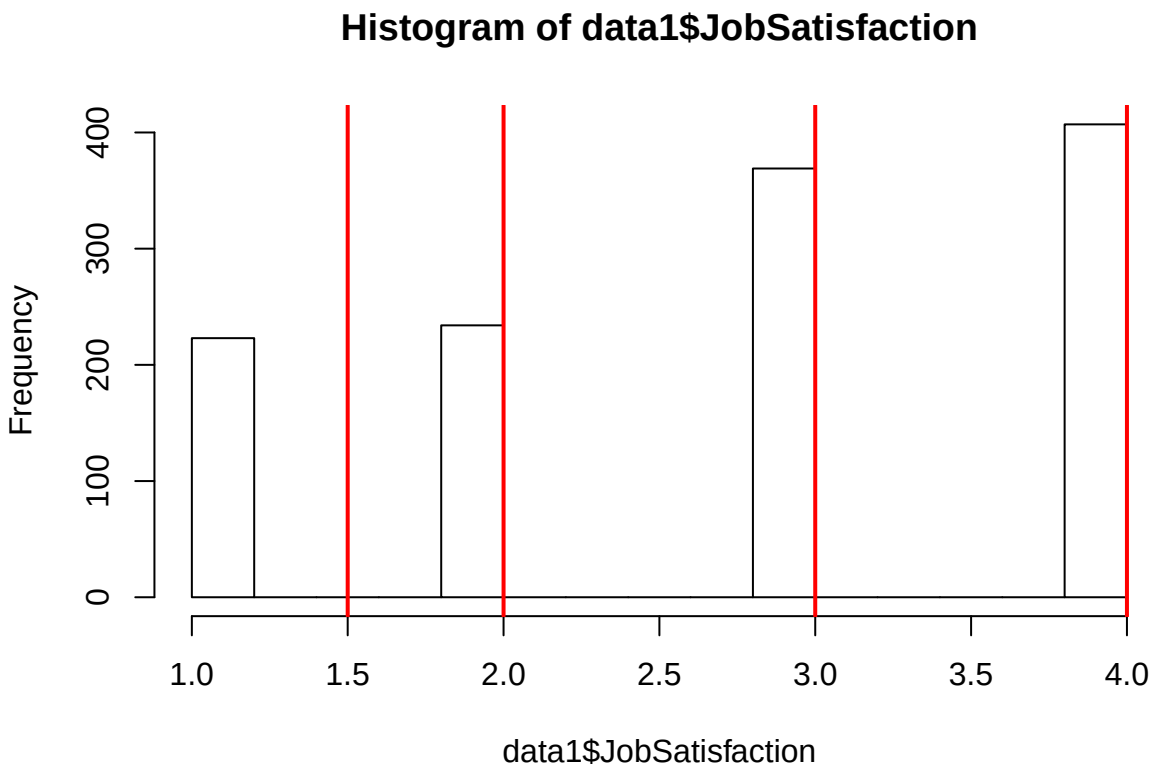
```
hist(data1$YearsWithCurrManager)
abline(v = c(4), col = "red", lwd =2)
```



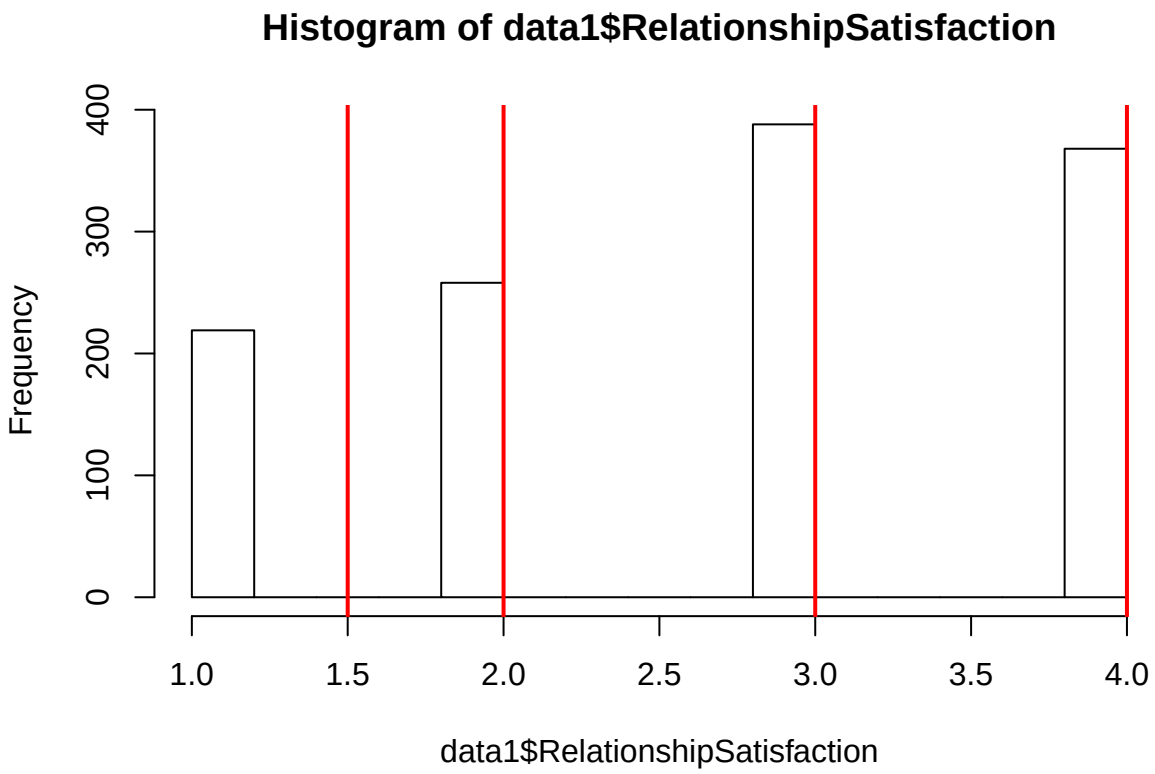

```
data1$YearsWithCurrManager <- cut(data1$YearsWithCurrManager,breaks = c(min(data1$YearsWithCurrManager)-
cor(as.numeric(data1$YearsWithCurrManager),my_data$YearsWithCurrManager,method = "spearman")
```

```
## [1] 0.855894
```

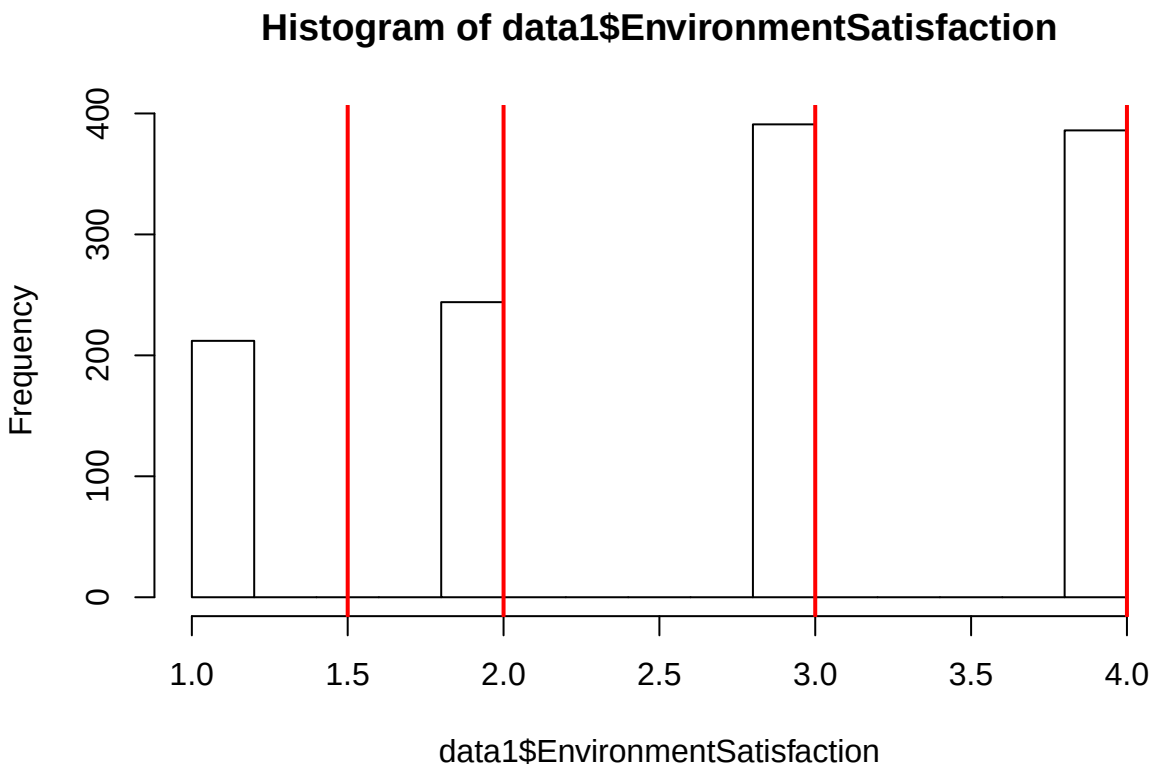
```
hist(data1$JobSatisfaction)
abline(v = c(1.5,2,3,4), col = "red", lwd =2)
```




```
hist(data1$RelationshipSatisfaction)
abline(v = c(1.5,2,3,4), col = "red", lwd =2)
```

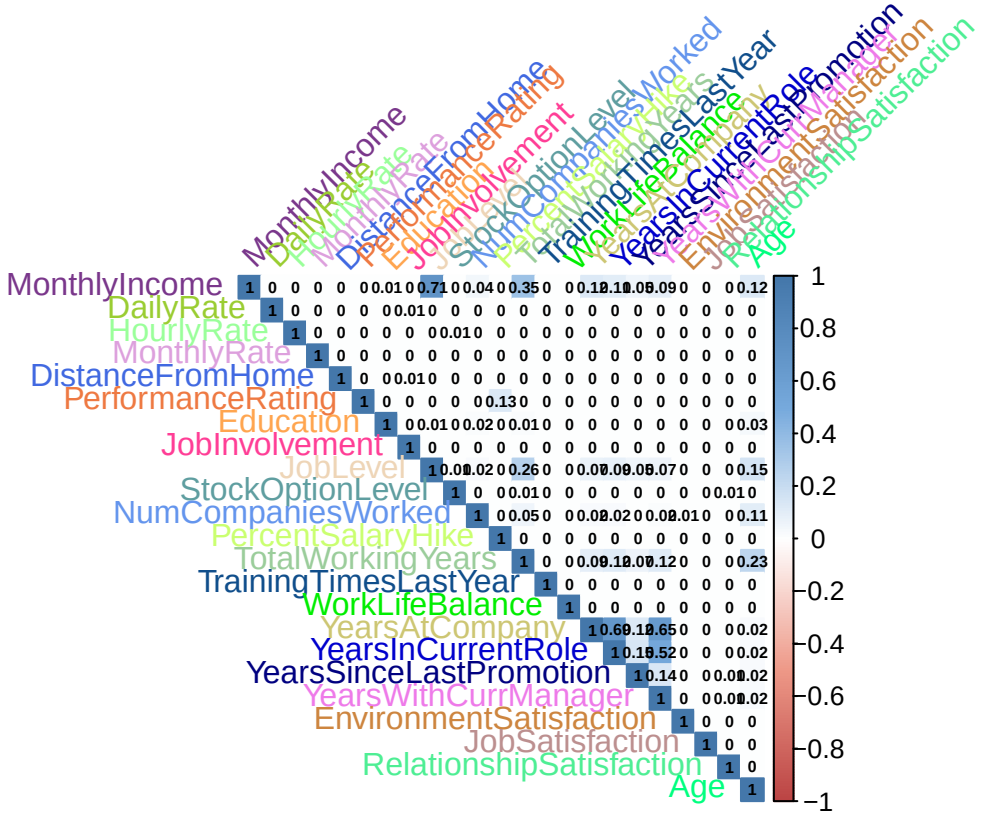



```r
hist(data1$EnvironmentSatisfaction)
abline(v = c(1.5,2,3,4), col = "red", lwd =2)
```



HeatMap

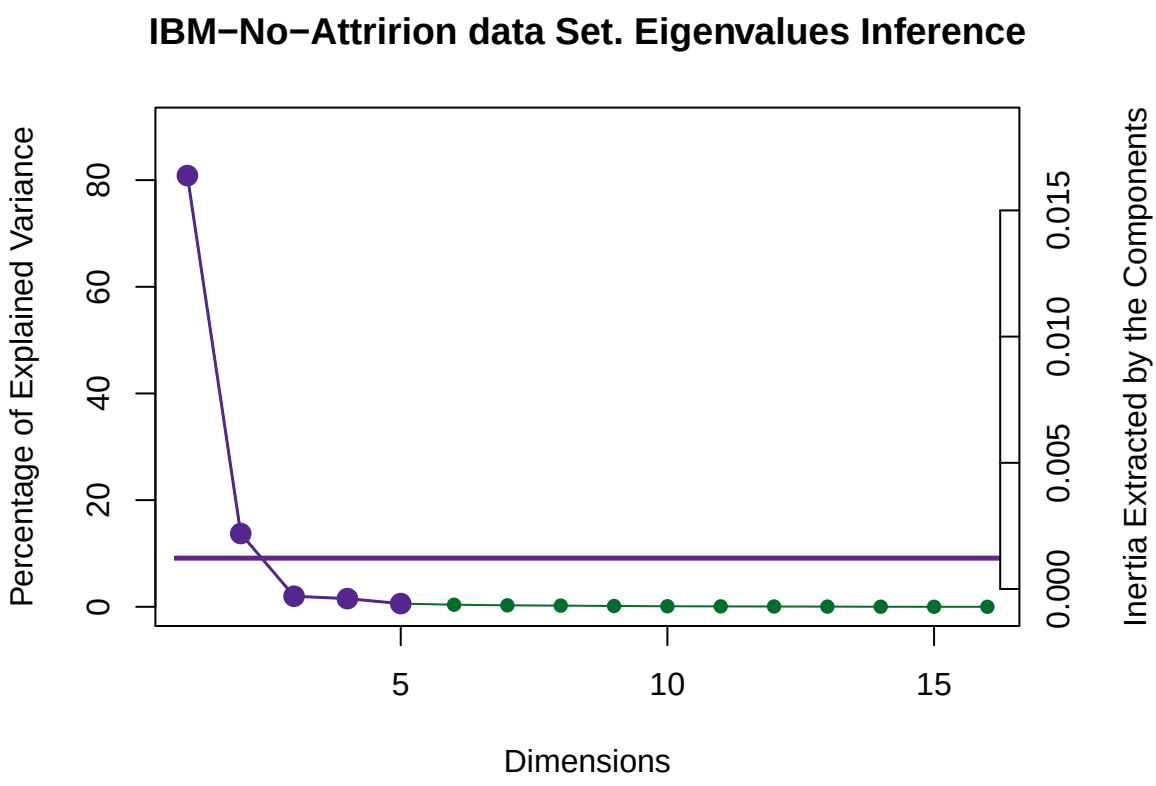
```
color4Var1 <- prettyGraphs::prettyGraphsColorSelection(ncol(data1))
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corrMatBurt.list1 <- phi2Mat4BurtTable(data1)
corr4MCA.r1 <- corrplot::corrplot( as.matrix(corrMatBurt.list1$phi2.mat), method="color", col=col(200),
tl.srt = 45, #Text label color and rotation
number.cex = .5,
diag = TRUE  )
```



Scree plot

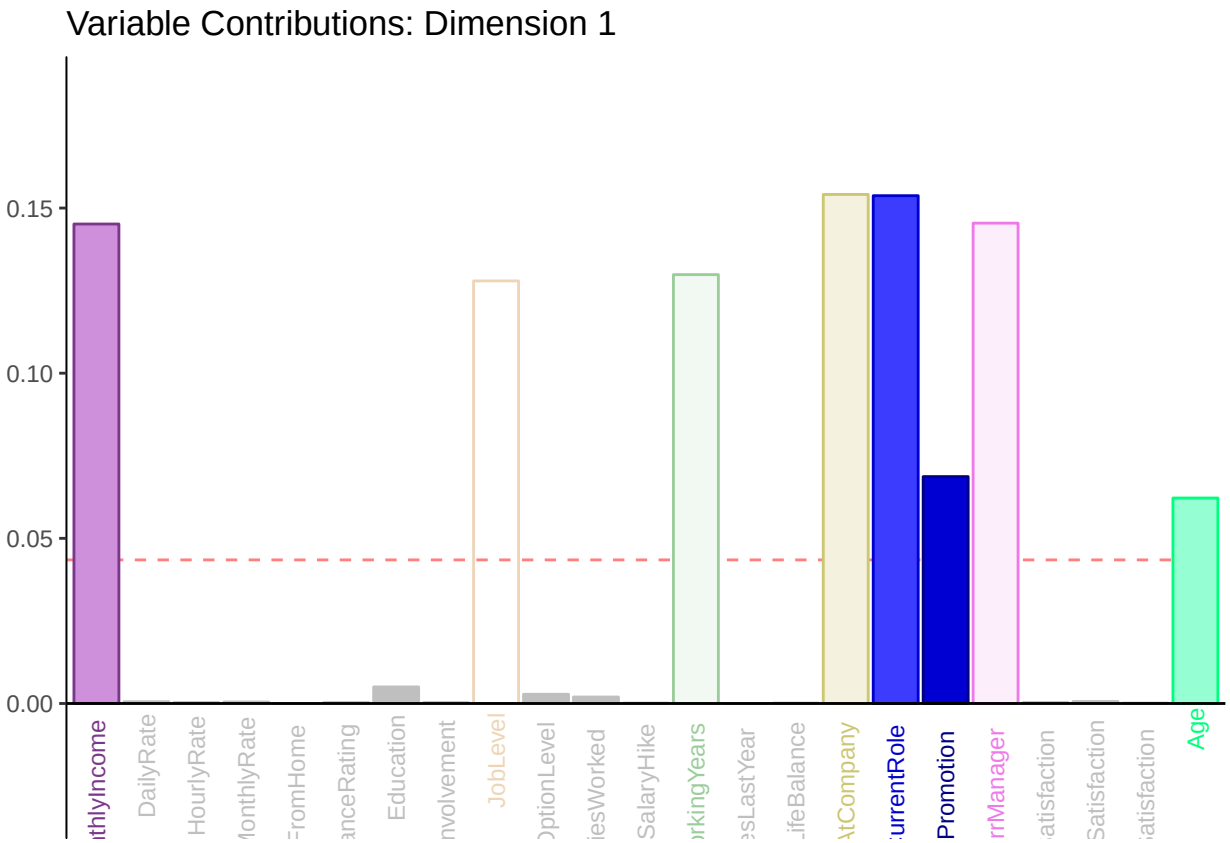
```
resMCA.sym <- epMCA(data1 ,make_data_nominal = TRUE ,DESIGN = my_data$Department ,make_design_nominal =

resMCA.asym <- epMCA(data1 ,make_data_nominal = TRUE ,DESIGN = my_data$Department ,make_design_nominal =

resMCA.inf <- epMCA.inference.battery(data1, make_data_nominal = TRUE, DESIGN = my_data$Department ,mak

resMCA.sym1 <- epMCA(data1 ,make_data_nominal = TRUE ,DESIGN = my_data$Gender ,make_design_nominal = TRU

resMCA.asym1 <- epMCA(data1 ,make_data_nominal = TRUE ,DESIGN = my_data$Gender ,make_design_nominal = TR

resMCA.inf1 <- epMCA.inference.battery(data1, make_data_nominal = TRUE, DESIGN = my_data$Gender ,make_d
```

```
PlotScree(ev = resMCA.sym$ExPosition.Data$eigs,
          p.ev =  resMCA.inf$Inference.Data$components$p.vals,
          title = 'IBM-No-Attririon data Set. Eigenvalues Inference',
          plotKaiser = TRUE
)
```

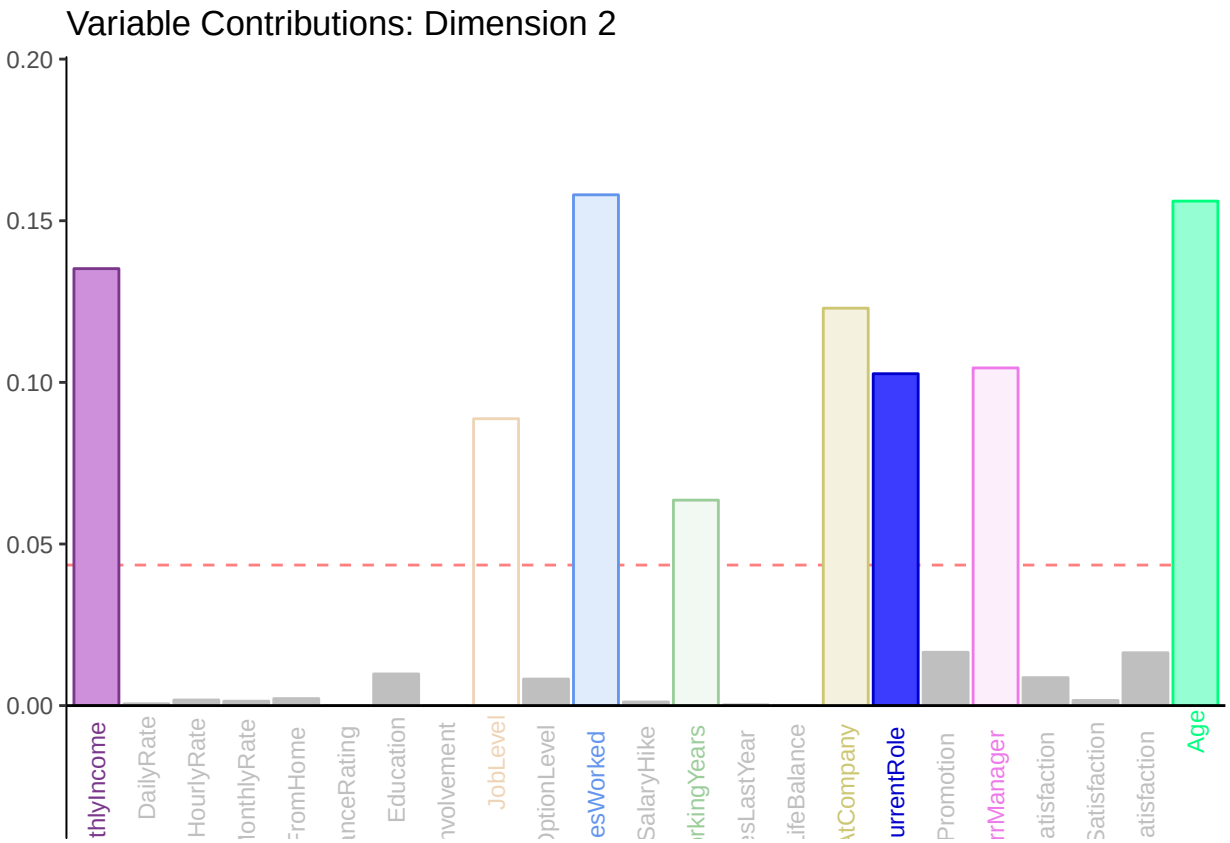


4.2 Variable contributions

```
color <- prettyGraphs::prettyGraphsColorSelection(ncol(data1))
cJ <- resMCA.sym$ExPosition.Data$cj
varCtr <- data4PCCAR::ctr4Variables(cJ)
rownames(color) <- rownames(varCtr)
varCtr1 <- varCtr[,1]
names(varCtr1) <- rownames(varCtr)
a0005.Var.ctr1 <- PrettyBarPlot2(varCtr1, main = 'Variable Contributions: Dimension 1', ylim = c(-.03,1
print(a0005.Var.ctr1)
```

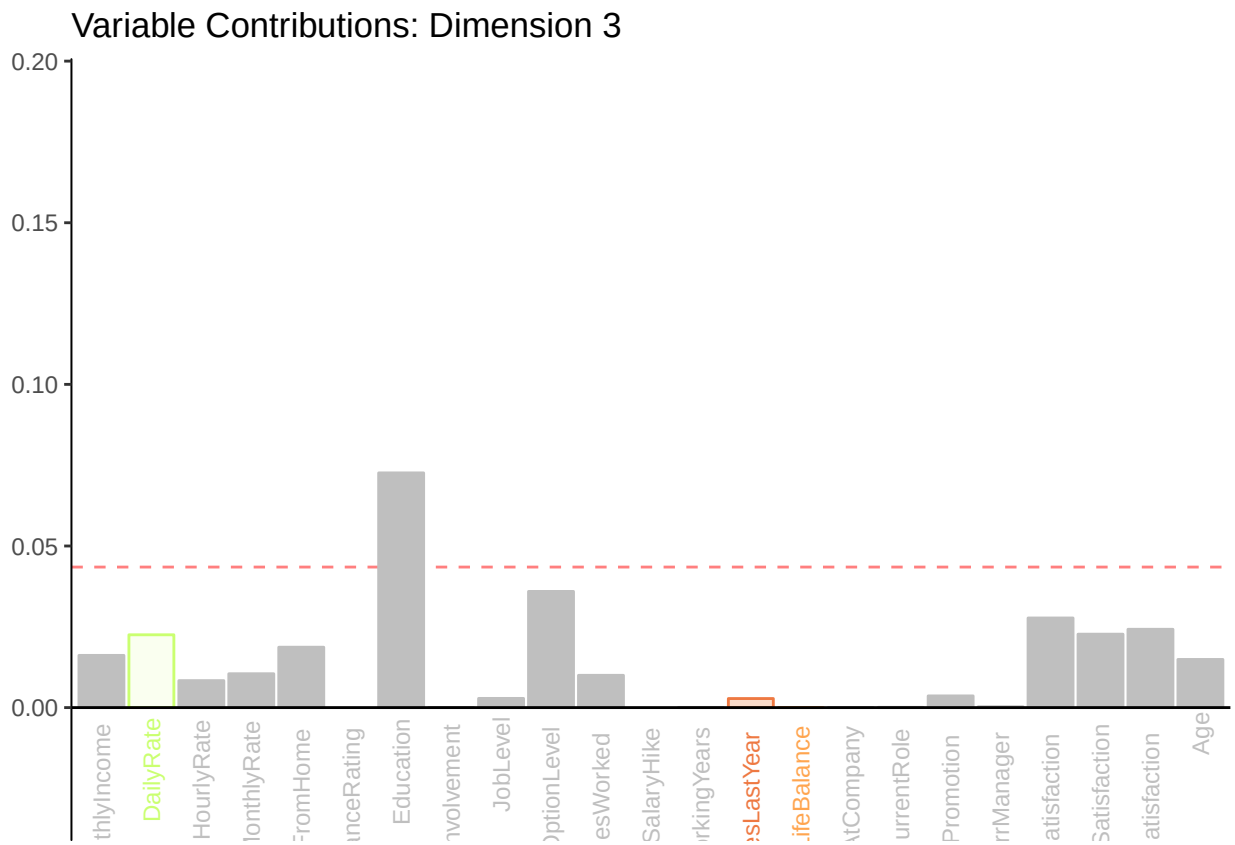



```
varCtr2 <- varCtr[,2]
names(varCtr2) <- rownames(varCtr)
a0006.Var.ctr2 <- PrettyBarPlot2(varCtr2,
main = 'Variable Contributions: Dimension 2', ylim = c(-.03, 1.2*max(varCtr2)),threshold = 1 / nrow(varC
print(a0006.Var.ctr2)
```



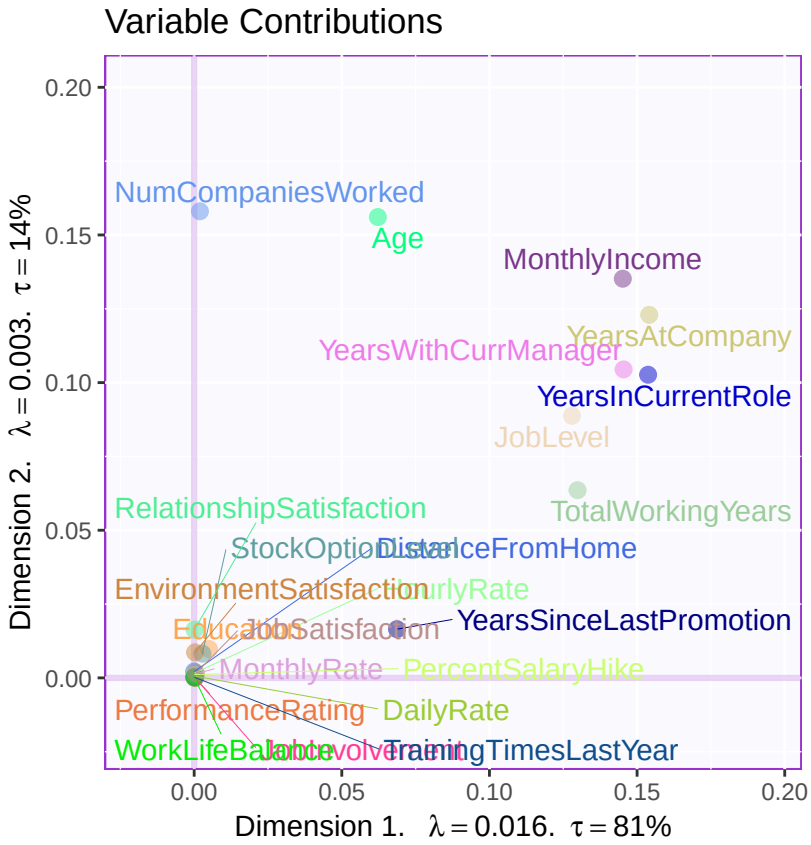

```
varCtr3 <- varCtr[,3]
names(varCtr3) <- rownames(varCtr)
a0006.Var.ctr3 <- PrettyBarPlot2(varCtr3,
main = 'Variable Contributions: Dimension 3', ylim = c(-.03, 1.2*max(varCtr2)),threshold = 1 / nrow(varC
)
print(a0006.Var.ctr3)
```

```
## Warning: Removed 2 rows containing missing values (position_stack).
```

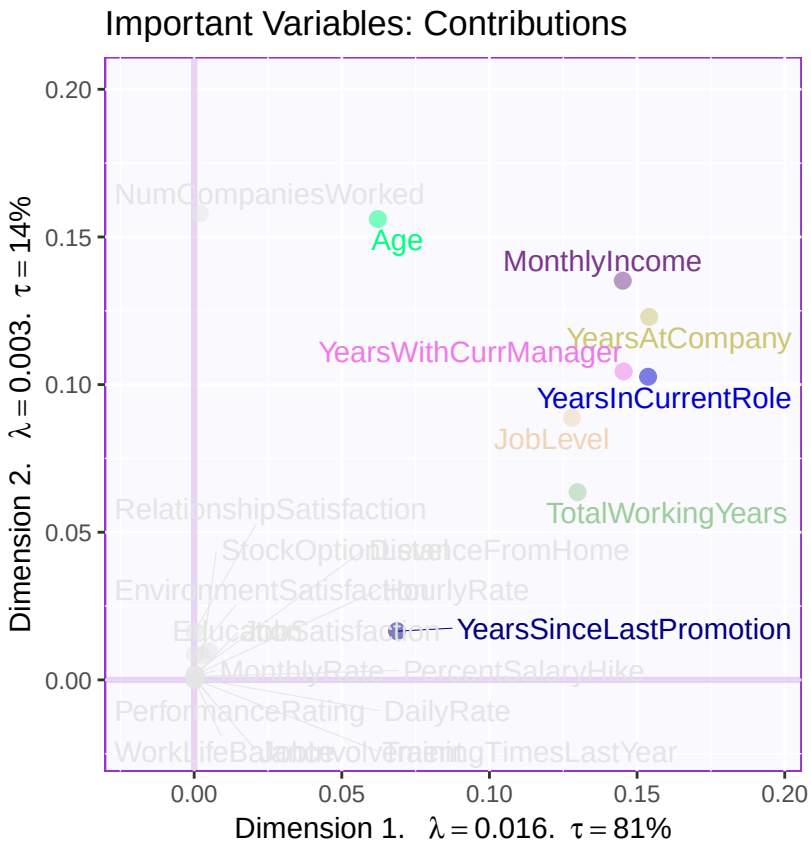


Pseudo factor plots with variable contributions

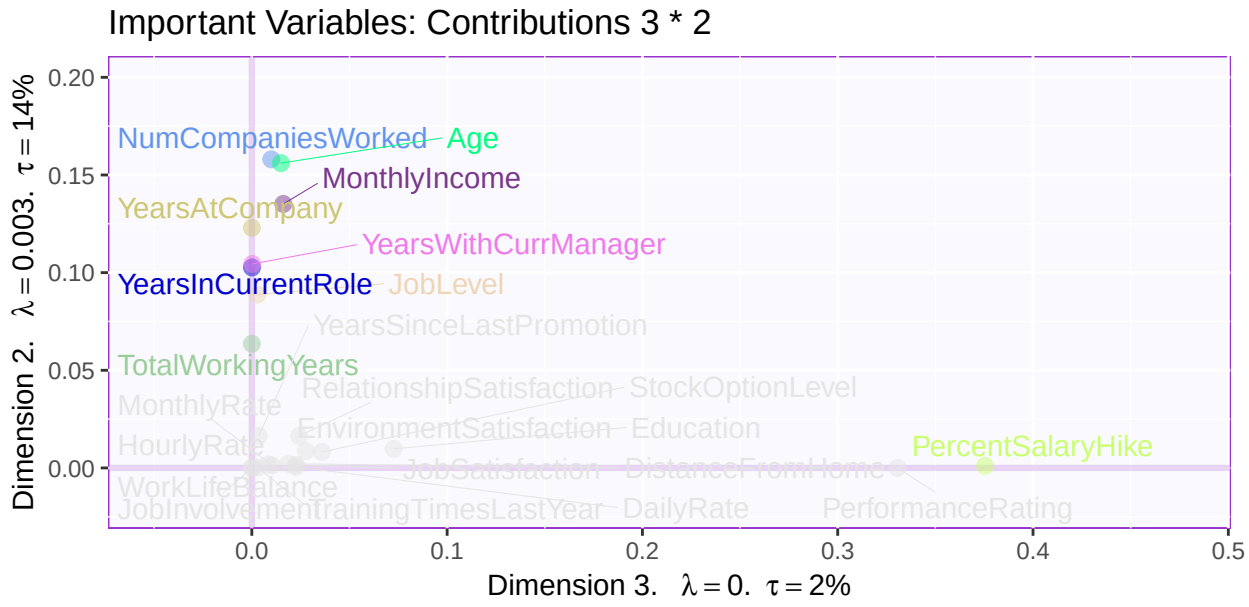
```
ctrV12 <- PTCA4CATA::createFactorMap(X = varCtr,
title = "Variable Contributions",
                        col.points = color,
                        col.labels = color,
                        alpha.points = 0.5,cex = 2.5,
                        alpha.labels = 1,
                        text.cex = 4,
                        font.face = "plain",
                        font.family = "sans")
ctr.labels <- createxyLabels.gen(
1,2, lambda = resMCA.sym$ExPosition.Data$eigs, tau = resMCA.sym$ExPosition.Data$t
)
a0007.Var.ctr12 <- ctrV12$zeMap + ctr.labels #
print(a0007.Var.ctr12)
```




```
absCtrVar <- as.matrix(varCtr) %*% diag(resMCA.sym$ExPosition.Data$eigs)
varCtr12 <- (absCtrVar[,1] + absCtrVar[,2]) /
(resMCA.sym$ExPosition.Data$eigs[1] + resMCA.sym$ExPosition.Data$eigs[2])
importantVar <- (varCtr12 >= 1 / length(varCtr12))
col4ImportantVar <- color
col4NS <- 'gray90'
col4ImportantVar[!importantVar] <- col4NS
ctrV12.imp <- PTCA4CATA::createFactorMap(X = varCtr,
title = "Important Variables: Contributions",
                        col.points = col4ImportantVar,
                        col.labels = col4ImportantVar,
                        alpha.points = 0.5,
                        cex = 2.5,
                        alpha.labels = 1,
                        text.cex = 4,
                        font.face = "plain",
                        font.family = "sans")
a0008.Var.ctr12.imp <- ctrV12.imp$zeMap + ctr.labels #
print(a0008.Var.ctr12.imp)
```

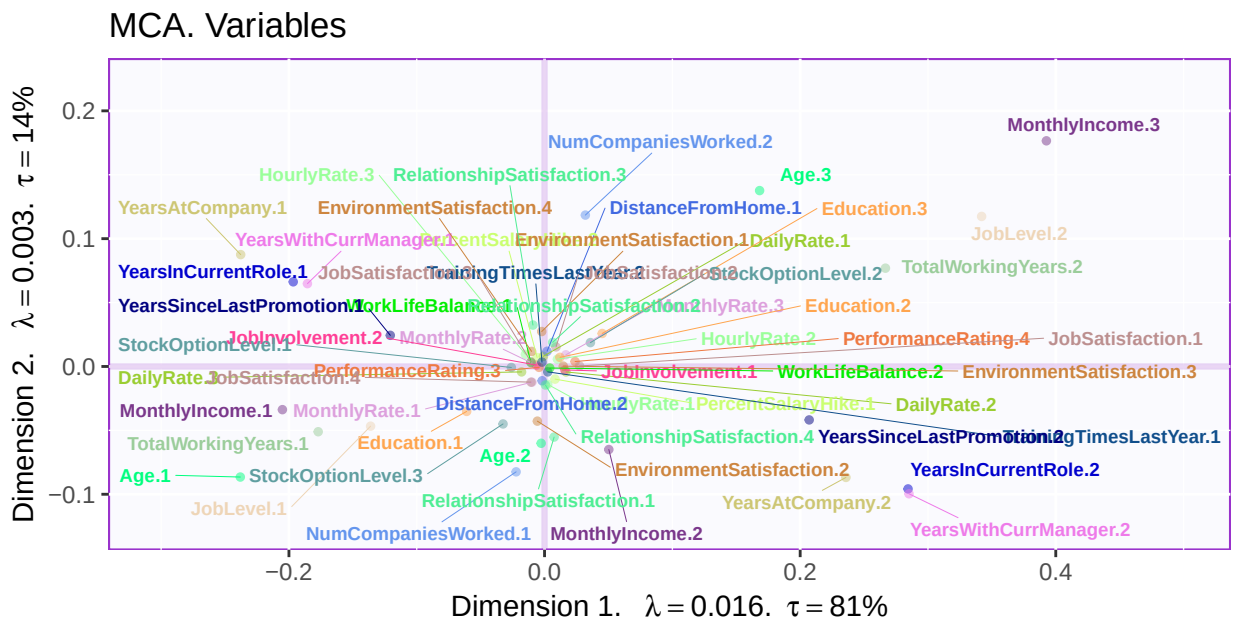



```
varCtr23 <- (absCtrVar[,3] + absCtrVar[,2]) / (resMCA.sym$ExPosition.Data$eigs[3] + resMCA.sym$ExPositi
importantVar23 <- (varCtr23 >= 1 / length(varCtr23))
col4ImportantVar23 <- color
col4NS <- 'gray90'
col4ImportantVar23[!importantVar23] <- col4NS
ctrV23.imp <- PTCA4CATA::createFactorMap(X = varCtr,
axis1 = 3, axis2 = 2,
                        title = "Important Variables: Contributions 3 * 2",
                        col.points = col4ImportantVar23,
                        col.labels = col4ImportantVar23,
                        alpha.points = 0.5,
                        cex = 2.5,
                        alpha.labels = 1,
                        text.cex = 4,
                        font.face = "plain",
                        font.family = "sans")
ctr.labels23 <- createxyLabels.gen(
3,2, lambda = resMCA.sym$ExPosition.Data$eigs, tau = resMCA.sym$ExPosition.Data$t
)
a0009.Var.ctr23.imp <- ctrV23.imp$zeMap + ctr.labels23 #
print(a0009.Var.ctr23.imp)
```

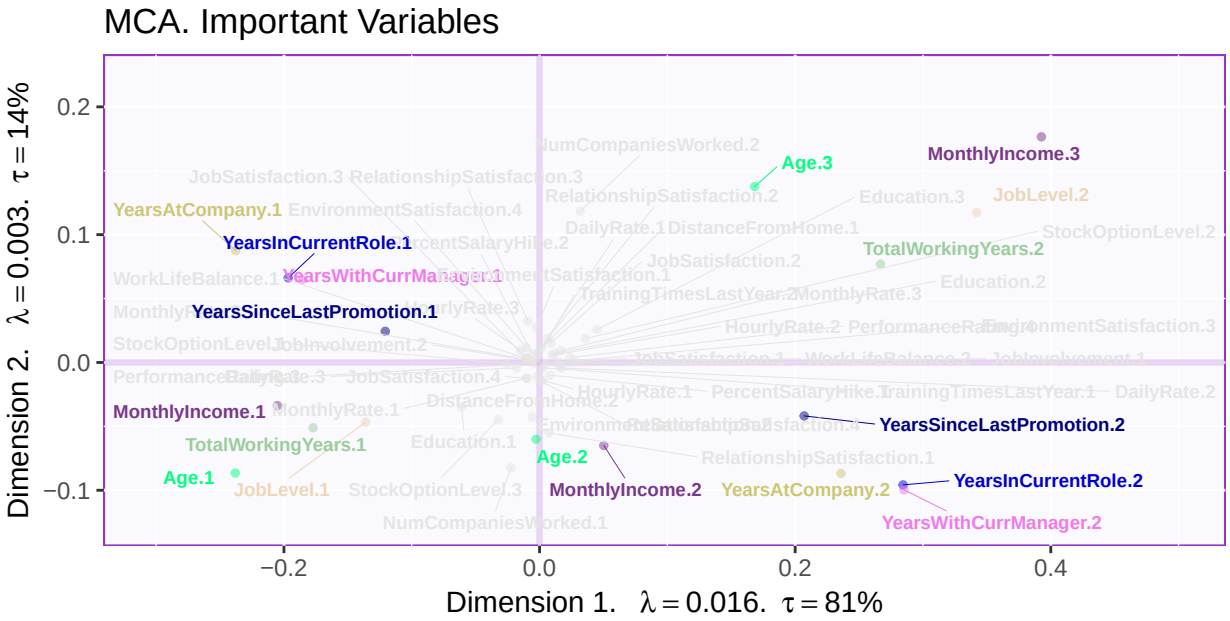


4.3 Variable Map for MCA

```
axis1 = 1
axis2 = 2
Fj1 <- resMCA.sym$ExPosition.Data$fj
col4Levels <- data4PCCAR::coloringLevels( rownames(resMCA.sym$ExPosition.Data$fj), color)
col4Labels <- col4Levels$color4Levels
# generate the set of maps
BaseMap.Fj <- createFactorMap(X = Fj1 , # resMCA$ExPosition.Data$fj,
                              axis1 = axis1, axis2 = axis2,
                              title = 'MCA. Variables',
                              col.points = col4Labels, cex = 1,
                              col.labels = col4Labels, text.cex = 2.5,
                              force = 2)
# add labels
labels4MCA <- createxyLabels.gen(x_axis = axis1, y_axis = axis2, lambda = resMCA.sym$ExPosition.Data$ei
tau = resMCA.sym$ExPosition.Data$t)
# make the maps
b0002.BaseMap.Fj <- BaseMap.Fj$zeMap + labels4MCA
b0003.BaseMapNoDot.Fj <- BaseMap.Fj$zeMap_background +
BaseMap.Fj$zeMap_text + labels4MCA
print(b0002.BaseMap.Fj)
```

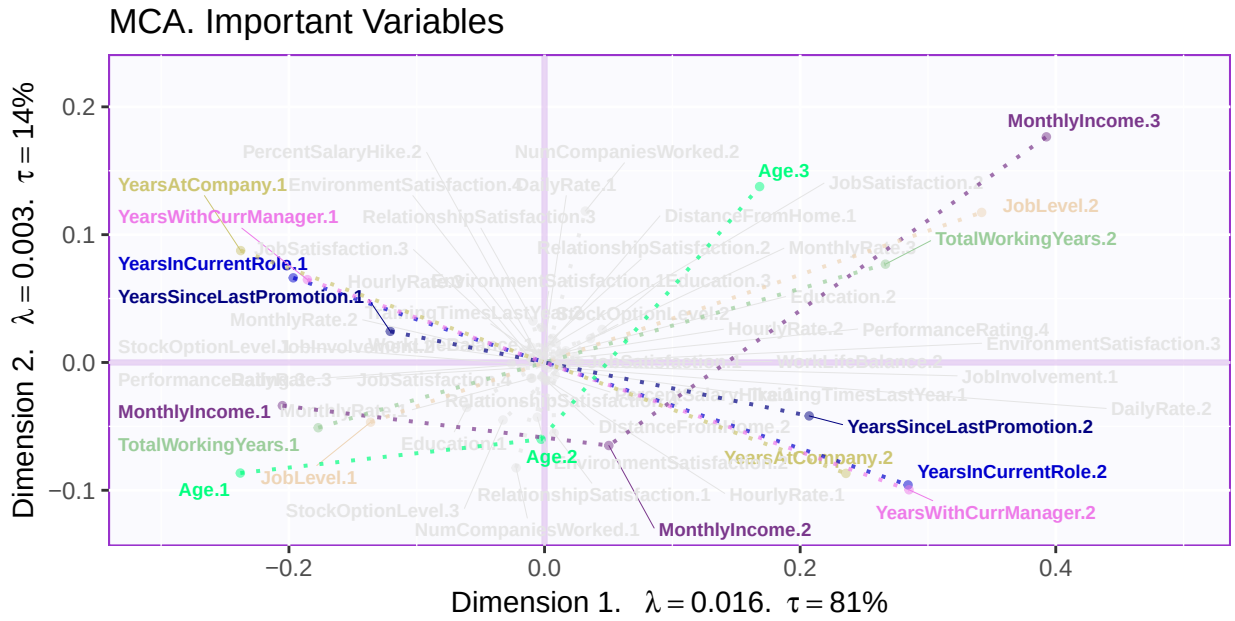



```
col4Levels.imp <- data4PCCAR::coloringLevels(rownames(Fj1), col4ImportantVar)
BaseMap.Fj.imp <- createFactorMap(X = Fj1 , # resMCA$ExPosition.Data$fj,
                                  axis1 = axis1, axis2 = axis2,
title = 'MCA. Important Variables', col.points = col4Levels.imp$color4Levels,
cex = 1,
col.labels = col4Levels.imp$color4Levels,
                                text.cex = 2.5,
force = 2)
b0010.BaseMap.Fj <- BaseMap.Fj.imp$zeMap + labels4MCA
print(b0010.BaseMap.Fj)
```



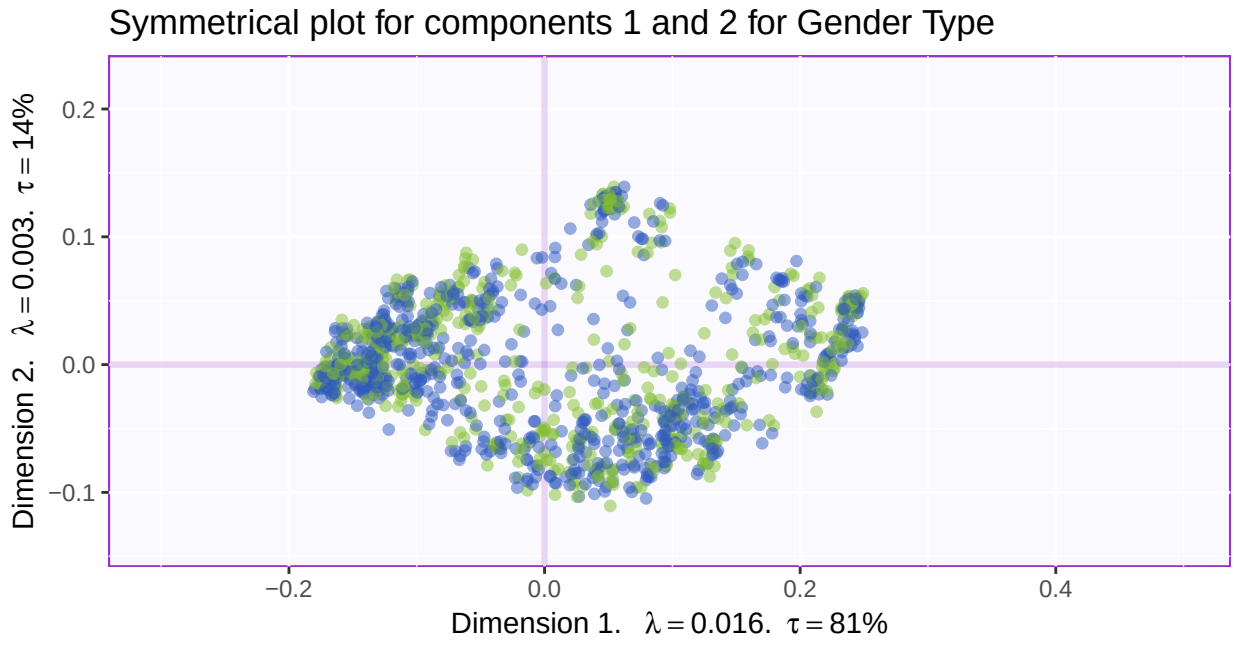
Map with important variables and lines

```
lines4J <- addLines4MCA(Fj1, col4Var = col4Levels.imp$color4Variables, size = .7)
b0020.BaseMap.Fj <- b0010.BaseMap.Fj + lines4J
print( b0020.BaseMap.Fj)
```



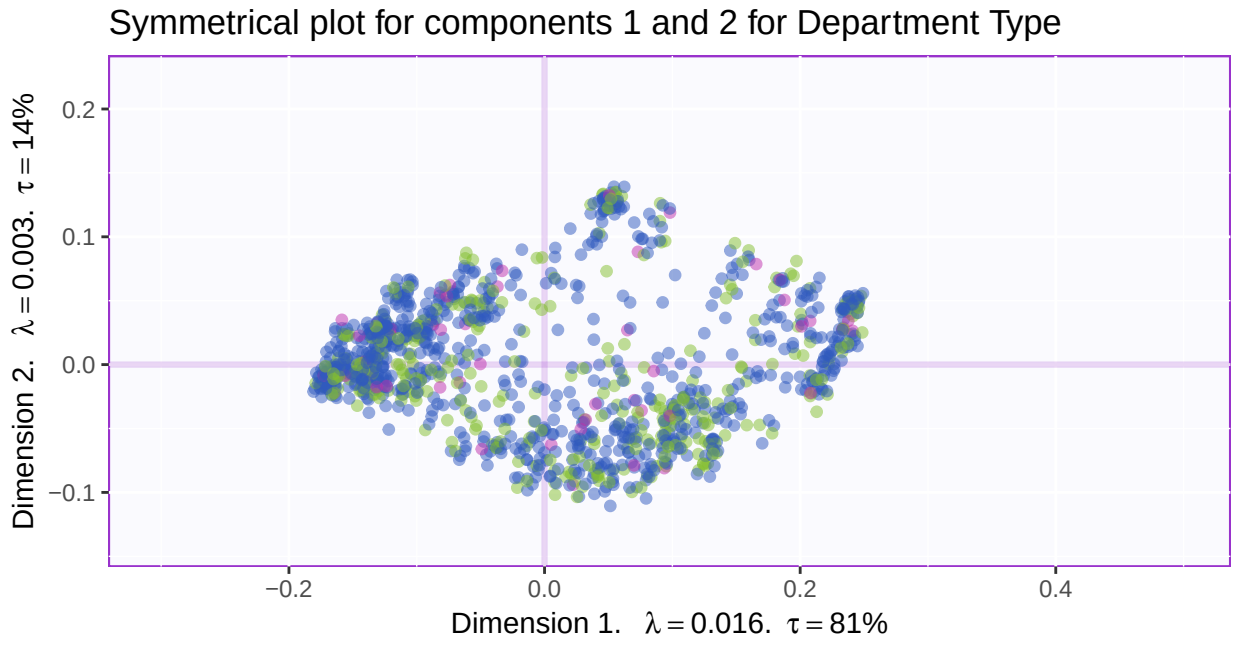
Biplot for symmetrical plot for component 1 and 2 for Gender datatype

```
col4J1 <- prettyGraphsColorSelection(NCOL(data1))
symMap1  <- createFactorMapIJ(resMCA.sym1$ExPosition.Data$fi,resMCA.sym1$ExPosition.Data$fj,
                              col.points.i = resMCA.sym1$Plotting.Data$fi.col,
                              col.points.j = "black",
                              col.labels.i = resMCA.sym1$Plotting.Data$fi.col ,
                              col.labels.j = "black" ,
                             cex.i = 2.5, pch.i = 20,
                              pch.j = 21,  cex.j = 2.5,text.cex.j =2, axis1 = 1,axis2 = 2, title = "Symm
                              alpha.axes = 0.2,alpha.points.i = 1)


labels4MCA1 <- createxyLabels(resCA = resMCA.sym1, x_axis = 1,y_axis = 2)

map.IJ.sym1 <- symMap1$baseMap + symMap1$I_points +labels4MCA1
print(map.IJ.sym1)
```



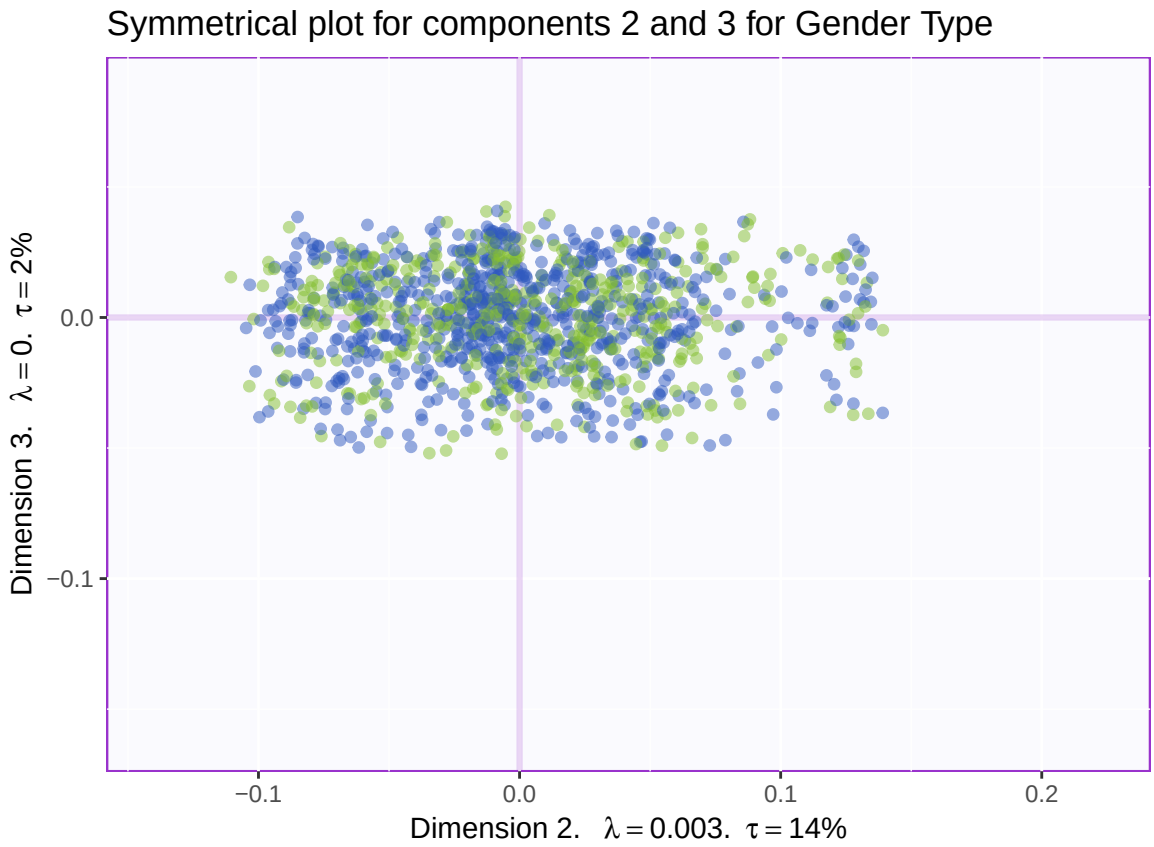
Biplot for symmetrical plot for component 1 and 2 for Department datatype

```
symMap2  <- createFactorMapIJ(resMCA.sym$ExPosition.Data$fi,resMCA.sym$ExPosition.Data$fj,
                              col.points.i = resMCA.sym$Plotting.Data$fi.col,
                              col.points.j = "black",
                              col.labels.i = resMCA.sym$Plotting.Data$fi.col ,
                              col.labels.j = "black" ,
                             cex.i = 2.5, pch.i = 20,
                              pch.j = 21,  text.cex.j =2, axis1 = 1,axis2 = 2, title = "Symmetrical plot
                              alpha.axes = 0.2,alpha.points.i = 1)

labels4MCA2 <- createxyLabels(resCA = resMCA.sym, x_axis = 1, y_axis = 2)

map.IJ.sym11 <- symMap2$baseMap + symMap2$I_points+labels4MCA2
print(map.IJ.sym11)
```



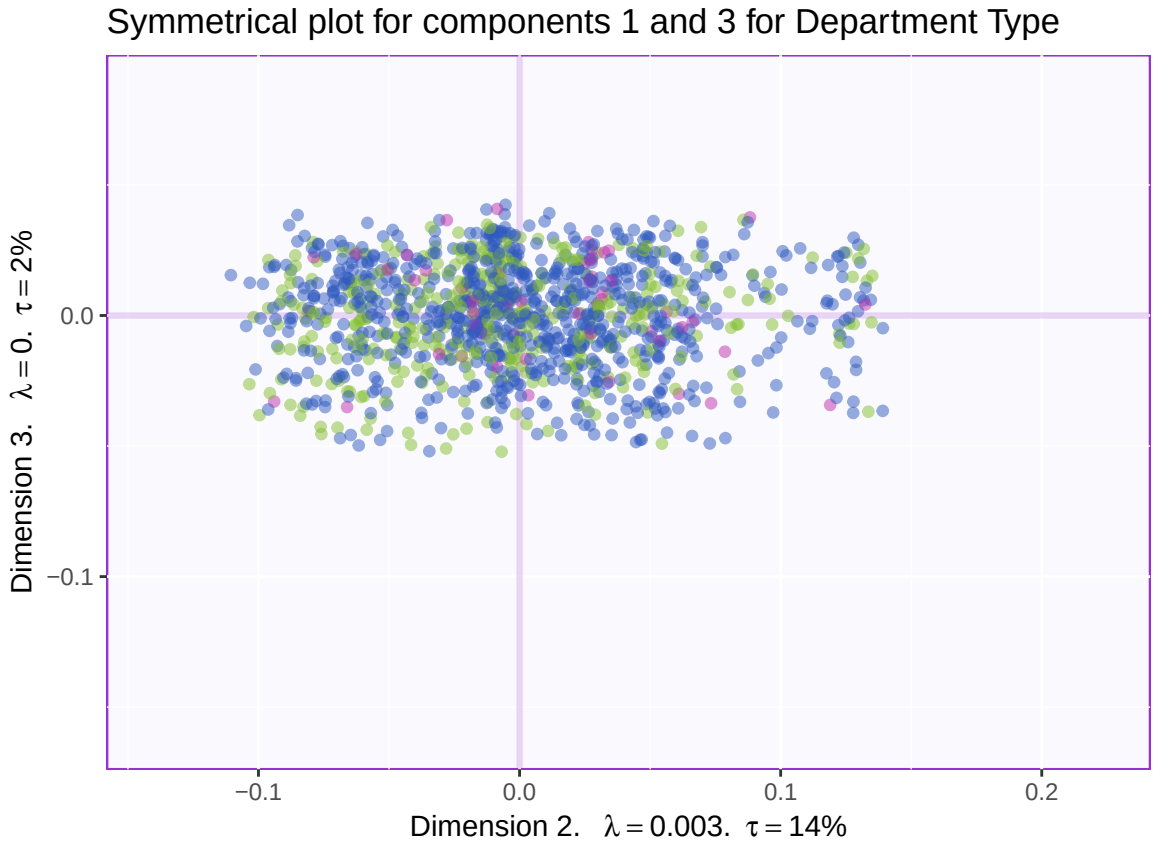
Biplot for symmetrical plot for component 2 and 3 for Gender datatype

```
symMap3  <- createFactorMapIJ(resMCA.sym1$ExPosition.Data$fi,resMCA.sym1$ExPosition.Data$fj,
                              col.points.i = resMCA.sym1$Plotting.Data$fi.col,
                              col.points.j = "black",
                              col.labels.i = resMCA.sym1$Plotting.Data$fi.col ,
                              col.labels.j = "black" ,
                             cex.i = 2.5, pch.i = 20,
                              pch.j = 21,  text.cex.j =2, axis1 = 2,axis2 = 3, title = "Symmetrical plot
                              alpha.axes = 0.2,alpha.points.i = 1)

labels4MCA3 <- createxyLabels(resCA = resMCA.sym1, x_axis = 2, y_axis = 3)

map.IJ.sym12 <- symMap3$baseMap + symMap3$I_points + labels4MCA3
print(map.IJ.sym12)
```



Biplot for symmetrical plot for component 2 and 3 for Department datatype

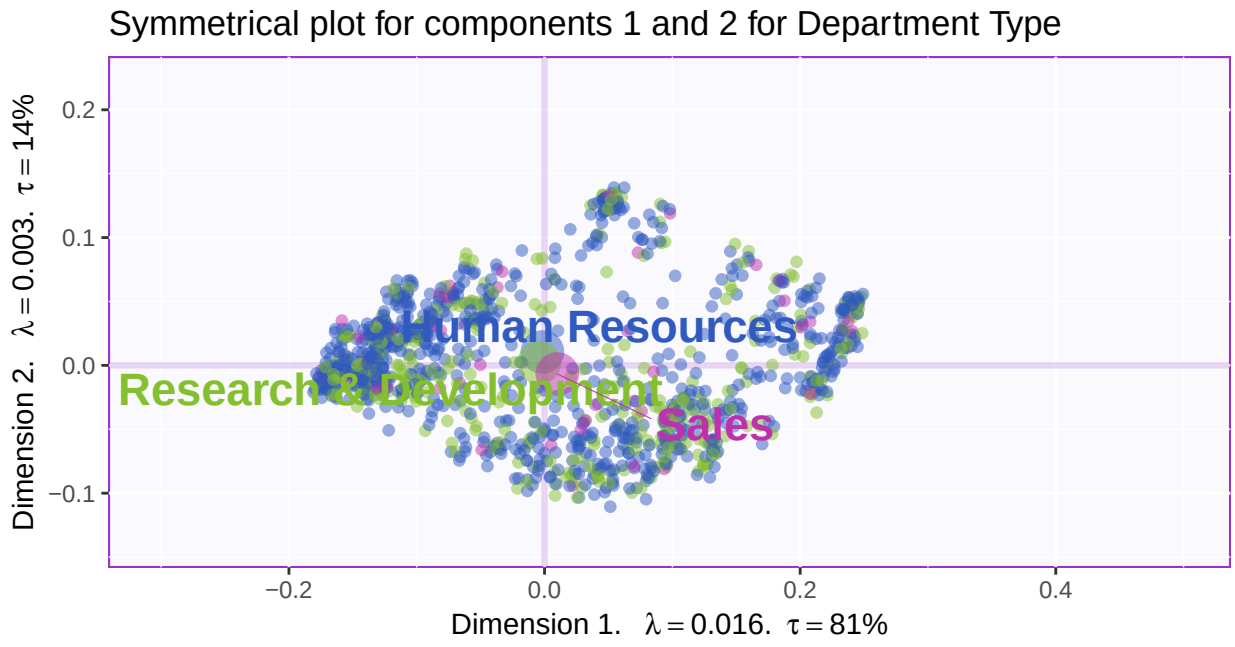
```
symMap4  <- createFactorMapIJ(resMCA.sym$ExPosition.Data$fi,resMCA.sym$ExPosition.Data$fj,
                              col.points.i = resMCA.sym$Plotting.Data$fi.col,
                              col.points.j = "black",
                              col.labels.i = resMCA.sym$Plotting.Data$fi.col ,
                              col.labels.j = "black" ,
                             cex.i = 2.5, pch.i = 20,
                              pch.j = 21,  text.cex.j =2, axis1 = 2,axis2 = 3, title = "Symmetrical plot
                              alpha.axes = 0.2,alpha.points.i = 1)

labels4MCA4 <- createxyLabels(resCA = resMCA.sym, x_axis = 2,y_axis = 3)

map.IJ.sym13 <- symMap4$baseMap + symMap4$I_points + labels4MCA4
print(map.IJ.sym13)
```



4.4 Bootstrap Interval

```
constraints.mca <- minmaxHelper(mat1 = resMCA.sym$ExPosition.Data$fi, mat2  = resMCA.sym$ExPosition.Data
BootCube.Gr1 <- Boot4Mean(resMCA.asym$ExPosition.Data$fi,
                                    design = my_data$Department,
                                    niter = 100,
                                    suppressProgressBar = TRUE)
#_____________________________________________________________________
# Bootstrap ratios ----
bootRatios.Gr1 <- boot.ratio.test(BootCube.Gr1$BootCube)
#**********************************************************************
# Mean Map
#  create the map for the means
#  get the means by groups

dataMeans1 <- getMeans(resMCA.asym$ExPosition.Data$fi, my_data$Department)
# a vector of color for the means
col4data1 <- resMCA.asym$Plotting.Data$fi.col
col4Means1 <- unique(col4data1)
# the map
MapGroup1 <- createFactorMap(dataMeans1,
                                      # use the constraint from the main map
                                      constraints = constraints.mca,
                                      col.points = col4Means1,
```

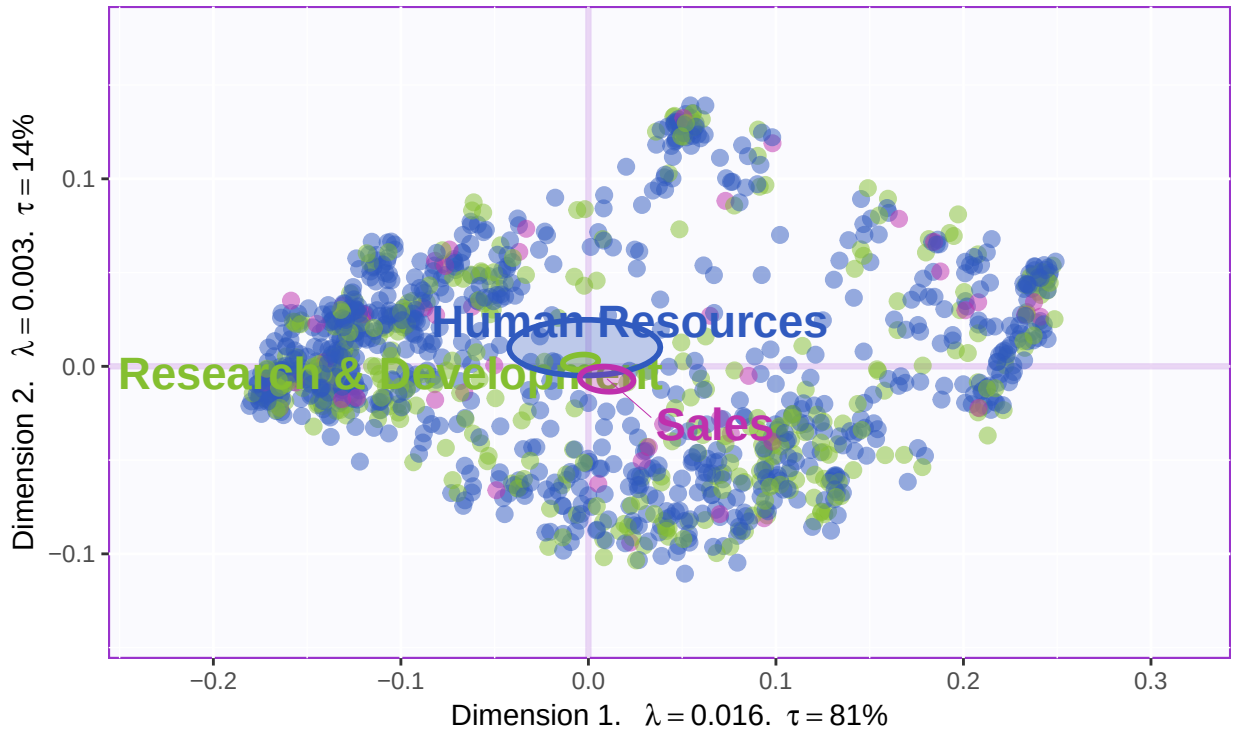
```
                                    cex = 7,  # size of the dot (bigger)
                                    col.labels = col4Means1,
                                    text.cex = 6)
# The map with observations and group means
a003.Map.I.withMeans1 <- map.IJ.sym11+
  MapGroup1$zeMap_dots + MapGroup1$zeMap_text
print(a003.Map.I.withMeans1)
```



C.I for the mean of the factor Map I

```
my_data1.Imap1 <- PTCA4CATA::createFactorMap(
  resMCA.sym$ExPosition.Data$fi,
  col.points = resMCA.sym$Plotting.Data$fi.col,
  display.labels = FALSE,
  alpha.points = .5
)
label4Map1 <- createxyLabels.gen(1,2,
                                 lambda =resMCA.sym$ExPosition.Data$eigs,
                                 tau = resMCA.sym$ExPosition.Data$t)
a002.Map.I1 <- my_data1.Imap1$zeMap + label4Map1

#_____________________________________________________________________
# Create the ellipses
# Bootstrapped CI ----
#_____________________________________________________________________
# Create Confidence Interval Plots
```

```
# use function MakeCIEllipses from package PTCA4CATA
GraphElli1 <- PTCA4CATA::MakeCIEllipses(BootCube.Gr1$BootCube[,1:2,],
                                        names.of.factors = c("Dimension 1","Dimension 2"),
                                        col = col4Means1,
                                        p.level = .95
)
#-------------------------------------------------------------------
# create the I-map with Observations, means and confidence intervals
#
a004.Map.I.withCI1 <-  a002.Map.I1 + MapGroup1$zeMap_text +  GraphElli1
#-------------------------------------------------------------------
# plot it!
print(a004.Map.I.withCI1)
```

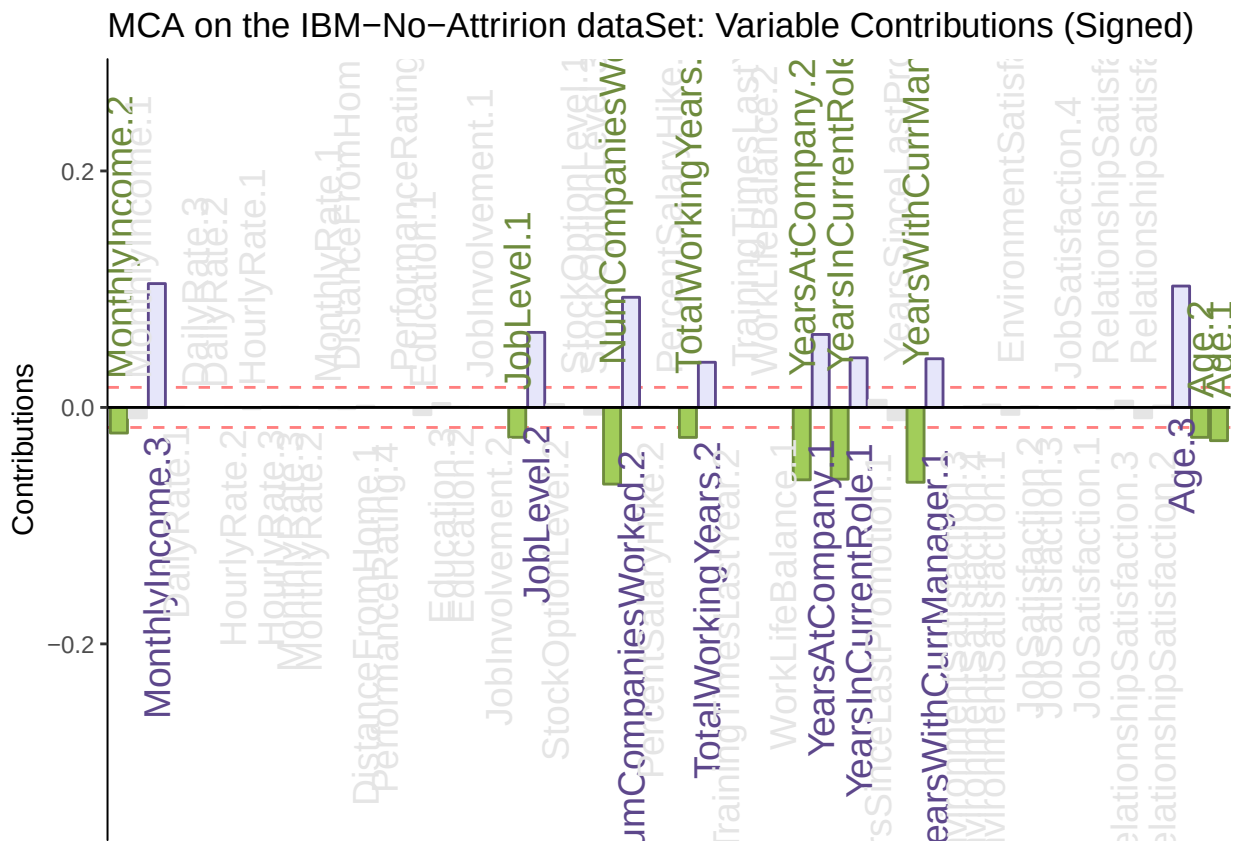


4.5 Contribution for variables

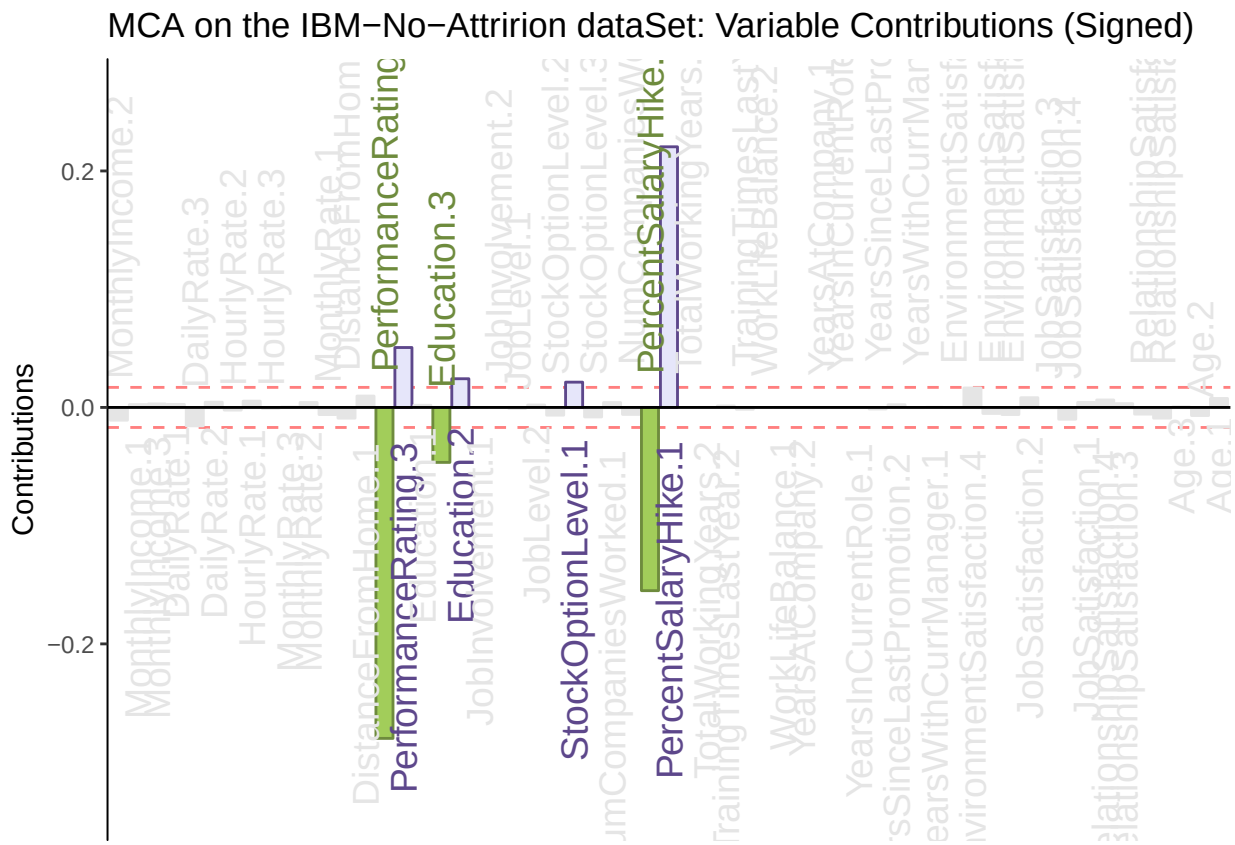
```
signed.ctrJ2 <- resMCA.sym$ExPosition.Data$cj * sign(resMCA.sym$ExPosition.Data$fj)
b003.ctrJ.s.111 <- PrettyBarPlot2(signed.ctrJ2[,1],
                                  threshold = 1 / NROW(signed.ctrJ2),
                                  font.size = 5,
                                 # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                  main = 'MCA on the IBM-No-Attririon data Set: Variable Contributions (S
                                  ylab = 'Contributions',
                                  ylim = c(1.2*min(signed.ctrJ2), 1.2*max(signed.ctrJ2))
```

```
)
print(b003.ctrJ.s.111)
```


```
b004.ctrJ.s.211 <- PrettyBarPlot2(signed.ctrJ2[,2],
                                  threshold = 1 / NROW(signed.ctrJ2),
                                  font.size = 5,
                                  # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                  main = 'MCA on the IBM-No-Attririon dataSet: Variable Contributions (Sig
                                  ylab = 'Contributions',
                                  ylim = c(1.2*min(signed.ctrJ2), 1.2*max(signed.ctrJ2))
)
print(b004.ctrJ.s.211)
```

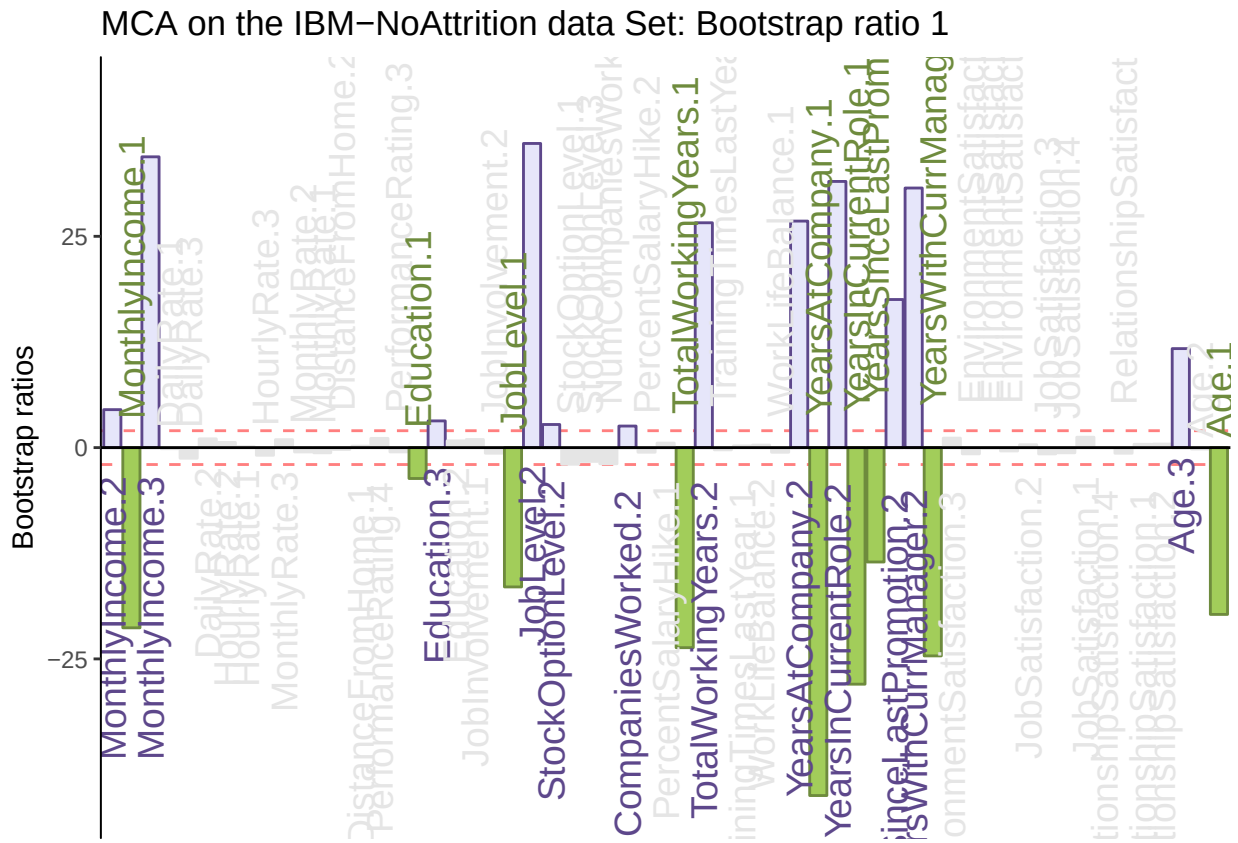



```
b004.ctrJ.s.311 <- PrettyBarPlot2(signed.ctrJ2[,3],
                                  threshold = 1 / NROW(signed.ctrJ2),
                                  font.size = 5,
                                 # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                  main = 'MCA on the IBM-No-Attririon dataSet: Variable Contributions (Si
                                  ylab = 'Contributions',
                                  ylim = c(1.2*min(signed.ctrJ2), 1.2*max(signed.ctrJ2))
)
print(b004.ctrJ.s.311)
```

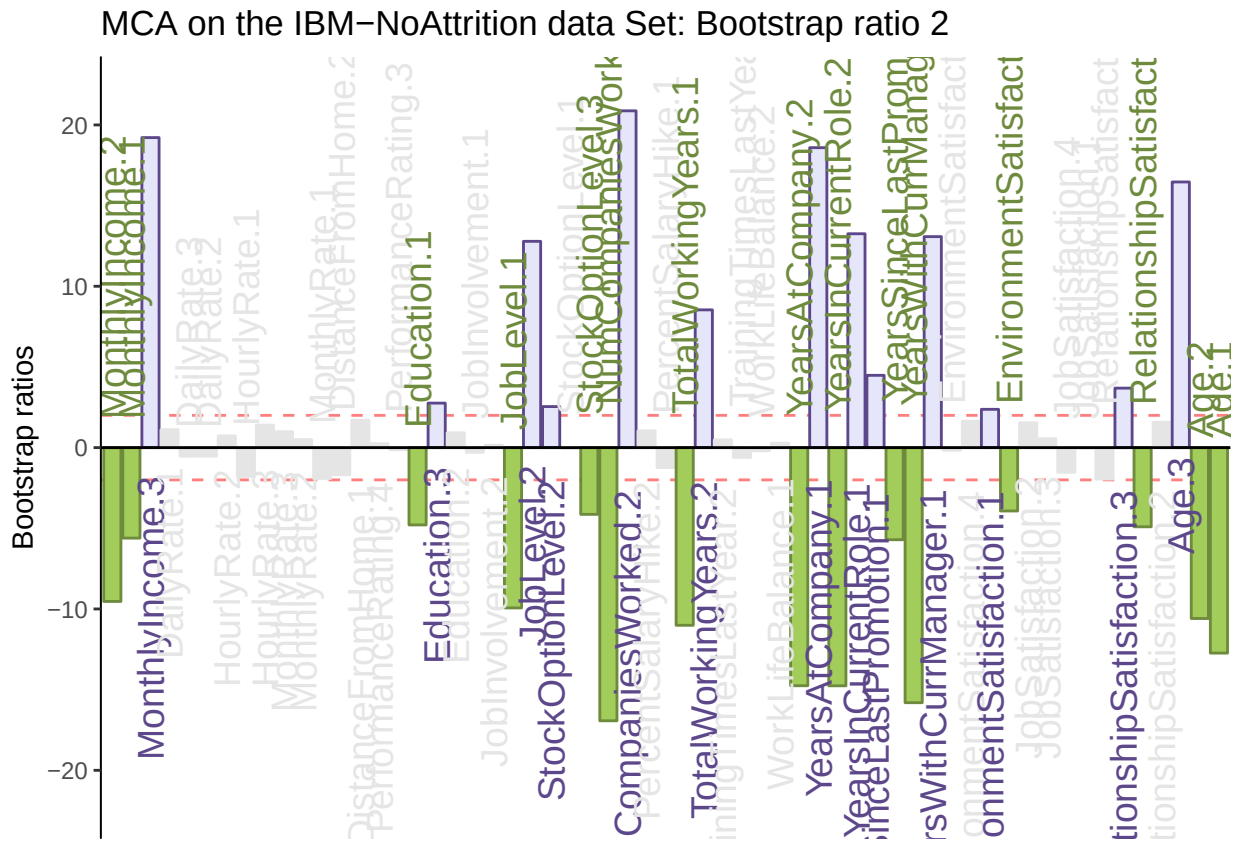


4.6 Bootstrap Ratios for Variables

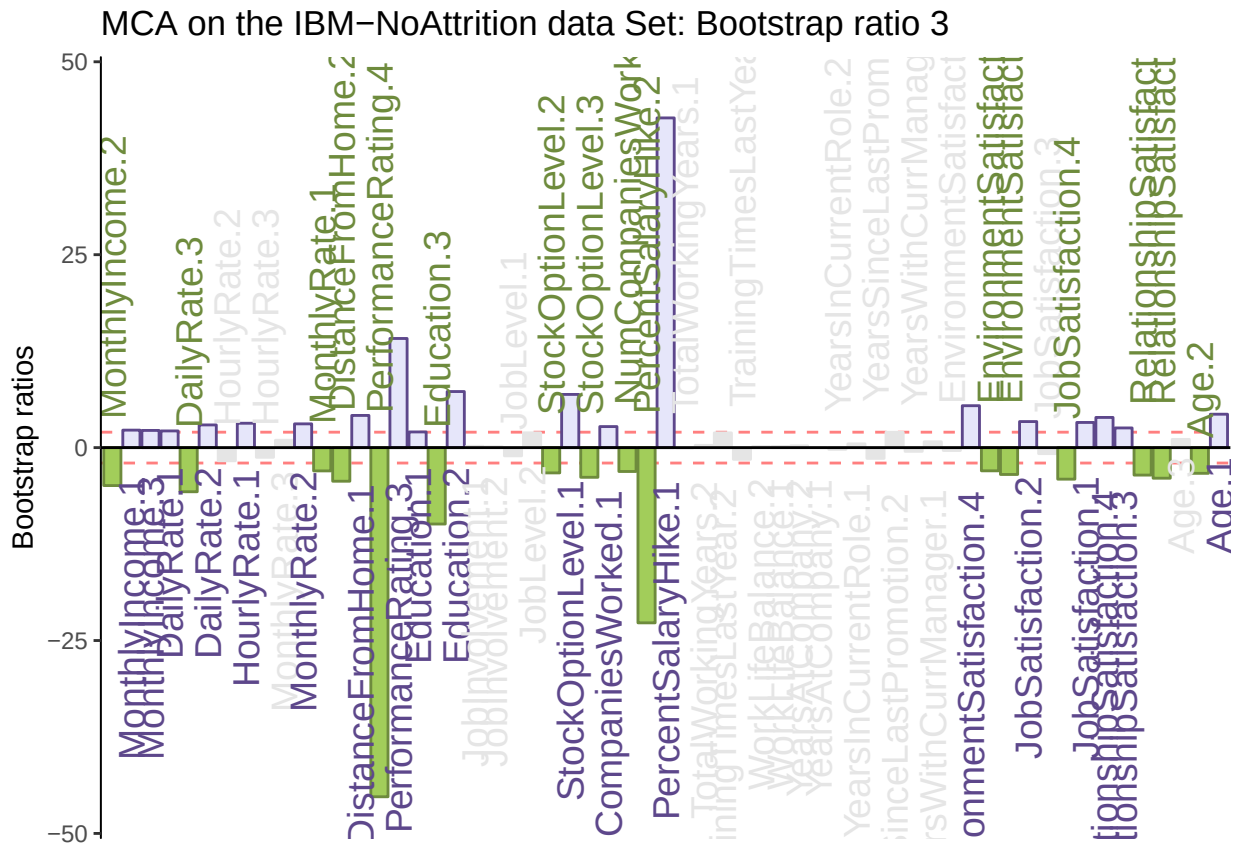
```
BR2 <- resMCA.inf$Inference.Data$fj.boots$tests$boot.ratios
ba001.BR111 <- PrettyBarPlot2(BR2[,1],
                              threshold = 2,
                              font.size = 5,
                              #color4bar = gplots::col2hex(col4J.ibm),
                              main = paste0( 'MCA on the IBM-NoAttrition data Set: Bootstrap ratio ',1),
                              ylab = 'Bootstrap ratios'
                              #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR111)
```




```
#
ba002.BR211 <- PrettyBarPlot2(BR2[,2],
                              threshold = 2,
                              font.size = 5,
                              #color4bar = gplots::col2hex(col4J.ibm),
                              main = paste0(
                                'MCA on the IBM-NoAttrition data Set: Bootstrap ratio ',2),
                              ylab = 'Bootstrap ratios'
)
print(ba002.BR211)
```




```
ba002.BR311 <- PrettyBarPlot2(BR2[,3],
                              threshold = 2,
                              font.size = 5,
                              main = paste0(
                                'MCA on the IBM-NoAttrition data Set: Bootstrap ratio ',3),
                              ylab = 'Bootstrap ratios'
)
print(ba002.BR311)
```



Summary

- For loadings we see that Monthly Income, Job Level, Number of Years are the important variables for the first component and for second component Percentage Salary, Percentage Salary Hike are an important component
- For the factor plot the gender type does not show any significant inference as the mean are almost overlapping each other and for the department type similar results can be seen as R&D, Sales and HR all intersect their mean confidence Intervals.

```
options(knitr.duplicate.label = 'allow')
```

5 Correspondance Analysis

Correspondence Analysis is a generalized principal component analysis tailored for the analysis of qualitative data. Also commonly known as reciprocal averaging is a multivariate statistical technique proposed by Herman Otto. It is conceptually similar to principal component analysis, but applies to categorical rather than continuous data. In a similar manner to principal component analysis, it provides a means of displaying or summarising a set of data in two-dimensional graphical form.This is a descriptive/exploratory technique designed to analyze simple two-way and multi-way tables containing some measure of correspondence between the rows and columns. The results provide information which is similar in nature to those produced by Factor Analysis techniques, and they allow you to explore the structure of categorical variables included in the table.Originally, ca was created to analyze contingency tables, but, ca is so versatile that it is used with a lot of other data table types.

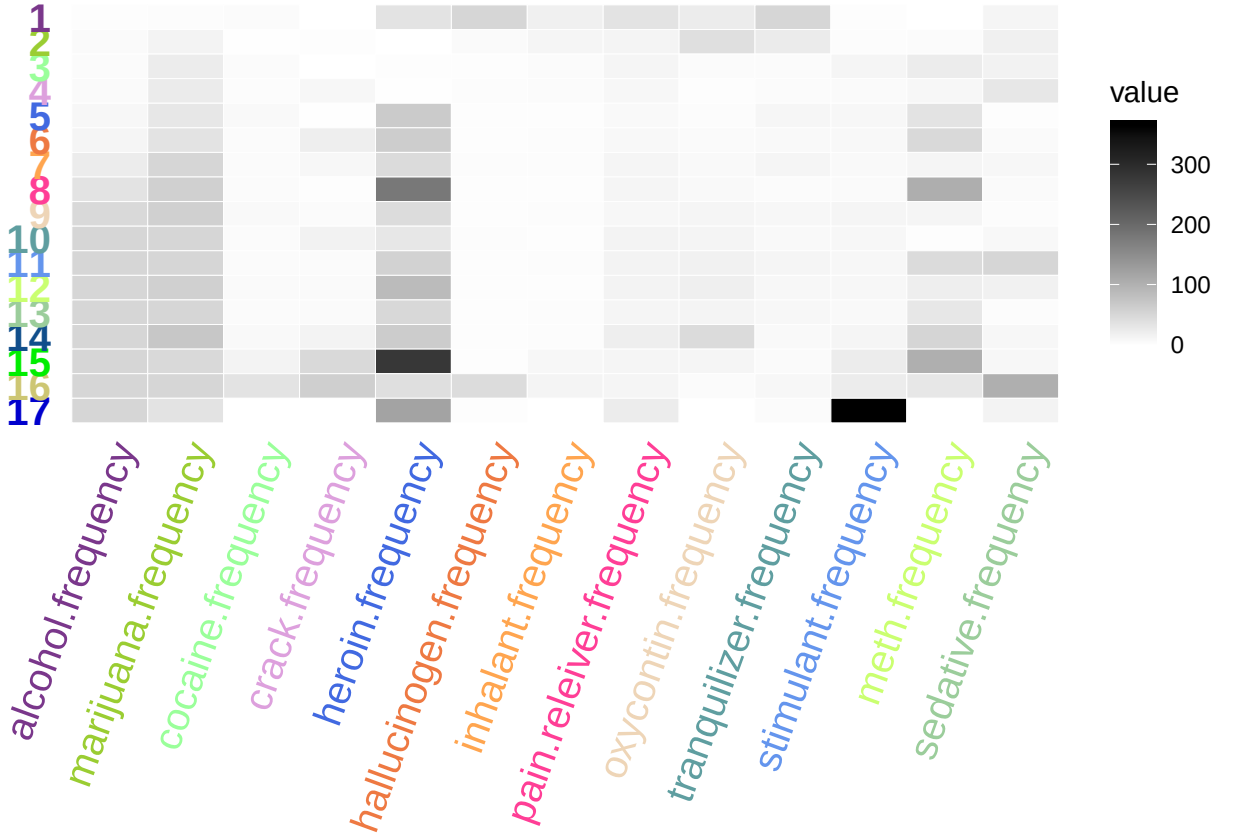
Dataset

The dataset is about Drug-use-by age which consists of 17 observations and 28 variables.This dataset describes about the variation of drugs people consume across all age groups. The dataset consists of both the frequency and the percentage of the drug. As,in statistics, a contingency table (also known as a cross tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. So, I dropped the variables showing the percentage count of drug use and kept only the variables displaying the median of the frequency count of drug use across all age group.

Heat Map

A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors.

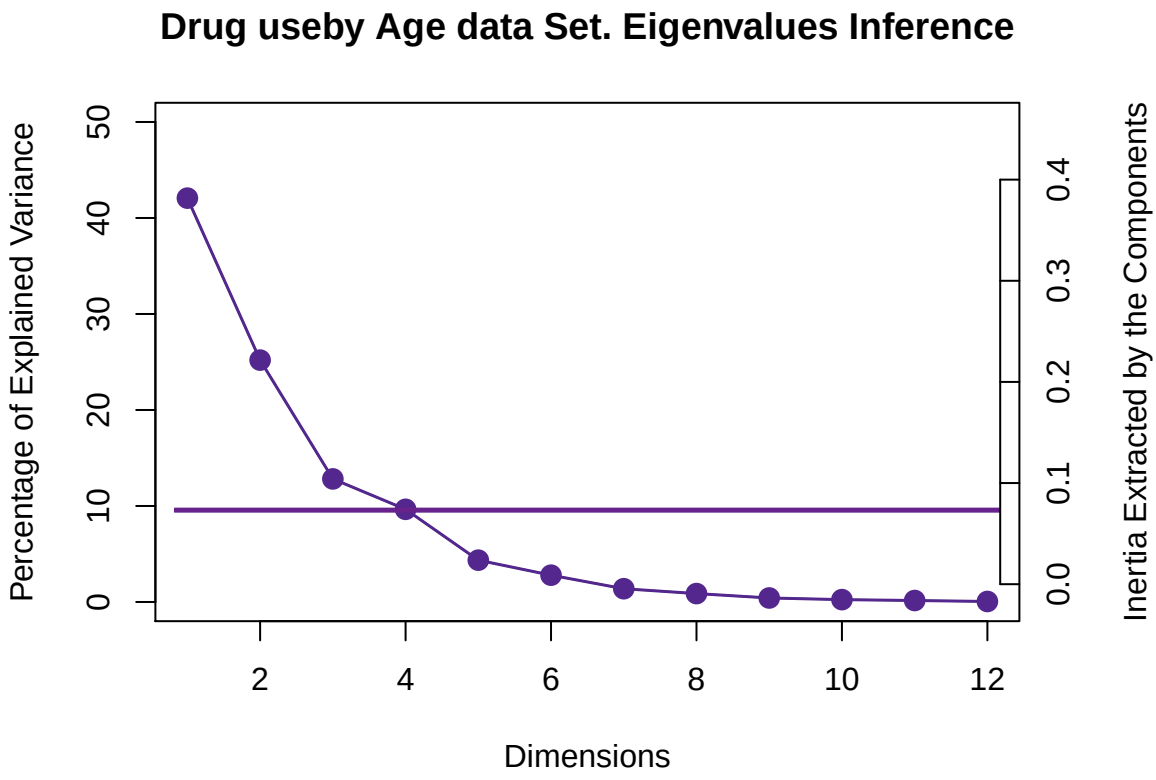
```
col4J.drugs <- prettyGraphsColorSelection(NCOL(x))
heatMapIJ <- makeggHeatMap4CT(x,
                  colorAttributes = col4J.drugs,
                  fontSize.x = 15)
print(heatMapIJ)
```



Scree plot

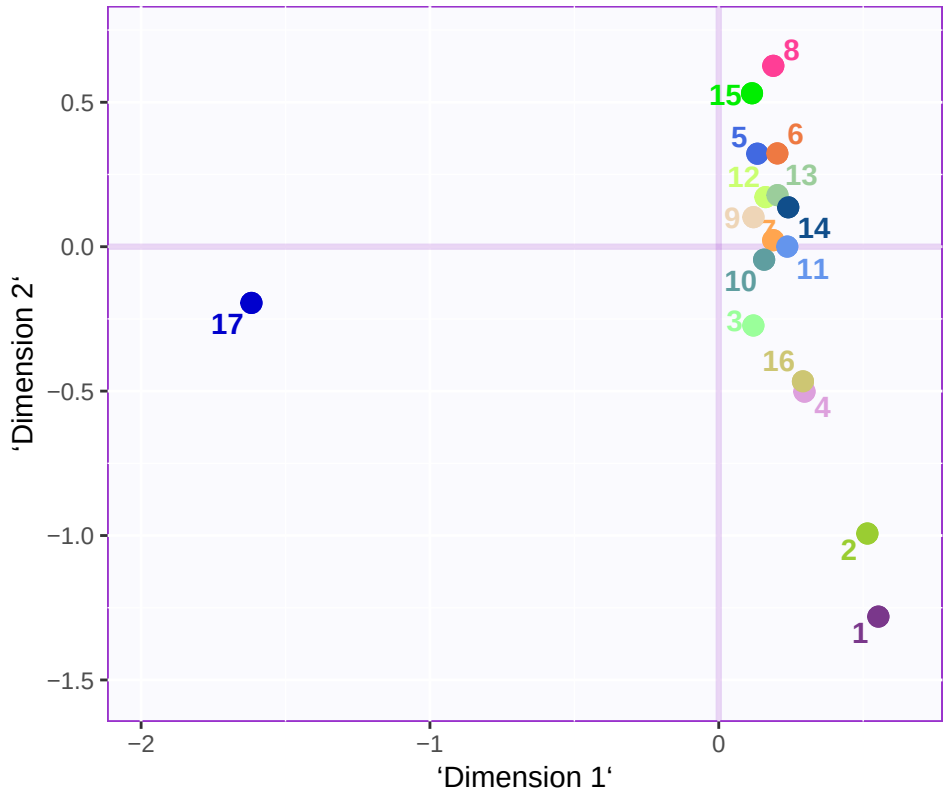
A Scree Plot is a simple line segment plot that shows the fraction of total variance in the data as explained or represented by each PC.(`In the PCA literature, the plot is called a 'Scree' Plot because it often looks like a 'scree' slope, where rocks have fallen down and accumulated on the side of a mountain.`)The scree plot shows the eigenvalues, the amount of information on each component. The number of components (the dimensionality of the factor space) is min(nrow(DATA), ncol(DATA)) minus 1.The scree plot is used to determine how many of the components should be interpreted. * `plot` draws the line that connects all data points by `type = "l"` * The first `points` function draws round purple dots. * The second `points` function draws black circles around the dots (just to make it prettier).

```
library(PTCA4CATA)
PlotScree(ev = resCA.sym$ExPosition.Data$eigs,
          p.ev =  rescainf$Inference.Data$components$p.vals,
          title = 'Drug useby Age data Set. Eigenvalues Inference',
          plotKaiser = TRUE
)
```

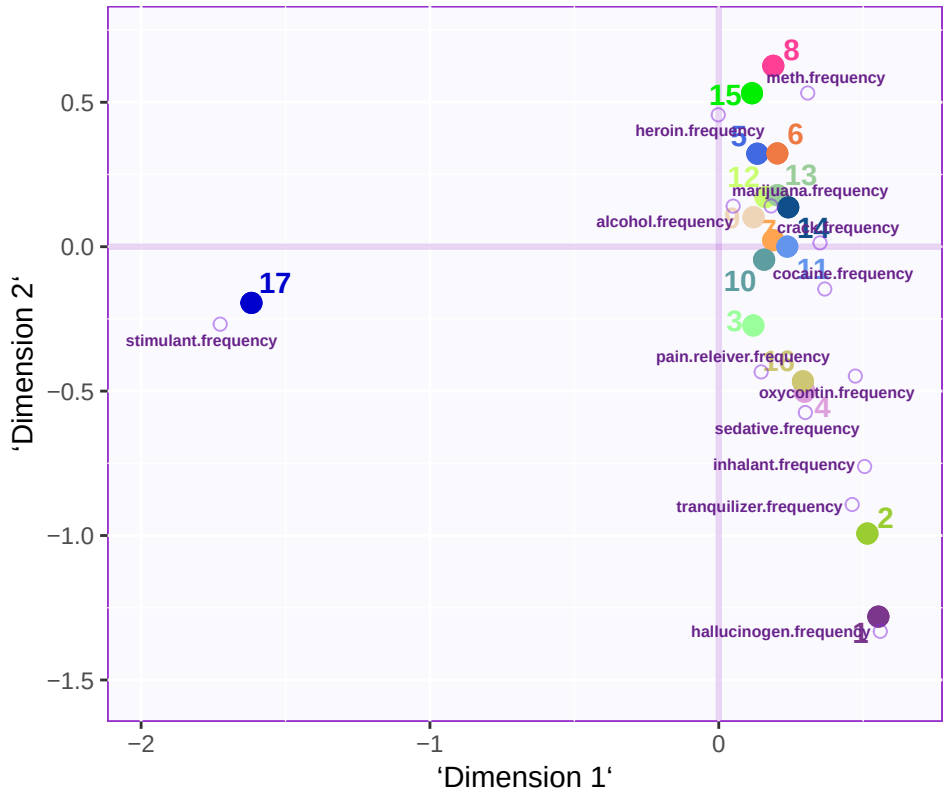


5.1 Factor Map for Symmetrical Graph

```
baseMap.i1 <- createFactorMap(Fi, constraints = constraints.sym,
                              col.points = color4drugs,
                             col.labels = color4drugs , cex = 5, text.cex = 4, pch = 20,
                             display.labels = TRUE , alpha.axes = 0.2,alpha.points = 1
)

print(baseMap.i1$zeMap + baseMap.i1$zeMap_dots)
```

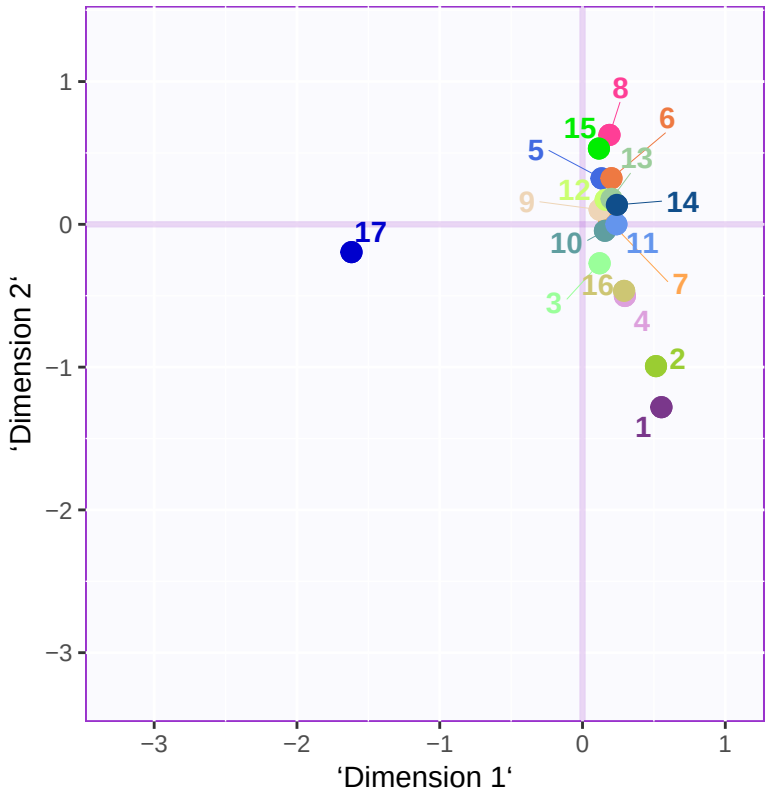



```
baseMap.j1 <- createFactorMap(Fj, constraints = constraints.sym,
                              color.points = "lightgreen",text.cex = 2,cex = 2,pch = 21)
print(baseMap.i1$zeMap + baseMap.i1$zeMap_dots + baseMap.j1$zeMap_dots + baseMap.j1$zeMap_text)
```

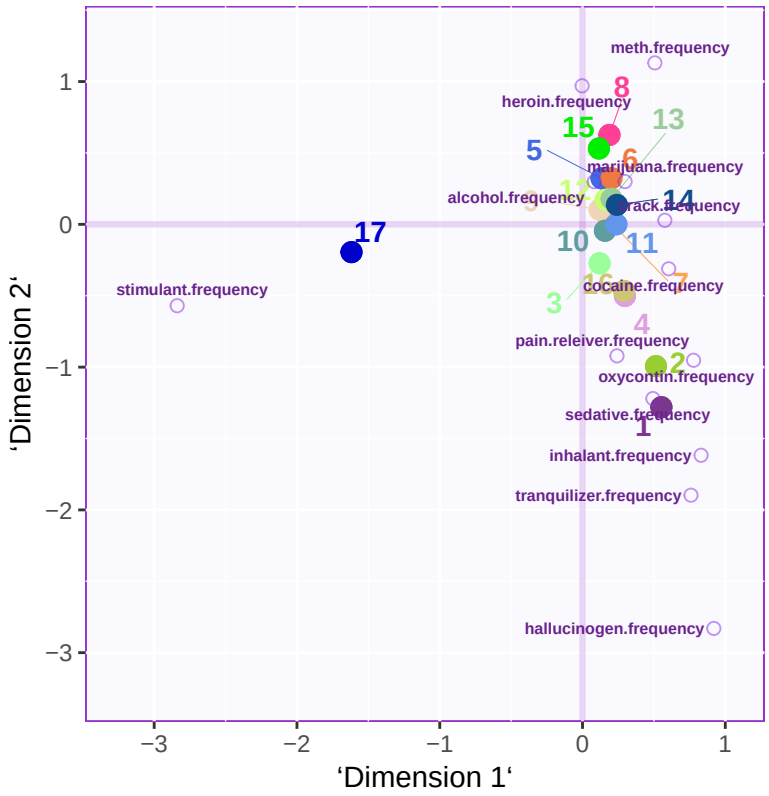


5.2 Factor Map for Asymmetrical Graph

```
baseMap.i2 <- createFactorMap(Fi, constraints = constraints.asym,
                              col.points = color4drugs,
                             col.labels = color4drugs , cex = 5, text.cex = 4, pch = 20,
                             display.labels = TRUE , alpha.axes = 0.2,alpha.points = 1
)
print(baseMap.i2$zeMap + baseMap.i2$zeMap_dots )
```

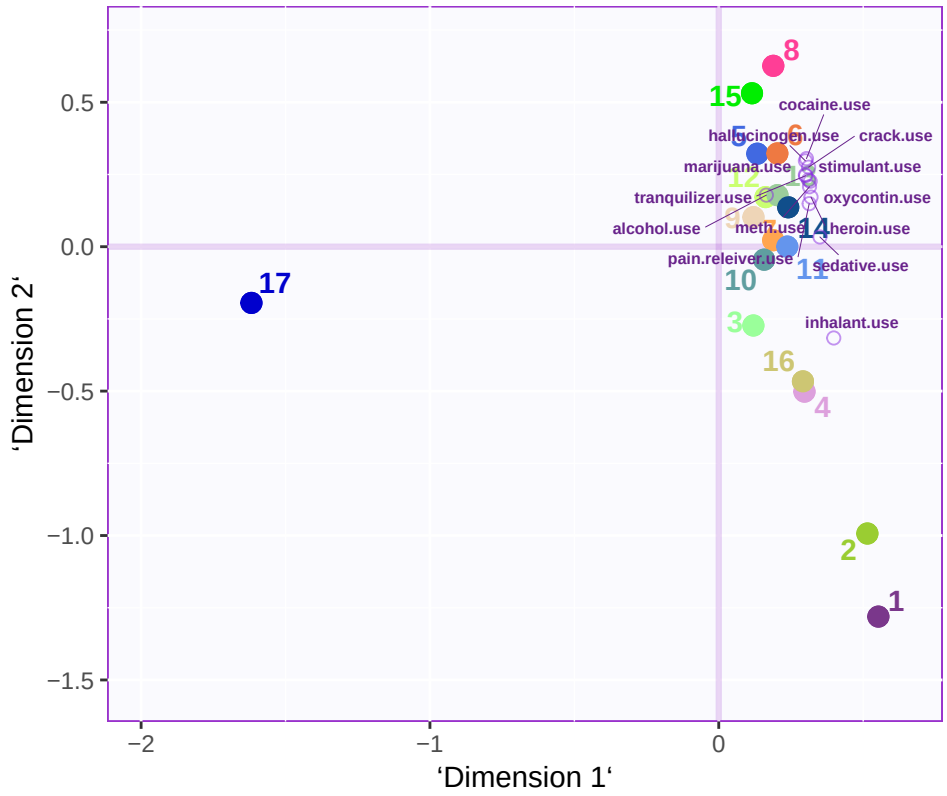



```
baseMap.j2 <- createFactorMap(Fj.a, constraints = constraints.asym,
                              color.points = "lightgreen",text.cex = 2,cex = 2,pch = 21)

print(baseMap.i2$zeMap + baseMap.i2$zeMap_dots+ baseMap.j2$zeMap_dots + baseMap.j2$zeMap_text)
```



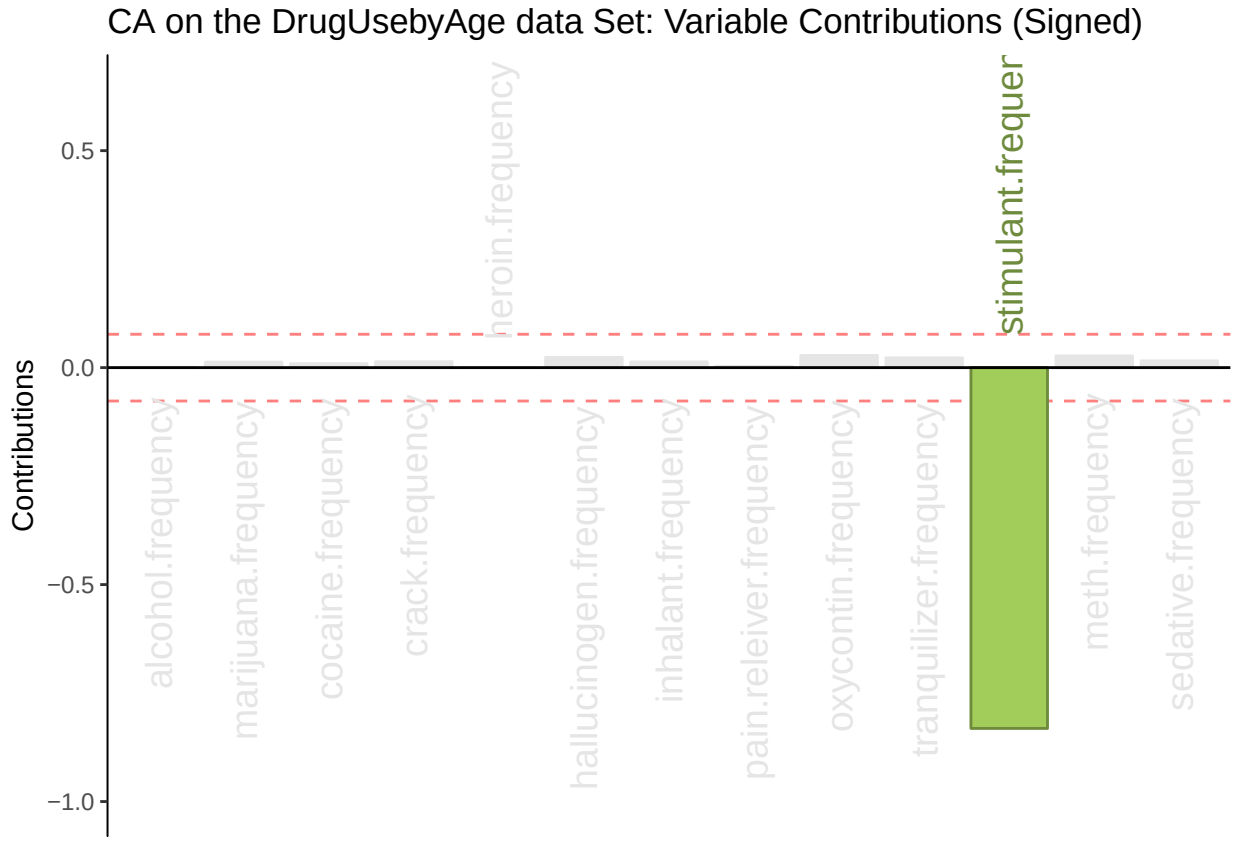
Factor Map for Supplementary Observations

```
HA.sup <- supplementaryRows(SUP.DATA = datasupp, res = resCA.sym)
HA.sup1 <- supplementaryCols(SUP.DATA = datasupp, res = resCA.sym)
baseMap.i11 <- createFactorMap(HA.sup1$fjj, constraints = constraints.sup,
                               color.points = "lightgreen",text.cex = 2,cex = 2,pch = 21)
print(baseMap.i1$zeMap + baseMap.i1$zeMap_dots + baseMap.i11$zeMap_dots + baseMap.i11$zeMap_text)
```

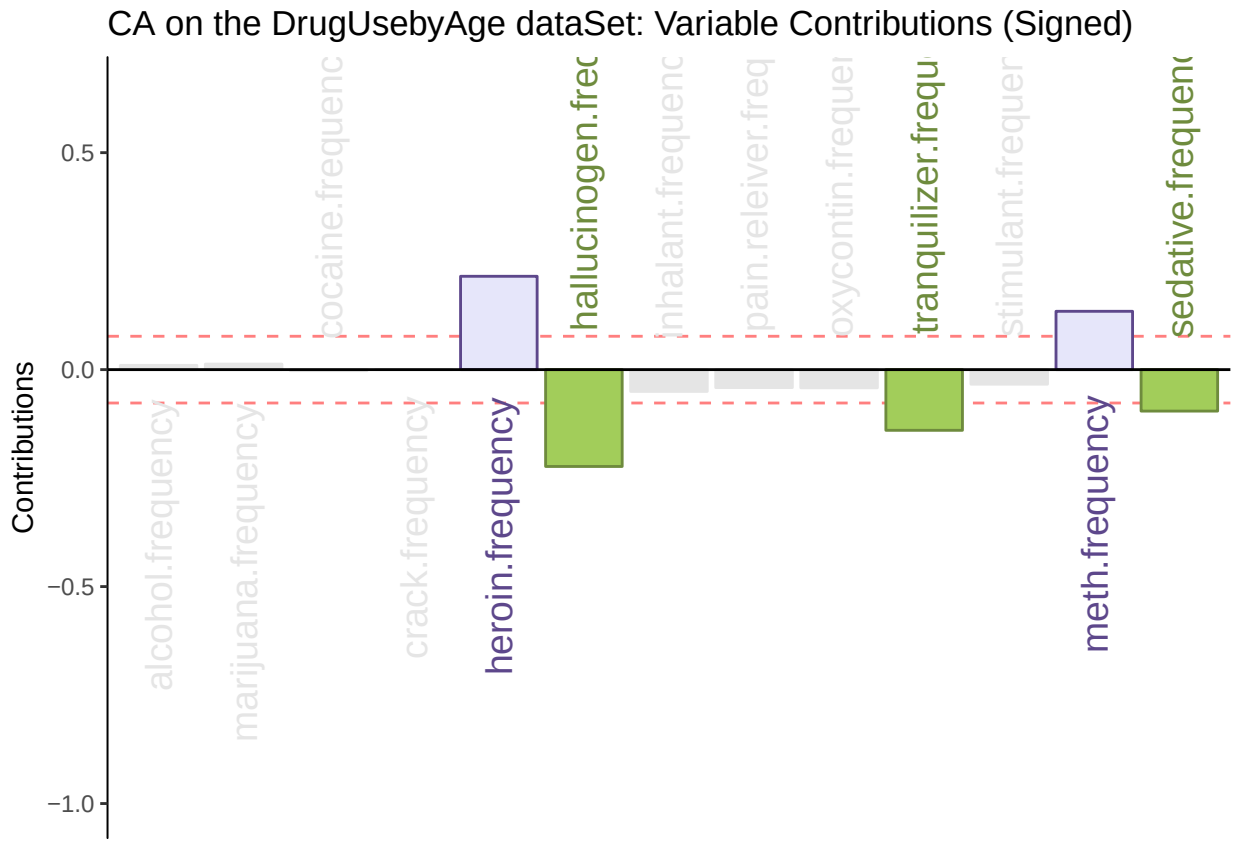


5.3 Contribution Bars for variables

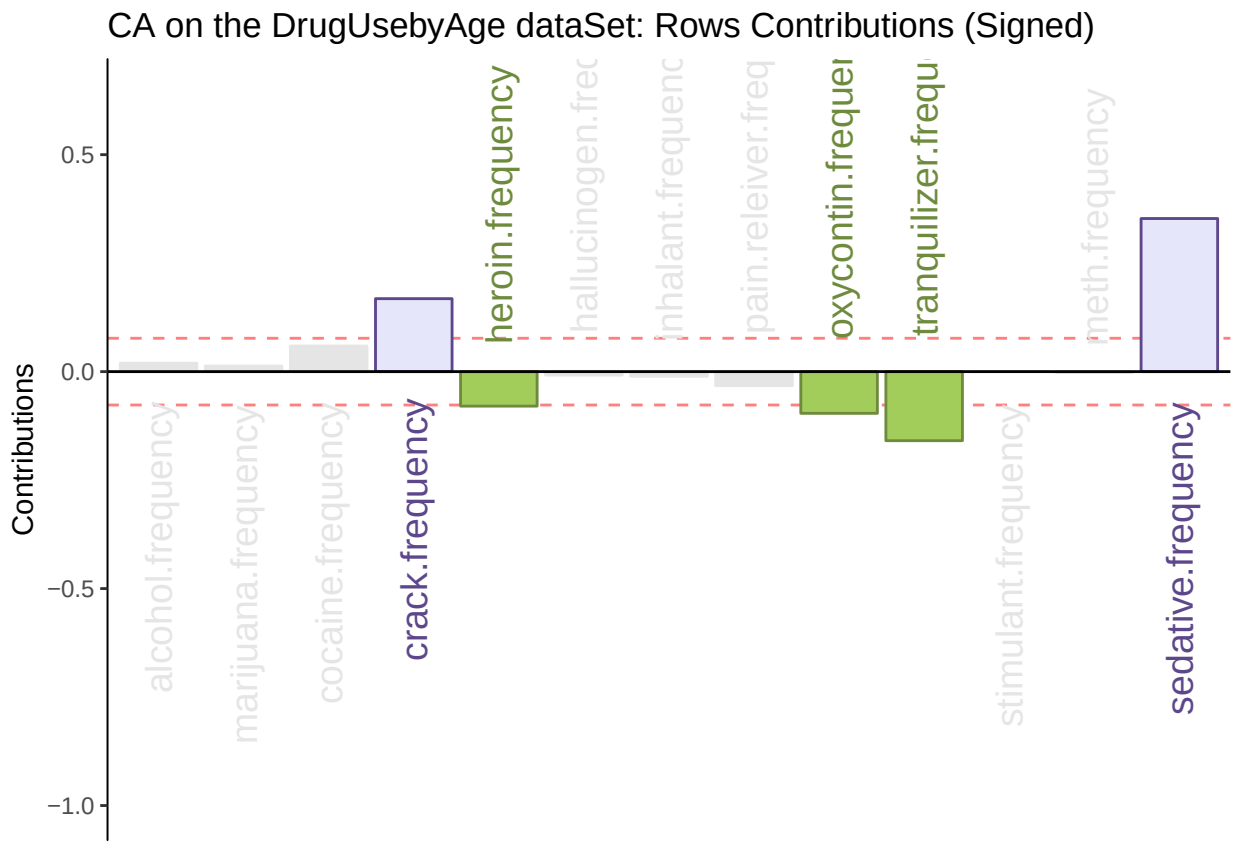
```
signed.ctrJ1 <- resCA.sym$ExPosition.Data$cj * sign(resCA.sym$ExPosition.Data$fj)
b003.ctrJ.s.11 <- PrettyBarPlot2(signed.ctrJ1[,1],
                                 threshold = 1 / NROW(signed.ctrJ1),
                                 font.size = 5,
                                 #color4bar = gplots::col2hex(color4drugs), # we need hex code
                                 main = 'CA on the DrugUsebyAge data Set: Variable Contributions (Signed
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ1), 1.2*max(signed.ctrJ1))
)
print(b003.ctrJ.s.11)
```




```
b004.ctrJ.s.21 <- PrettyBarPlot2(signed.ctrJ1[,2],
                                 threshold = 1 / NROW(signed.ctrJ1),
                                 font.size = 5,
                                 #color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'CA on the DrugUsebyAge dataSet: Variable Contributions (Signed)',
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ1), 1.2*max(signed.ctrJ1))
)
print(b004.ctrJ.s.21)
```



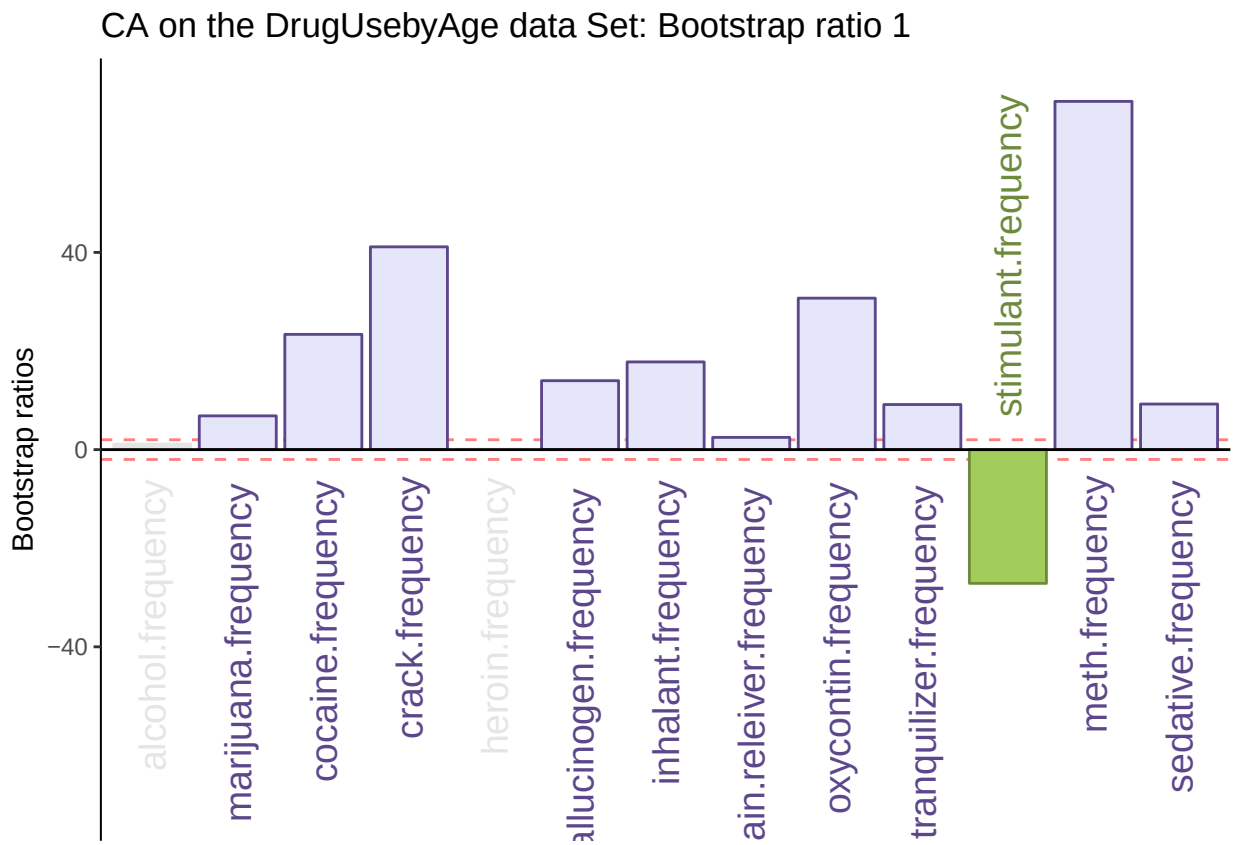

```
b004.ctrJ.s.31 <- PrettyBarPlot2(signed.ctrJ1[,3],
                                 threshold = 1 / NROW(signed.ctrJ1),
                                 font.size = 5,
                                #color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'CA on the DrugUsebyAge dataSet: Rows Contributions (Signed)',
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ1), 1.2*max(signed.ctrJ1))
)
print(b004.ctrJ.s.31)
```



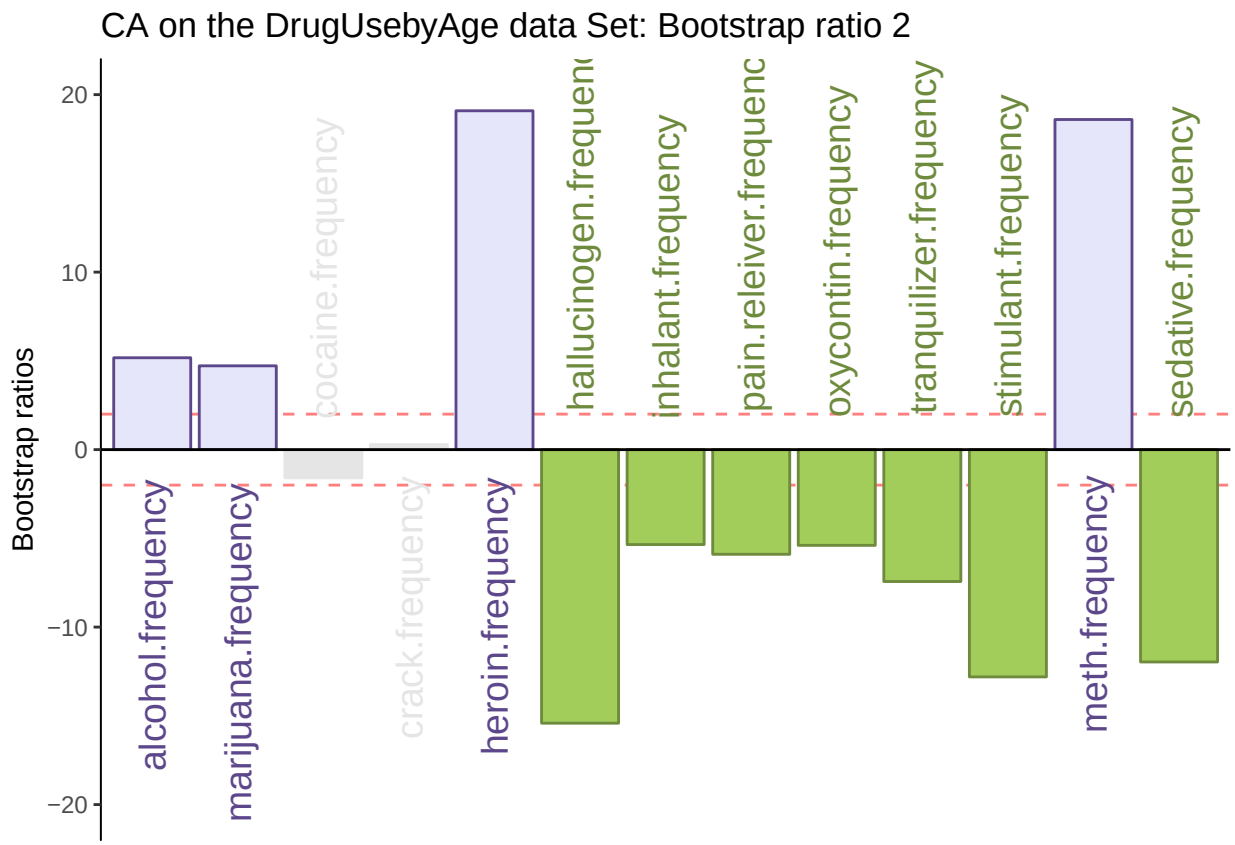
5.4 Bootstrap Ratios for variables

For the bootstrap ratios we can say that those having B.R greater than 2 are significant.

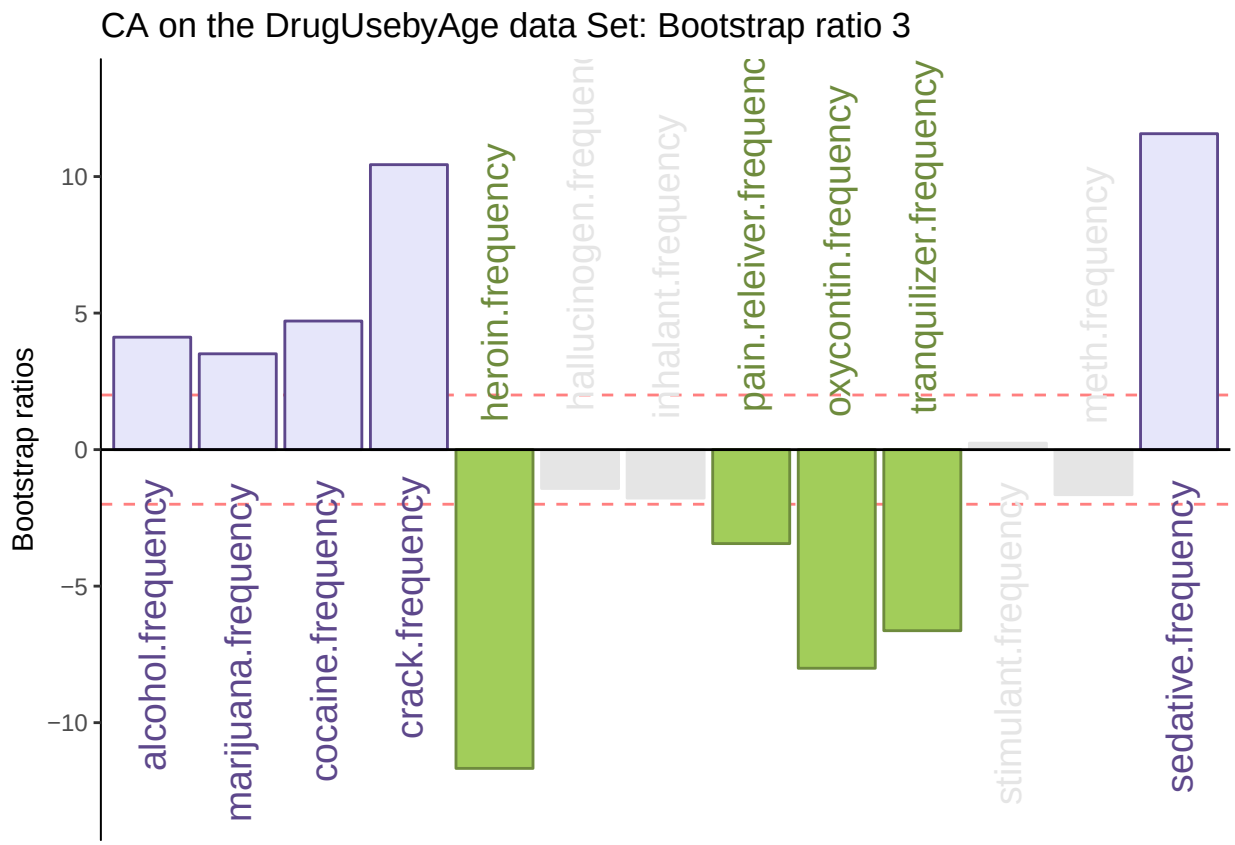
```
BR1 <- rescainf$Inference.Data$fj.boots$tests$boot.ratios
laDim1 = 1
ba001.BR11 <- PrettyBarPlot2(BR1[,laDim1],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0( 'CA on the DrugUsebyAge data Set: Bootstrap ratio ',laDim1),
                             ylab = 'Bootstrap ratios'
                             #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR11)
```




```
#
laDim2 = 2
ba002.BR21 <- PrettyBarPlot2(BR1[,laDim2],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0(
                               'CA on the DrugUsebyAge data Set: Bootstrap ratio ',laDim2),
                             ylab = 'Bootstrap ratios'
)
print(ba002.BR21)
```




```
laDim3 = 3
ba002.BR31 <- PrettyBarPlot2(BR1[,laDim3],
                             threshold = 2,
                             font.size = 5,
                             main = paste0(
                               'CA on the DrugUsebyAge data Set: Bootstrap ratio ',laDim3),
                             ylab = 'Bootstrap ratios'
)
print(ba002.BR31)
```



5.5 Summary

From the above Data Analysis we can infer that :

From the symmetrical plot, we can say that - Old aged people above the age of 65years are generally heavy on stimulants than the average use

- Kids around the age of 12 were consuming more hallucinogen drug than the average use
- Teenagers around the age of 13 were consuming more tranquilizer drug than the average use
- Teenagers around the age of 14 were consuming more sedative drug than the average use
- Teenagers around the age of 16 were consuming more heroin drug than the average use
- Teenagers around the age of 17 were consuming more heroin drug than the average use
- Youngsters around the age of 20 were consuming more alcholol and marijuana than the average use
- People around the age of 35-49 were consuming more heroin and meth drug than the average use
- People around the age of 22-23 were consuming more cocaine and crack drug than the average use
- Youngsters/People around the age of 24-34 were consuming more marijuana than the average use
- People around the 65+ age were consuming more stimulant than the average use

```
options(knitr.duplicate.label = 'allow')
```

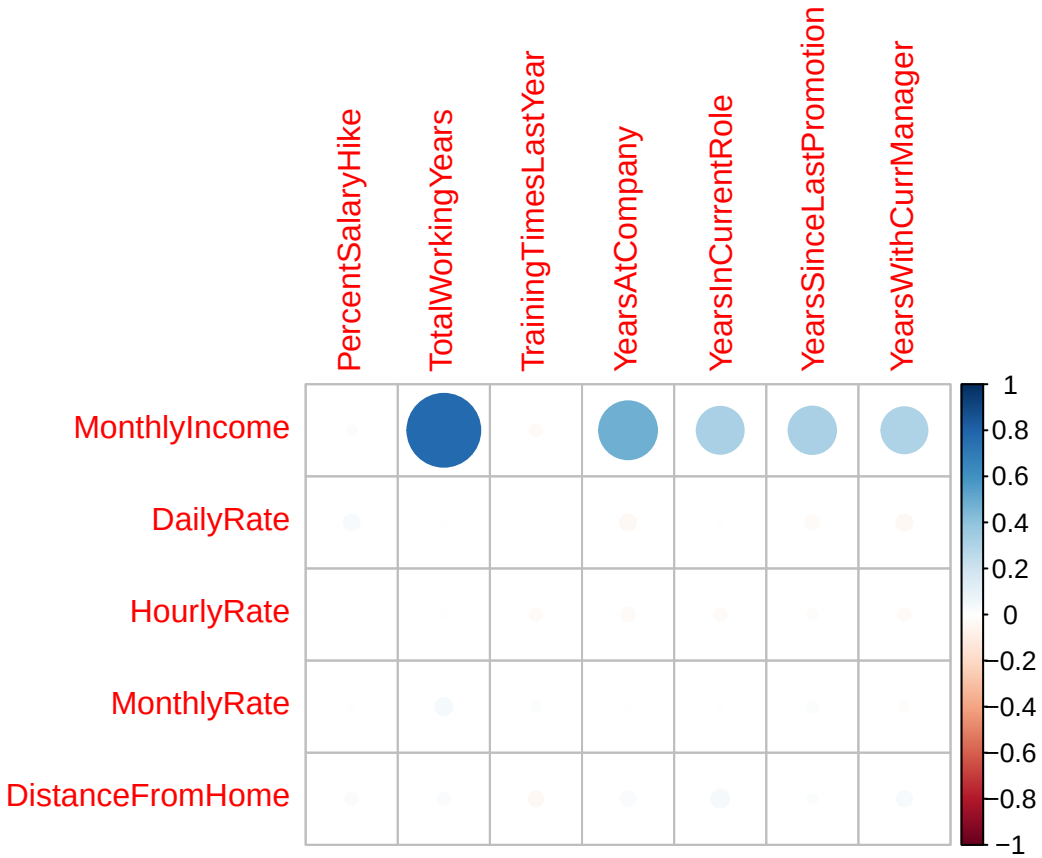
6 PLS

Partial least square (PLS) methods (also sometimes called projection to latent structures) relate the information present in two data tables that collect measurements on the same set of observations. PLS methods proceed by deriving latent variables which are (optimal) linear combinations of the variables of a data table. When the goal is to find the shared information between two tables, the approach is equivalent to a correlation problem and the technique is then called partial least square correlation (PLSC) (also sometimes called PLS-SVD). In this case there are two sets of latent variables (one set per table), and these latent variables are required to have maximal covariance.

```
options(knitr.duplicate.label = 'allow')
data <- read.csv("IBM-HR-Emplyee-NoAttrition.csv")
data1 <- data[,11:15]
data2 <- data[,22:29]
data2 <- data2[ -c(4) ]
#data$Gender <- as.numeric(as.factor(data$Gender))
#data$Department <- as.numeric(as.factor(data$Department))
#data$JobRole <- as.numeric(as.factor(data$JobRole))
#data$EducationField <- as.numeric(as.factor(data$EducationField))
design1 <- data$Gender
design2 <- data$Department
design3 <- data$JobRole
design4 <- data$EducationField
```

Correlation Plot

```
corrplot::corrplot(cor(data1,data2))
```



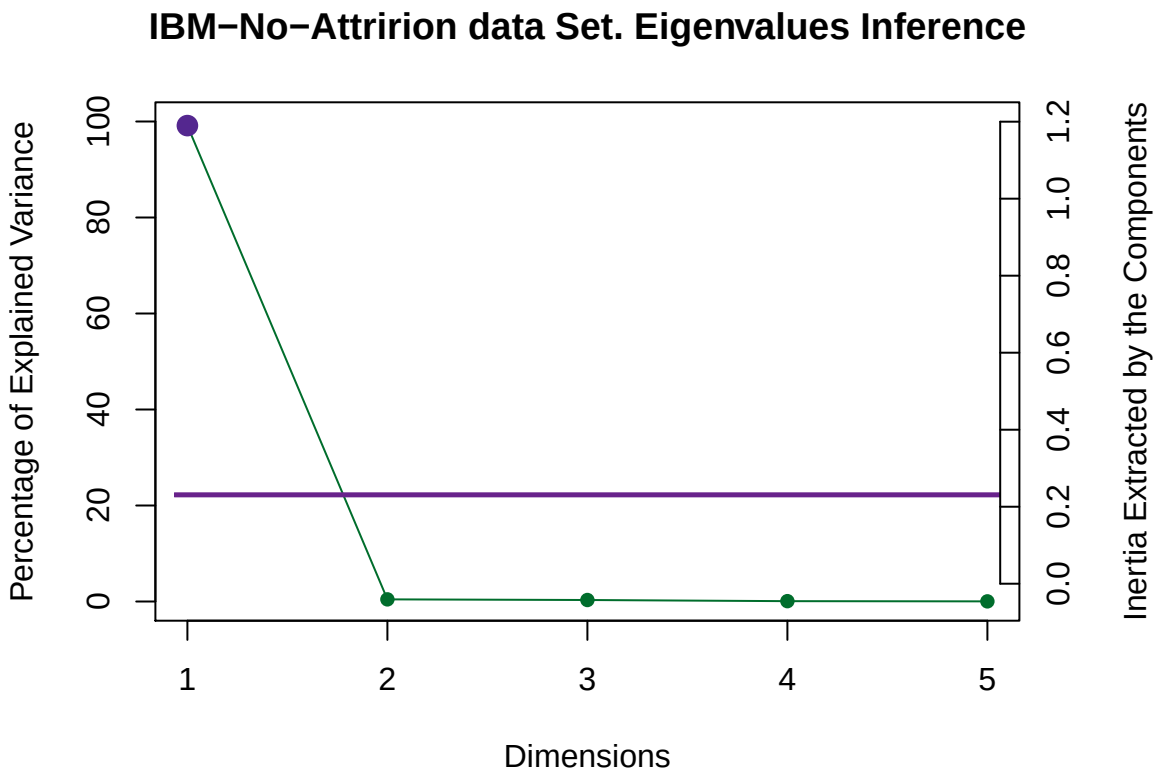
Scree Plot

A Scree Plot is a simple line segment plot that shows the fraction of total variance in the data as explained or represented by each PC.(`In the PCA literature, the plot is called a 'Scree' Plot because it often looks like a 'scree' slope, where rocks have fallen down and accumulated on the side of a mountain.`)The scree plot shows the eigenvalues, the amount of information on each component. The number of components (the dimensionality of the factor space) is min(nrow(DATA), ncol(DATA)) minus 1.

```
library(TExPosition)
library(data4PCCAR)
resPLSC <- tepPLS(data1,data2,DESIGN = design1,graphs = FALSE)
```

```
resPerm4PLSC <- perm4PLSC(data1, # First Data matrix
                          data2, # Second Data matrix
                          nIter = 1000 # How mny iterations
                          )
```

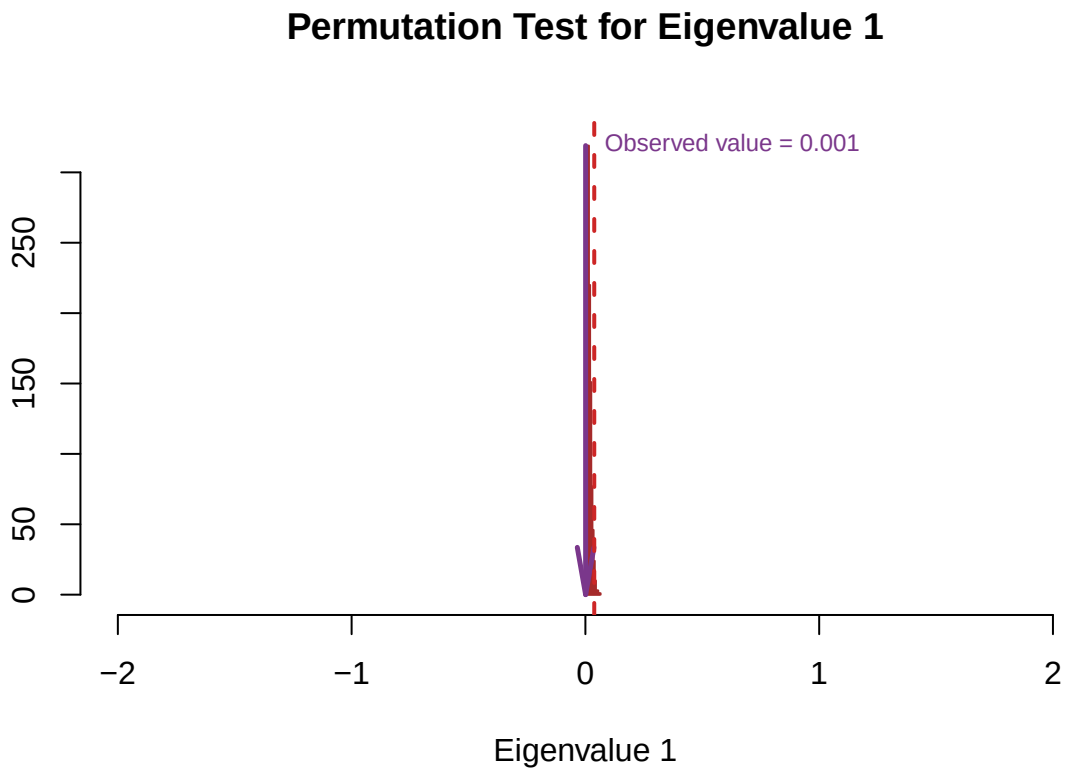
```
PlotScree(ev = resPLSC$TExPosition.Data$eigs,
          p.ev = resPerm4PLSC$pEigenvalues,
          title = 'IBM-No-Attririon data Set. Eigenvalues Inference',
          plotKaiser = TRUE
)
```



Permutation test for eigen-values

```
zeDim = 1
pH1 <- prettyHist(
  distribution = resPerm4PLSC$permEigenvalues[,zeDim],
```

```
          observed = resPerm4PLSC$pEigenvalues[zeDim],
          xlim = c(-2, 2), # needs to be set by hand
          breaks = 20,
          border = "brown",
          main = paste0("Permutation Test for Eigenvalue ",zeDim),
          xlab = paste0("Eigenvalue ",zeDim),
          ylab = "",
          counts = FALSE,
          cutoffs = c( 0.975))
```




```
b005.PermTest <- recordPlot()
```

6.1 Factor Maps for latent Variables

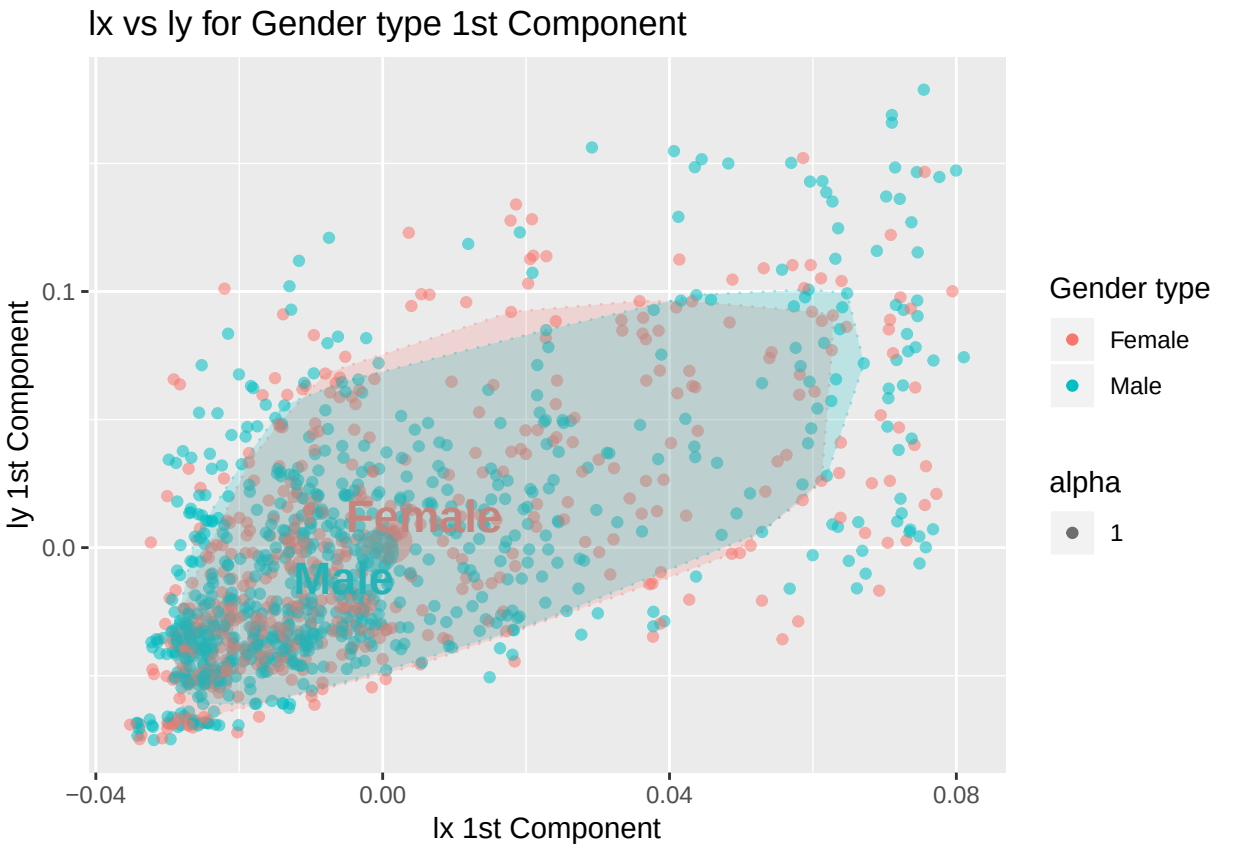
lx vs ly for Gender type (1st Component)

```
library(data4PCCAR)
library(PTCA4CATA)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(ggplot2)
library(ExPosition)
library(InPosition)
constraints1 <- minmaxHelper(mat1 = resPLSC$TExPosition.Data$lx, mat2  = resPLSC$TExPosition.Data$ly)
```

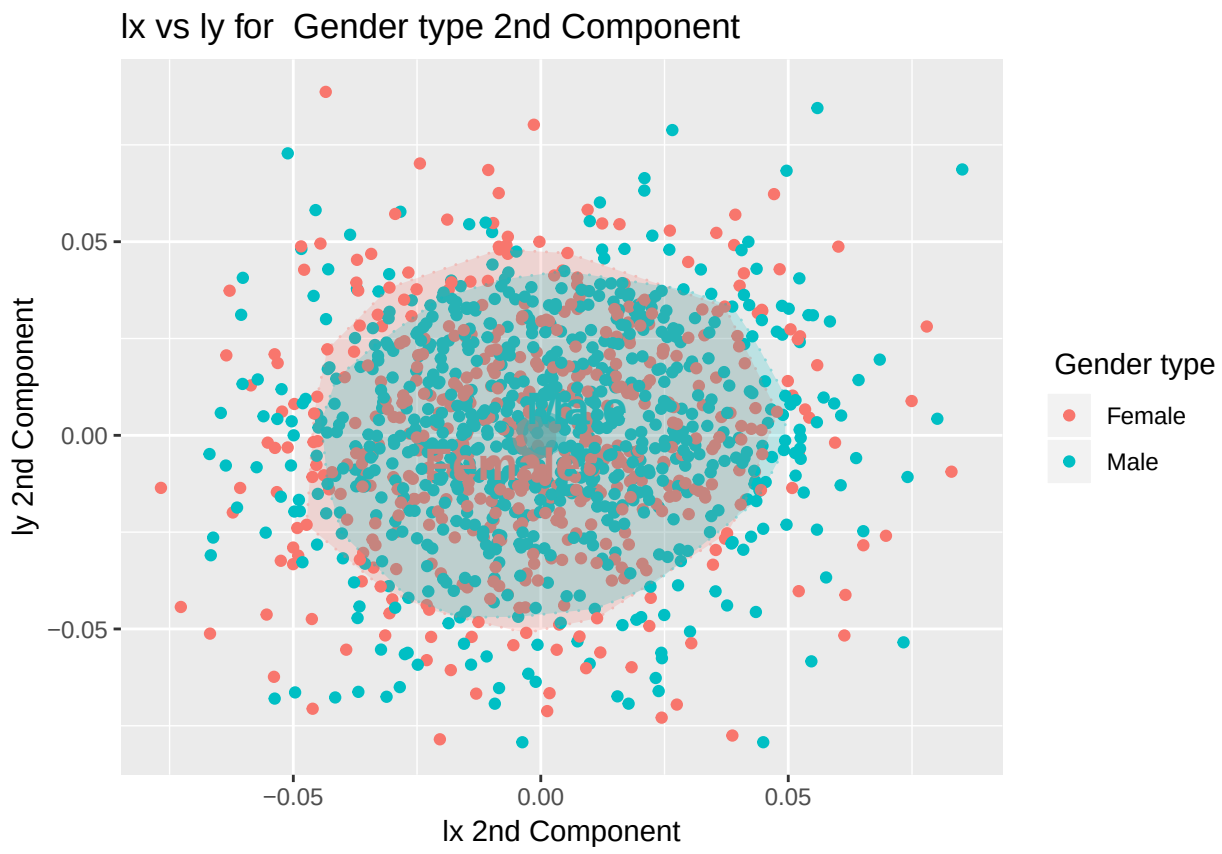
```
df <- as.data.frame(cbind(resPLSC$TExPosition.Data$lx[,1],resPLSC$TExPosition.Data$ly[,1]))
dataMeans <- getMeans(df, design1)
col4data <- factor(design1)
col4Means <- c("#F8766D", "#00BFC4")
MapGroup <- createFactorMap(dataMeans,
                            # use the constraint from the main map
                            constraints = constraints1,
                            col.points = col4Means,
                            cex = 7,  # size of the dot (bigger)
                            col.labels = col4Means,
                            text.cex = 6)

GraphTI.Hull <- PTCA4CATA::MakeToleranceIntervals(df,
                                                  design = design1,
                                                  # line below is needed
                                                  names.of.factors =  c("Dim1","Dim2"), # needed
                                                  col = col4Means,
                                                  line.size = .50,
                                                  line.type = 3,
                                                  alpha.ellipse = .2,
                                                  alpha.line    = .4,
                                                  p.level       = .75)

z <- ggplot(data, aes(x=resPLSC$TExPosition.Data$lx[,1], y=resPLSC$TExPosition.Data$ly[,1])) + geom_poi
  ylab("ly 1st Component") +
  ggtitle("lx vs ly for Gender type 1st Component") + scale_color_discrete(name = "Gender type", labels
print(z)
```



lx vs ly for Gender type (2nd Component)

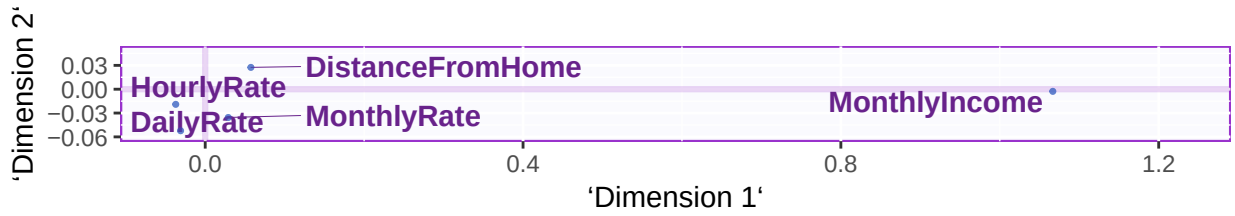
```
df1 <- as.data.frame(cbind(resPLSC$TExPosition.Data$lx[,2],resPLSC$TExPosition.Data$ly[,2]))
dataMeans1 <- getMeans(df1, design1)
col4data1 <- factor(design1)
col4Means1 <-  c("#F8766D", "#00BFC4")
MapGroup1 <- createFactorMap(dataMeans1,
                             # use the constraint from the main map
                             constraints = constraints1,
                             col.points = col4Means1,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means1,
                             text.cex = 6)

GraphTI.Hull1 <- PTCA4CATA::MakeToleranceIntervals(df1,
                                                   design = design1,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means1,
                                                   line.size = .50,
                                                   line.type = 3,
                                                   alpha.ellipse = .2,
                                                   alpha.line    = .4,
                                                   p.level       = .75)

z1 <- ggplot(data, aes(x=resPLSC$TExPosition.Data$lx[,2], y=resPLSC$TExPosition.Data$ly[,2])) + geom_po
  ylab("ly 2nd Component") +
```

```
  ggtitle("lx vs ly for  Gender type 2nd Component") + scale_color_discrete(name = " Gender type", label
print(z1)
```

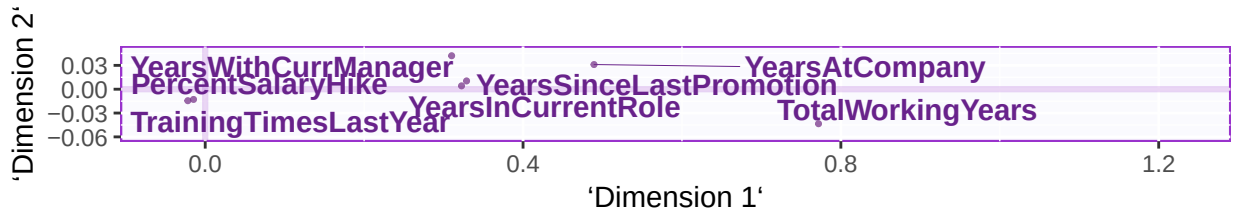


6.2 Factor Maps for J

```
constraints <- minmaxHelper(mat1 = resPLSC$TExPosition.Data$fi, mat2  = resPLSC$TExPosition.Data$fj)
baseMap.j2 <- createFactorMap(resPLSC$TExPosition.Data$fi,constraints = constraints,
                              col.points = resPLSC$Plotting.Data$fi.col, axis1 = 1, axis2 = 2,
                              cex = 1, pch = 20,
                              display.labels = TRUE
)
a2 <- baseMap.j2$zeMap + baseMap.j2$zeMap_dots
print(a2)
```




```
baseMap.j3 <- createFactorMap(resPLSC$TExPosition.Data$fj,constraints = constraints,
                              col.points = resPLSC$Plotting.Data$fj.col, axis1 = 1, axis2 = 2,
                              cex = 1, pch = 20,
                              display.labels = TRUE
)
a1 <- baseMap.j3$zeMap + baseMap.j3$zeMap_dots
print(a1)
```

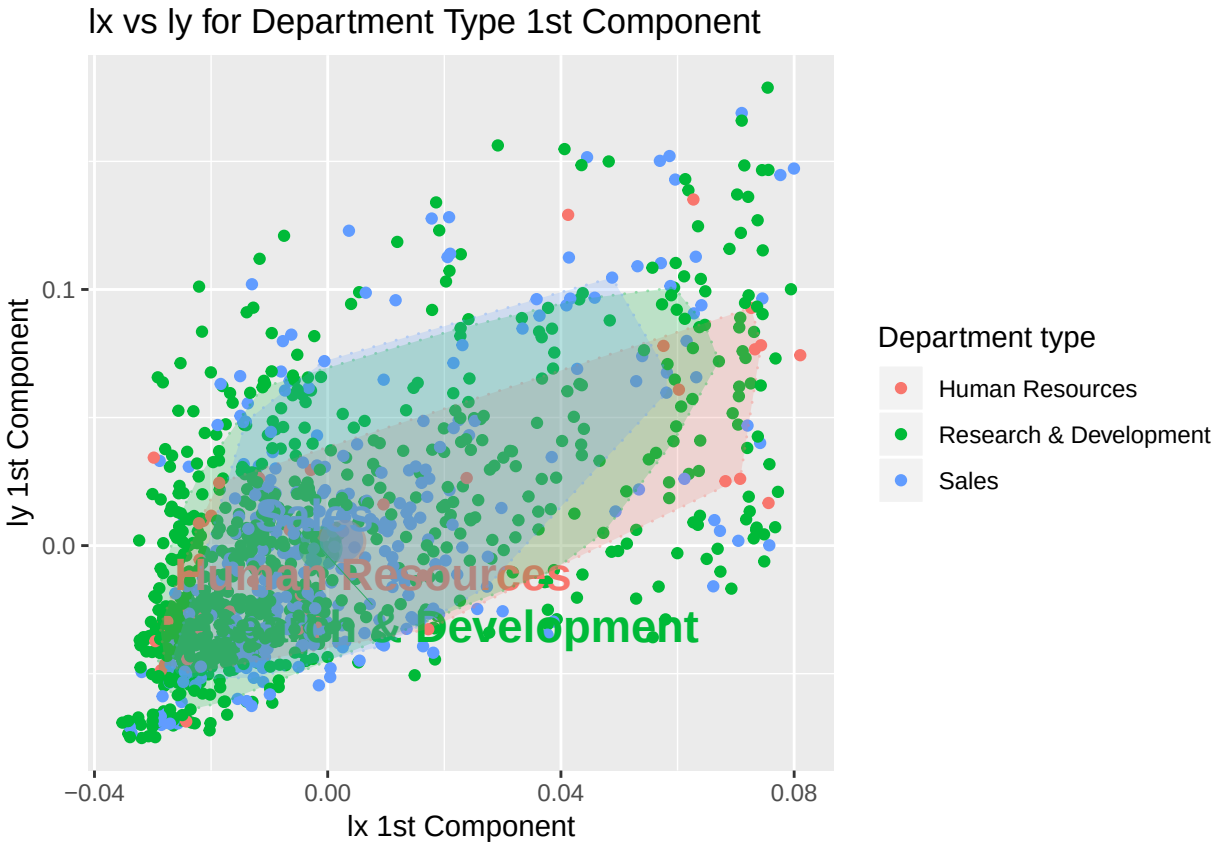


Departmenttype(1st Component)

```
resPLSC1 <- tepPLS(data1,data2,DESIGN = design2,graphs = FALSE)
```

```
constraints2 <- minmaxHelper(mat1 = resPLSC1$TExPosition.Data$lx, mat2  = resPLSC1$TExPosition.Data$ly)
df2 <- as.data.frame(cbind(resPLSC1$TExPosition.Data$lx[,1],resPLSC1$TExPosition.Data$ly[,1]))
dataMeans2 <- getMeans(df2, design2)
col4data2 <- factor(design2)
col4Means2 <-  c("#F8766D" ,"#00BA38" ,"#619CFF")
MapGroup2 <- createFactorMap(dataMeans2,
                             # use the constraint from the main map
                             constraints = constraints2,
                             col.points = col4Means2,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means2,
                             text.cex = 6)
GraphTI.Hull2 <- PTCA4CATA::MakeToleranceIntervals(df2,
                                                   design = design2,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means2,
                                                   line.size = .50,
                                                   line.type = 3,
                                                   alpha.ellipse = .2,
                                                   alpha.line    = .4,
                                                   p.level       = .75)
```

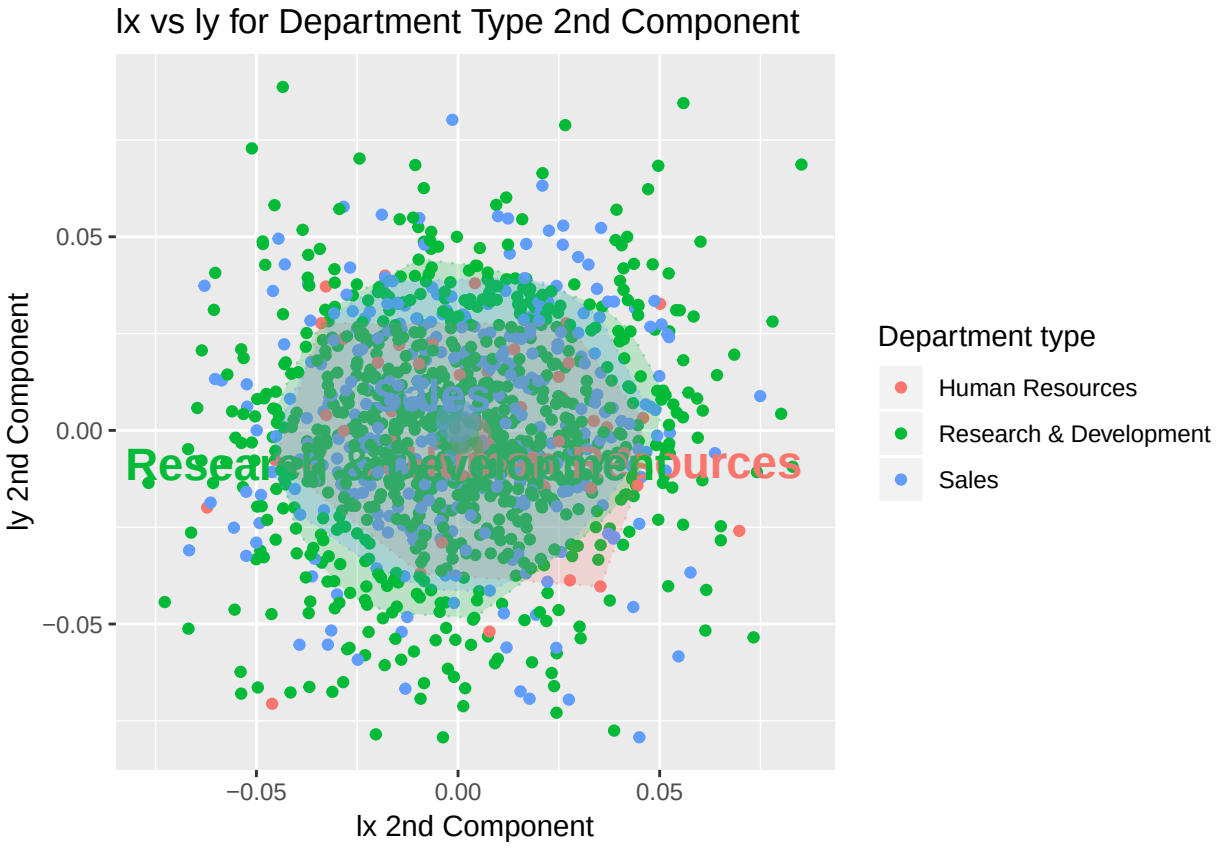
```
z2 <- ggplot(data, aes(x=resPLSC1$TExPosition.Data$lx[,1], y=resPLSC1$TExPosition.Data$ly[,1])) + geom_
  ylab("ly 1st Component") +
  ggtitle("lx vs ly for Department Type 1st Component") + scale_color_discrete(name = "Department type"
print(z2)
```



lx vs ly for Department type(2st Component)

```
df3 <- as.data.frame(cbind(resPLSC1$TExPosition.Data$lx[,2],resPLSC1$TExPosition.Data$ly[,2]))
dataMeans3 <- getMeans(df3, design2)
col4data3 <- factor(design2)
col4Means3 <- c("#F8766D" ,"#00BA38" ,"#619CFF")
MapGroup3 <- createFactorMap(dataMeans3,
                             # use the constraint from the main map
                             constraints = constraints2,
                             col.points = col4Means3,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means3,
                             text.cex = 6)
BootCube.Gr3 <- Boot4Mean(df3,
                          design = design2,
                          niter = 1233,
                          suppressProgressBar = TRUE)

GraphElli3 <- PTCA4CATA::MakeCIEllipses(BootCube.Gr3$BootCube[,1:2,],
                             names.of.factors = c("Dimension 1","Dimension 2"),
                             col = col4Means3,
```

```
                              p.level = .95
)

GraphTI.Hull3 <- PTCA4CATA::MakeToleranceIntervals(df3,
                                                   design = design2,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means3,
                                                   line.size = .50,
                                                   line.type = 3,
                                                   alpha.ellipse = .2,
                                                   alpha.line    = .4,
                                                   p.level       = .75)


z3 <- ggplot(data, aes(x=resPLSC1$TExPosition.Data$lx[,2], y=resPLSC1$TExPosition.Data$ly[,2])) + geom_
  ylab("ly 2nd Component") +
  ggtitle("lx vs ly for Department Type 2nd Component") + scale_color_discrete(name = "Department type"
print( z3 )
```

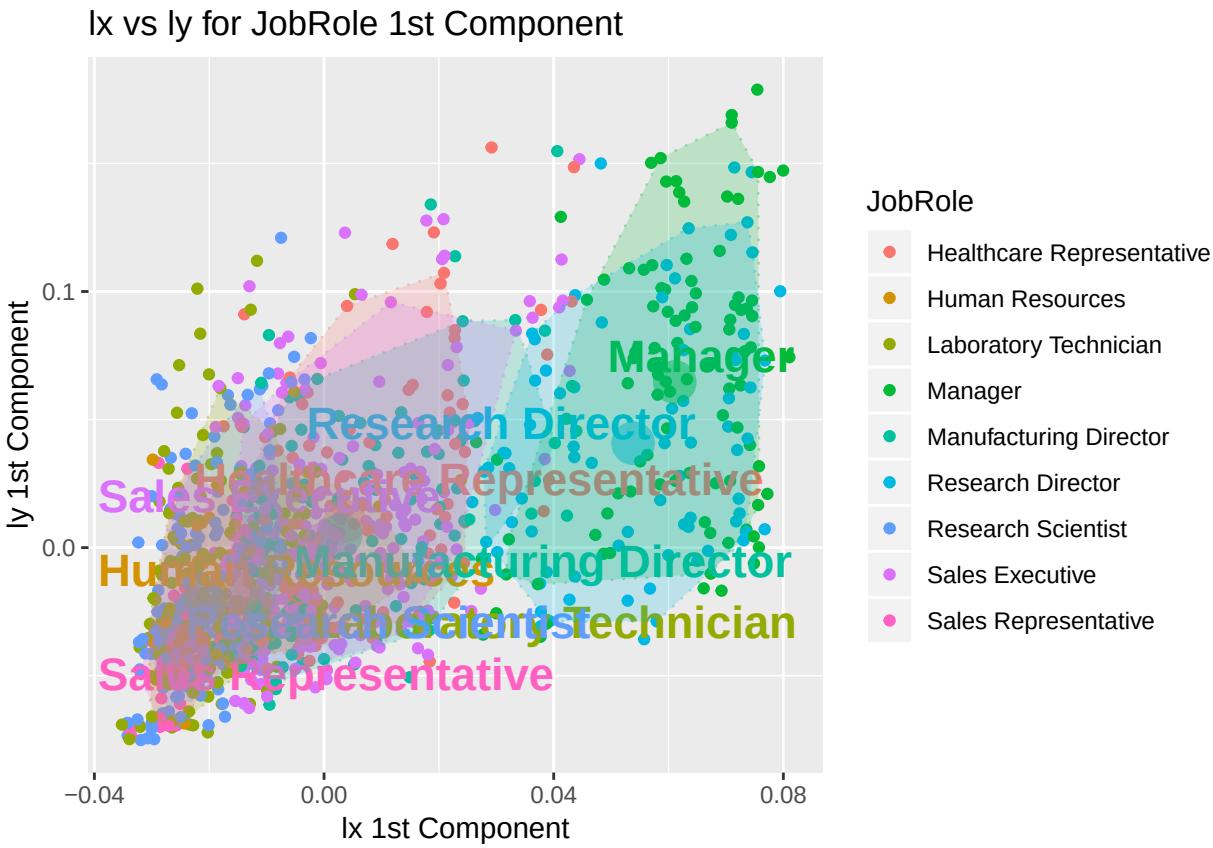


lx vs ly for JObRole(1st Component)

```
resPLSC2 <- tepPLS(data1,data2,DESIGN = design3,graphs = FALSE)
```

```
constraints3 <- minmaxHelper(mat1 = resPLSC2$TExPosition.Data$lx, mat2  = resPLSC2$TExPosition.Data$ly)
df4 <- as.data.frame(cbind(resPLSC2$TExPosition.Data$lx[,1],resPLSC2$TExPosition.Data$ly[,1]))
dataMeans4 <- getMeans(df4, design3)
```

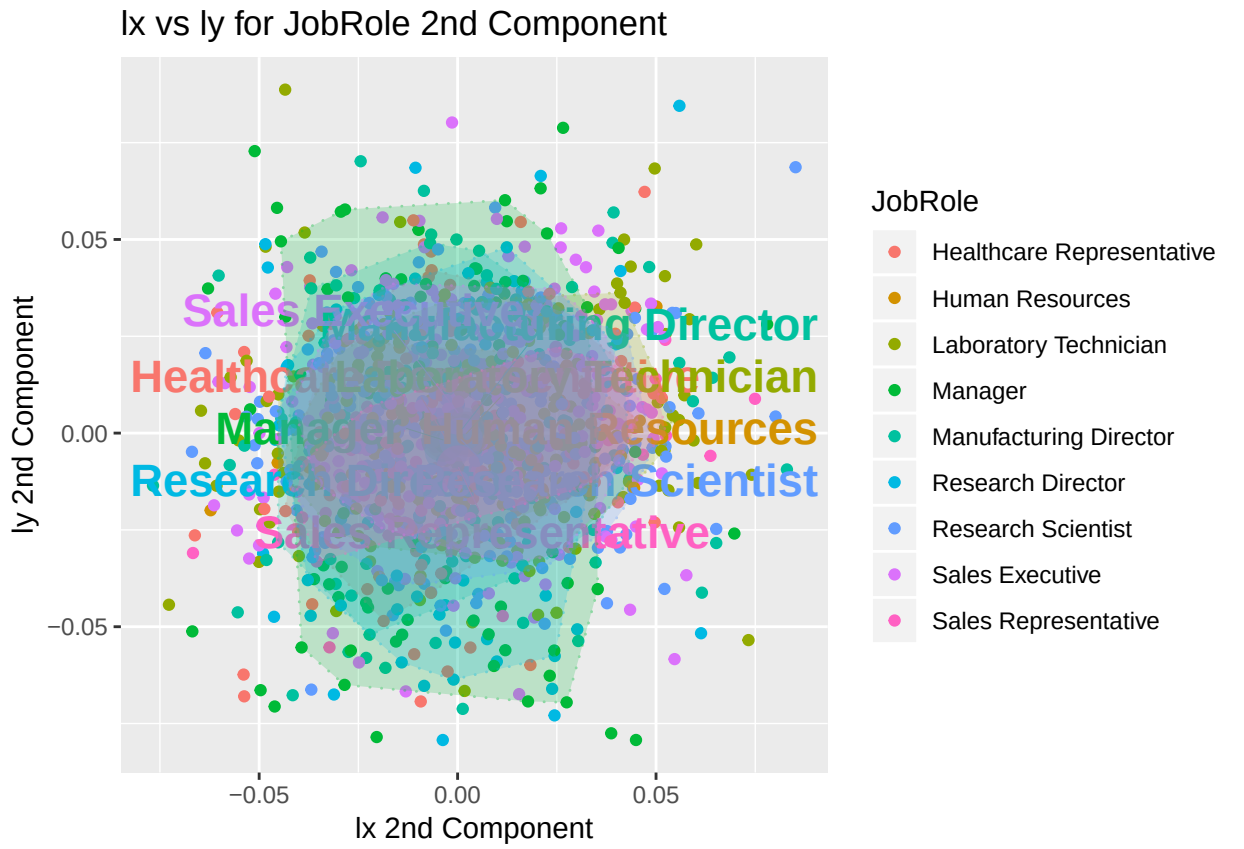
```r
col4data4 <- factor(design3)
col4Means4 <- c("#F8766D", "#D39200" ,"#93AA00" ,"#00BA38", "#00C19F", "#00B9E3", "#619CFF" ,"#DB72FB",
MapGroup4 <- createFactorMap(dataMeans4,
                             # use the constraint from the main map
                             constraints = constraints3,
                             col.points = col4Means4,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means4,
                             text.cex = 6)
BootCube.Gr4 <- Boot4Mean(df4,
                          design = design3,
                          niter = 100,
                          suppressProgressBar = TRUE)

GraphElli4 <- PTCA4CATA::MakeCIEllipses(BootCube.Gr4$BootCube[,1:2,],
                             names.of.factors = c("Dimension 1","Dimension 2"),
                             col = col4Means4,
                             p.level = .95
)

GraphTI.Hull4 <- PTCA4CATA::MakeToleranceIntervals(df4,
                                                   design = design3,
                                                     # line below is needed
                                                     names.of.factors =  c("Dim1","Dim2"), # needed
                                                     col = col4Means4,
                                                     line.size = .50,
                                                     line.type = 3,
                                                     alpha.ellipse = .2,
                                                     alpha.line    = .4,
                                                     p.level       = .75)

z4 <- ggplot(data, aes(x = resPLSC2$TExPosition.Data$lx[,1], y = resPLSC2$TExPosition.Data$ly[,1])) +ge
  ylab("ly 1st Component") + ggtitle("lx vs ly for JobRole 1st Component") + GraphTI.Hull4   #GraphElli

print(z4)
```



lx vs ly for JObRole(2nd Component)

```
df5 <- as.data.frame(cbind(resPLSC2$TExPosition.Data$lx[,2],resPLSC2$TExPosition.Data$ly[,2]))
dataMeans5 <- getMeans(df5, design3)
col4data5 <- factor(design3)
col4Means5 <- c("#F8766D", "#D39200" ,"#93AA00" ,"#00BA38", "#00C19F", "#00B9E3", "#619CFF" ,"#DB72FB",
MapGroup5 <- createFactorMap(dataMeans5,
                             # use the constraint from the main map
                             constraints = constraints3,
                             col.points = col4Means5,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means5,
                             text.cex = 6)
GraphTI.Hull5 <- PTCA4CATA::MakeToleranceIntervals(df5,
                                                   design = design3,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means5,
                                                   line.size = .50,
                                                   line.type = 3,
                                                   alpha.ellipse = .2,
                                                   alpha.line    = .4,
                                                   p.level       = .75)


z5 <- ggplot(data, aes(x = resPLSC2$TExPosition.Data$lx[,2], y = resPLSC2$TExPosition.Data$ly[,2])) +
  geom_point(aes(color = factor(design3))) + MapGroup5$zeMap_dots + MapGroup5$zeMap_text + scale_color_
```

```
  ylab("ly 2nd Component") +
  ggtitle("lx vs ly for JobRole 2nd Component") + GraphTI.Hull5

print(z5)
```

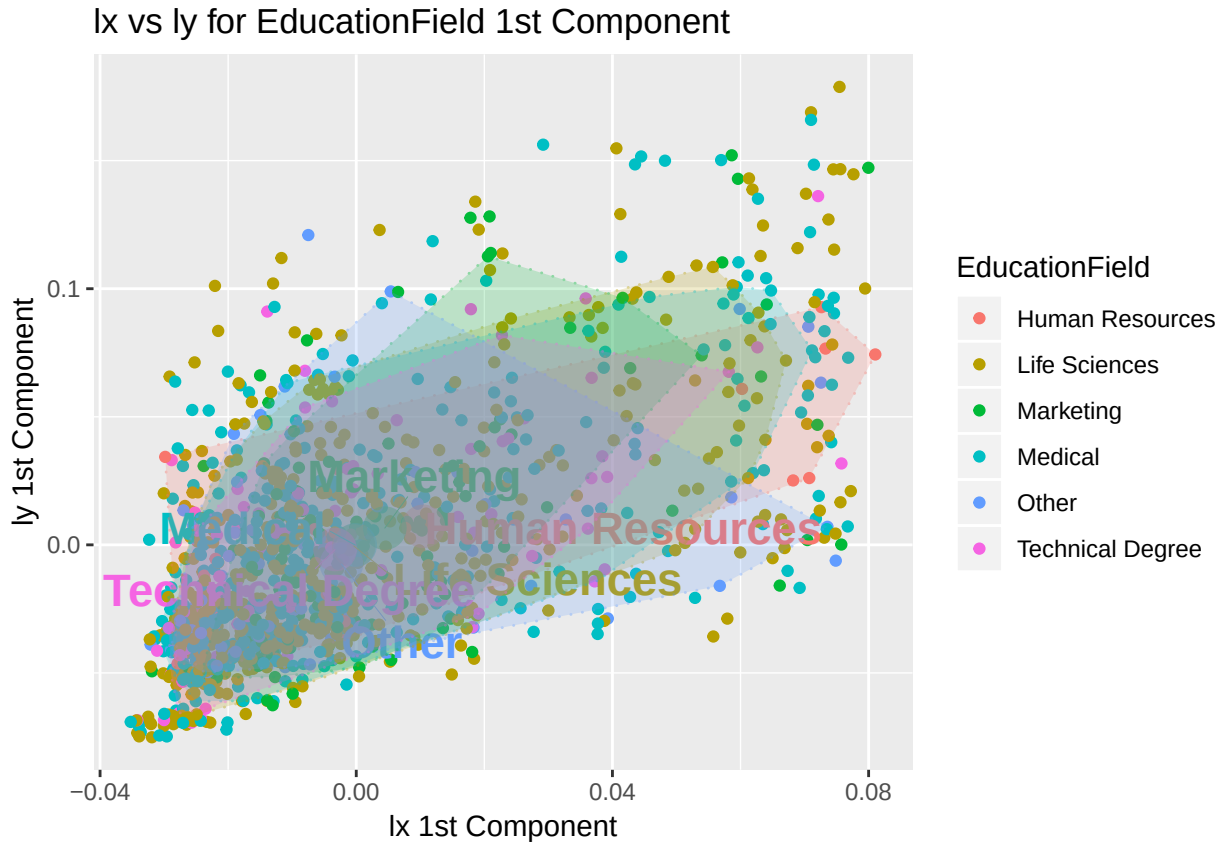


lx vs ly for EducationField(1st Component)

```
resPLSC3 <- tepPLS(data1,data2,DESIGN = design4,graphs = FALSE)
```

```
constraints4 <- minmaxHelper(mat1 = resPLSC3$TExPosition.Data$lx, mat2  = resPLSC3$TExPosition.Data$ly)
df6 <- as.data.frame(cbind(resPLSC3$TExPosition.Data$lx[,1],resPLSC3$TExPosition.Data$ly[,1]))
dataMeans6 <- getMeans(df6, design4)
col4data6 <- factor(design4)
col4Means6 <- c("#F8766D" ,"#B79F00" ,"#00BA38" ,"#00BFC4" ,"#619CFF", "#F564E3")
MapGroup6 <- createFactorMap(dataMeans6,
                             # use the constraint from the main map
                             constraints = constraints4,
                             col.points = col4Means6,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means6,
                             text.cex = 6)
GraphTI.Hull6 <- PTCA4CATA::MakeToleranceIntervals(df6,
                                                   design = design4,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means6,
                                                   line.size = .50,
```

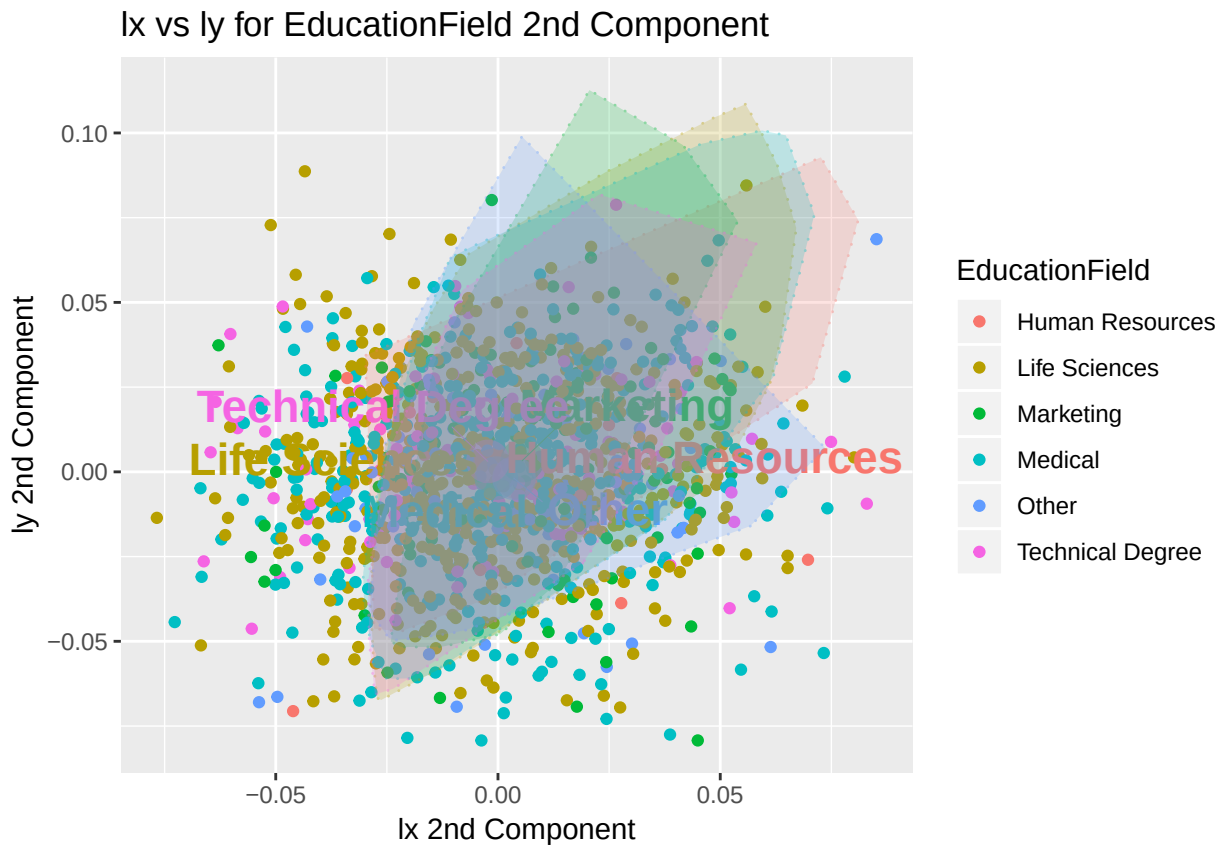
```
                                                    line.type = 3,
                                                    alpha.ellipse = .2,
                                                    alpha.line    = .4,
                                                    p.level       = .75)



z6 <- ggplot(data, aes(x=resPLSC3$TExPosition.Data$lx[,1], y=resPLSC3$TExPosition.Data$ly[,1])) + geom_
  ylab("ly 1st Component") +
  ggtitle("lx vs ly for EducationField 1st Component") + GraphTI.Hull6
print(z6)
```



lx vs ly for EducationField(2nd Component)

```
df7 <- as.data.frame(cbind(resPLSC3$TExPosition.Data$lx[,2],resPLSC3$TExPosition.Data$ly[,2]))
dataMeans7 <- getMeans(df7, design4)
col4data7 <- factor(design4)
col4Means7 <- c("#F8766D" ,"#B79F00" ,"#00BA38" ,"#00BFC4" ,"#619CFF", "#F564E3")
MapGroup7 <- createFactorMap(dataMeans7,
                             # use the constraint from the main map
                             constraints = constraints4,
                             col.points = col4Means7,
                             cex = 7,  # size of the dot (bigger)
                             col.labels = col4Means7,
                             text.cex = 6)
```

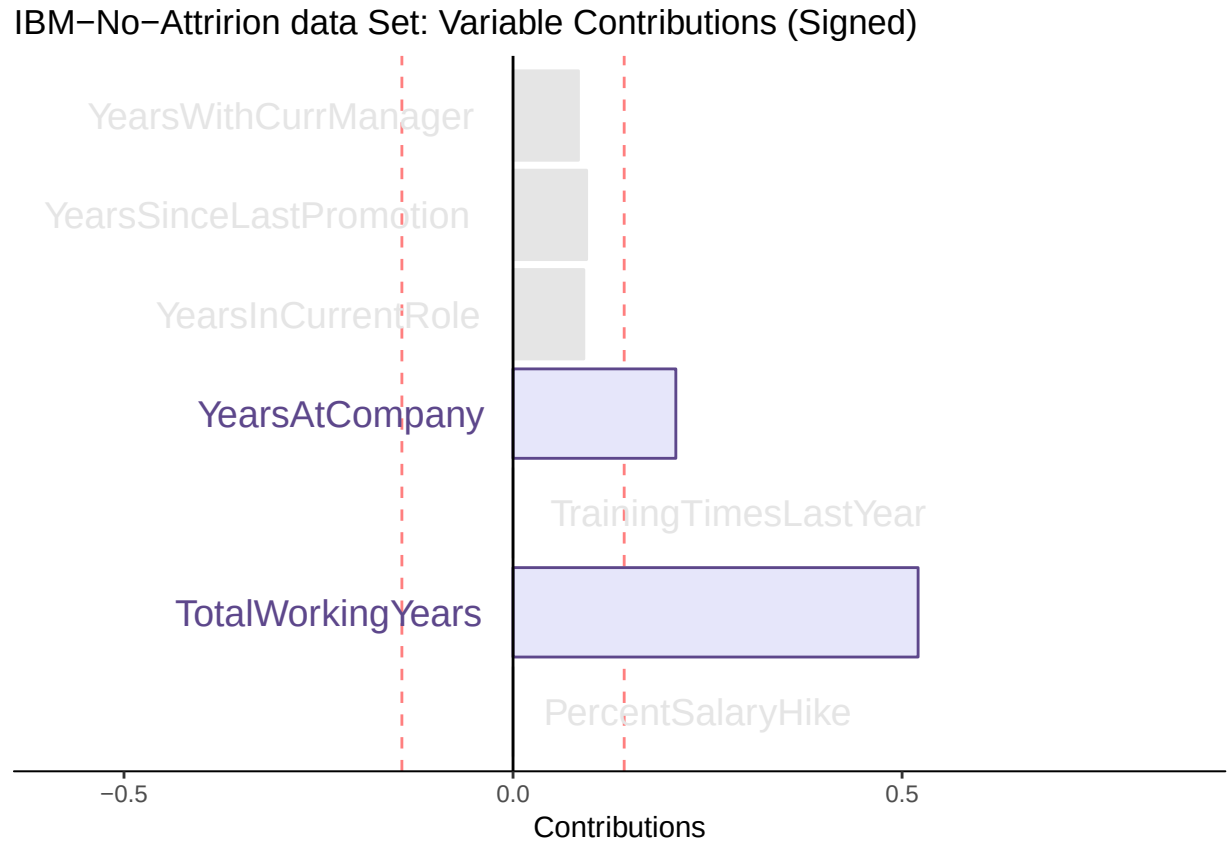
```
GraphTI.Hull7 <- PTCA4CATA::MakeToleranceIntervals(df6,
                                                   design = design4,
                                                   # line below is needed
                                                   names.of.factors =  c("Dim1","Dim2"), # needed
                                                   col = col4Means6,
                                                   line.size = .50,
                                                   line.type = 3,
                                                   alpha.ellipse = .2,
                                                   alpha.line    = .4,
                                                   p.level       = .75)


z7 <- ggplot(data, aes(x=resPLSC3$TExPosition.Data$lx[,2], y=resPLSC3$TExPosition.Data$ly[,2])) + geom_
  ylab("ly 2nd Component") +
  ggtitle("lx vs ly for EducationField 2nd Component") + GraphTI.Hull7

print(z7)
```



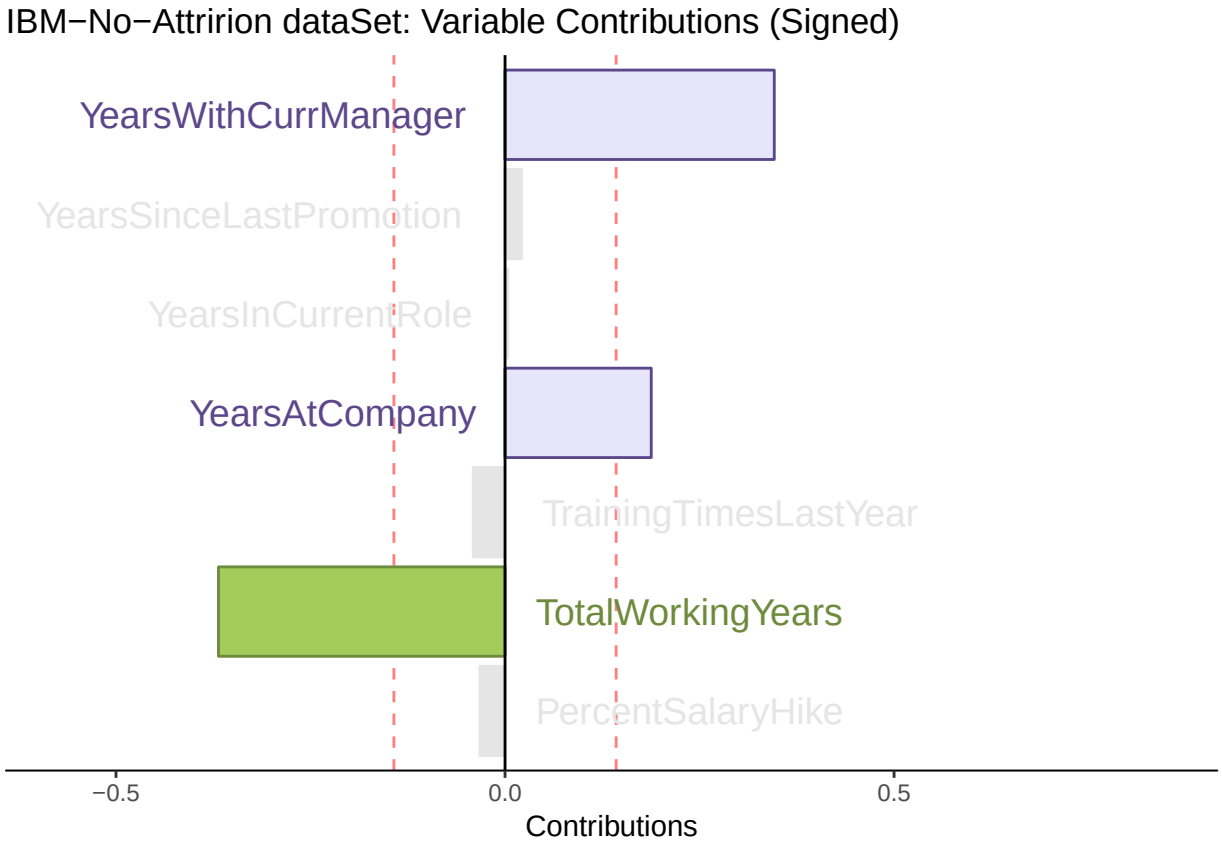
6.3 Contributions for Variables

```
signed.ctrJ <- resPLSC$TExPosition.Data$cj * sign(resPLSC$TExPosition.Data$fj)
b003.ctrJ.s.1 <- PrettyBarPlot2(signed.ctrJ[,1],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
```

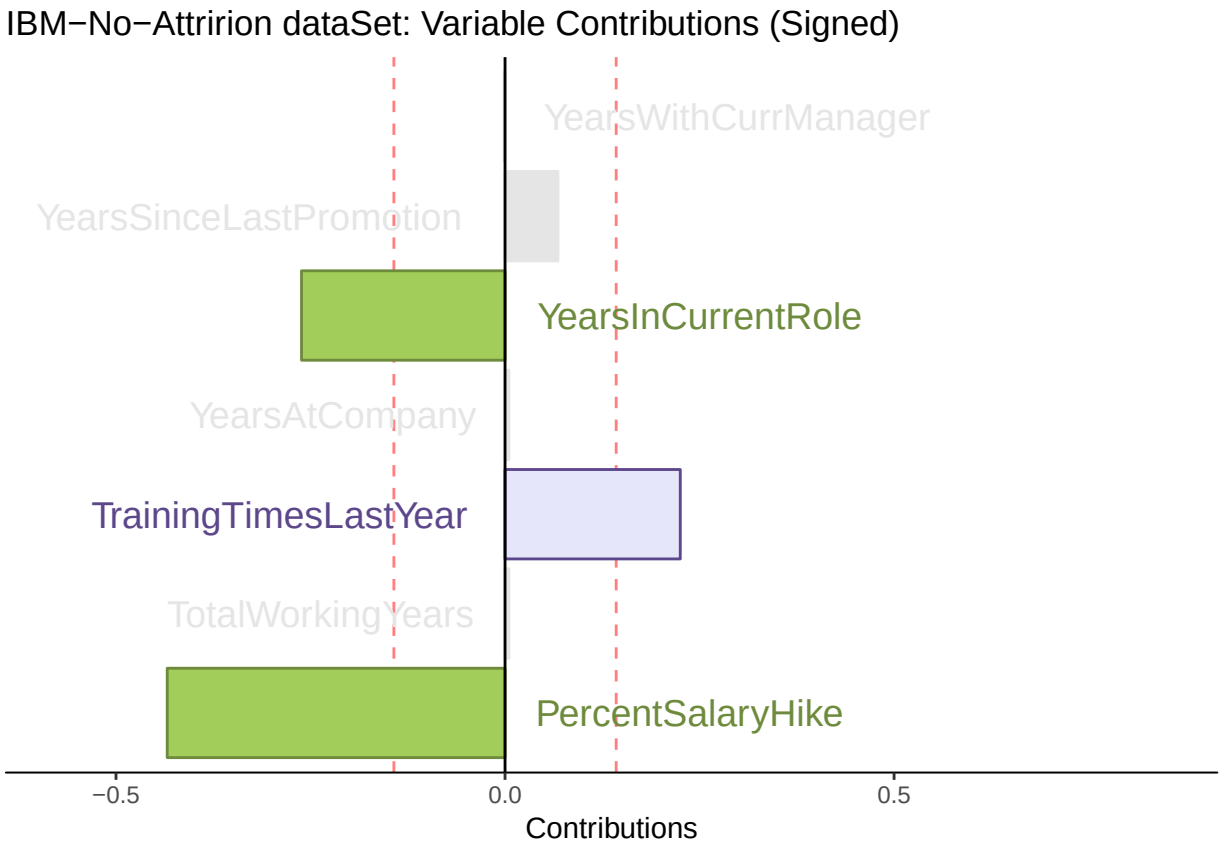
```
                                # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'IBM-No-Attririon data Set: Variable Contributions (Signed)',
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ)), horizontal = FALSE
)
print(b003.ctrJ.s.1)
```




```
b004.ctrJ.s.2 <- PrettyBarPlot2(signed.ctrJ[,2],
                                 threshold = 1 / NROW(signed.ctrJ),
                                 font.size = 5,
                                # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'IBM-No-Attririon dataSet: Variable Contributions (Signed)',
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ)), horizontal = FALSE
)
print(b004.ctrJ.s.2)
```

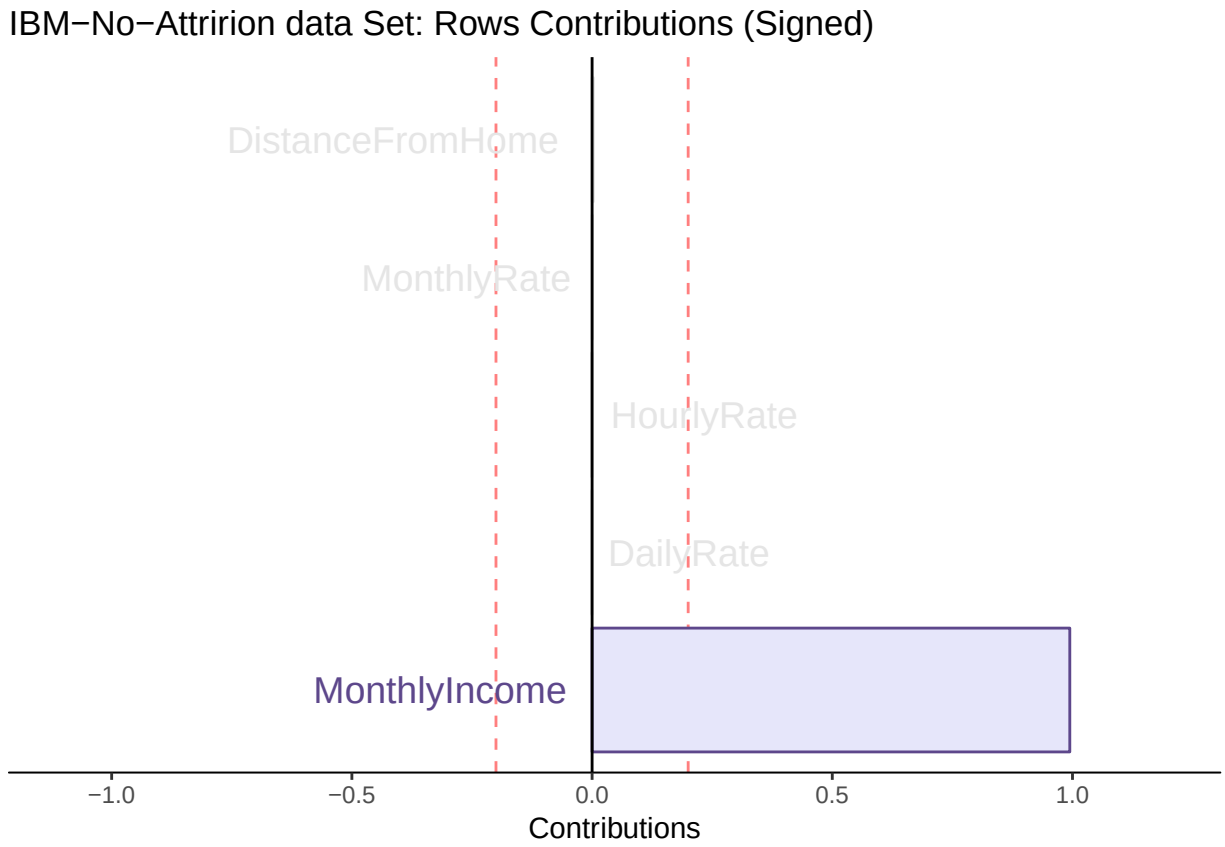



```
b004.ctrJ.s.3 <- PrettyBarPlot2(signed.ctrJ[,3],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                               # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'IBM-No-Attririon dataSet: Variable Contributions (Signed)',
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ)),horizontal = FALSE
)
print(b004.ctrJ.s.3)
```

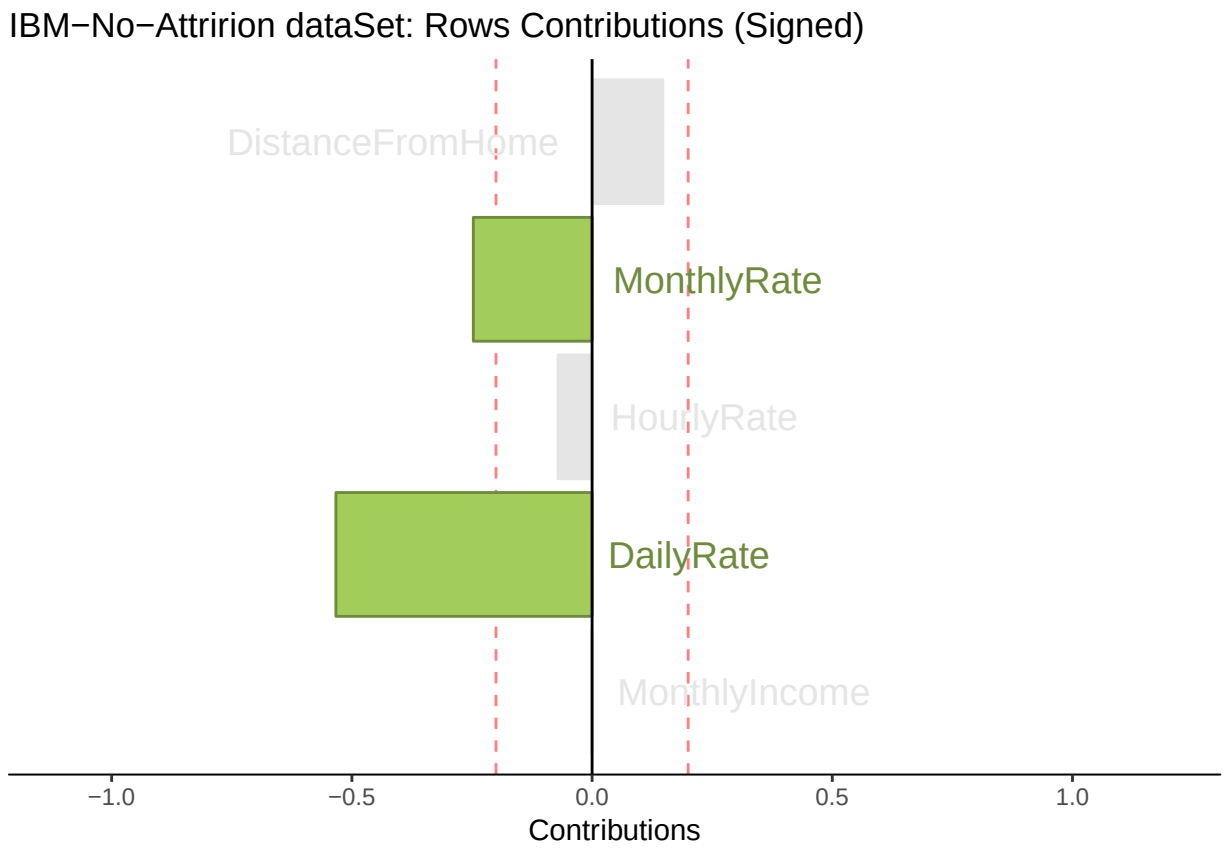


6.4 Contribution for Rows

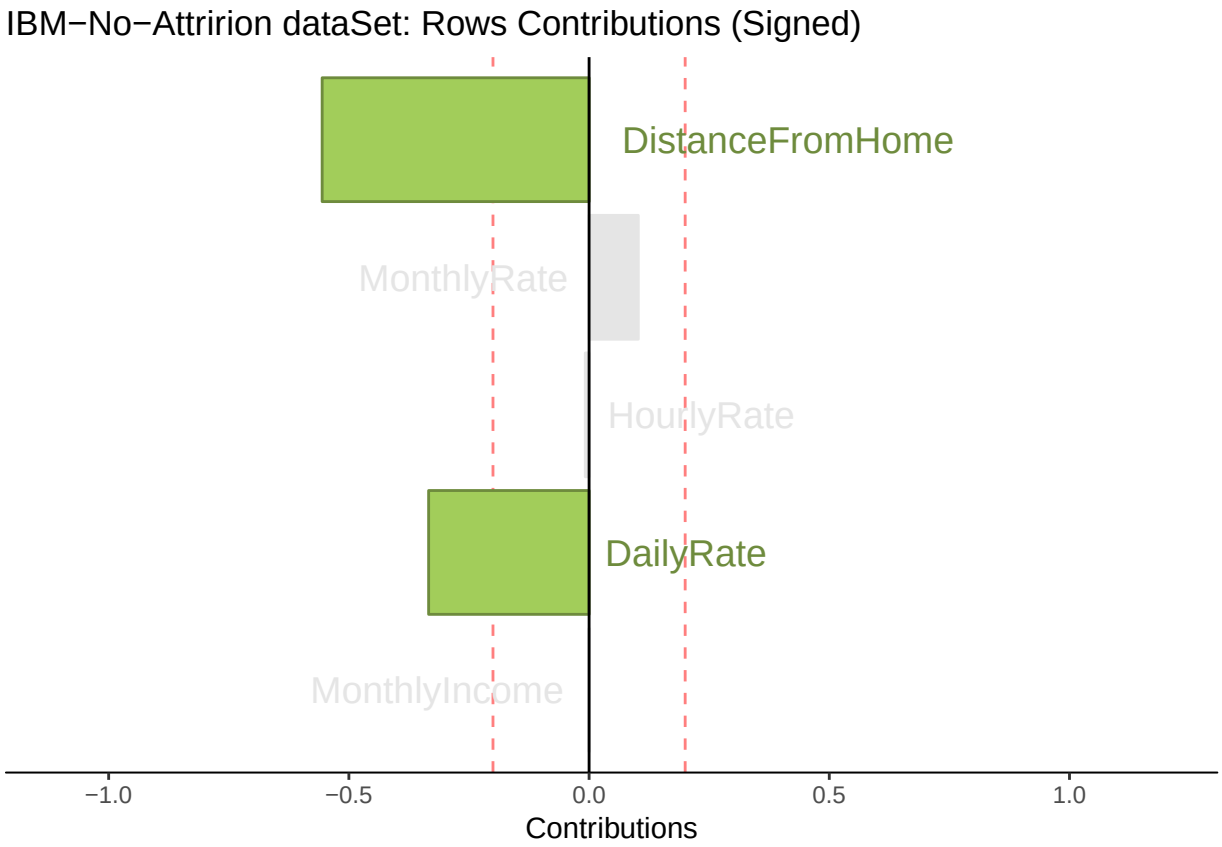
```
signed.ctri <- resPLSC$TExPosition.Data$ci * sign(resPLSC$TExPosition.Data$fi)
b003.ctri.s.1 <- PrettyBarPlot2(signed.ctri[,1],
                                threshold = 1 / NROW(signed.ctri),
                                font.size = 5,
                               # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'IBM-No-Attririon data Set: Rows Contributions (Signed)',
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctri), 1.2*max(signed.ctri)), horizontal = FALSE
)
print(b003.ctri.s.1)
```




```
b004.ctri.s.2 <- PrettyBarPlot2(signed.ctri[,2],
                                threshold = 1 / NROW(signed.ctri),
                                font.size = 5,
                               # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'IBM-No-Attririon dataSet: Rows Contributions (Signed)',
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctri), 1.2*max(signed.ctri)) , horizontal = FALS
)
print(b004.ctri.s.2)
```

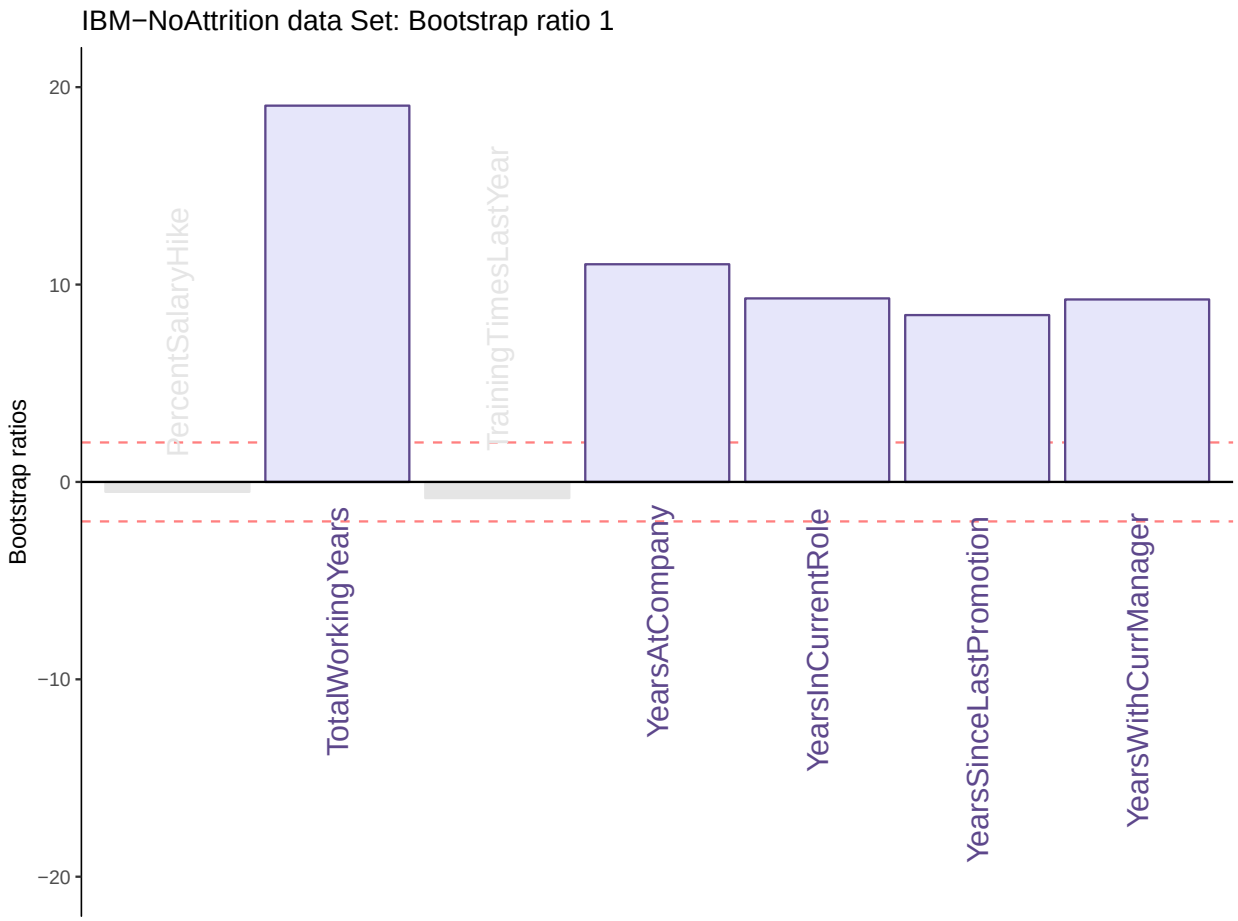



```
b004.ctri.s.3 <- PrettyBarPlot2(signed.ctri[,3],
                                threshold = 1 / NROW(signed.ctri),
                                font.size = 5,
                               # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'IBM-No-Attririon dataSet: Rows Contributions (Signed)',
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctri), 1.2*max(signed.ctri)),horizontal = FALSE
)
print(b004.ctri.s.3)
```

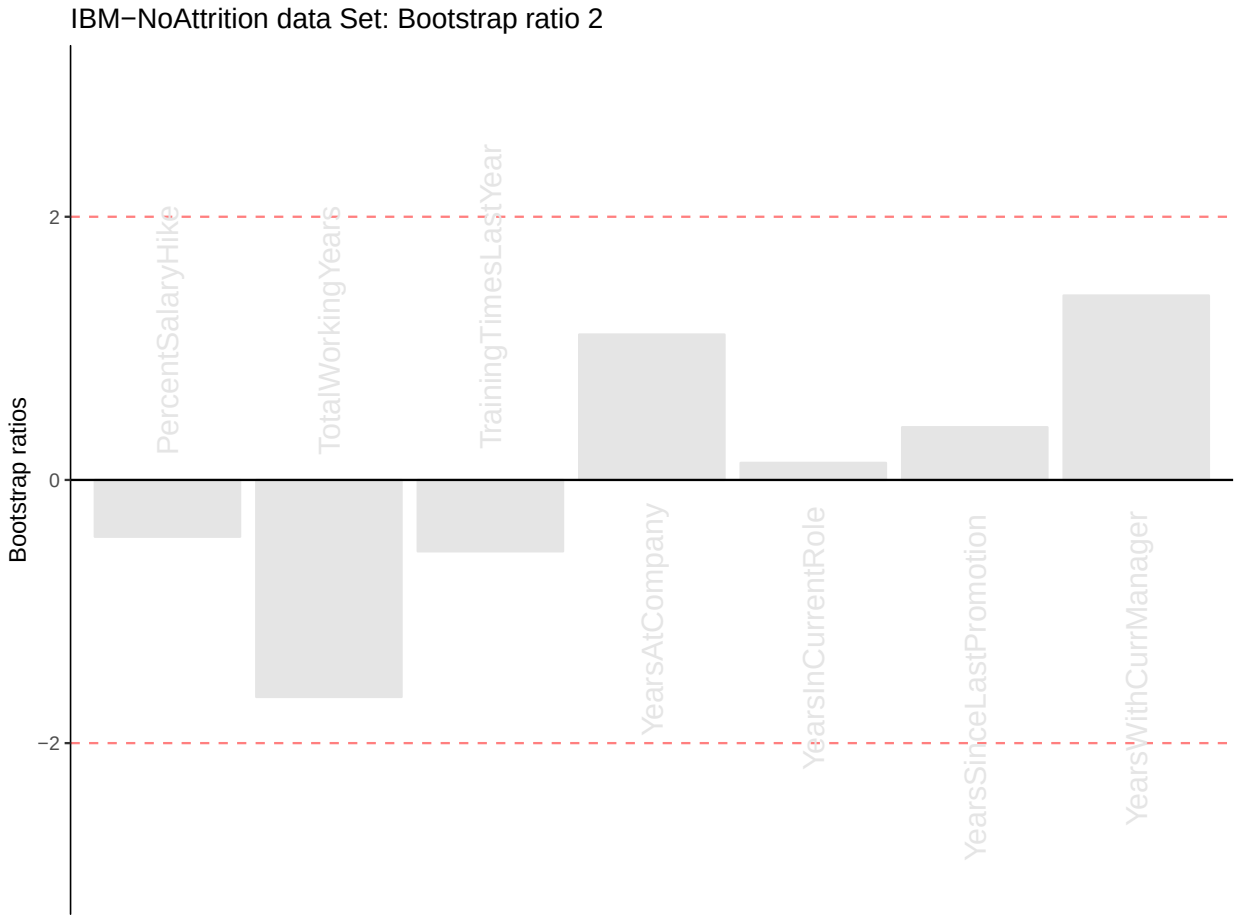


6.5 Bootstrap Ratios for Variables

```
resBoot4PLSC <- Boot4PLSC(data1, # First Data matrix
                          data2, # Second Data matrix
                          nIter = 1000, # How many iterations
                          Fi = resPLSC$TExPosition.Data$fi,
                          Fj = resPLSC$TExPosition.Data$fj,
                          nf2keep = 3,
                          critical.value = 2,
                          # To be implemented later
                          # has no effect currently
                          alphaLevel = .05)
print(resBoot4PLSC)
laDim = 1
ba001.BR1 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.j[,laDim],
                            threshold = 2,
                            font.size = 5,
                            #color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0( 'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
                            #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR1)
```



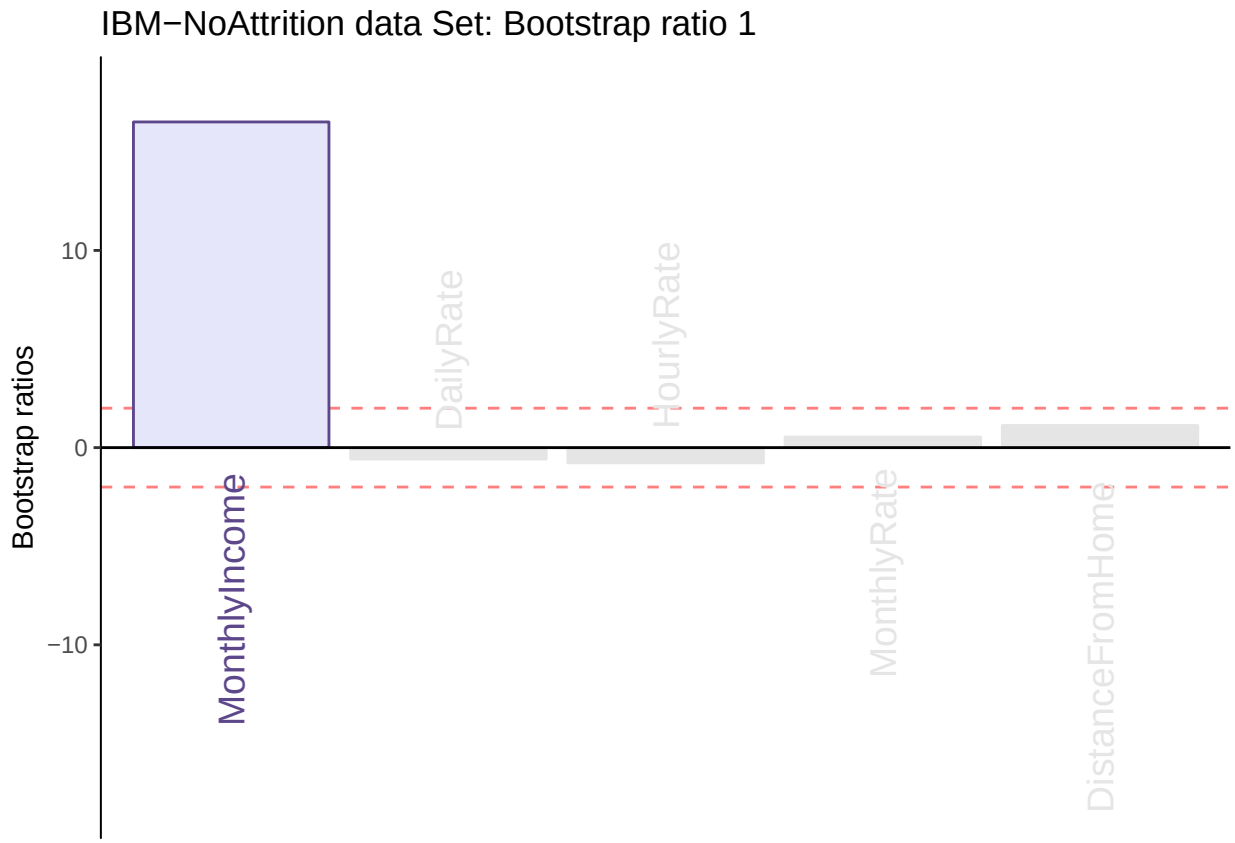

```
#
laDim = 2
ba002.BR2 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.j[,laDim],
                            threshold = 2,
                            font.size = 5,
                            #color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0(
                              'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
)
print(ba002.BR2)
```



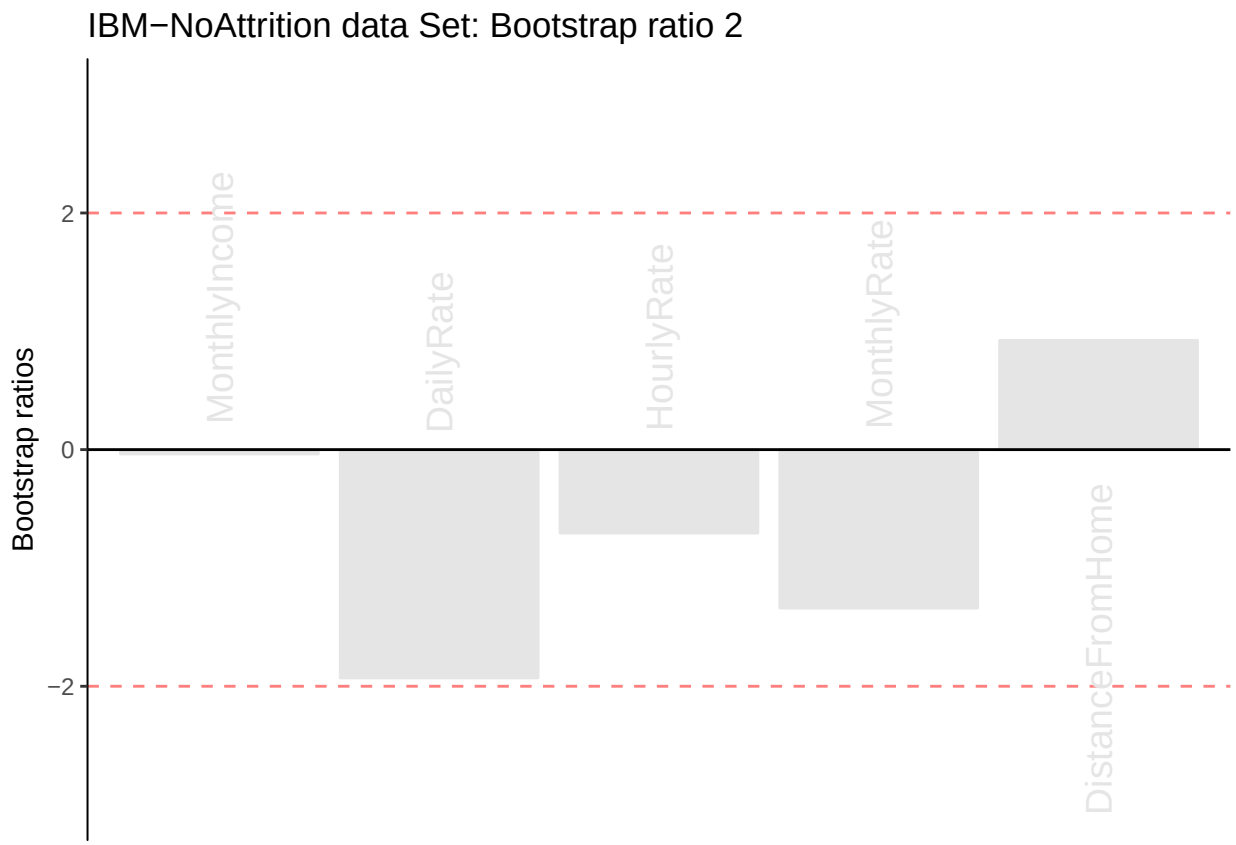

```
laDim = 3
ba002.BR3 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.j[,laDim],
                            threshold = 2,
                            font.size = 5,
                            main = paste0(
                              'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
)
print(ba002.BR3)
```

6.6 Bootstrap Ratios for Rows

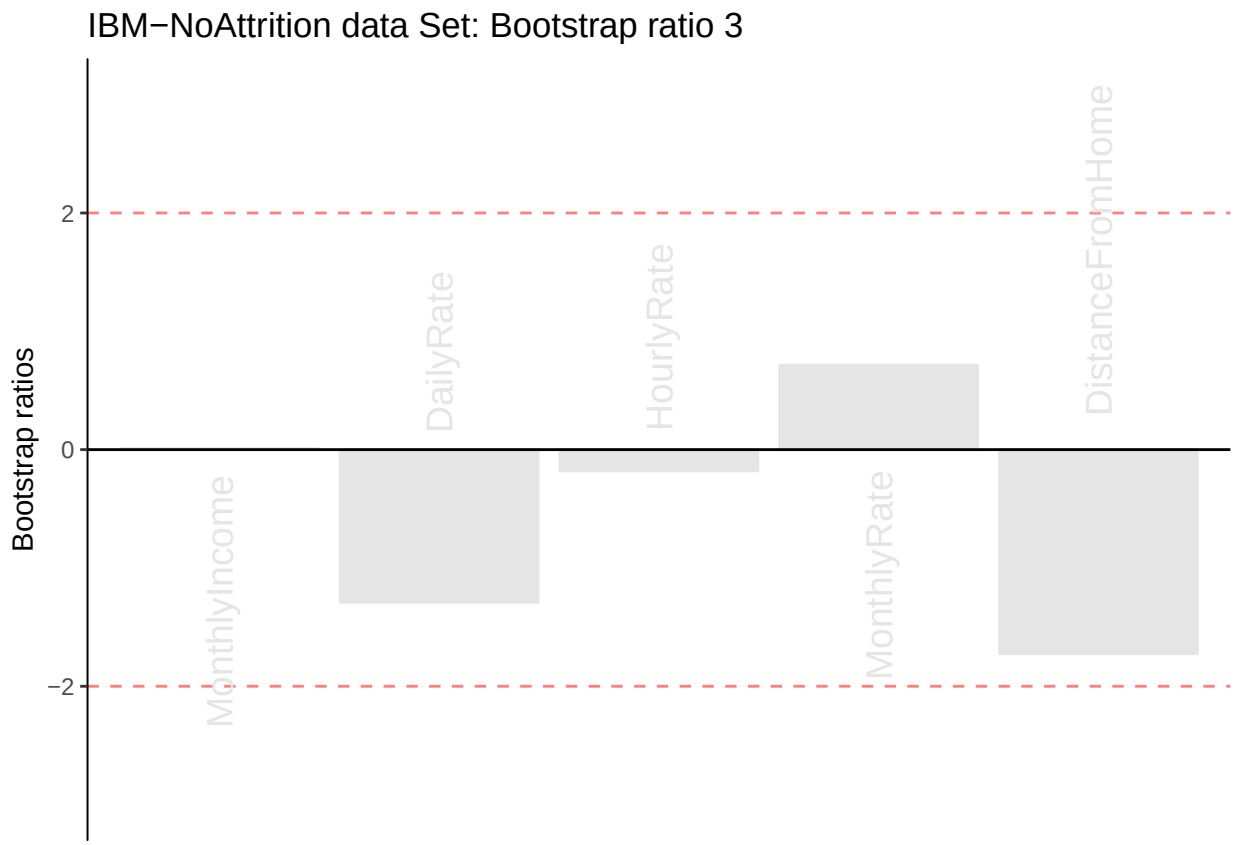
```
laDim = 1
ba001.BR11 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.i[,laDim],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0( 'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                             ylab = 'Bootstrap ratios'
                             #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR11)
```




```
#
laDim = 2
ba002.BR21 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.i[,laDim],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0(
                               'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                             ylab = 'Bootstrap ratios'
)
print(ba002.BR21)
```




```
laDim = 3
ba002.BR31 <- PrettyBarPlot2(resBoot4PLSC$bootRatios.i[,laDim],
                             threshold = 2,
                             font.size = 5,
                             main = paste0(
                               'IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                             ylab = 'Bootstrap ratios'
)
print(ba002.BR31)
```



6.7 Summary

Component 1: -No significant difference between the gender type Females and Males

-No significant difference between the department types HR , Sales and Research & Development

-Managers and the Research Directors have high monthly income and high years of experience for the Job Role

-No significant difference for the Education field type

Component 2: -No significant difference between Females and Males

-No significant difference between the department types HR ,Sales and Research & Development

-No significant difference between the Job Role of different people

-No significant difference for the Education field type

```
options(knitr.duplicate.label = 'allow')
```

7 BADA

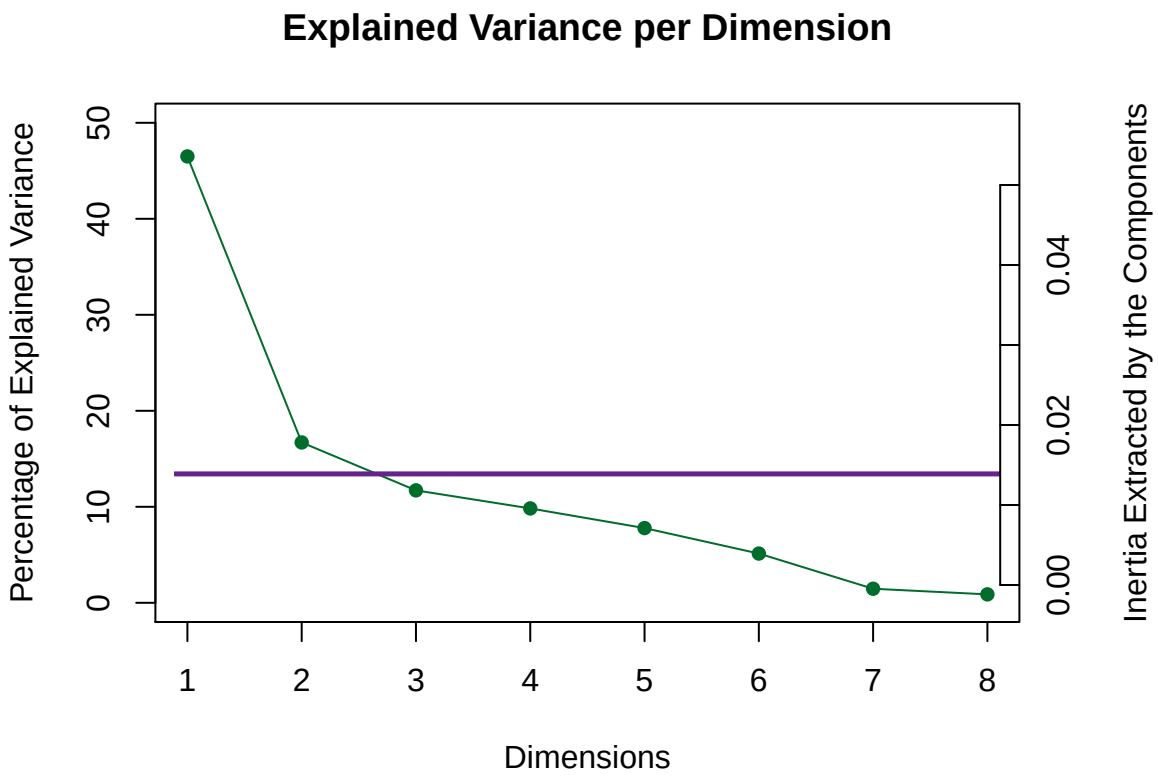
Barycentric discriminant analysis (BADA) is a robust version of discriminant analysis that is used to assign, to pre-defined groups (also called categories), observations described by multiple variables.The goal of BADA is to combine the measurements to create new variables (called components or discriminant variables) that best separate the categories. These discriminant variables are also used to assign the original observations

or new observations to the a-priori defined categories.Barycentric discriminant analysis is a robust version of discriminant analysis that is used when multiple measurements describe a set of observations in which each observation be- longs to one category (i.e., group) from a set of a priori defined categories. BADA combines the original variables to create new variables that best separate the groups and that can also be used to optimally assign old or new observations to these categories. The quality of the performance is evaluated by cross-validation techniques that estimate the performance of the classification model for new observations. BADA is a very versatile technique that can be declined in several different varieties that can handle, for example, qualitative data and data structured in blocks. This versatility make BADA particularly suited for the analysis of multi-modal and Big data.

```
datae <- data[,8]
datar <- data[,5]
resBADA <- tepBADA(DATA = data1,
                   scale = 'SS1', center = TRUE,
                   DESIGN = datae,
                   make_design_nominal = TRUE,
                   group.masses = NULL,
                   weights = NULL, graphs =  FALSE)

resBADA1 <- tepBADA(DATA = data1,
                   scale = 'SS1', center = TRUE,
                   DESIGN = datar,
                   make_design_nominal = TRUE,
                   group.masses = NULL,
                   weights = NULL, graphs =  FALSE)
```

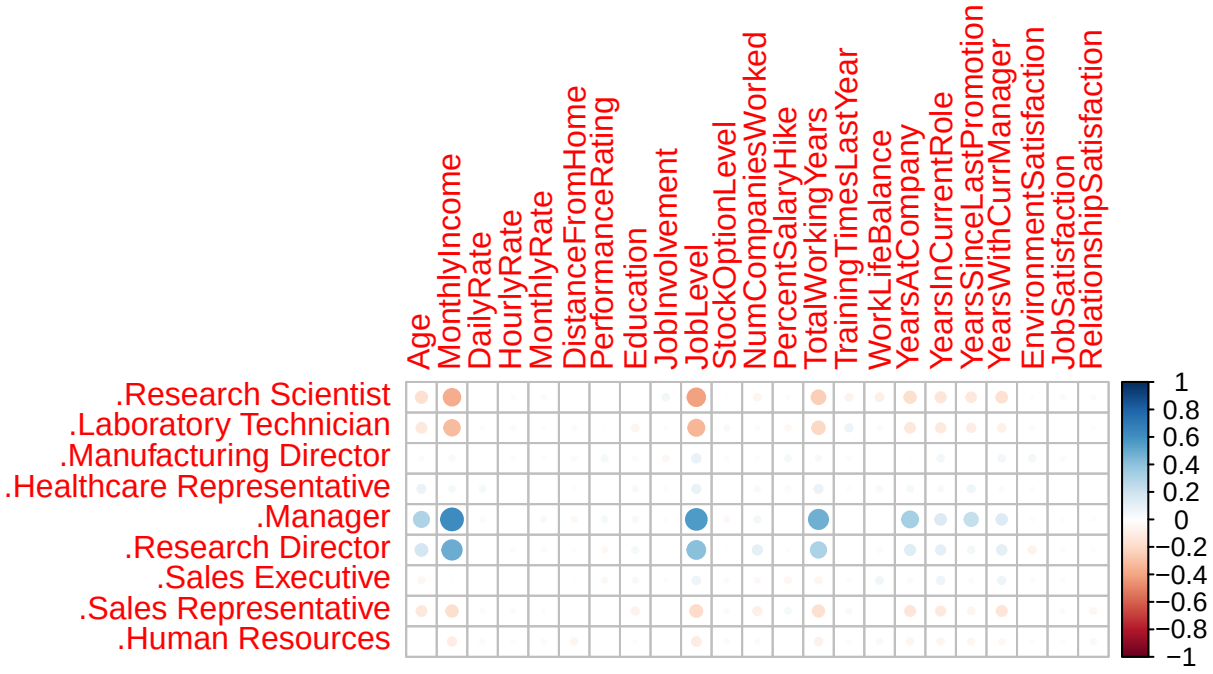
ScreePlot

```
PlotScree(ev = resBADA$TExPosition.Data$eigs,
          p.ev = NULL, max.ev = NULL, alpha = 0.05,
          col.ns = "#006D2C", col.sig = "#54278F",
          title = "Explained Variance per Dimension",plotKaiser = TRUE)
```



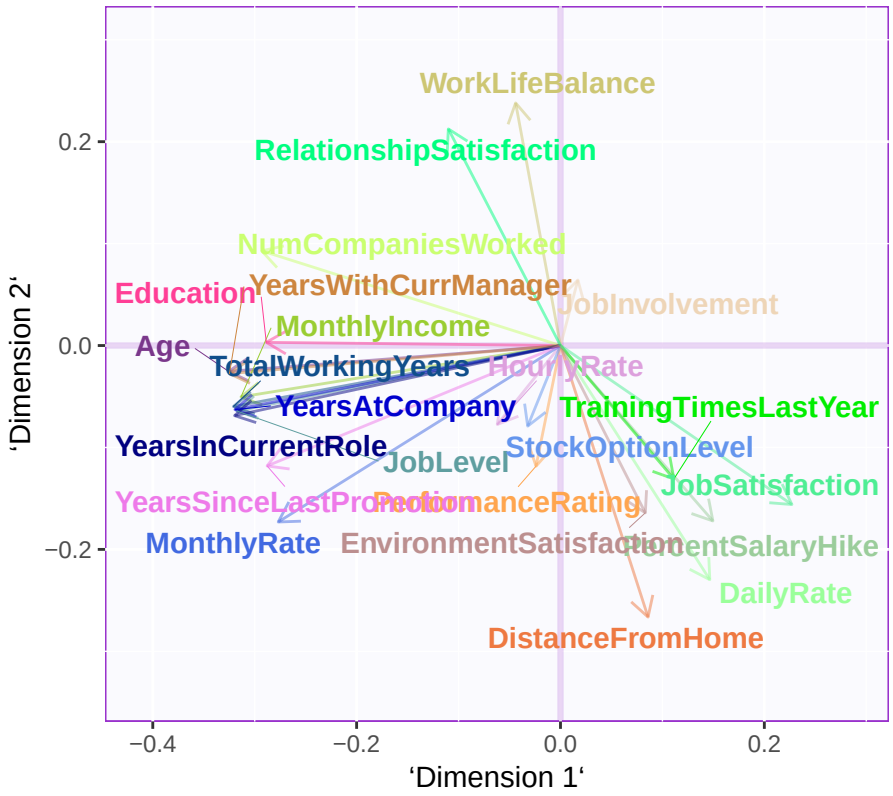
HeatMap

```
color4Var <- prettyGraphs::prettyGraphsColorSelection(ncol(data1))
data2 <- makeNominalData(as.matrix(data[,8]))
corrplot::corrplot(cor(data2,data1))
```



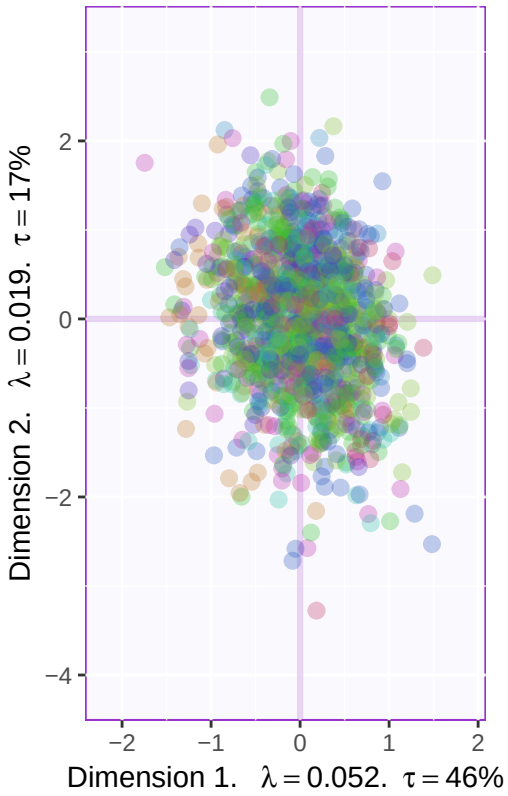
7.1 Factor Map J

```
col4Var <- prettyGraphsColorSelection(NCOL(data1))
baseMap.j <- PTCA4CATA::createFactorMap(Fj,
                            col.points   = col4Var,
                            alpha.points =  .3,
                            col.labels   = col4Var)
# A graph for the J-set
aggMap.j <- baseMap.j$zeMap_background + # background layer
  baseMap.j$zeMap_dots + baseMap.j$zeMap_text # dots & labels
# We print this Map with the following code
zeLines <- ggplot2::annotate("segment", x = c(0), y = c(0),
                     xend = Fj[,1],
                     yend = Fj[,2],
                     color = col4Var,
                     alpha = .5,
                     arrow = arrow(length = unit(.3, "cm") ) )
# Create the map by adding background, labels, and arrows:
aggMap.j.arrows <- baseMap.j$zeMap_background +
                                      zeLines + baseMap.j$zeMap_text
print(aggMap.j.arrows)
```

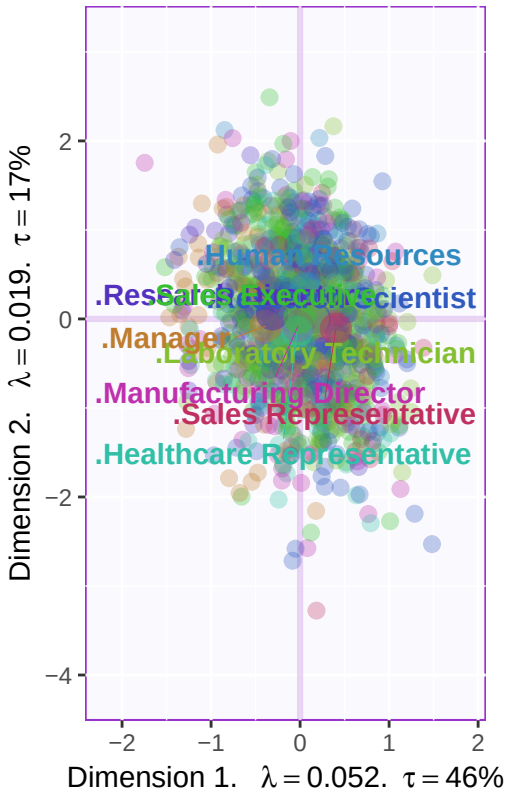


7.2 Factor Map I

```
baseMap.i <- PTCA4CATA::createFactorMap(Fi,
                                        col.points   = resBADA$Plotting.Data$fii.col,
                                        alpha.points =  .3)
labels4BADA <- createxyLabels.gen(x_axis = 1, y_axis = 2, lambda = resBADA$TExPosition.Data$eigs,
tau = resBADA$TExPosition.Data$t)

# Plain map with color for the I-set
aggMap.i <- baseMap.i$zeMap_background + baseMap.i$zeMap_dots + labels4BADA
#------------------------------------------------------------------------
print(aggMap.i)
```

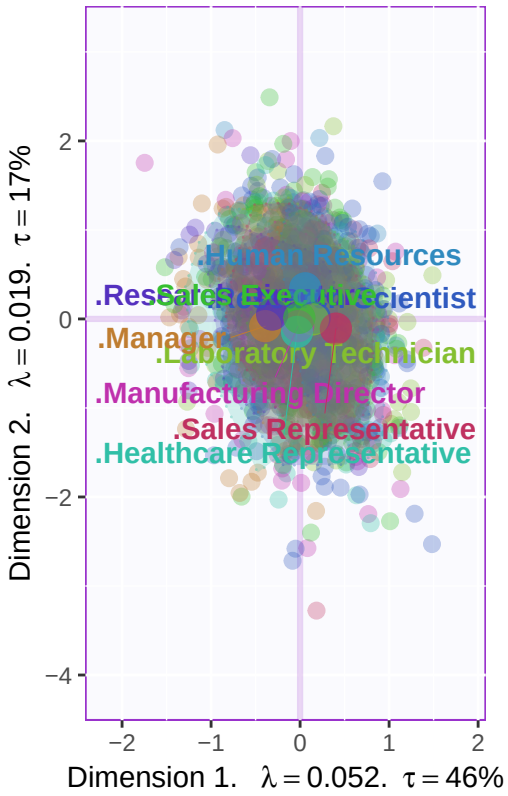



```
col4data <- resBADA$Plotting.Data$fii.col
col4Means <- unique(col4data)
# create the map for the means
MapGroup    <- PTCA4CATA::createFactorMap(Fk,
                                axis1 = 1, axis2 = 2,
                                constraints = baseMap.i$constraints,
                                title = NULL,
                                col.points = col4Means,
                                display.points = TRUE,
                                pch = 19, cex = 5,
                                display.labels = TRUE,
                                col.labels = col4Means,
                                text.cex = 4,
                                font.face = "bold",
                                font.family = "sans",
                                col.axes = "darkorchid",
                                alpha.axes = 0.2,
                                width.axes = 1.1,
                              col.background = adjustcolor("lavender",
                                                           alpha.f = 0.2),
                              force = 1, segment.size = 0)
# The map with observations and group means
aggMap.i.withMeans <- aggMap.i+
  MapGroup$zeMap_dots + MapGroup$zeMap_text
#-------------------------------------------------------------------------
print(aggMap.i.withMeans)
```



Create 75% Confidence interval polygons

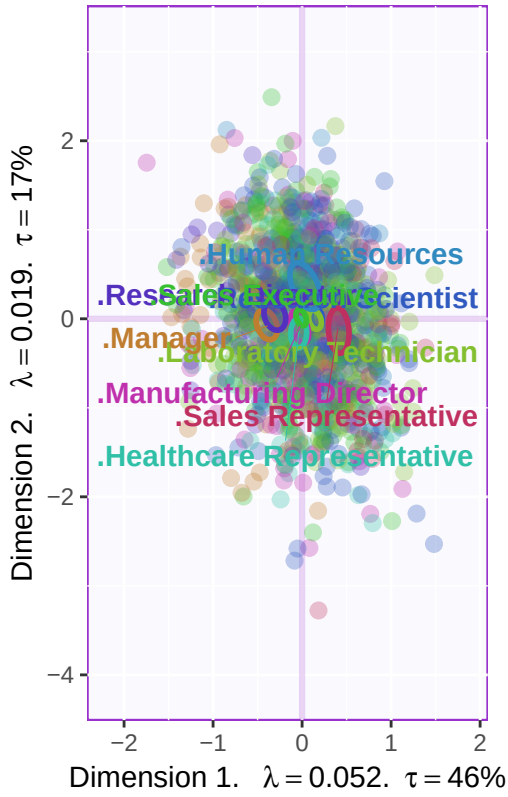
```
GraphTI.Hull.90 <- MakeToleranceIntervals(Fi,
                                  as.factor(datae),
                                  names.of.factors = c("Dim1","Dim2"),
                                  col = unique(col4data),
                                  line.size = .5, line.type = 3,
                                  alpha.ellipse = .2,
                                  alpha.line = .4,
                                  p.level = .75, # 75% TI
                                  type = 'hull' #
                                          # use 'hull' for convex hull
)
#-------------------------------------------------------------------------
# Create the map
aggMap.i.withHull <- aggMap.i +
  GraphTI.Hull.90 + MapGroup$zeMap_dots +
  MapGroup$zeMap_text +  MapGroup$zeMap_dots
#-------------------------------------------------------------------------

print(aggMap.i.withHull)
```



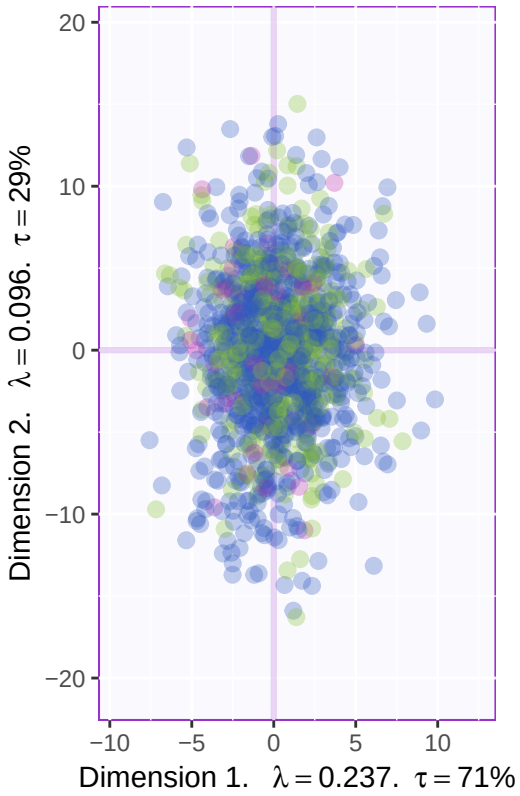
Inferences

```
resBADA.inf <- tepBADA.inference.battery(DATA = data1,
                    scale = 'SS1', center = TRUE,
                    DESIGN = datae,
                    make_design_nominal = TRUE,
                    group.masses = NULL,
                    weights = NULL,
                    graphs = FALSE,
                    k = 2,
                    test.iters = 100,
                    critical.value = 2)
#-------------------------------------------------------------------
# Confusion matrices
# To be saved as table
fixedCM   <-   resBADA.inf$Inference.Data$loo.data$fixed.confuse
looedCM   <- resBADA.inf$Inference.Data$loo.data$loo.confuse

#-------------------------------------------------------------------
# Create Confidence Interval Plots
BootCube <- resBADA.inf$Inference.Data$boot.data$fi.boot.data$boots
dimnames(BootCube)[[2]] <- c("Dimension 1","Dimension 2")
# use function MakeCIEllipses from package PTCA4CATA
GraphElli <- MakeCIEllipses(BootCube[,1:2,],
                    names.of.factors = c("Dimension 1","Dimension 2"),
                    col = col4Means,
                    p.level = .95
```

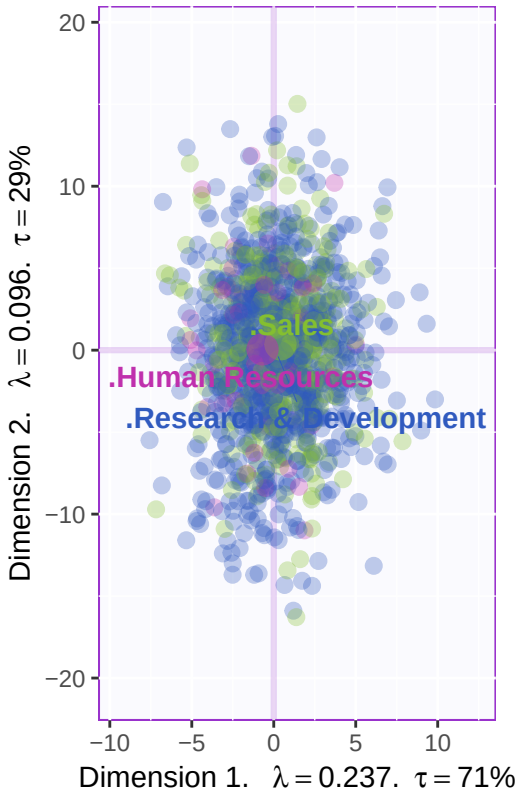
```
)
#--------------------------------------------------------------------
# create the I-map with Observations, means and confidence intervals
#
aggMap.i.withCI <-  aggMap.i +  GraphElli + MapGroup$zeMap_text
#--------------------------------------------------------------------

print(aggMap.i.withCI)
```




```
baseMap.i1 <- PTCA4CATA::createFactorMap(Fi1,
                                         col.points   = resBADA1$Plotting.Data$fii.col,
                                         alpha.points =  .3)
labels4BADA1 <- createxyLabels.gen(x_axis = 1, y_axis = 2, lambda = resBADA1$TExPosition.Data$eigs,
tau = resBADA1$TExPosition.Data$t)

# Plain map with color for the I-set
aggMap.i1 <- baseMap.i1$zeMap_background + baseMap.i1$zeMap_dots + labels4BADA1
#--------------------------------------------------------------------
print(aggMap.i1)
```

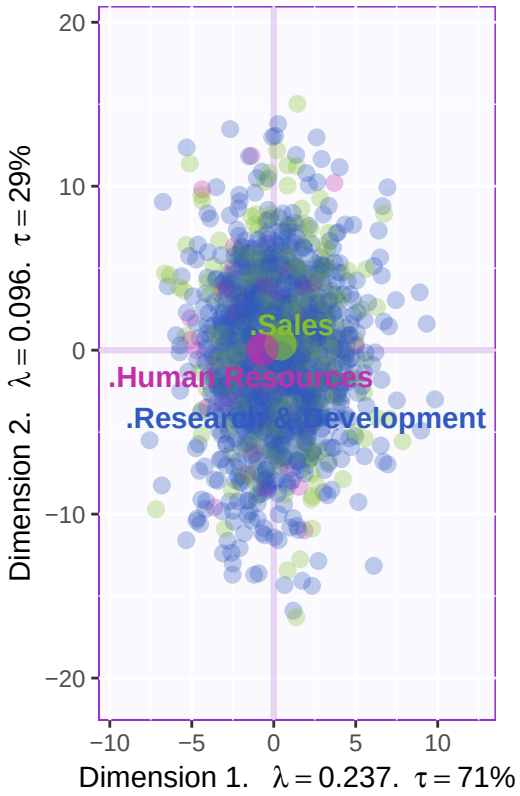



```
col4data1 <- resBADA1$Plotting.Data$fii.col
col4Means1 <- unique(col4data1)
# create the map for the means
MapGroup1    <- PTCA4CATA::createFactorMap(Fk1,
                                  axis1 = 1, axis2 = 2,
                                  constraints = baseMap.i$constraints,
                                  title = NULL,
                                  col.points = col4Means1,
                                  display.points = TRUE,
                                  pch = 19, cex = 5,
                                  display.labels = TRUE,
                                  col.labels = col4Means1,
                                  text.cex = 4,
                                  font.face = "bold",
                                  font.family = "sans",
                                  col.axes = "darkorchid",
                                  alpha.axes = 0.2,
                                  width.axes = 1.1,
                                col.background = adjustcolor("lavender",
                                                             alpha.f = 0.2),
                                force = 1, segment.size = 0)
# The map with observations and group means
aggMap.i.withMeans1 <- aggMap.i1 +
  MapGroup1$zeMap_dots + MapGroup1$zeMap_text
#-----------------------------------------------------------------------
print(aggMap.i.withMeans1)
```



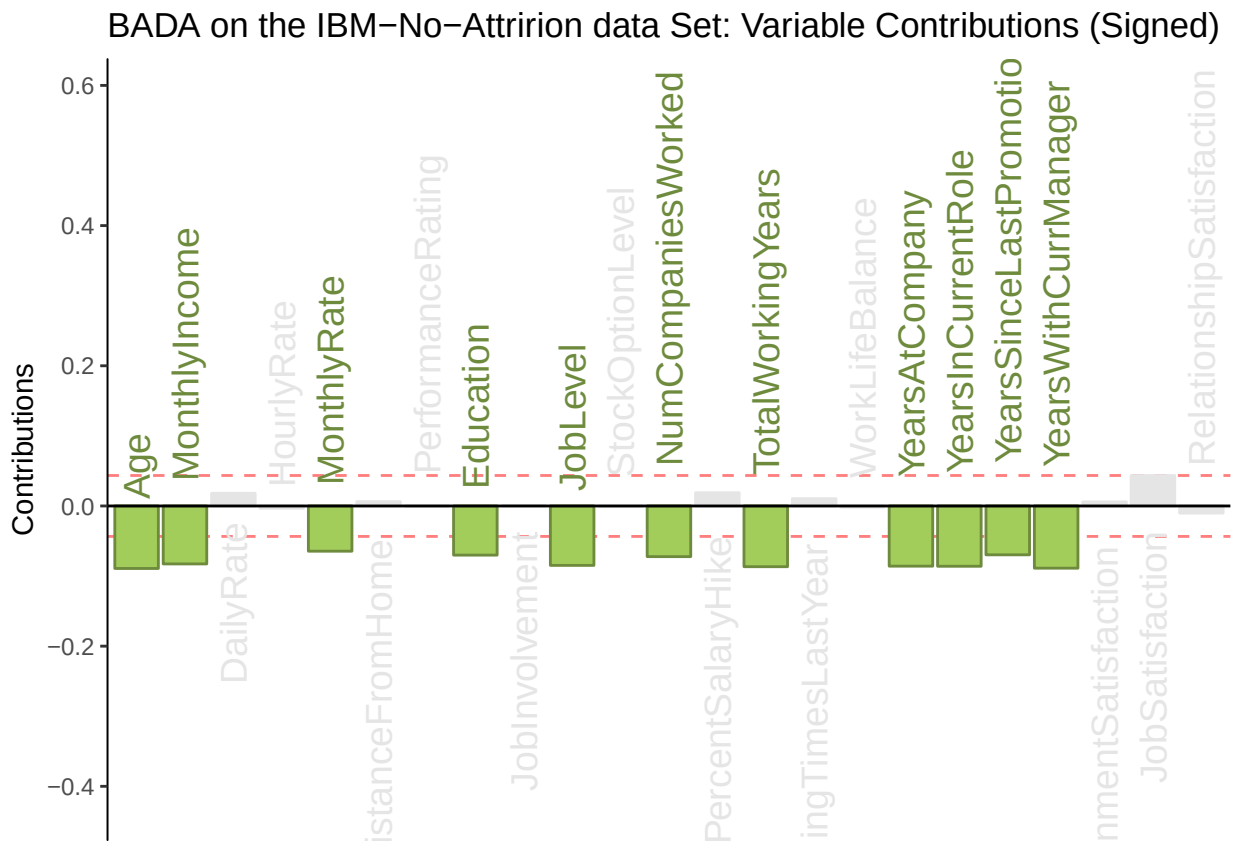
Create 75% Confidence interval polygons

```
GraphTI.Hull.901 <- MakeToleranceIntervals(Fi1,
                                  as.factor(datar),
                                  names.of.factors = c("Dim1","Dim2"),
                                  col = unique(col4data),
                                  line.size = .5, line.type = 3,
                                  alpha.ellipse = .2,
                                  alpha.line = .4,
                                  p.level = .75, # 75% TI
                                  type = 'hull' #
                                          # use 'hull' for convex hull
)
#-------------------------------------------------------------------------
# Create the map
aggMap.i.withHull1 <- aggMap.i1 +
  GraphTI.Hull.901 + MapGroup1$zeMap_dots +
  MapGroup1$zeMap_text +  MapGroup1$zeMap_dots
#-------------------------------------------------------------------------

print(aggMap.i.withHull1)
```

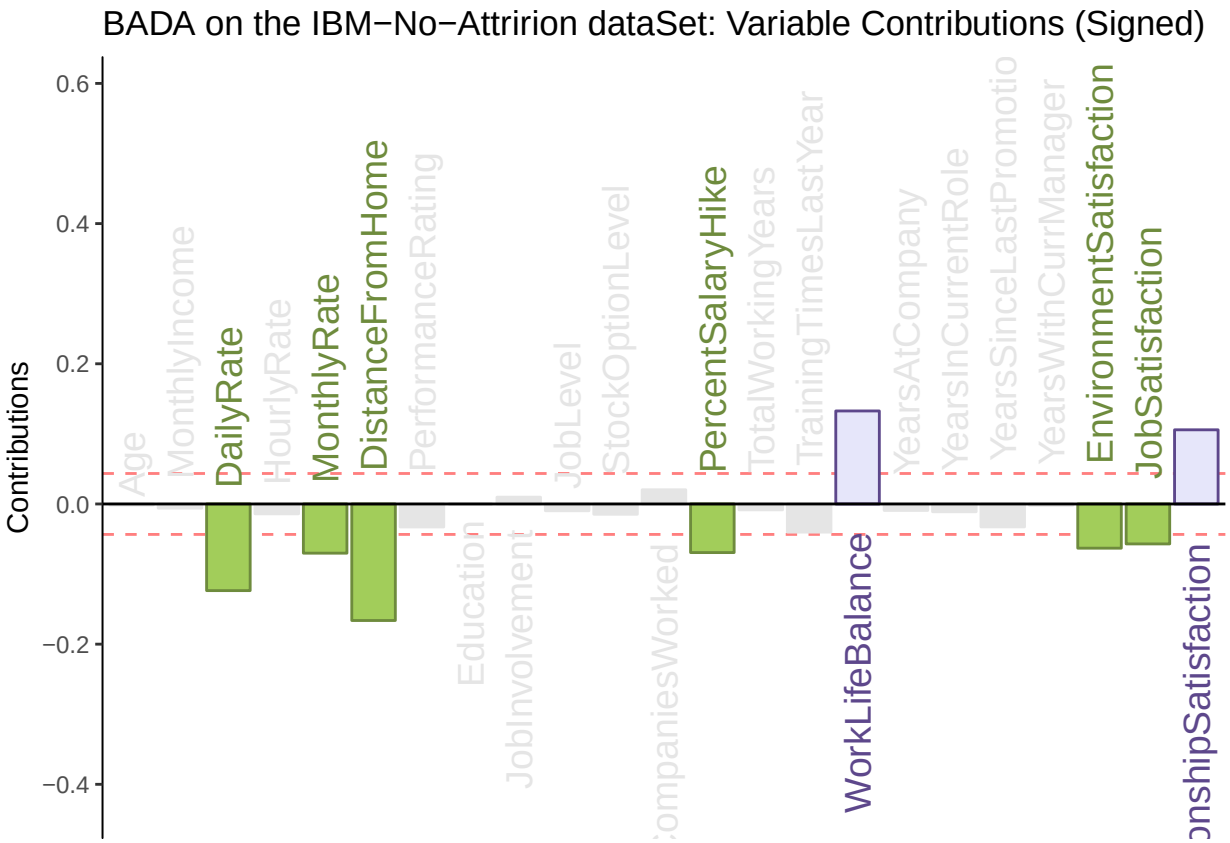


7.3 Contribution

```
signed.ctrJ1 <- resBADA$TExPosition.Data$cj * sign(resBADA$TExPosition.Data$fj)
b003.ctrJ.s.11 <- PrettyBarPlot2(signed.ctrJ1[,1],
                                 threshold = 1 / NROW(signed.ctrJ1),
                                 font.size = 5,
                                 # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'BADA on the IBM-No-Attririon data Set: Variable Contributions (S
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ1), 1.2*max(signed.ctrJ1))
)
print(b003.ctrJ.s.11)
```

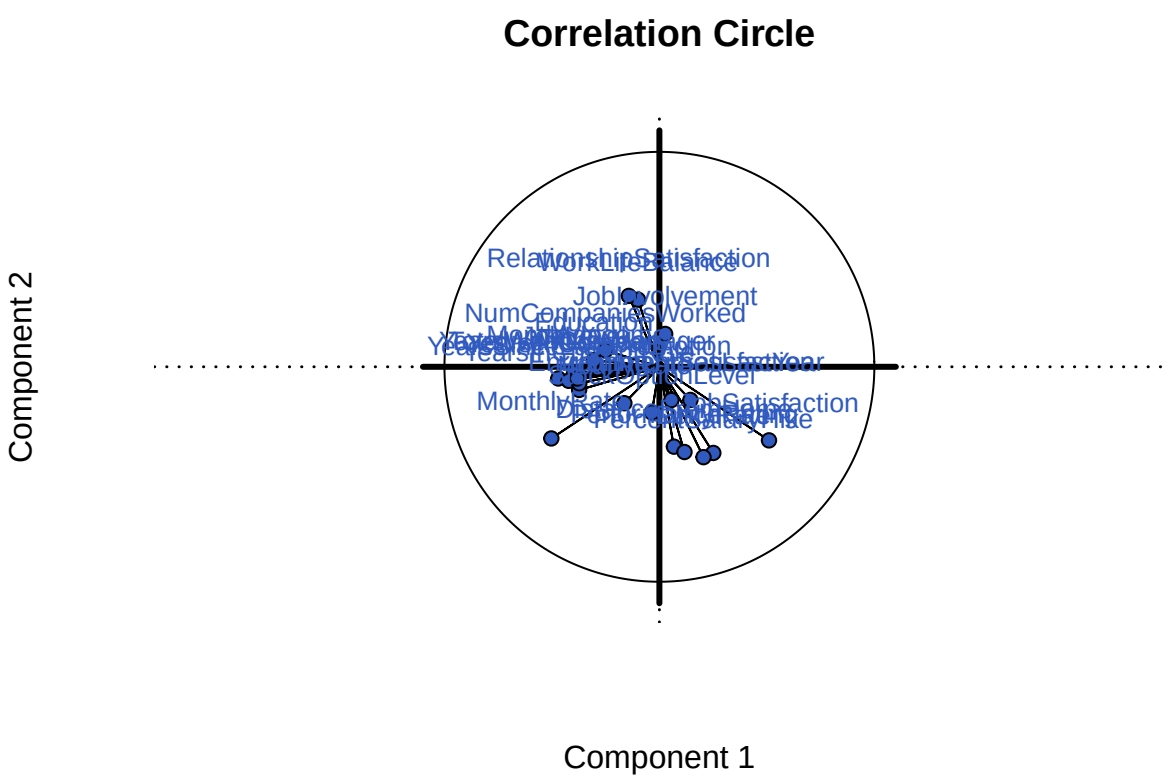



```
b004.ctrJ.s.21 <- PrettyBarPlot2(signed.ctrJ1[,2],
                                 threshold = 1 / NROW(signed.ctrJ1),
                                 font.size = 5,
                                 # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                 main = 'BADA on the IBM-No-Attririon dataSet: Variable Contributions (S
                                 ylab = 'Contributions',
                                 ylim = c(1.2*min(signed.ctrJ1), 1.2*max(signed.ctrJ1))
)
print(b004.ctrJ.s.21)
```



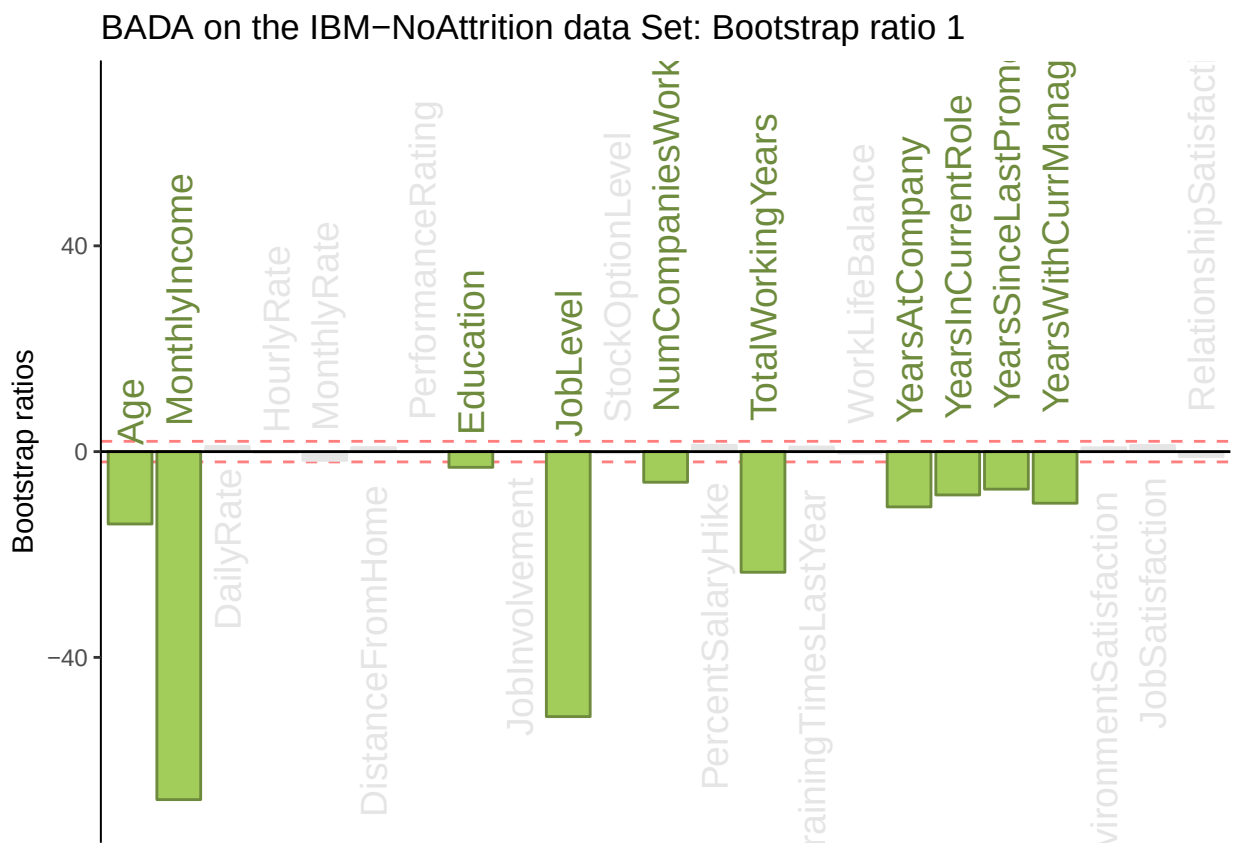
Correlation Circle

```
correlationCircle <- correlationPlotter(data_matrix = data1 , factor_scores = resBADA$TExPosition.Data$
```

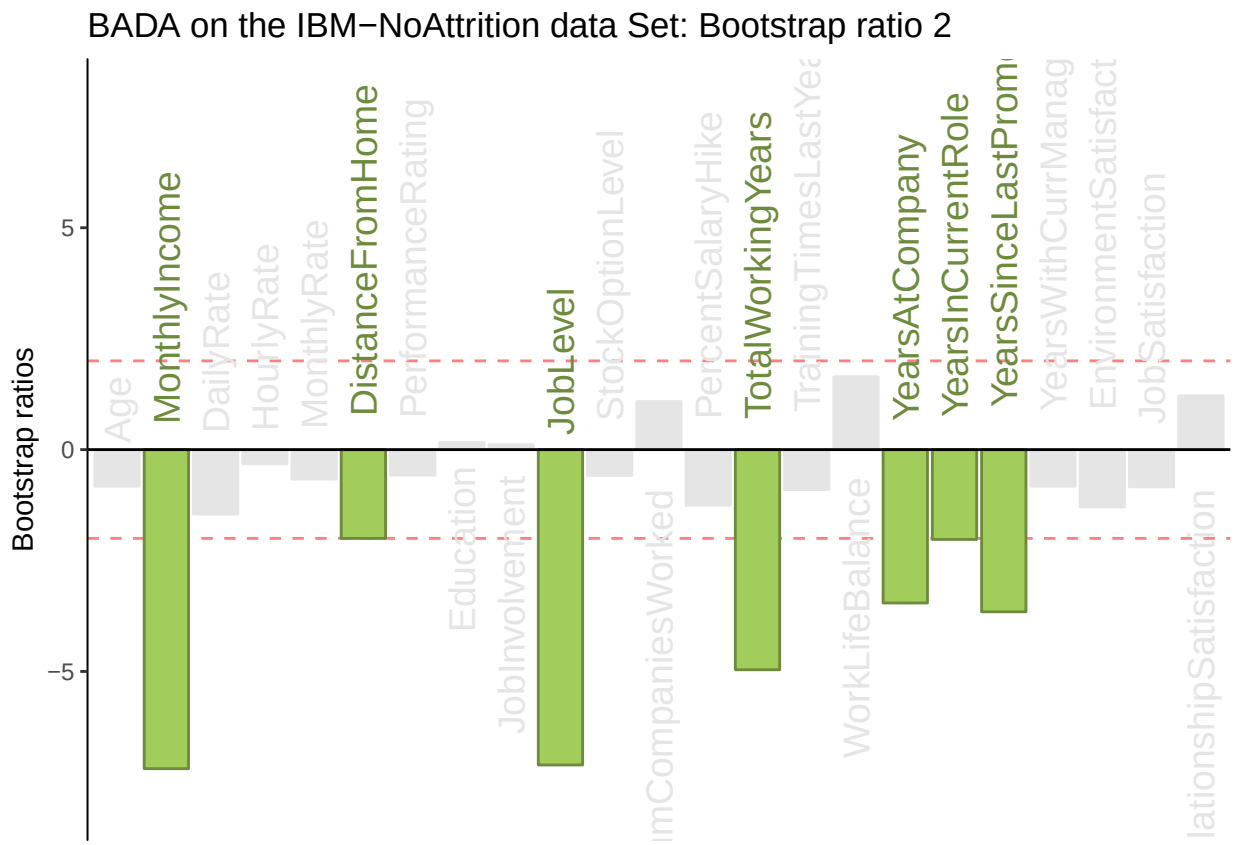


7.4 Bootstrap Ratios

```
BR1 <- resBADA.inf$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
laDim = 1
ba001.BR11 <- PrettyBarPlot2(BR1[,laDim],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0( 'BADA on the IBM-NoAttrition data Set: Bootstrap ratio ',laD
                             ylab = 'Bootstrap ratios'
                             #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR11)
```




```
#
laDim = 2
ba002.BR21 <- PrettyBarPlot2(BR1[,laDim],
                             threshold = 2,
                             font.size = 5,
                             #color4bar = gplots::col2hex(col4J.ibm),
                             main = paste0(
                               'BADA on the IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                             ylab = 'Bootstrap ratios'
)
print(ba002.BR21)
```



Random (LOO) confusion matrix

```
#-----------------------------------------------------------------------
# Confusion matrices
# To be saved as table
fixedCM1   <-   resBADA.inf$Inference.Data$loo.data$fixed.confuse
looedCM1   <- resBADA.inf$Inference.Data$loo.data$loo.confuse
```

7.5 Summary

- No significant inference observed in BADA as both the design variables Department and Job Role cannot be significantly diffrentiated from their mean Confidence interval plot.

```
options(knitr.duplicate.label = 'allow')
```

8 DiCA

The main idea behind DCA is to represent each group by the sum of its observations and to perform a simple CA on the groups by variables matrix. The original observations are then projected as supplementary elements and each observation is assigned to the closest group. The comparison between the a priori and the a posteriori classifications can be used to assess the quality of the discrimination. A similar procedure can be used to assign new observations to categories. The stability of the analysis can be evalu- ated using cross-validation techniques such as jackknifing or bootstrapping.

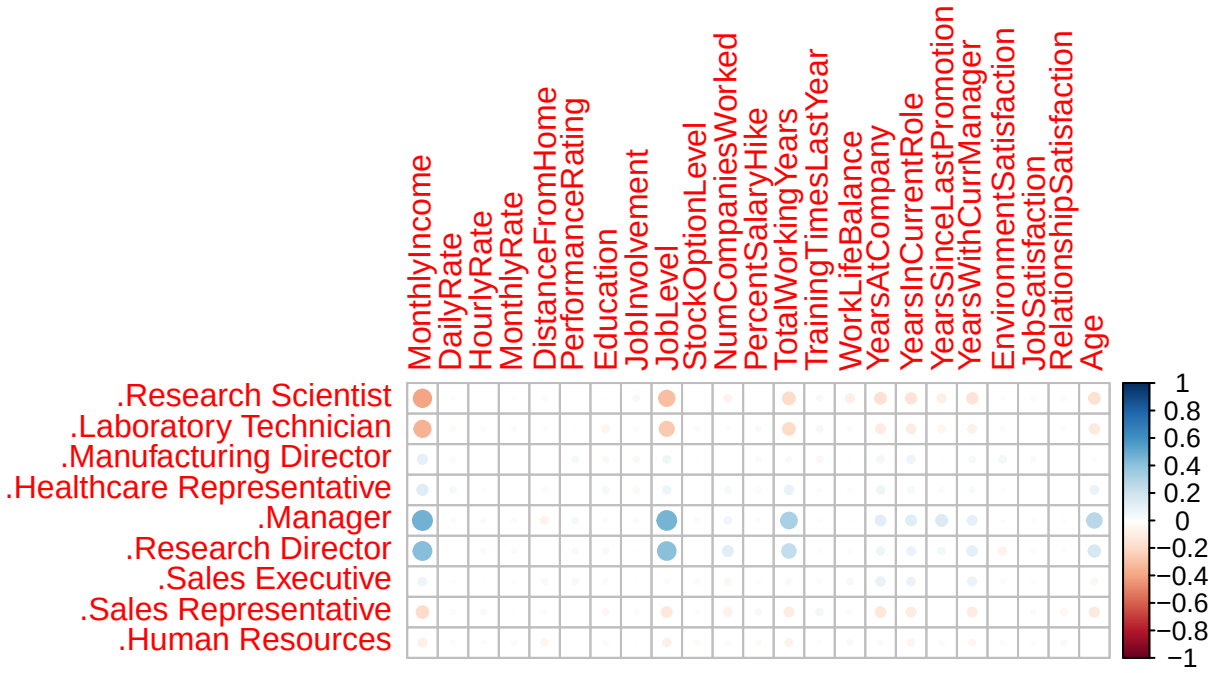
```
options(knitr.duplicate.label = 'allow')
my_data <- read.csv("IBM-HR-Emplyee-NoAttrition.csv")
cols <- colnames(my_data)
rownames(my_data) <- my_data$Subj
data1 <- my_data[,11:32]
data1$Age <- my_data$Age
```

Distribution of the quantitative data

Binning the quantitative data in the similar fashion as MCA

8.1 Heatmap

```
color4Var <- prettyGraphs::prettyGraphsColorSelection(ncol(data1))
data2 <- makeNominalData(as.matrix(my_data[,8]))
data3<- data.matrix(data1)
corrplot::corrplot(cor(data2,data3))
```




```
datae <- my_data$JobRole
datar <- my_data$Department
resDICA <- tepDICA(DATA = data1,
                   DESIGN = datae,
                   make_design_nominal = TRUE,
                   make_data_nominal = TRUE,
                   group.masses = NULL,
```

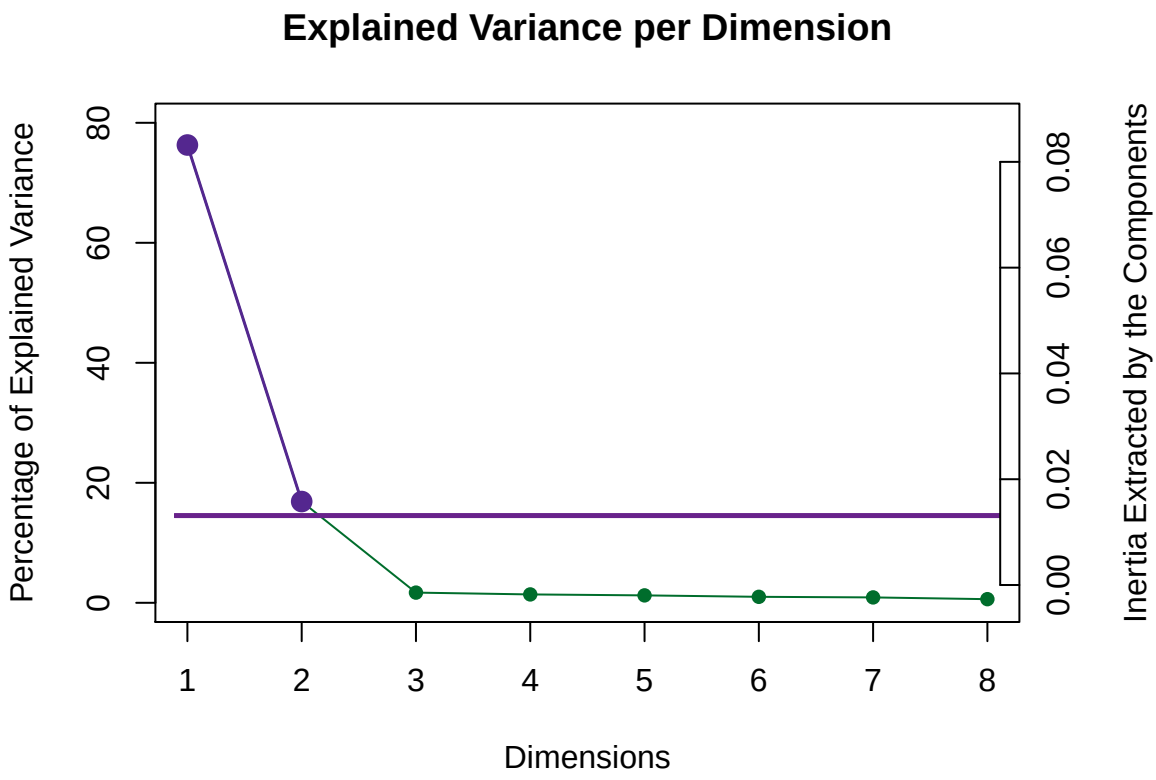
```
                    weights = NULL, graphs =  FALSE)

resDICA1 <- tepDICA(DATA = data1,
                    DESIGN = datar,
                    make_design_nominal = TRUE,
                    make_data_nominal = TRUE,
                    group.masses = NULL,
                    weights = NULL, graphs =  FALSE)
```

ScreePlot for DICA

```
resDICA.inf1 <- tepDICA.inference.battery(DATA = data2,
                    DESIGN = datae,
                    make_design_nominal = TRUE,
                    make_data_nominal = TRUE,
                    group.masses = NULL,
                    weights = NULL,
                    graphs = FALSE,
                    k = 2,
                    test.iters = 100,
                    critical.value = 2)
```

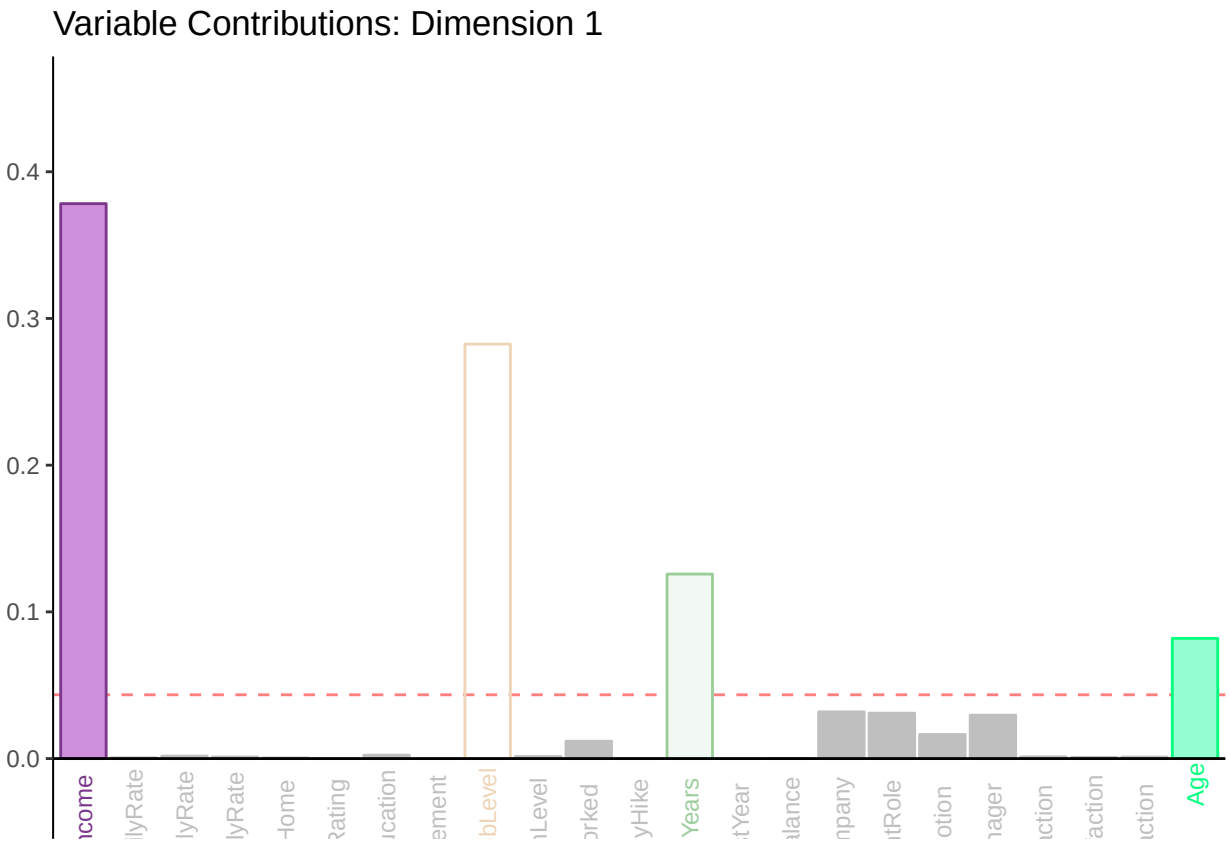
```
## [1] "It is estimated that your iterations will take 0.59 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note of the
## ================================================================================
```

```
PlotScree(ev = resDICA$TExPosition.Data$eigs,
          p.ev = resDICA.inf1$Inference.Data$components$p.vals, max.ev = NULL, alpha = 0.05,
          col.ns = "#006D2C", col.sig = "#54278F",
          title = "Explained Variance per Dimension",plotKaiser = TRUE)
```

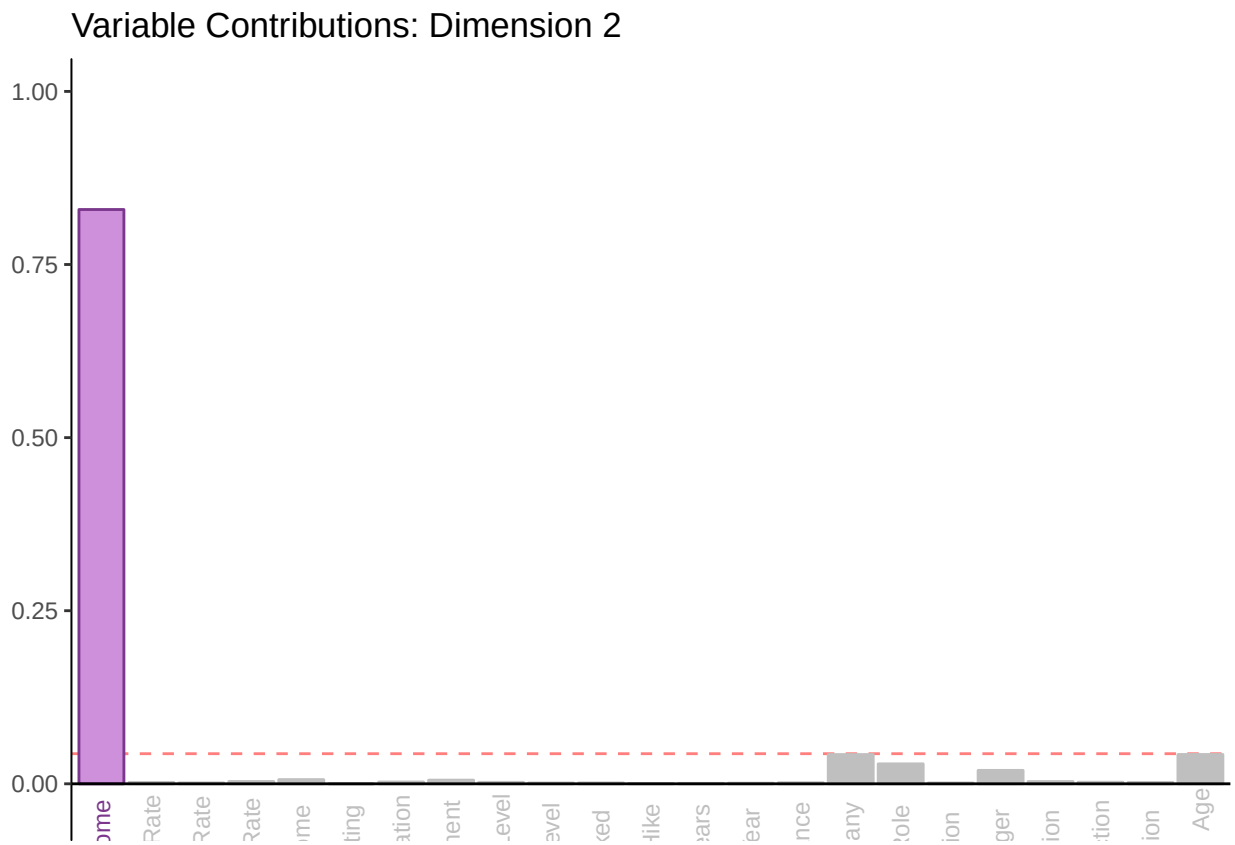


8.2 Variable contributions

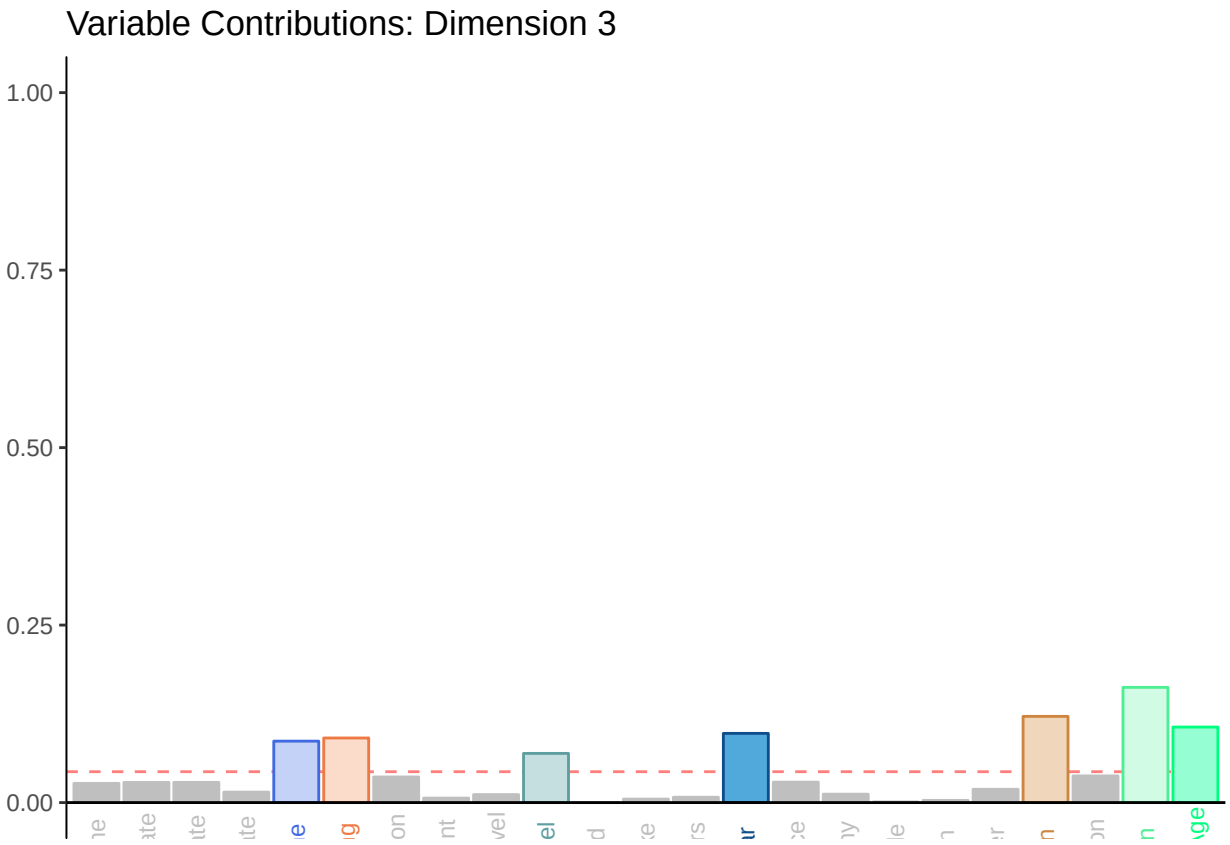
```
color <- prettyGraphs::prettyGraphsColorSelection(ncol(data1))
cJ <- resDICA$TExPosition.Data$cj
varCtr <- data4PCCAR::ctr4Variables(cJ)
rownames(color) <- rownames(varCtr)
varCtr1 <- varCtr[,1]
names(varCtr1) <- rownames(varCtr)
a0005.Var.ctr1 <- PrettyBarPlot2(varCtr1, main = 'Variable Contributions: Dimension 1', ylim = c(-.03,1
print(a0005.Var.ctr1)
```




```
varCtr2 <- varCtr[,2]
names(varCtr2) <- rownames(varCtr)
a0006.Var.ctr2 <- PrettyBarPlot2(varCtr2,
main = 'Variable Contributions: Dimension 2', ylim = c(-.03, 1.2*max(varCtr2)),threshold = 1 / nrow(varC
print(a0006.Var.ctr2)
```

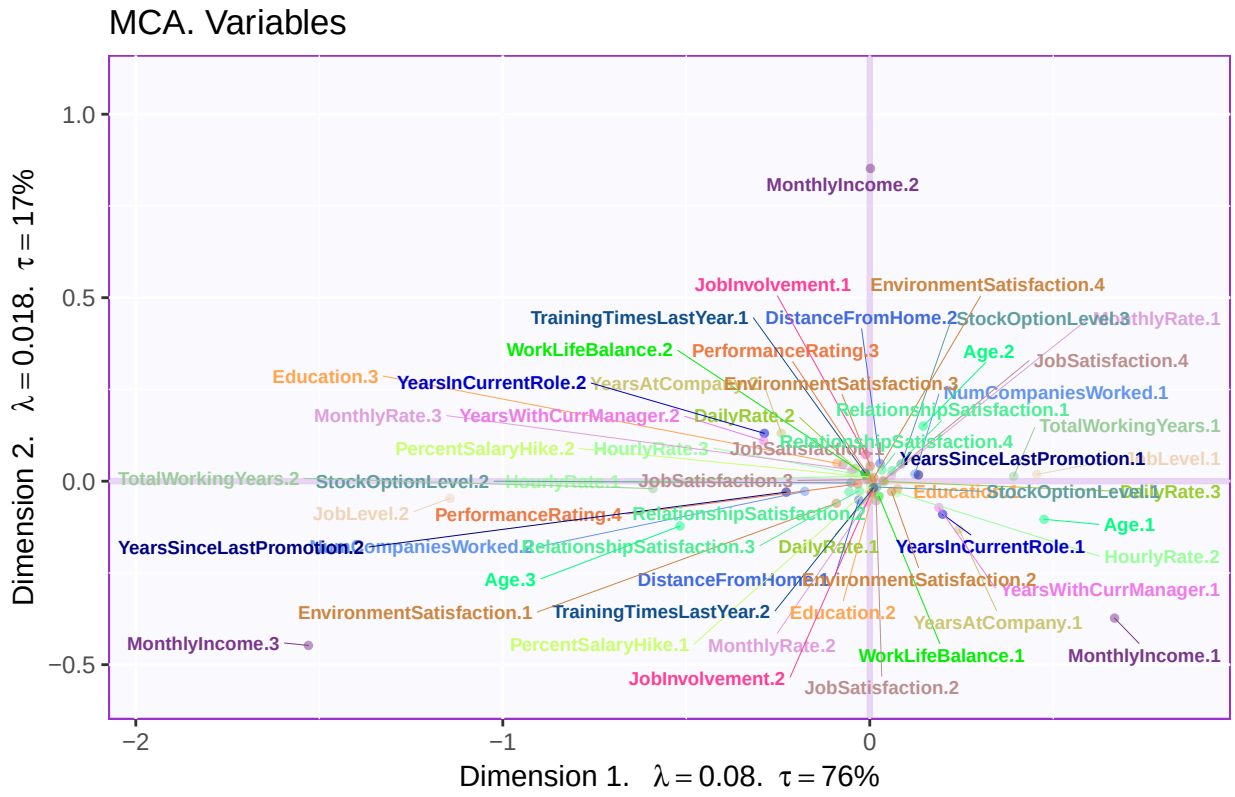



```
varCtr3 <- varCtr[,3]
names(varCtr3) <- rownames(varCtr)
a0006.Var.ctr3 <- PrettyBarPlot2(varCtr3,
main = 'Variable Contributions: Dimension 3', #ylim = c(-.03, 1.2*max(varCtr2)),
threshold = 1 / nrow(varCtr), color4bar = gplots::col2hex(color)
)
print(a0006.Var.ctr3)
```

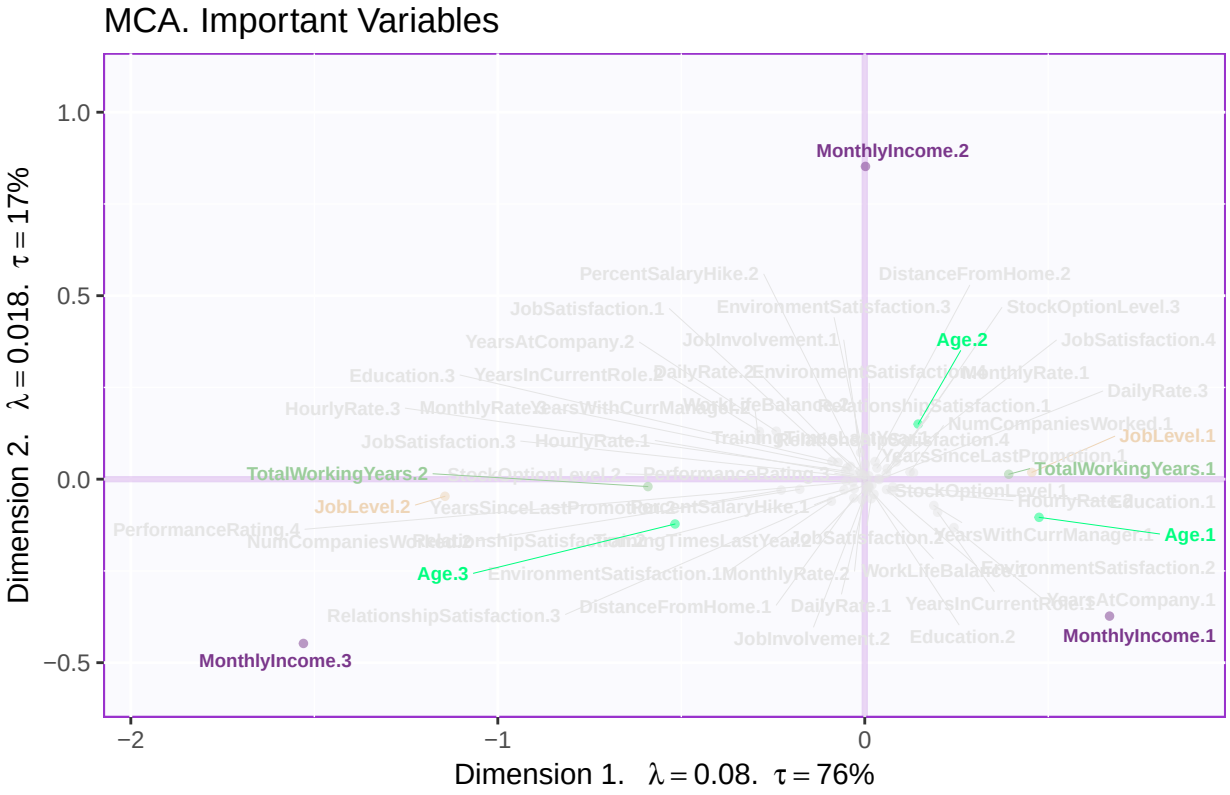


Variable Map

```
axis1 = 1
axis2 = 2
Fj <- resDICA$TExPosition.Data$fj
col4Levels <- data4PCCAR::coloringLevels( rownames(resDICA$TExPosition.Data$fj), color)
col4Labels <- col4Levels$color4Levels

BaseMap.Fj <- createFactorMap(X = Fj , # resMCA$ExPosition.Data$fj,
                              axis1 = axis1, axis2 = axis2,
                              title = 'MCA. Variables',
                              col.points = col4Labels, cex = 1,
                              col.labels = col4Labels, text.cex = 2.5,
                              force = 2)
labels4MCA <- createxyLabels.gen(x_axis = axis1, y_axis = axis2, lambda = resDICA$TExPosition.Data$eigs
tau = resDICA$TExPosition.Data$t)

b0002.BaseMap.Fj <- BaseMap.Fj$zeMap + labels4MCA
b0003.BaseMapNoDot.Fj <- BaseMap.Fj$zeMap_background +
BaseMap.Fj$zeMap_text + labels4MCA
print(b0002.BaseMap.Fj)
```

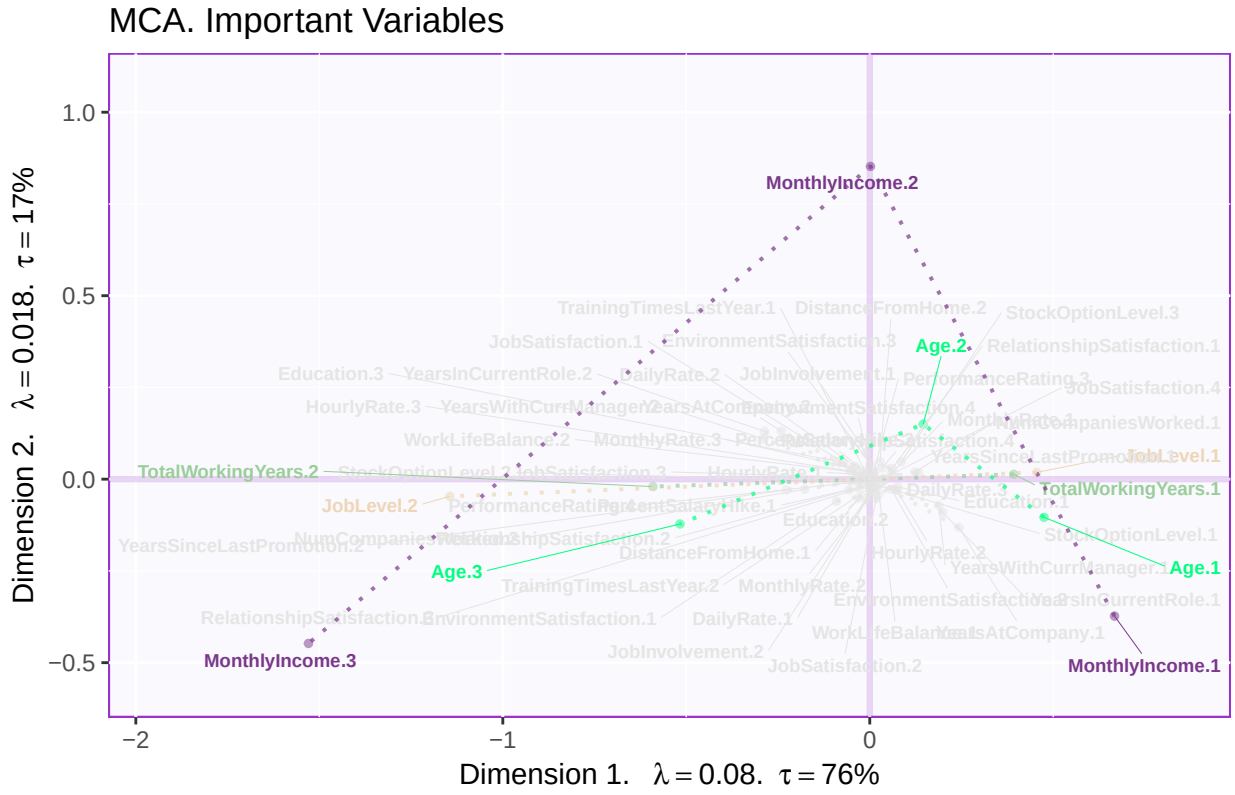



```
absCtrVar <- as.matrix(varCtr) %*% diag(resDICA$TExPosition.Data$eigs)
varCtr12 <- (absCtrVar[,1] + absCtrVar[,2]) /(resDICA$TExPosition.Data$eigs[1] +resDICA$TExPosition.Data
importantVar <- (varCtr12 >= 1 / length(varCtr12))
col4ImportantVar <- color
col4NS <- 'gray90'
col4ImportantVar[!importantVar] <- col4NS
col4Levels.imp <- data4PCCAR::coloringLevels(rownames(Fj), col4ImportantVar)
BaseMap.Fj.imp <- createFactorMap(X = Fj , # resMCA$ExPosition.Data$fj,
                                  axis1 = axis1, axis2 = axis2,
title = 'MCA. Important Variables', col.points = col4Levels.imp$color4Levels,
cex = 1,
col.labels = col4Levels.imp$color4Levels,
                               text.cex = 2.5,
force = 2)
b0010.BaseMap.Fj <- BaseMap.Fj.imp$zeMap + labels4MCA
print(b0010.BaseMap.Fj)
```



Map with important variables and lines

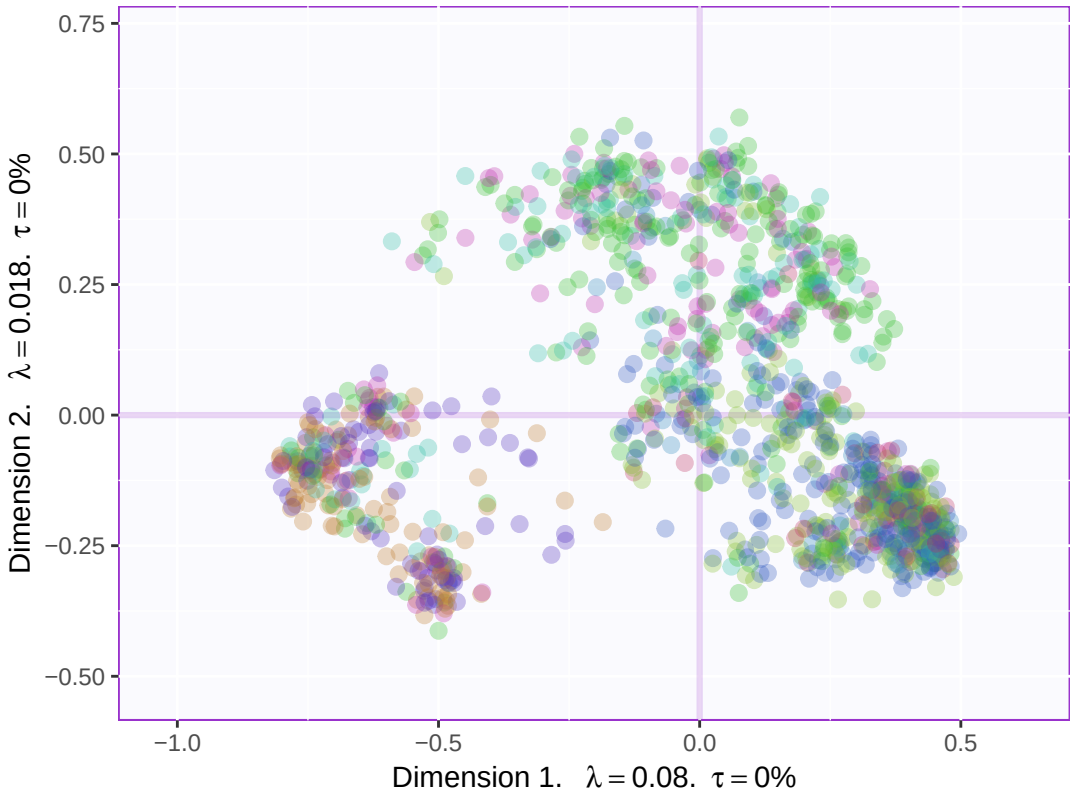
```
lines4J <- addLines4MCA(Fj, col4Var = col4Levels.imp$color4Variables, size = .7)
b0020.BaseMap.Fj <- b0010.BaseMap.Fj + lines4J
print( b0020.BaseMap.Fj)
```



8.3 Factor Map I

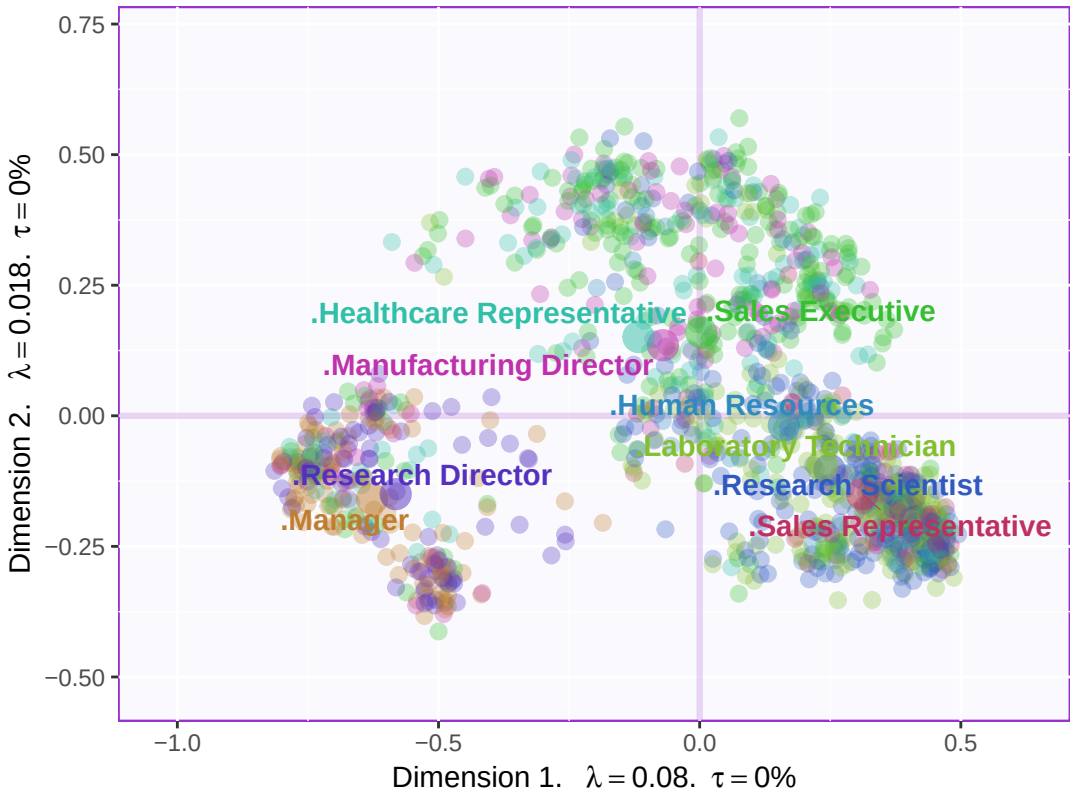
Job Role

```
baseMap.i1 <- PTCA4CATA::createFactorMap(resDICA$TExPosition.Data$fii,
                                         col.points   = resDICA$Plotting.Data$fii.col,
                                         alpha.points =  .3)
label4Map <- createxyLabels.gen(1,2,
                                lambda = resDICA$TExPosition.Data$eigs,
                                tau = resDICA$TExPosition.Data$eigs )

# Plain map with color for the I-set
aggMap.i1 <- baseMap.i1$zeMap_background + baseMap.i1$zeMap_dots + label4Map
#-----------------------------------------------------------------------
print(aggMap.i1)
```



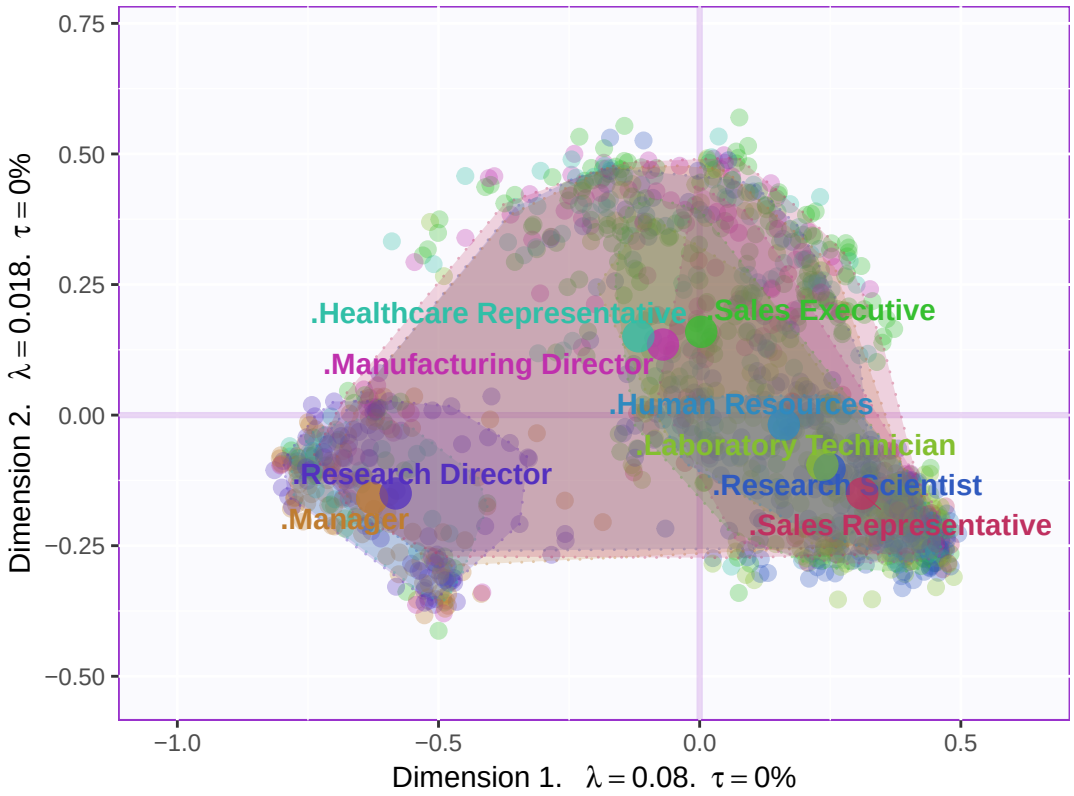

```
col4data1 <- resDICA$Plotting.Data$fii.col
col4Means1 <- unique(col4data1)

MapGroup1    <- PTCA4CATA::createFactorMap(resDICA$TExPosition.Data$fi,
                                  axis1 = 1, axis2 = 2,
                                  constraints = baseMap.i1$constraints,
                                  title = NULL,
                                  col.points = col4Means1,
                                  display.points = TRUE,
                                  pch = 19, cex = 5,
                                  display.labels = TRUE,
                                  col.labels = col4Means1,
                                  text.cex = 4,
                                  font.face = "bold",
                                  font.family = "sans",
                                  col.axes = "darkorchid",
                                  alpha.axes = 0.2,
                                  width.axes = 1.1,
                               col.background = adjustcolor("lavender",
                                                            alpha.f = 0.2),
                               force = 1, segment.size = 0)

aggMap.i.withMeans1 <- aggMap.i1+
  MapGroup1$zeMap_dots + MapGroup1$zeMap_text
#------------------------------------------------------------------------
```

```
print(aggMap.i.withMeans1)
```



Create 75% Tolerance interval polygons

```
GraphTI.Hull.901 <- MakeToleranceIntervals(resDICA$TExPosition.Data$fii,
                                   as.factor(datae),
                                   names.of.factors = c("Dim1","Dim2"),
                                   col = unique(col4data1),
                                   line.size = .5, line.type = 3,
                                   alpha.ellipse = .2,
                                   alpha.line = .4,
                                   p.level = .75, # 75% TI
                                   type = 'hull' #
                                           # use 'hull' for convex hull
)
aggMap.i.withHull1 <- aggMap.i1 +
  GraphTI.Hull.901 + MapGroup1$zeMap_dots +
  MapGroup1$zeMap_text +  MapGroup1$zeMap_dots

print(aggMap.i.withHull1)
```



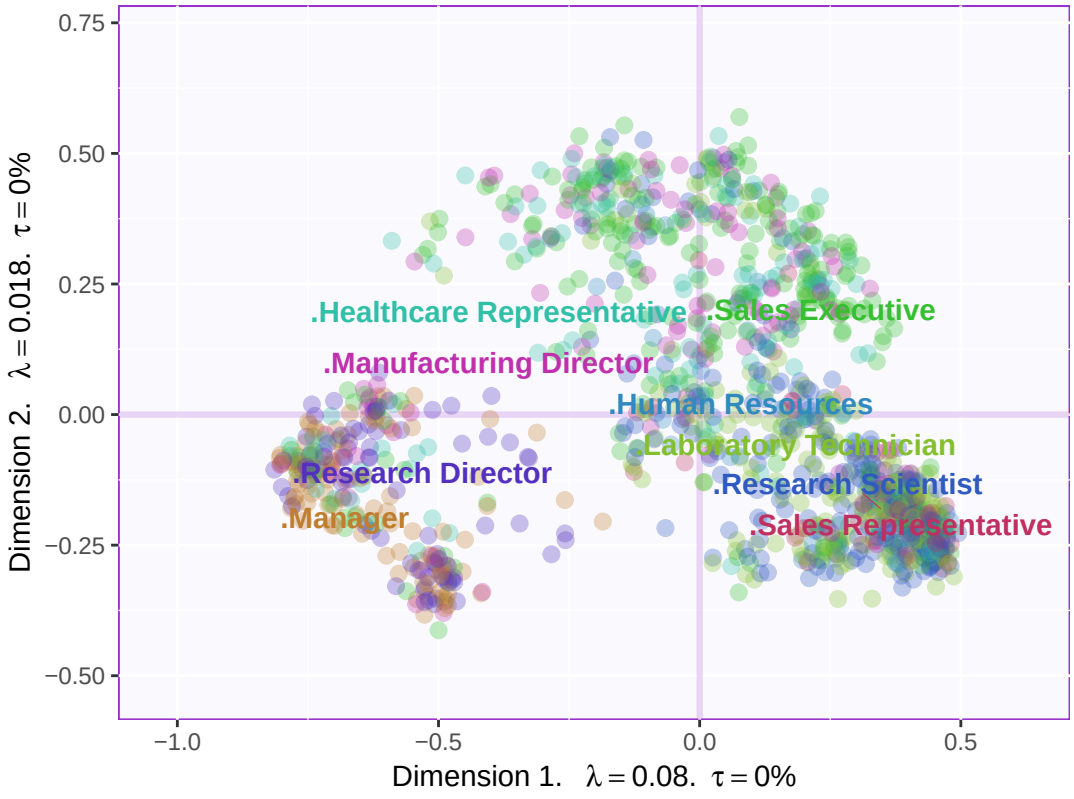
Inferences

```
fixedCM1   <-   resDICA.inf1$Inference.Data$loo.data$fixed.confuse
looedCM1   <- resDICA.inf1$Inference.Data$loo.data$loo.confuse
```

```
BootCube1 <- resDICA.inf1$Inference.Data$boot.data$fi.boot.data$boots
dimnames(BootCube1)[[2]] <- c("Dimension 1","Dimension 2")

GraphElli1 <- MakeCIEllipses(BootCube1[,1:2,],
                  names.of.factors = c("Dimension 1","Dimension 2"),
                  col = col4Means1,
                  p.level = .95
)


aggMap.i.withCI1 <-  aggMap.i1 +  GraphElli1 + MapGroup1$zeMap_text

print(aggMap.i.withCI1)
```

```
## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed
```

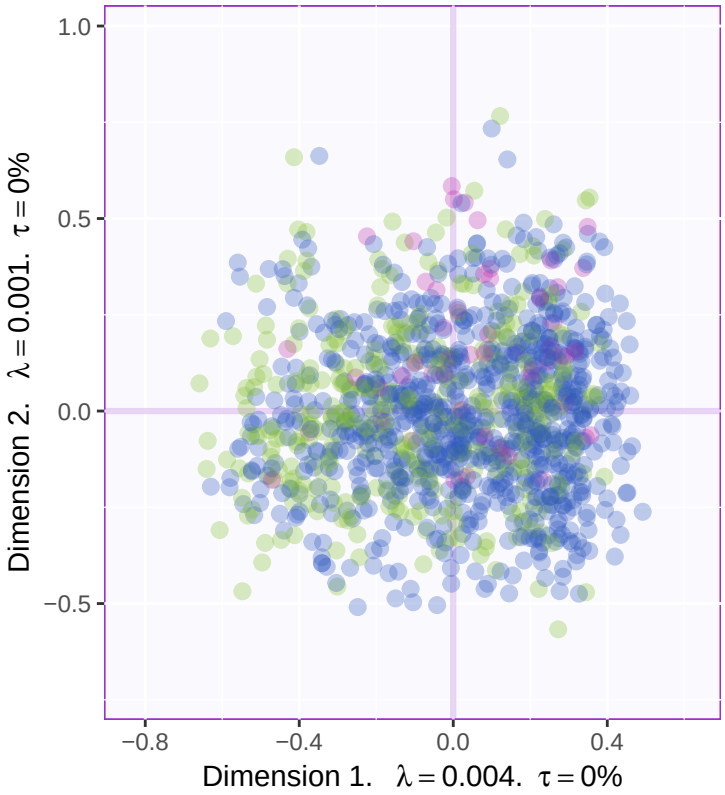
```
## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Removed 100 rows containing non-finite values (stat_ellipse).

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed
```



Department

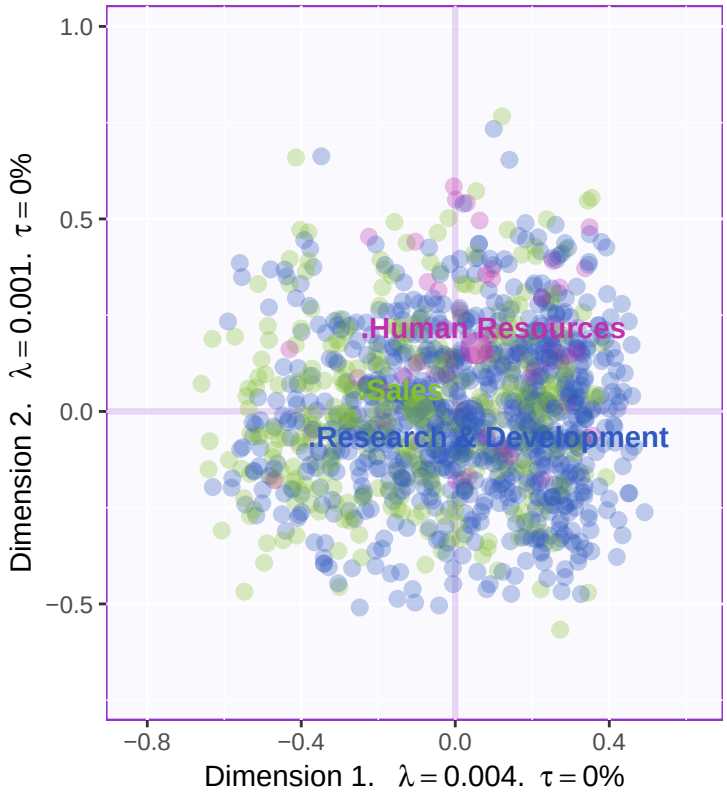
```
baseMap.i11 <- PTCA4CATA::createFactorMap(resDICA1$TExPosition.Data$fii,
                                          col.points    = resDICA1$Plotting.Data$fii.col,
                                          alpha.points =  .3)
label4Map1 <- createxyLabels.gen(1,2,
                                 lambda = resDICA1$TExPosition.Data$eigs,
                                 tau = resDICA1$TExPosition.Data$eigs )

# Plain map with color for the I-set
```

```
aggMap.i11 <- baseMap.i11$zeMap_background + baseMap.i11$zeMap_dots + label4Map1
#-------------------------------------------------------------------------
# print this Map
print(aggMap.i11)
```




```
col4data <- resDICA1$Plotting.Data$fii.col
col4Means <- unique(col4data)

MapGroup   <- PTCA4CATA::createFactorMap(resDICA1$TExPosition.Data$fi,
                                axis1 = 1, axis2 = 2,
                                constraints = baseMap.i1$constraints,
                                title = NULL,
                                col.points = col4Means,
                                display.points = TRUE,
                                pch = 19, cex = 5,
                                display.labels = TRUE,
                                col.labels = col4Means,
                                text.cex = 4,
                                font.face = "bold",
                                font.family = "sans",
                                col.axes = "darkorchid",
                                alpha.axes = 0.2,
                                width.axes = 1.1,
                            col.background = adjustcolor("lavender",
                                                         alpha.f = 0.2),
                            force = 1, segment.size = 0)
```

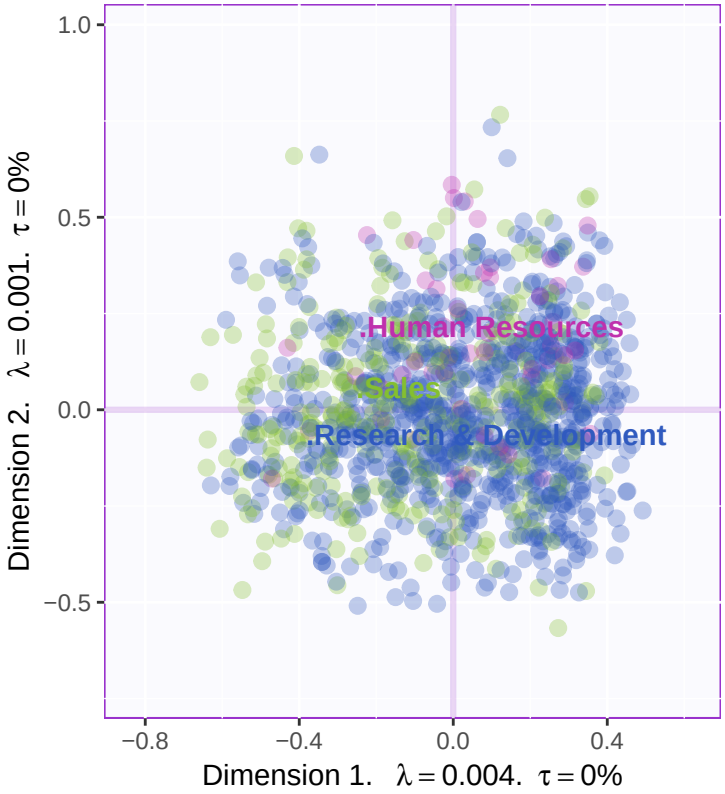
```
aggMap.i.withMeans <- aggMap.i11+
  MapGroup$zeMap_dots + MapGroup$zeMap_text
#-----------------------------------------------------------------------

print(aggMap.i.withMeans)
```




```
resDICA.inf <- tepDICA.inference.battery(DATA = data2,
                    DESIGN = datar,
                    make_design_nominal = TRUE,
                    make_data_nominal = TRUE,
                    group.masses = NULL,
                    weights = NULL,
                    graphs = FALSE,
                    k = 2,
                    test.iters = 100,
                    critical.value = 2)

BootCube <- resDICA.inf$Inference.Data$boot.data$fi.boot.data$boots
dimnames(BootCube)[[2]] <- c("Dimension 1","Dimension 2")

GraphElli <- MakeCIEllipses(BootCube1[,1:2,],
                   names.of.factors = c("Dimension 1","Dimension 2"),
                   col = col4Means,
                   p.level = .95
)
```

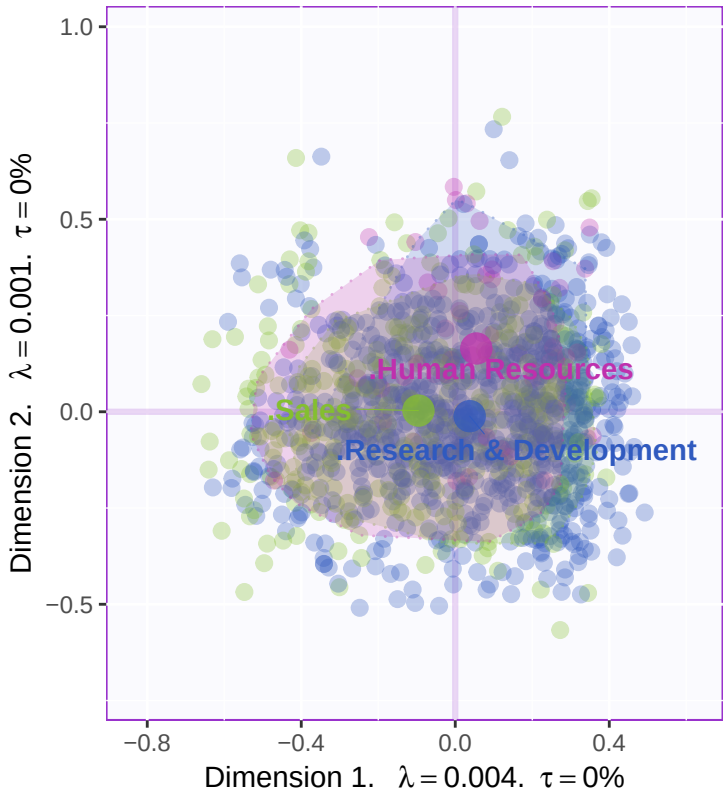
```
aggMap.i.withCI <-  aggMap.i11 +  GraphElli + MapGroup$zeMap_text

print(aggMap.i.withCI)
```

```
## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Removed 100 rows containing non-finite values (stat_ellipse).

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_ellipse()`:
## missing value where TRUE/FALSE needed
```



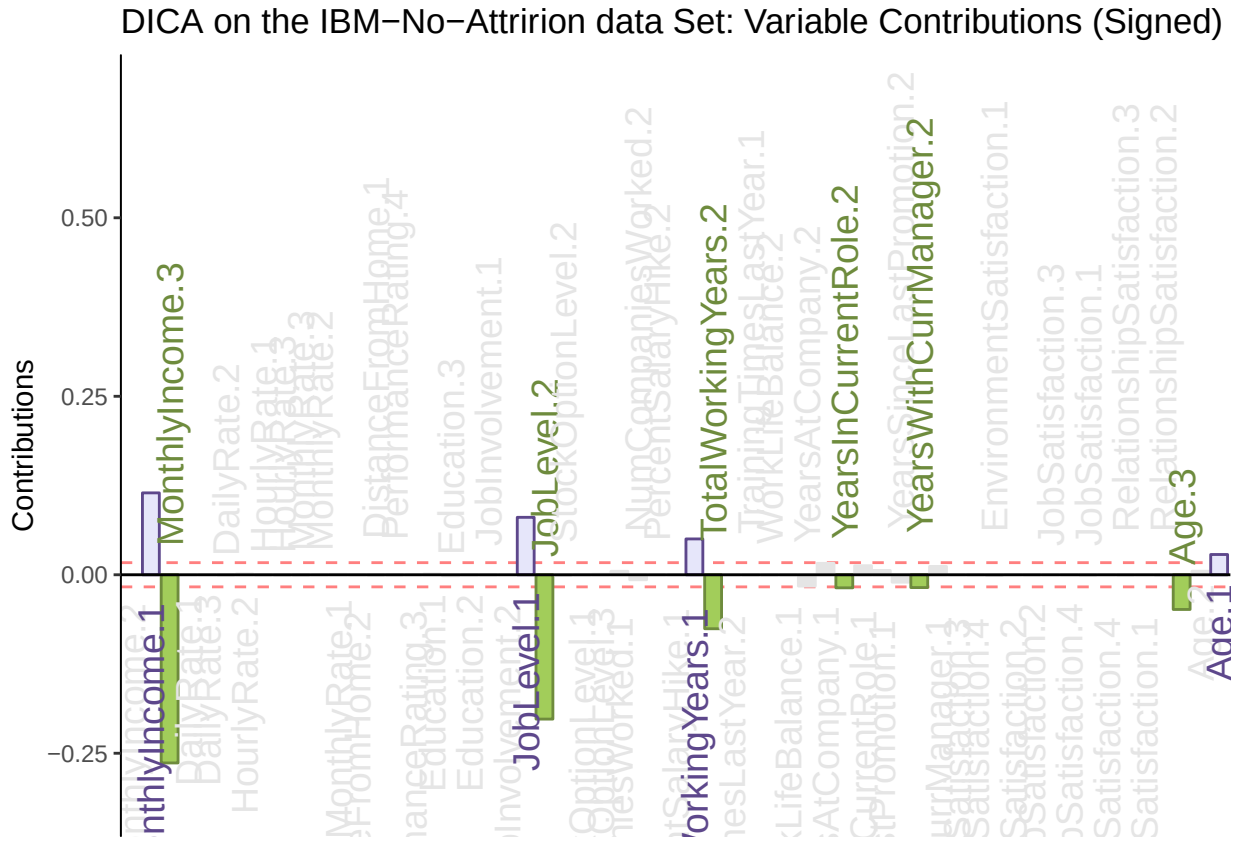
Create 75% Tolerance interval polygons

```
GraphTI.Hull <- MakeToleranceIntervals(resDICA1$TExPosition.Data$fii,
                                       as.factor(datar),
                                       names.of.factors = c("Dim1","Dim2"),
                                       col = unique(col4data),
                                       line.size = .5, line.type = 3,
                                       alpha.ellipse = .2,
                                       alpha.line = .4,
                                       p.level = .75, # 75% TI
                                       type = 'hull' #
                                              # use 'hull' for convex hull
)
aggMap.i.withHull <- aggMap.i11 +
  GraphTI.Hull + MapGroup$zeMap_dots +
  MapGroup$zeMap_text +  MapGroup$zeMap_dots

print(aggMap.i.withHull)
```

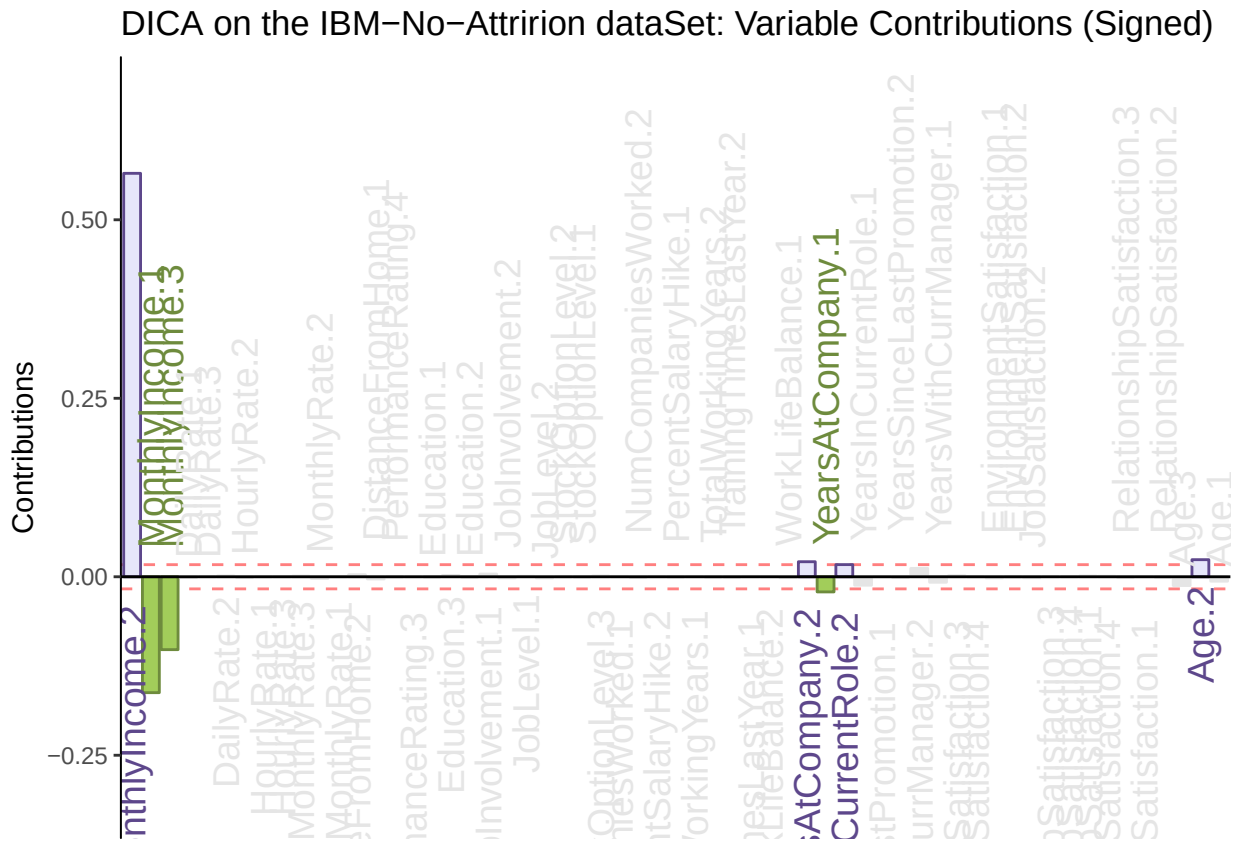


8.4 Contribution

```
signed.ctrJ <- resDICA$TExPosition.Data$cj * sign(resDICA$TExPosition.Data$fj)
b003.ctrJ.s.1 <- PrettyBarPlot2(signed.ctrJ[,1],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'DICA on the IBM-No-Attririon data Set: Variable Contributions (S
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b003.ctrJ.s.1)
```

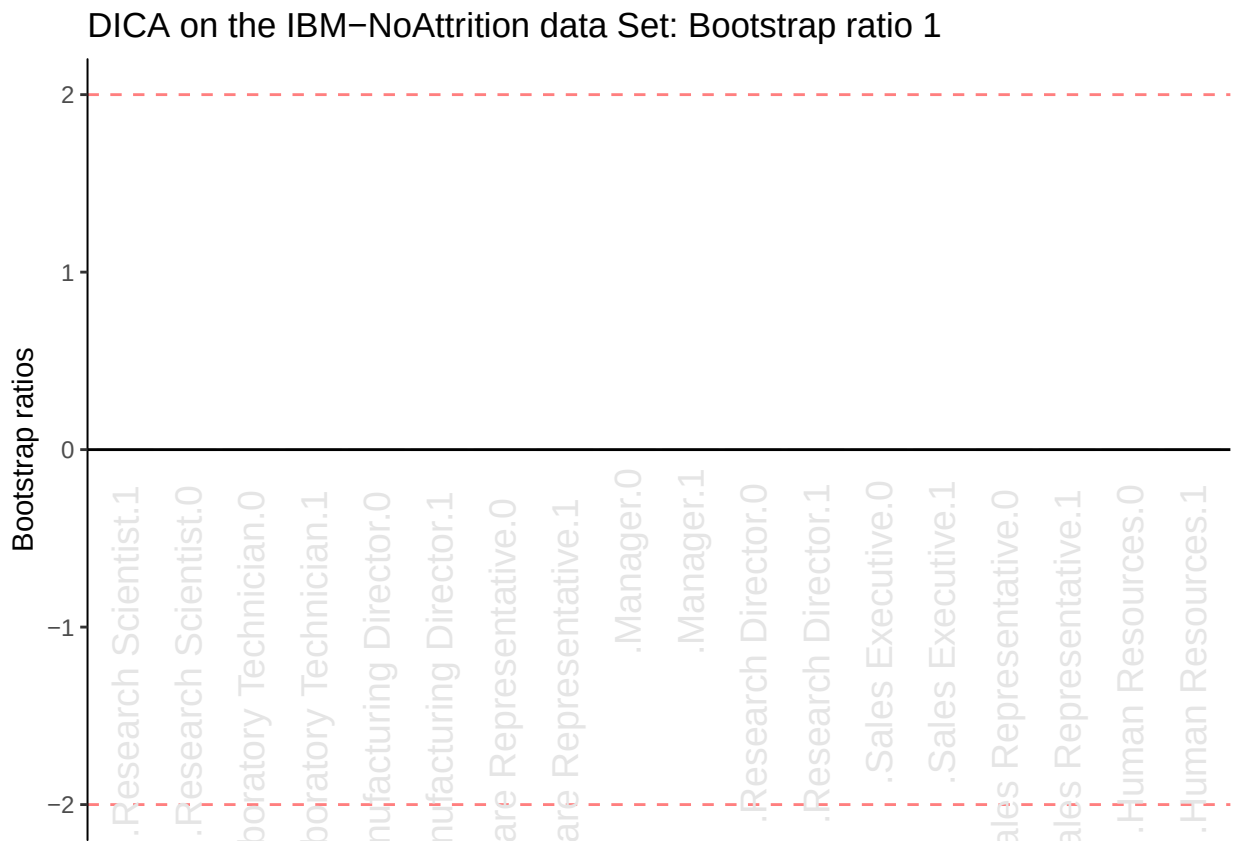



```
b004.ctrJ.s.2 <- PrettyBarPlot2(signed.ctrJ[,2],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                # color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'DICA on the IBM-No-Attririon dataSet: Variable Contributions (S
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b004.ctrJ.s.2)
```

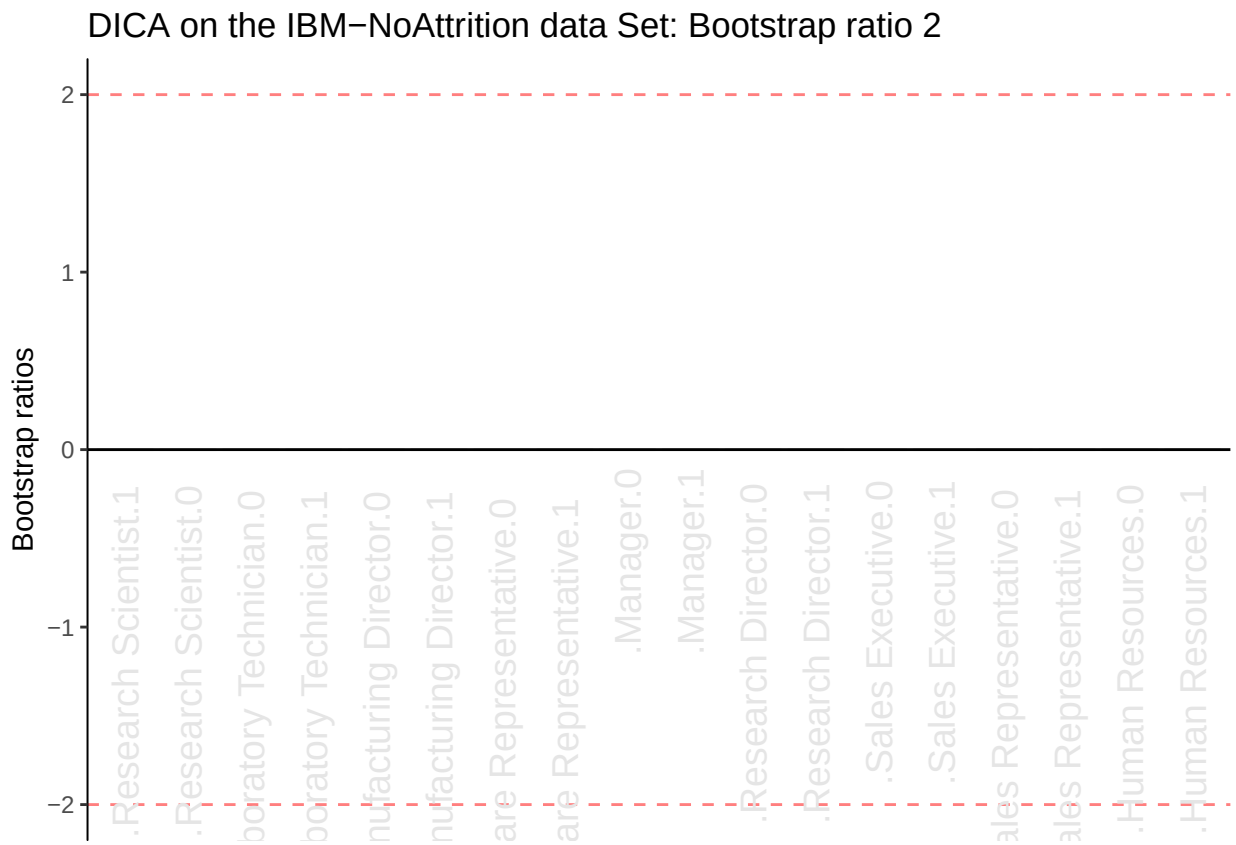


8.5 Bootstrap Ratios

```
BR <- resDICA.inf1$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
laDim = 1
ba001.BR1 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                            #color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0( 'DICA on the IBM-NoAttrition data Set: Bootstrap ratio ',laD
                            ylab = 'Bootstrap ratios'
                            #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR1)
```




```
#
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                            #color4bar = gplots::col2hex(col4J.ibm),
                            main = paste0(
                              'DICA on the IBM-NoAttrition data Set: Bootstrap ratio ',laDim),
                            ylab = 'Bootstrap ratios'
)
print(ba002.BR2)
```



8.6 Summary

Component 1: - We can say that the Research Director, Manager, Research Scientists have high Income and higher experience in comparison to other Job Roles. Also, we can say that the Sales People have least Income and less number of experience.

Component 2: - We can also say that component 2 divides HR vs Sales and Research & Development telling that the HR people are generally older with higher income.

```
options(knitr.duplicate.label = 'allow')
```

9 DiSTATIS

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

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

DataSet

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

```
for (i in 1:length(Design_Data)) {
  Design_Data[i] <- if (Design_Data[i] <= 1) 1 else if (Design_Data[i] <= 4 & Design_Data[i] >1) 2 else
}
```

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

```
color4mus <- c(1, 1, 2 ,2, 1 ,1 ,1 ,1, 1, 1 ,1, 1, 3, 2 ,3, 2, 3 ,1, 3 ,1, 2 ,2 ,3 ,2 ,2 ,1 ,1 ,1, 2 ,1
Judges <- paste0(Design_Data,1:length(Design_Data))
```

Create the set of distance matrices

```
DistanceCube <- DistatisR::DistanceFromSort(Sorting_Data)
```

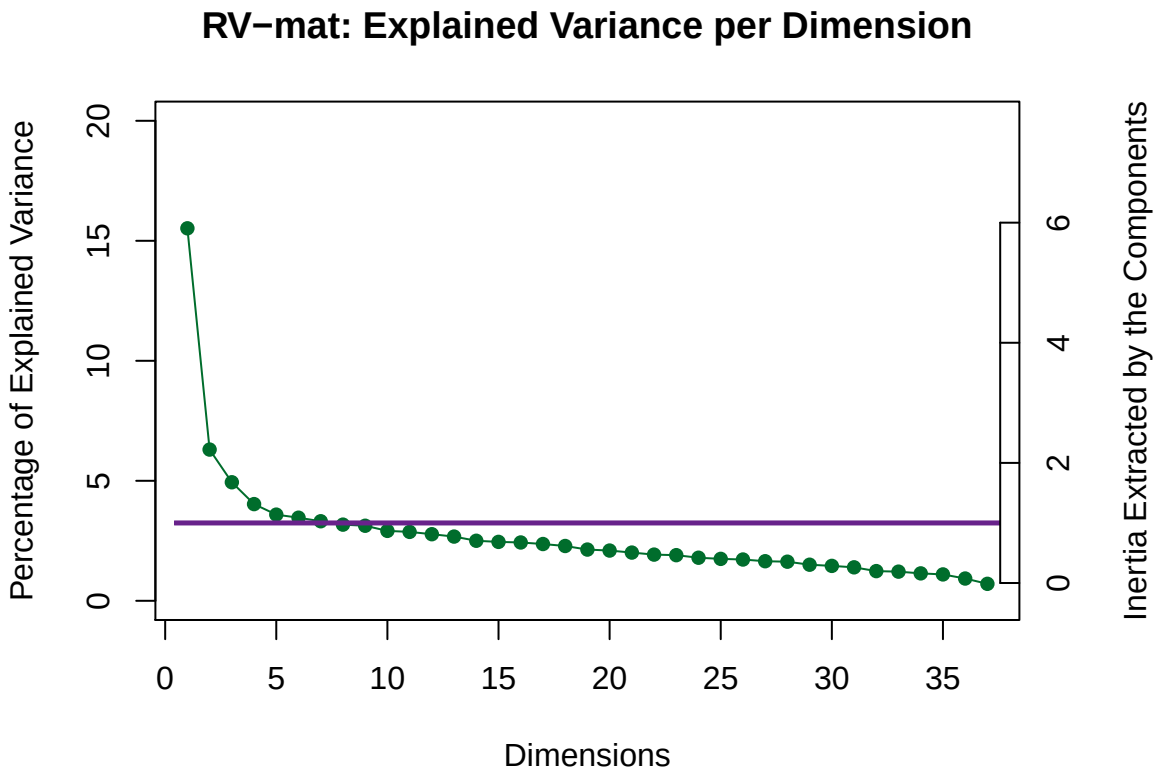
```
testDistatis <- DistatisR::distatis(DistanceCube)
```

```
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
```

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

ScreePlot

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



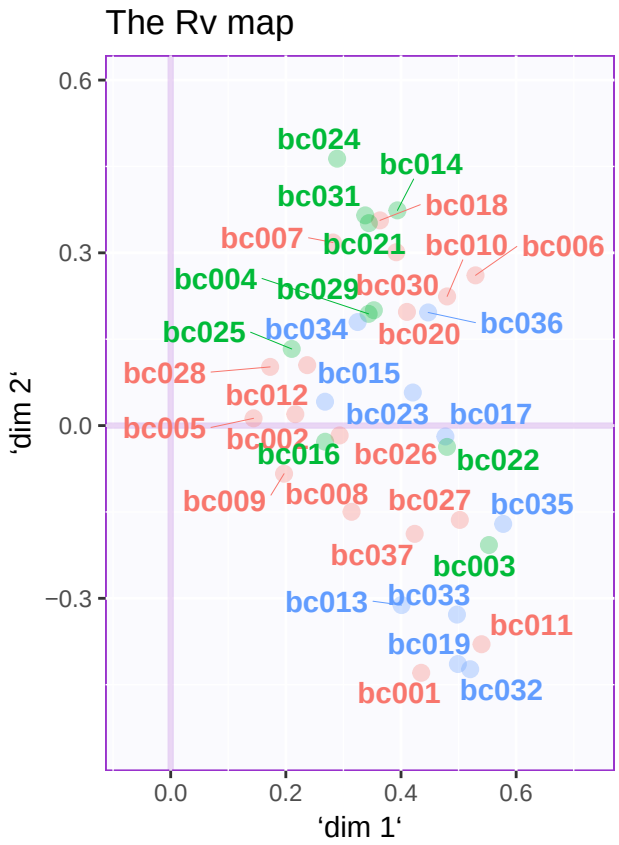
9.1 The Assessor Matrix

```
# Plot the assessor matrix
G <- testDistatis$res4Cmat$G
# Create a color scheme for the Composers
col4Non_musicions <- "#F8766D"
col4Moderate_musicions <- "#00BA38"
col4_musicions <- "#619CFF"
Judges[color4mus == 1 ] <- rep(col4Non_musicions,length(Judges))
```

```
## Warning in Judges[color4mus == 1] <- rep(col4Non_musicions,
## length(Judges)): number of items to replace is not a multiple of
## replacement length
```

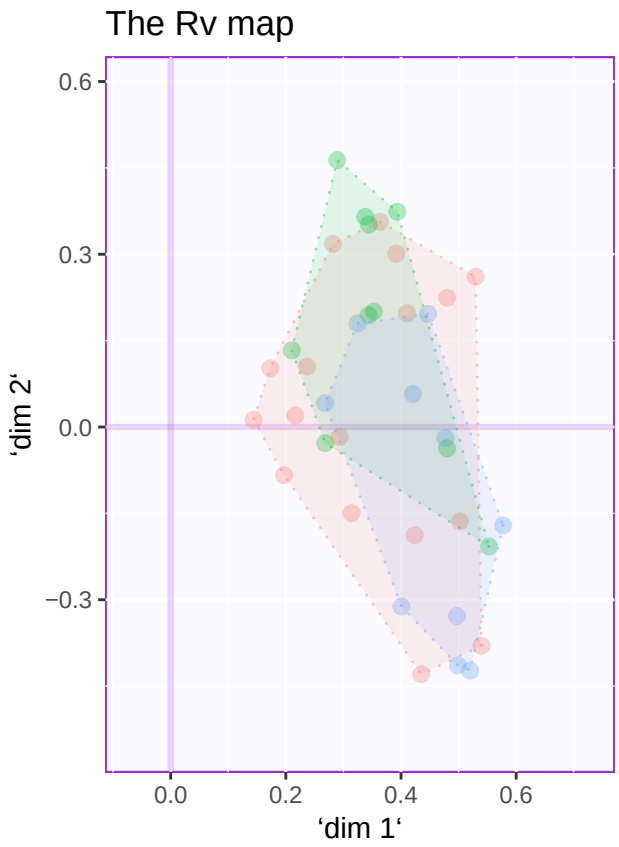
```
Judges[color4mus == 2 ] <- col4Moderate_musicions
Judges[color4mus == 3 ] <- col4_musicions
#------------------------------------------------------------------------
#------------------------------------------------------------------------
# A graph for the Judges set
baseMap.j <- PTCA4CATA::createFactorMap(G,
                                        title = 'The Rv map',
                                        col.points   = Judges,
                                        alpha.points =  .3,
                                        col.labels = Judges)
# A graph for the J-set
```

```
aggMap.j <- baseMap.j$zeMap_background + # background layer
            baseMap.j$zeMap_dots + baseMap.j$zeMap_text # dots & labels
# We print this Map with the following code
print(aggMap.j)
```



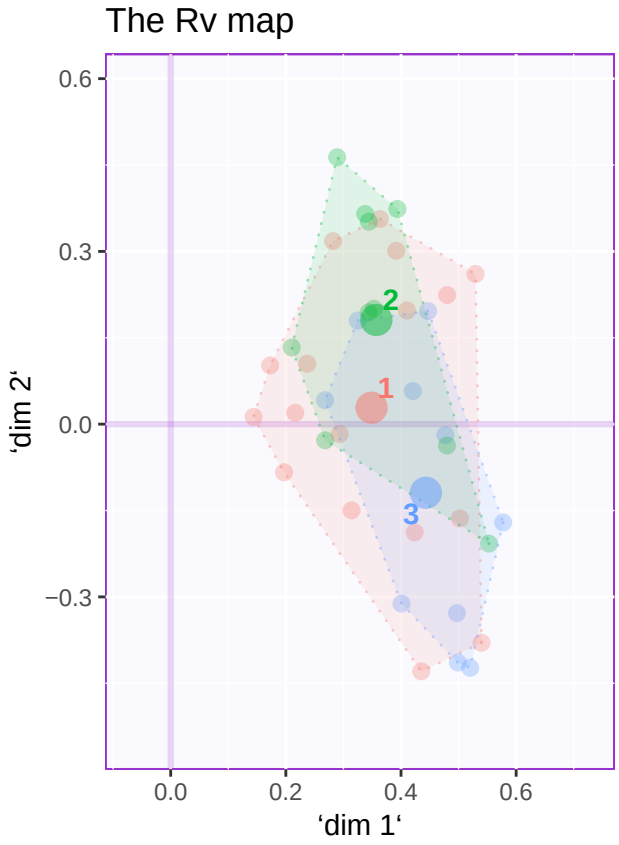

```
# Create 100% Tolerance interval polygons
#
GraphTJ.Hull.100 <- MakeToleranceIntervals(G,
                                    as.factor(Design_Data),
                                    names.of.factors = c("Dim1","Dim2"),
                                    col = unique(Judges),
               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

```
print(aggMap.j.withHull)
```

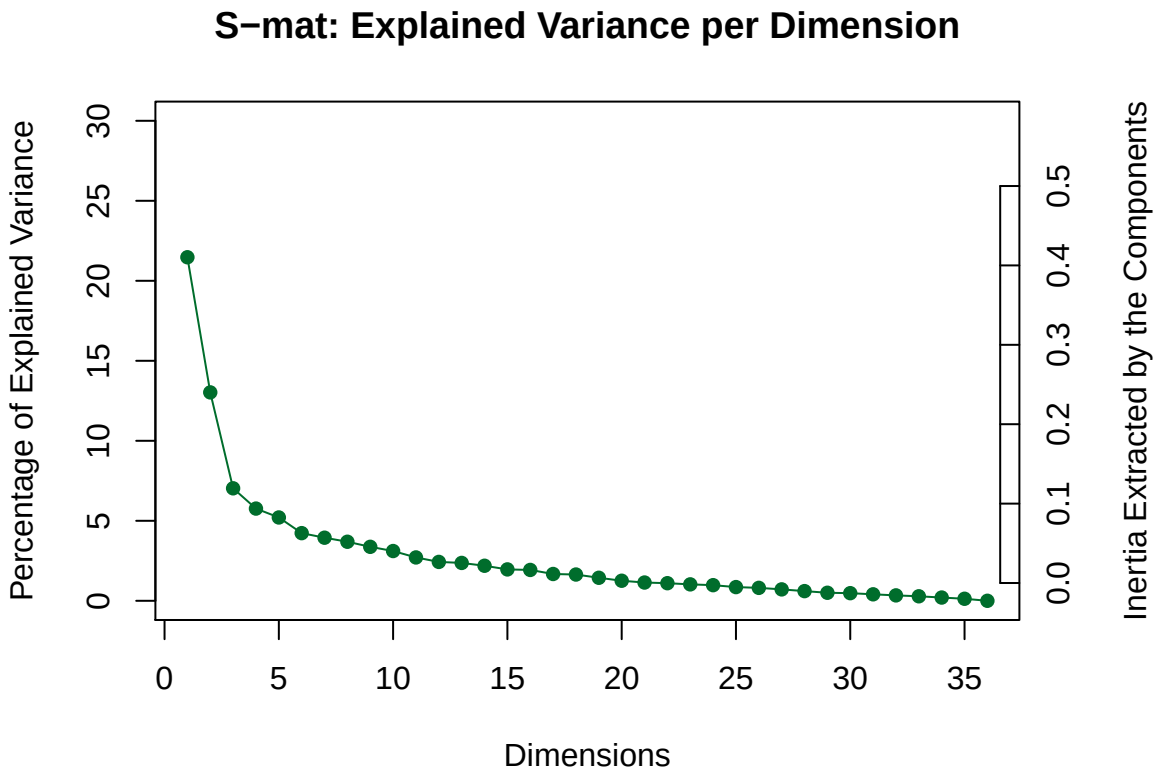



```
JudgesMeans.tmp <- aggregate(G, list(as.numeric(Design_Data)), mean) # compute the means
JudgesMeans <- JudgesMeans.tmp[,2:ncol(JudgesMeans.tmp )] # drop var 1
rownames(JudgesMeans) <- JudgesMeans.tmp[,1] # use var 1 to name the groups
#-------------------------------------------------------------------------
# a vector of color for the means
col4Means <- unique(Judges)
#-------------------------------------------------------------------------
# create the map for the means
MapGroup    <- PTCA4CATA::createFactorMap(JudgesMeans,
                                  axis1 = 1, axis2 = 2,
                                  constraints = baseMap.j$constraints,
                                          title = NULL,
                                          col.points = col4Means,
                                          display.points = TRUE,
                                          pch = 19, cex = 5,
                                          display.labels = TRUE,
                                          col.labels = col4Means,
                                          text.cex = 4,
                                          font.face = "bold",
                                          font.family = "sans",
                                          col.axes = "darkorchid",
                                          alpha.axes = 0.2,
                                          width.axes = 1.1,
                             col.background = adjustcolor("lavender",
```

```
                                              alpha.f = 0.2),
                                            force = 1, segment.size = 0)
# The map with observations and group means
aggMap.j.withMeans <- aggMap.j.withHull +
                      MapGroup$zeMap_dots + MapGroup$zeMap_text
#-----------------------------------------------------------------------
# plot it!
print(aggMap.j.withMeans)
```




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

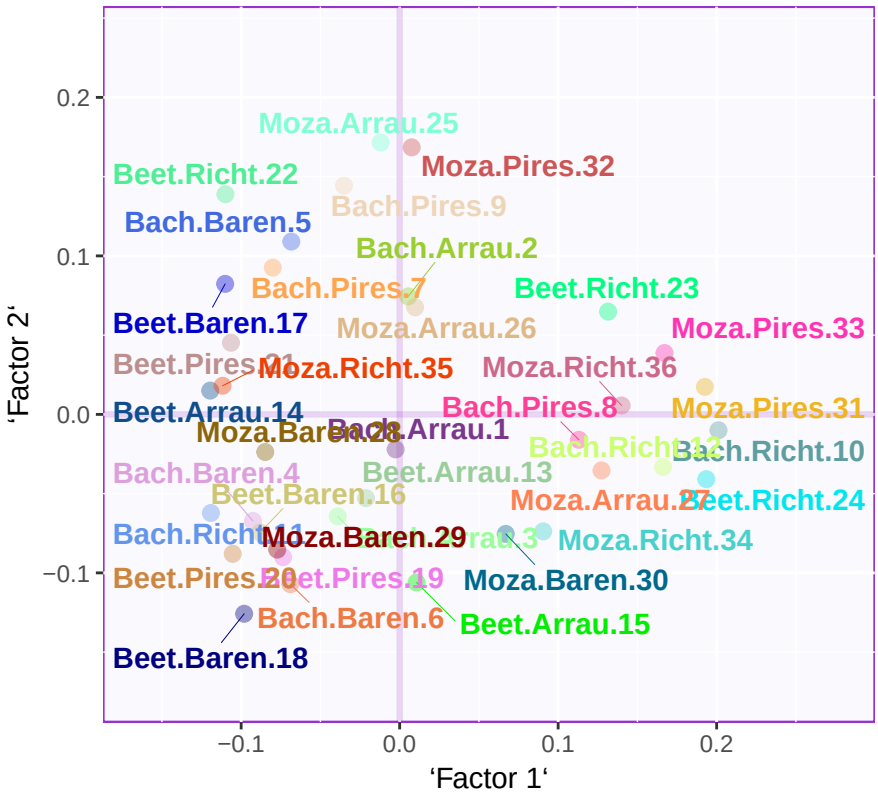



```
zeScree.S <- recordPlot()
```

9.2 I-Map

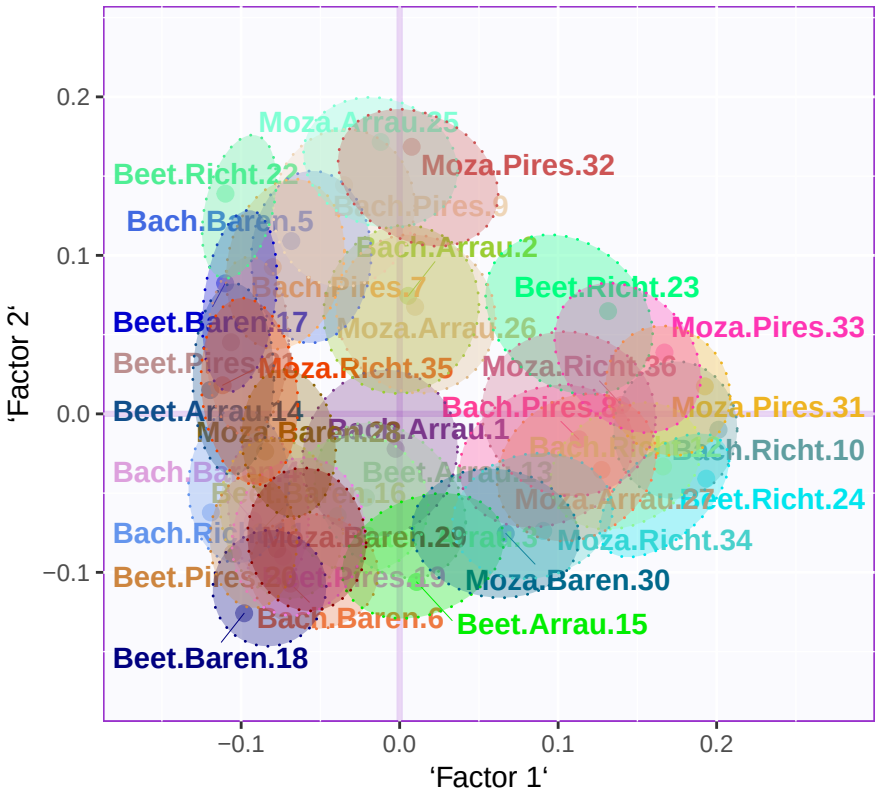
```
Fi <- testDistatis$res4Splus$F
col4Beers <- prettyGraphsColorSelection(nrow(Fi))
#  Use colors from prettyGraphs
#------------------------------------------------------------------------
# Graphs for the I set
#------------------------------------------------------------------------
# Create the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

```
# print this Map
print(aggMap.i)
```




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

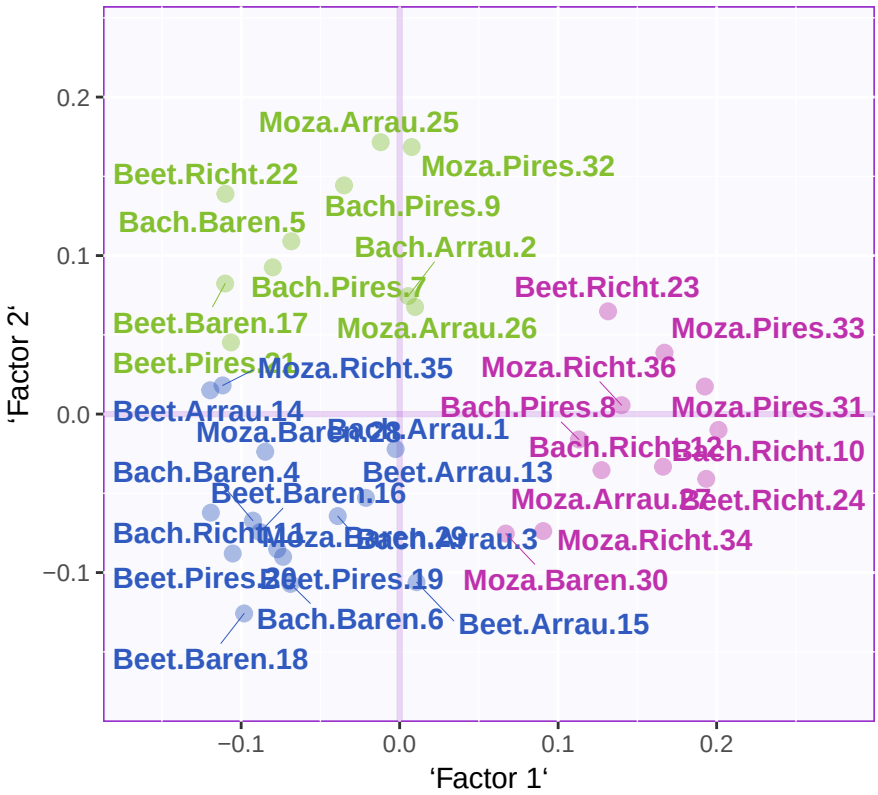
```
## Warning: Removed 3 rows containing missing values (geom_text_repel).
```




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

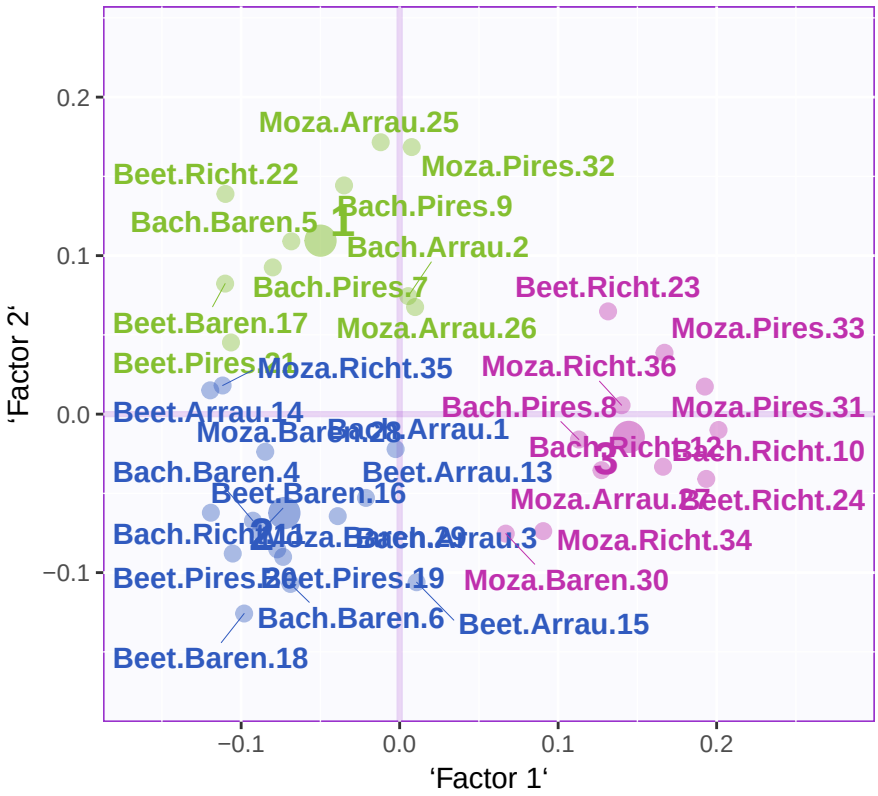
```
# Some classification now
# First plain k-means
set.seed(42)
beers.kMeans <- kmeans(x = Fi , centers = 3)
#------------------------------------------------------------------------
# Now to get a map by cluster:
col4Clusters  <- createColorVectorsByDesign(
               makeNominalData(
               as.data.frame(beers.kMeans$cluster)  ))
```

```
#=========================================================================
#-------------------------------------------------------------------------
# Graphs for the I set
#-------------------------------------------------------------------------
# Create the base map
# constraints4Fi <- lapply(minmaxHelper(Fi),'*',1.2)
baseMap.i.km <- PTCA4CATA::createFactorMap(Fi,
                                           col.points = col4Clusters$oc,
                                           col.labels = col4Clusters$oc,
                                           constraints = constraints4Fi,
                                           alpha.points =  .4)
#-------------------------------------------------------------------------
# We are  interested about the labels here
# so we will use dots and labels
#-------------------------------------------------------------------------
# Plain map with color for the I-set
aggMap.i.km <- baseMap.i.km$zeMap_background +
  baseMap.i.km$zeMap_dots + baseMap.i.km$zeMap_text
# print
print(aggMap.i.km)
```



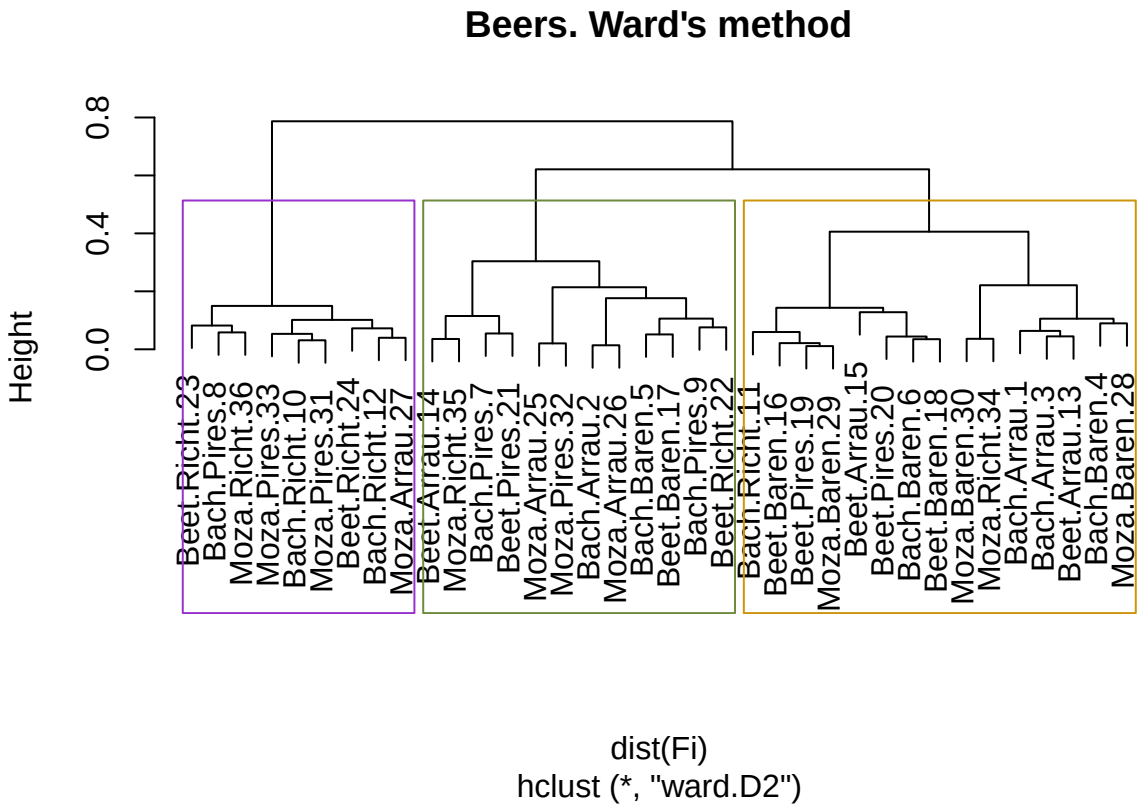

```
# get the color order in the c=good order
col4C <- col4Clusters$gc[sort(rownames(col4Clusters$gc),
                              index.return = TRUE)$ix]
# create the map for the means
map4Clusters    <- PTCA4CATA::createFactorMap(beers.kMeans$centers,
```

```
                                        axis1 = 1, axis2 = 2,
                                        constraints = constraints4Fi,
                                        title = NULL,
                                        col.points = col4C,
                                        display.points = TRUE,
                                        pch = 19, cex = 5,
                                        display.labels = TRUE,
                                        col.labels = col4C,
                                        text.cex = 6,
                                        font.face = "bold",
                                        font.family = "sans",
                                        col.axes = "darkorchid",
                                        alpha.axes = 0.2,
                                        width.axes = 1.1,
                                        col.background =
                              adjustcolor("lavender", alpha.f = 0.2),
                                        force = 1, segment.size = 0)
# The map with observations and group means
aggMap.i.withCenters <- aggMap.i.km  +
               map4Clusters$zeMap_dots + map4Clusters$zeMap_text
#
print(aggMap.i.withCenters)
```



9.3 Cluster Analysis

```
# A cluster analysis
beer.hc <- hclust(d = dist(Fi),
                  method = 'ward.D2' )

plot.tree <- plot(beer.hc,  main = "Beers. Ward's method")
hc.3.cl <- rect.hclust(beer.hc, k = 3,
               border = c('darkorchid',
                          'darkolivegreen4','darkgoldenrod3')
                         )
```




```
library(useful)
```

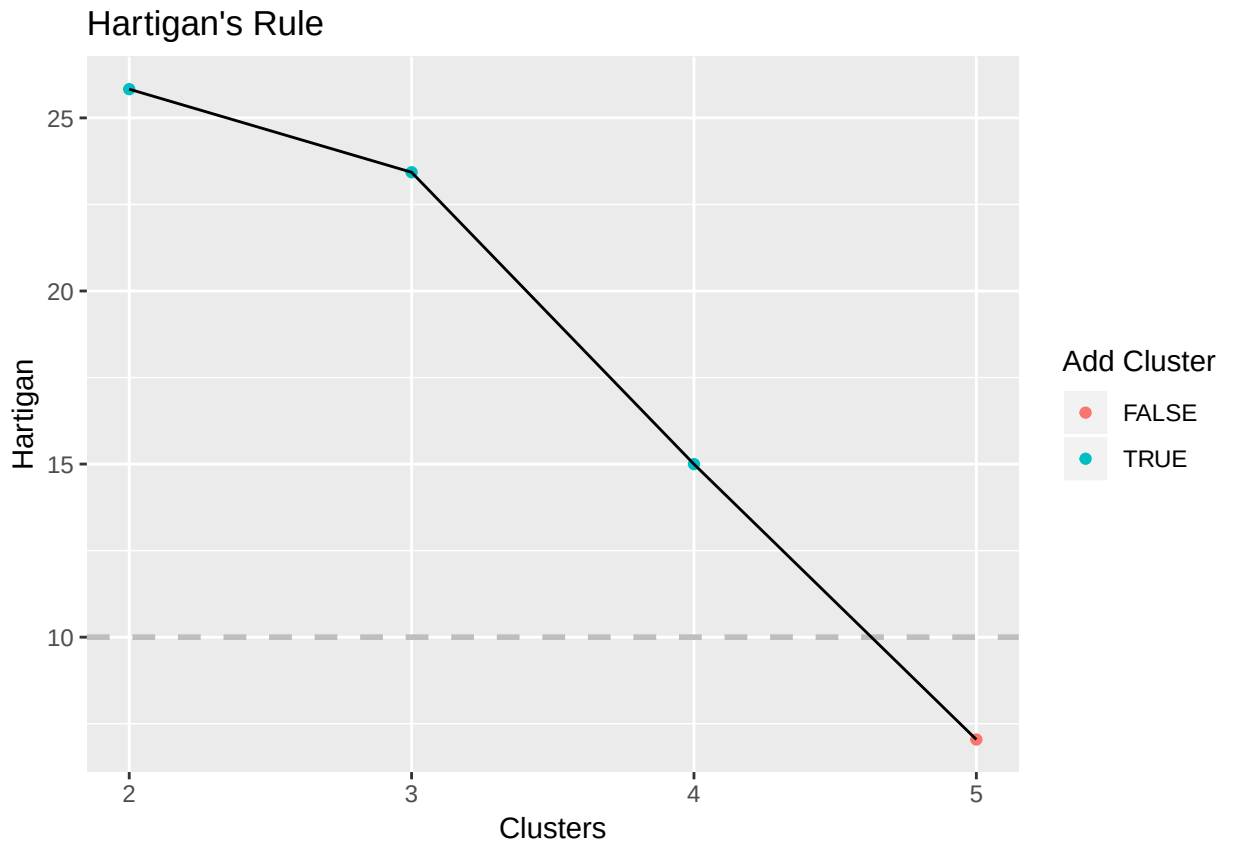
```
## Loading required package: ggplot2
```

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

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

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

```
print(plot.harti)
```



9.4 Summary

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

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

10 MFA

```
data <- read.csv("IBM-HR-Emplyee-NoAttrition.csv")
data1 <- data[,c(11:29)]
design <- data$JobRole
```

```
a <- as.matrix(c("1","1","1","1","1","2","2","2","2","2","2","2","3","3","3","3","3","3","3"))
a <- t(a)
colnames(a) <- colnames(data1)
resMFA <- mpMFA(data1 ,column.design = a,graphs = FALSE,DESIGN = data$JobRole)
```

```
## [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"
```

```
## Warning in sqrt(eigOut$values): NaNs produced
```

```
## [1] "Processing Complete"
```

```
resMFA1 <- mpMFA(data1 ,column.design = a,graphs = FALSE,DESIGN = data$Gender)
```

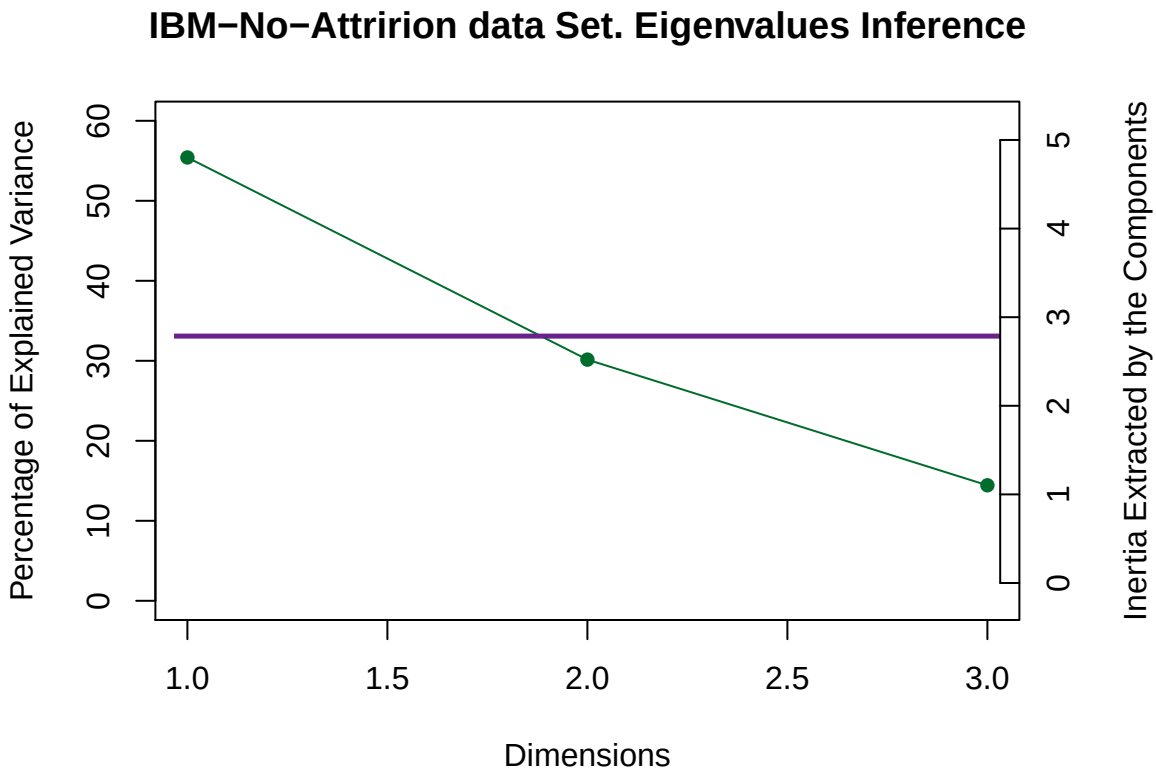
```
## [1] "Preprocessed the Rows of the data matrix using:  None"
## [1] "Preprocessed the Columns of the data matrix using:  Center_1Norm"
## [1] "Preprocessed the Tables of the data matrix using:  MFA_Normalization"
## [1] "Preprocessing Completed"
## [1] "Optimizing using:  None"
```

```
## Warning in sqrt(eigOut$values): NaNs produced
```

```
## [1] "Processing Complete"
```

10.1 PlotScree

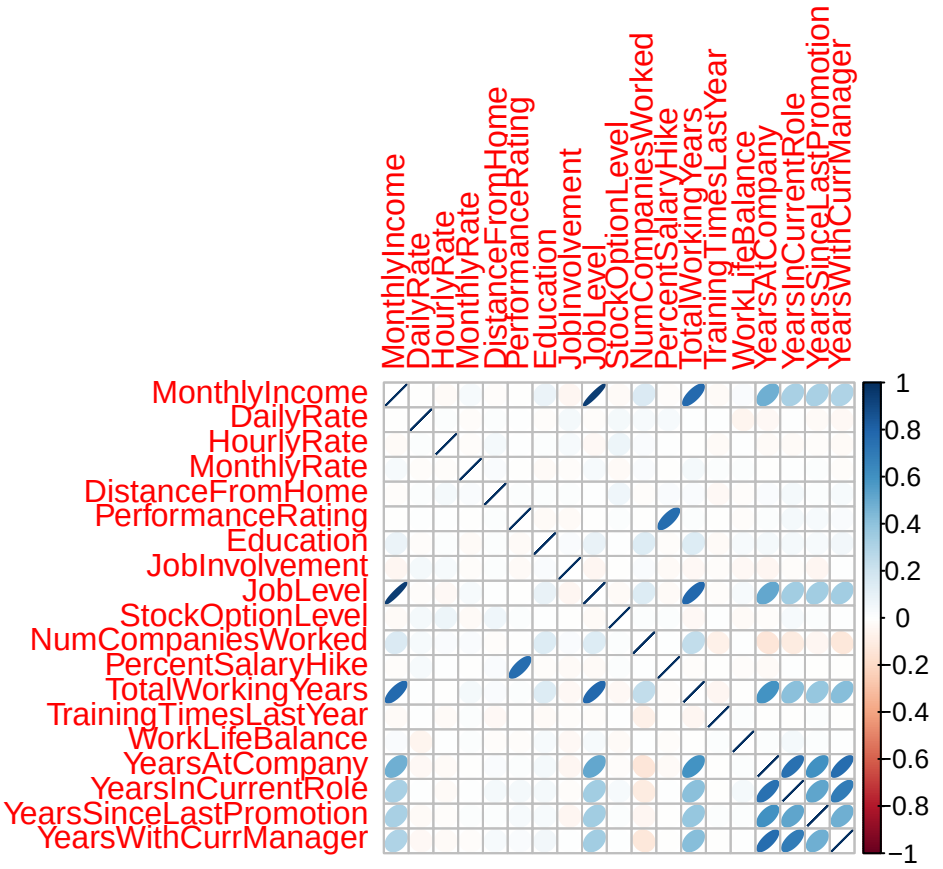
```
PlotScree(ev = resMFA$mexPosition.Data$InnerProduct$eigs,
          p.ev = NULL,
          title = 'IBM-No-Attririon data Set. Eigenvalues Inference',
          plotKaiser = TRUE
)
```




```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

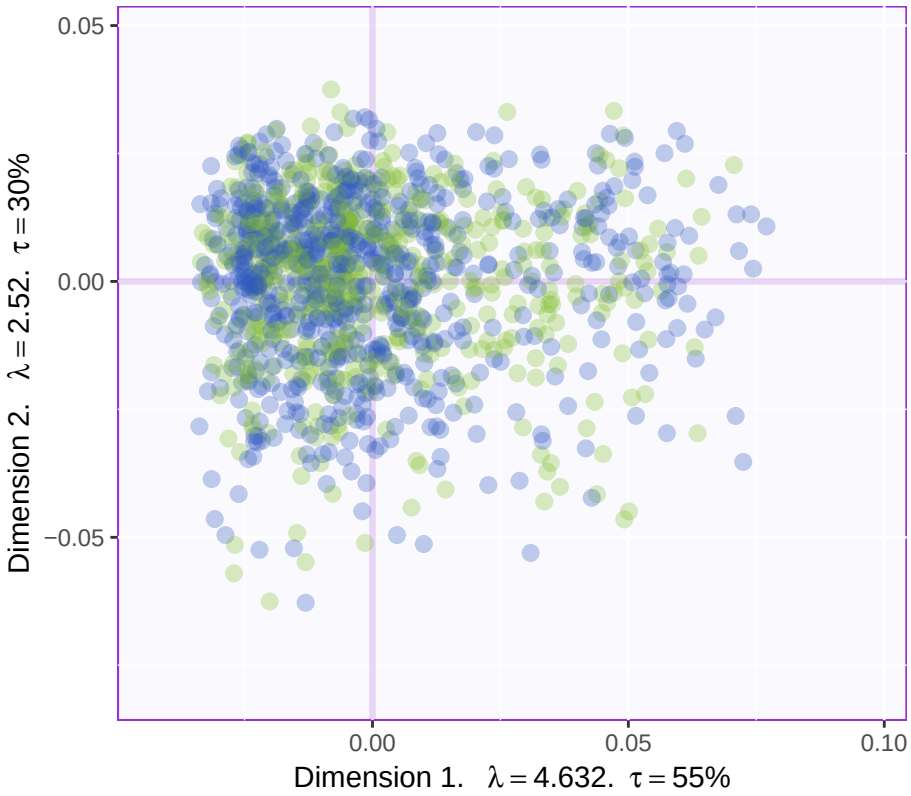
```
cor.my_data <- cor(data1)
corrplot(cor.my_data, method = "ellipse")
```



10.2 Factor Scores

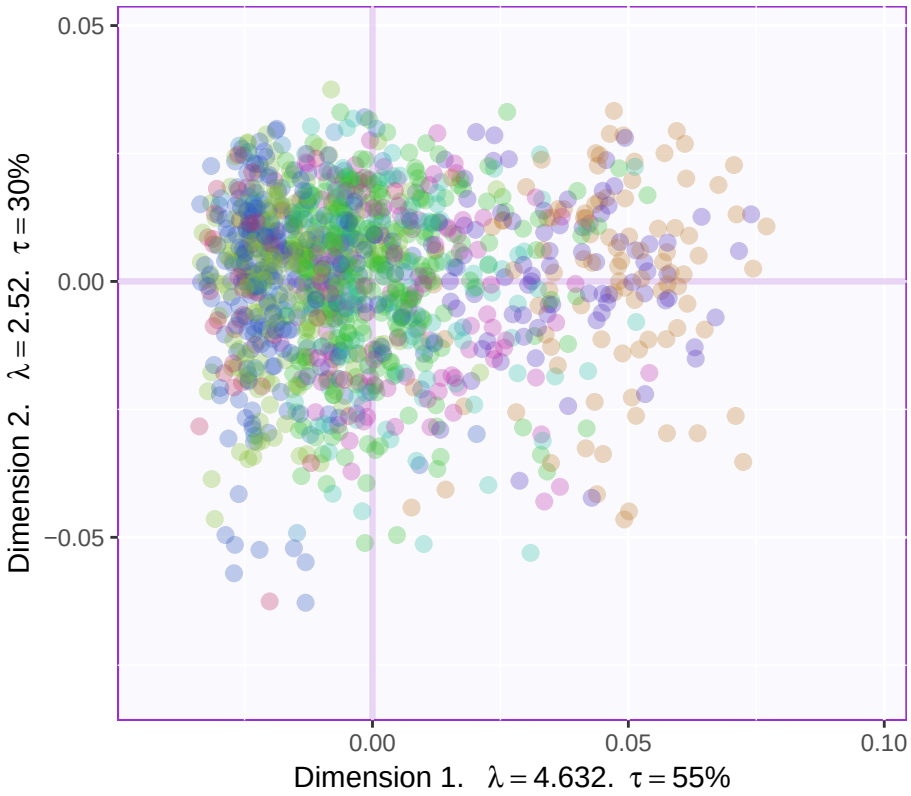
Gender

```
baseMap.j0 <- PTCA4CATA::createFactorMap(resMFA1$mexPosition.Data$Table$fi,col.points = resMFA1$Plottin
                                         col.labels = resMFA$Plotting.Data$fi.col,
                                         alpha.points =  .3,display.labels = FALSE)

label4Map0 <- createxyLabels.gen(1,2,
                                 lambda = resMFA1$mexPosition.Data$InnerProduct$eigs,
                                 tau = resMFA1$mexPosition.Data$InnerProduct$t )
# A graph for the J-set
b000.aggMap.j0 <- baseMap.j0$zeMap_background + # background layer
  baseMap.j0$zeMap_dots + baseMap.j0$zeMap_text + label4Map0
# We print this Map with the following code
print(b000.aggMap.j0)
```

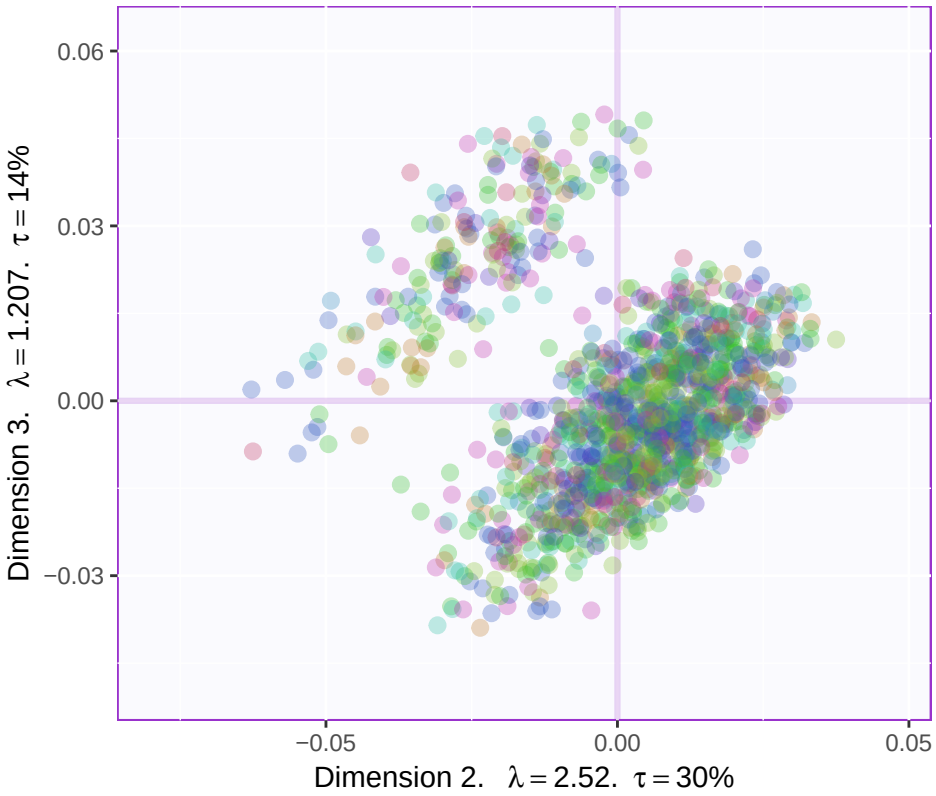


Job Role

```
baseMap.j <- PTCA4CATA::createFactorMap(resMFA$mexPosition.Data$Table$fi,col.points = resMFA$Plotting.Da
                                        col.labels = resMFA$Plotting.Data$fi.col,
                                        alpha.points =  .3,display.labels = FALSE)

label4Map <- createxyLabels.gen(1,2,
                                lambda = resMFA$mexPosition.Data$InnerProduct$eigs,
                                tau = resMFA$mexPosition.Data$InnerProduct$t )
# A graph for the J-set
b000.aggMap.j <- baseMap.j$zeMap_background + # background layer
  baseMap.j$zeMap_dots + baseMap.j$zeMap_text + label4Map
# We print this Map with the following code
print(b000.aggMap.j)
```

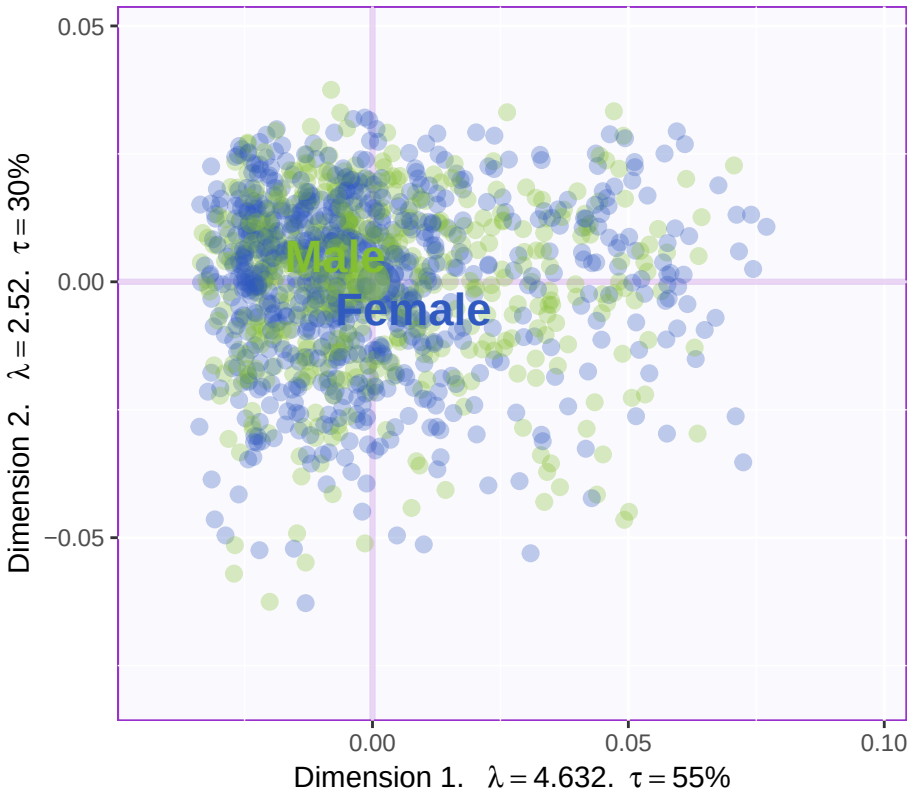



```
baseMap.j5 <- PTCA4CATA::createFactorMap(resMFA$mexPosition.Data$Table$fi,col.points = resMFA$Plotting.
                                         col.labels = resMFA$Plotting.Data$fi.col,
                                         alpha.points =  .3,display.labels = FALSE, axis1 = 2, axis2 = 3

label4Map5 <- createxyLabels.gen(2,3,
                                 lambda = resMFA$mexPosition.Data$InnerProduct$eigs,
                                 tau = resMFA$mexPosition.Data$InnerProduct$t )
# A graph for the J-set
b000.aggMap.j5 <- baseMap.j5$zeMap_background + # background layer
  baseMap.j5$zeMap_dots + baseMap.j5$zeMap_text + label4Map5
# We print this Map with the following code
print(b000.aggMap.j5)
```



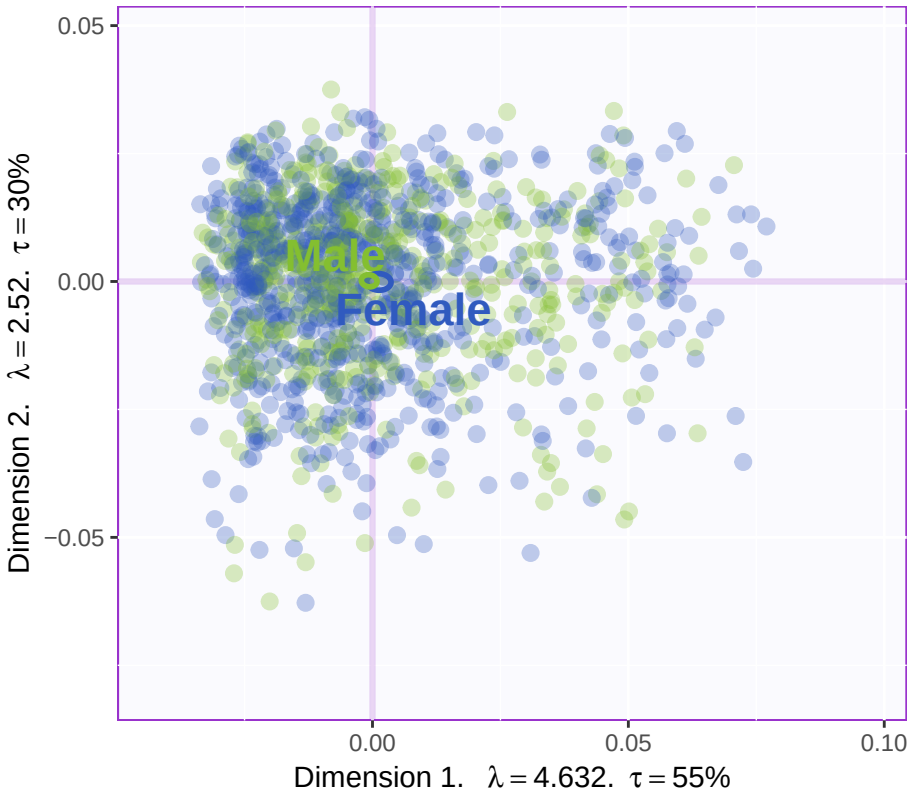

```
# Bootstrap for CI:
BootCube.Gr0 <- PTCA4CATA::Boot4Mean(resMFA1$mexPosition.Data$Table$fi,
                                     design = data$Gender,
                                     niter = 100,
                                     suppressProgressBar = TRUE)
#_____________________________________________________________________
# Bootstrap ratios ----
bootRatios.Gr0 <- boot.ratio.test(BootCube.Gr0$BootCube)
#**********************************************************************
# eigenvalues: MonteCarlo Approach ----
#
random.eigen0 <- data4PCCAR::monteCarlo.eigen(X = data1, nIter = 100)
#
# eigenvalues: Bootstrap approach
#
bootstrap.eigen0 <- data4PCCAR::boot.eigen(data1, nIter = 100)
# Mean Map
#  create the map for the means
#  get the means by groups

dataMeans0 <- PTCA4CATA::getMeans(resMFA1$mexPosition.Data$Table$fi, data$Gender)
# a vector of color for the means
col4data0 <- resMFA1$Plotting.Data$fi.col
col4Means0 <- unique(col4data0)
# the map
MapGroup0 <- PTCA4CATA::createFactorMap(dataMeans0,
```

```
                                    # use the constraint from the main map
                                    constraints = resMFA1$Plotting.Data$constraints,
                                    col.points = col4Means0,
                                    cex = 7,  # size of the dot (bigger)
                                    col.labels = col4Means0,
                                    text.cex = 6)
# The map with observations and group means
a003.Map.I.withMeans0 <- b000.aggMap.j0 +
  MapGroup0$zeMap_dots + MapGroup0$zeMap_text
print(a003.Map.I.withMeans0)
```



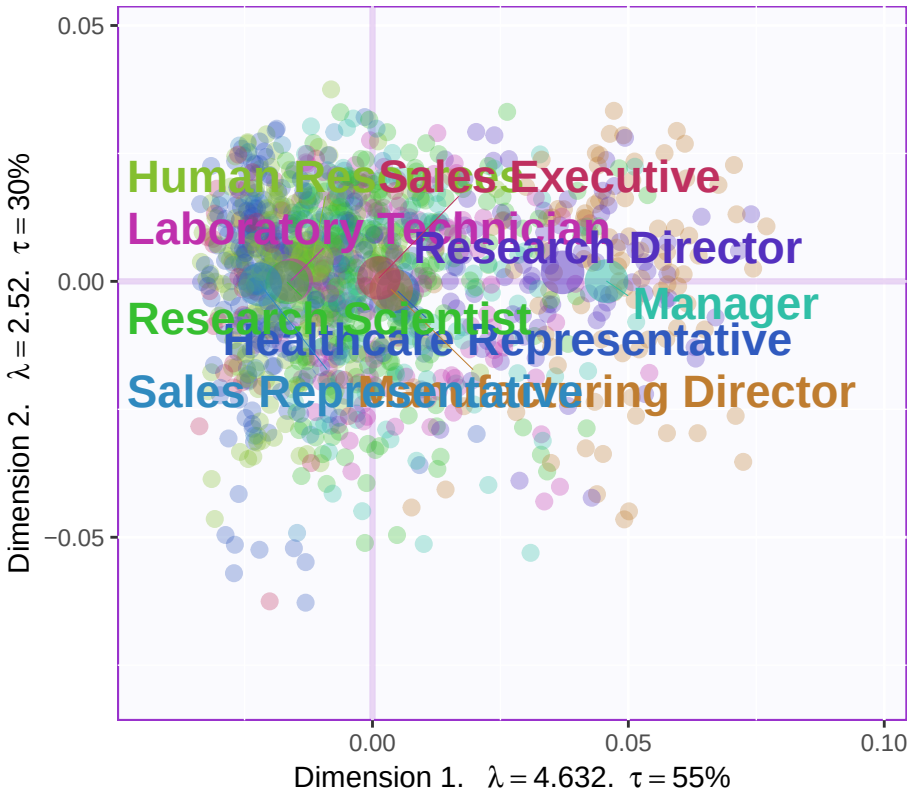

```
#_____________________________________________________________________
# Create the ellipses
# Bootstrapped CI ----
#_____________________________________________________________________
# Create Confidence Interval Plots
# use function MakeCIEllipses from package PTCA4CATA
GraphElli0 <- PTCA4CATA::MakeCIEllipses(BootCube.Gr0$BootCube[,1:2,],
                                    names.of.factors = c("Dimension 1","Dimension 2"),
                                    col = col4Means0,
                                    p.level = .95
)
#_____________________________________________________________________
# create the I-map with Observations, means and confidence intervals
#
a004.Map.I.withCI0 <-  b000.aggMap.j0 + MapGroup0$zeMap_text +  GraphElli0
```

```
#------------------------------------------------------------------------
# plot it!
print(a004.Map.I.withCI0)
```

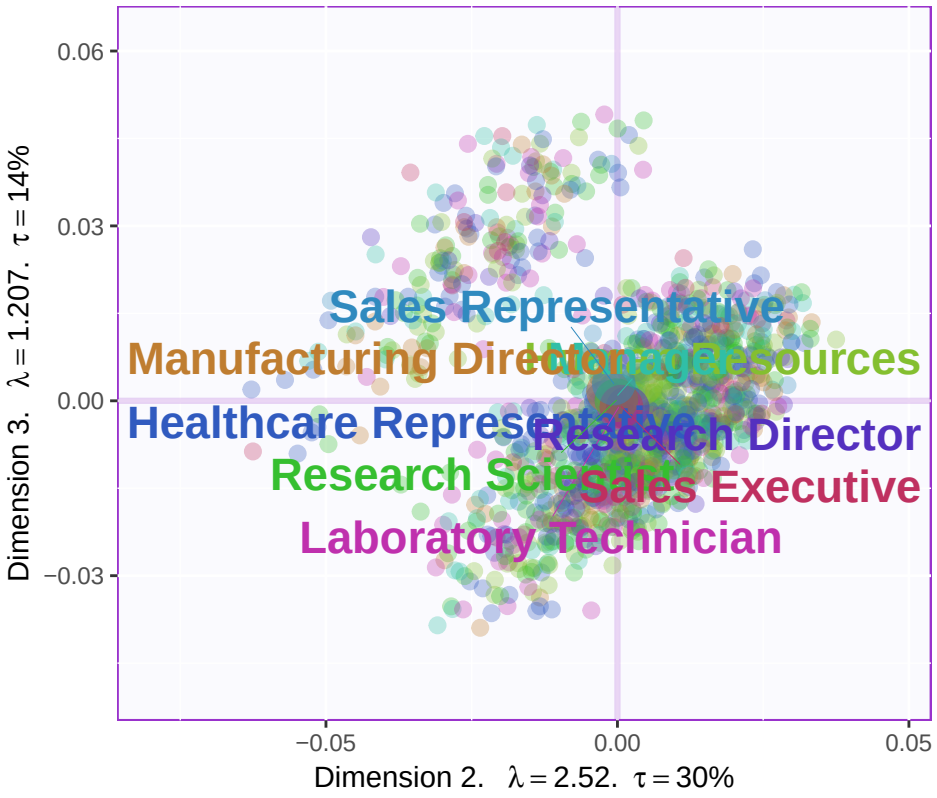



```
# Bootstrap for CI:
BootCube.Gr <- PTCA4CATA::Boot4Mean(resMFA$mexPosition.Data$Table$fi,
                                    design = data$JobRole,
                                    niter = 100,
                                    suppressProgressBar = TRUE)
#------------------------------------------------------------------------
# Bootstrap ratios ----
bootRatios.Gr <- boot.ratio.test(BootCube.Gr$BootCube)
#************************************************************************
# eigenvalues: MonteCarlo Approach ----
#
random.eigen <- data4PCCAR::monteCarlo.eigen(X = data1, nIter = 100)
#
# eigenvalues: Bootstrap approach
#
bootstrap.eigen <- data4PCCAR::boot.eigen(data1, nIter = 100)
# Mean Map
#  create the map for the means
#  get the means by groups

dataMeans <- PTCA4CATA::getMeans(resMFA$mexPosition.Data$Table$fi, data$JobRole)
# a vector of color for the means
```

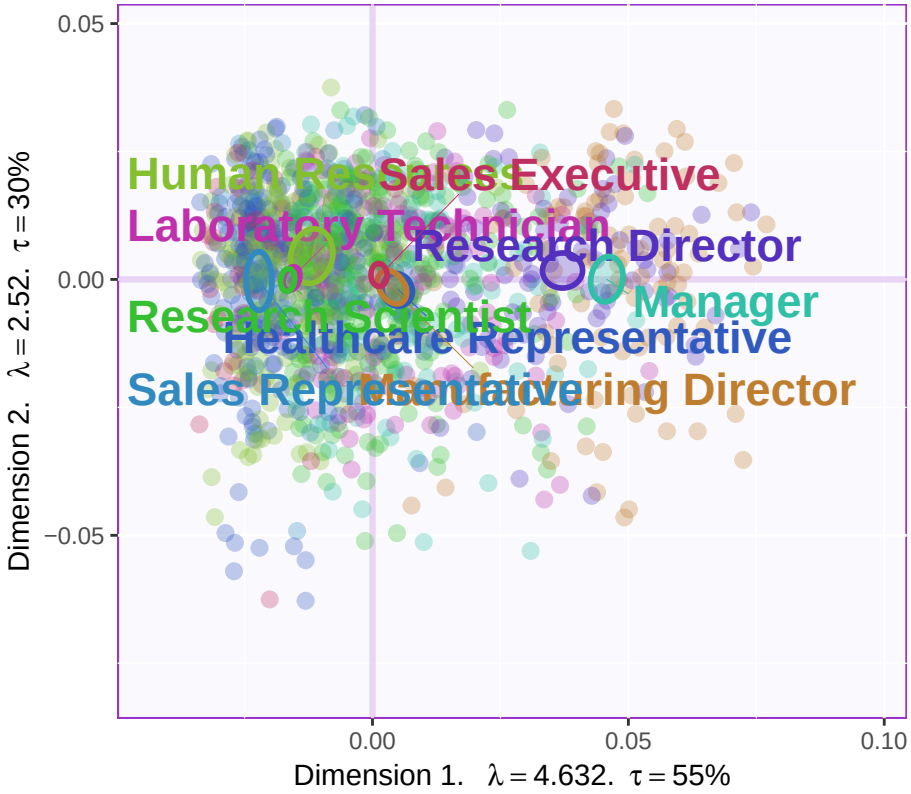
```
col4data <- resMFA$Plotting.Data$fi.col
col4Means <- unique(col4data)
# the map
MapGroup <- PTCA4CATA::createFactorMap(dataMeans,
                                       # use the constraint from the main map
                                       constraints = resMFA$Plotting.Data$constraints,
                                       col.points = col4Means,
                                       cex = 7,  # size of the dot (bigger)
                                       col.labels = col4Means,axis1 = 1, axis2 = 2,
                                       text.cex = 6)
MapGroup22 <- PTCA4CATA::createFactorMap(dataMeans,
                                       # use the constraint from the main map
                                       constraints = resMFA$Plotting.Data$constraints,
                                       col.points = col4Means,
                                       cex = 7,  # size of the dot (bigger)
                                       col.labels = col4Means,axis1 = 2, axis2 = 3,
                                       text.cex = 6)

# The map with observations and group means
a003.Map.I.withMeans <- b000.aggMap.j +
  MapGroup$zeMap_dots + MapGroup$zeMap_text
print(a003.Map.I.withMeans)
```



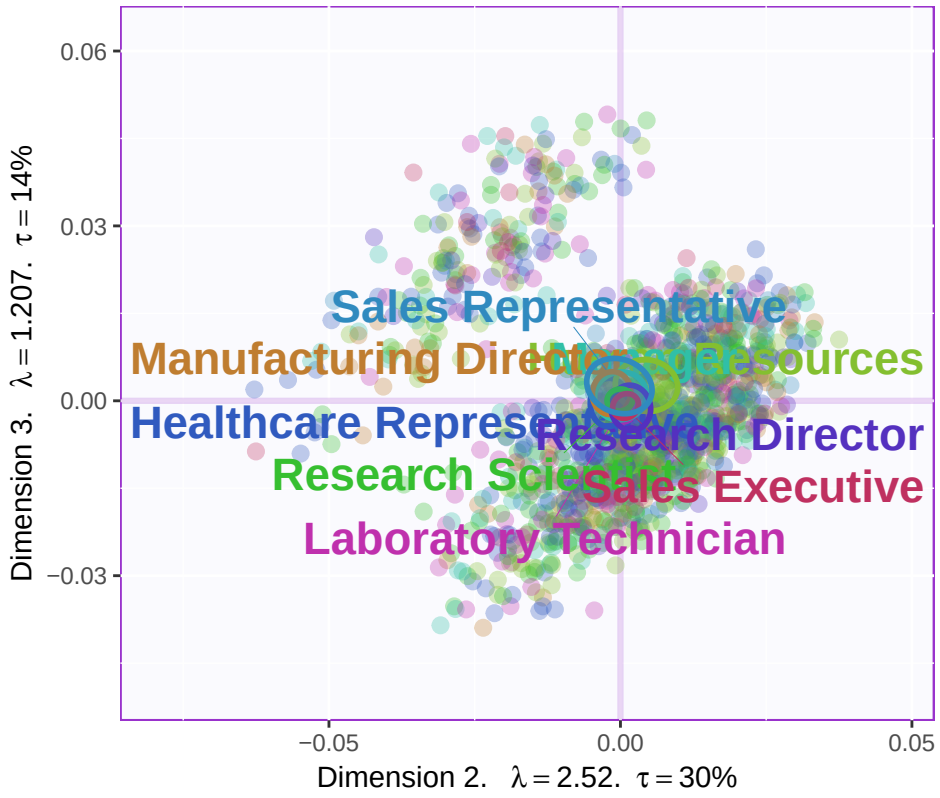

```
a003.Map.I.withMeans11 <- b000.aggMap.j5 +
  MapGroup22$zeMap_dots + MapGroup22$zeMap_text
print(a003.Map.I.withMeans11)
```




```
#_____________________________________________________________________
# Create the ellipses
# Bootstrapped CI ----
#_____________________________________________________________________
# Create Confidence Interval Plots
# use function MakeCIEllipses from package PTCA4CATA
GraphElli <- PTCA4CATA::MakeCIEllipses(BootCube.Gr$BootCube[,1:2,],
                                       names.of.factors = c("Dimension 1","Dimension 2"),
                                       col = col4Means,
                                       p.level = .95,axis1 = 1, axis2 = 2
)
GraphElli22 <- PTCA4CATA::MakeCIEllipses(BootCube.Gr$BootCube[,2:3,],
                                       names.of.factors = c("Dimension 2","Dimension 3"),
                                       col = col4Means,
                                       p.level = .95
)
#_____________________________________________________________________
# create the I-map with Observations, means and confidence intervals
#
a004.Map.I.withCI <-  b000.aggMap.j + MapGroup$zeMap_text +  GraphElli
print(a004.Map.I.withCI)
```




```
a004.Map.I.withCI11 <-  b000.aggMap.j5 + MapGroup22$zeMap_text +  GraphElli22
print(a004.Map.I.withCI11)
```



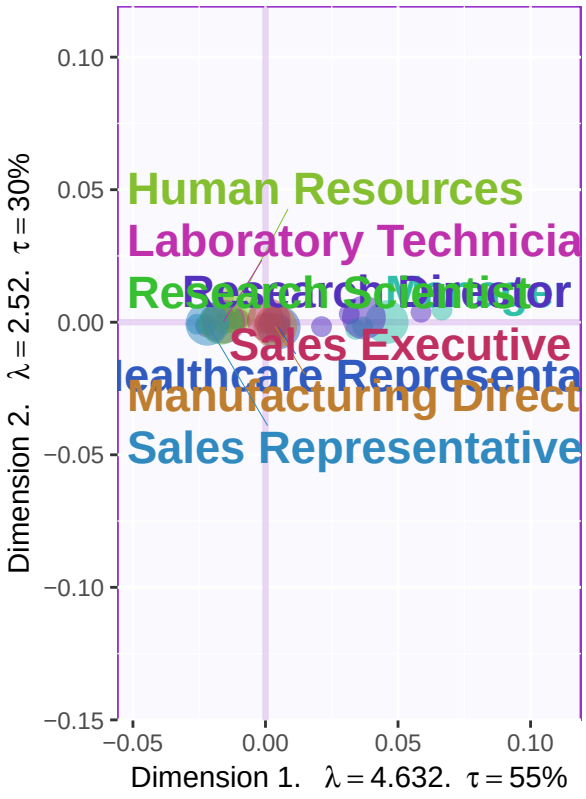

```
#_____________________________________________________________________
```

10.3 Partial Factor Scores

Job Role

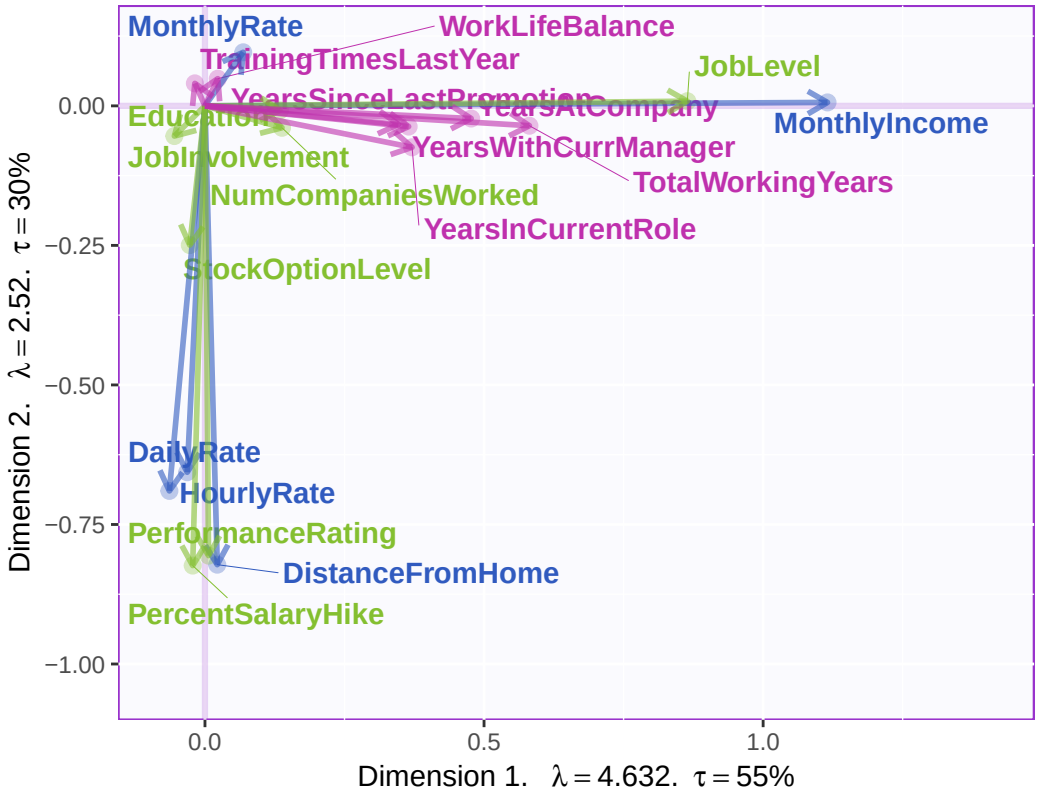
```
baseMap.j11 <- PTCA4CATA::createFactorMap(resMFA$mexPosition.Data$Table$partial.fi[1:1233,],
                                          col.labels = resMFA$Plotting.Data$fi.col,
                                          alpha.points =  .1,display.labels = FALSE)

dataMeans1 <- PTCA4CATA::getMeans(resMFA$mexPosition.Data$Table$partial.fi[1:1233,], data$JobRole)
dataMeans2 <- PTCA4CATA::getMeans(resMFA$mexPosition.Data$Table$partial.fi[1234:2466,], data$JobRole)
dataMeans3 <- PTCA4CATA::getMeans(resMFA$mexPosition.Data$Table$partial.fi[2467:3699,], data$JobRole)
#dataMeans4 <- dataMeans1 + dataMeans2 + dataMeans3

col4data1 <- resMFA$Plotting.Data$fi.col
col4Means1 <- unique(col4data1)
# the map
MapGroup1 <- PTCA4CATA::createFactorMap(dataMeans1,
                                        # use the constraint from the main map
                                        constraints = resMFA$Plotting.Data$constraints,
                                        col.points = col4Means1,
                                        cex = 3,  # size of the dot (bigger)
                                        col.labels = col4Means1,display.labels = FALSE,
                                        text.cex = 6)
MapGroup2 <- PTCA4CATA::createFactorMap(dataMeans2,
```

```
                                        # use the constraint from the main map
                                        constraints = resMFA$Plotting.Data$constraints,
                                        col.points = col4Means1,
                                        cex = 3,  # size of the dot (bigger)
                                        col.labels = col4Means1, display.labels = FALSE,
                                        text.cex = 6)
MapGroup3 <- PTCA4CATA::createFactorMap(dataMeans3,
                                        # use the constraint from the main map
                                        constraints = resMFA$Plotting.Data$constraints,
                                        col.points = col4Means1,
                                        cex = 3,  # size of the dot (bigger)
                                        col.labels = col4Means1,display.labels = FALSE,
                                        text.cex = 6)

label4Map1 <- createxyLabels.gen(1,2,
                                 lambda = resMFA$mexPosition.Data$InnerProduct$eigs,
                                 tau = resMFA$mexPosition.Data$InnerProduct$t )

# The map with observations and group means
a003.Map.I.withMeans1 <- baseMap.j11$zeMap_background + label4Map1 + MapGroup1$zeMap_dots + MapGroup1$z
print(a003.Map.I.withMeans1)
```



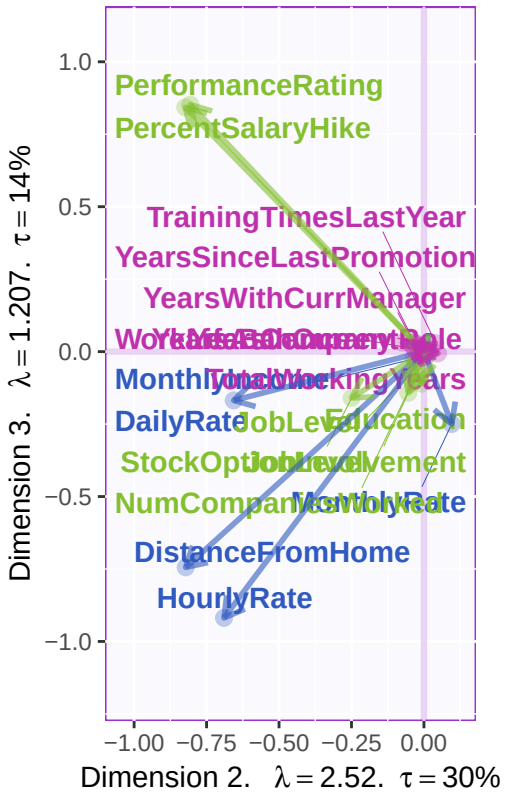
10.4 Loadings

```
baseMap.j2 <- PTCA4CATA::createFactorMap(resMFA$mexPosition.Data$Table$Q,col.points = resMFA$Plotting.Da
                                         col.labels =resMFA$Plotting.Data$fj.col ,
                                         alpha.points =  .3,display.labels = TRUE)
# arrows
zeArrows2 <- addArrows(resMFA$mexPosition.Data$Table$Q , col = resMFA$Plotting.Data$fj.col )
# A graph for the J-set
b000.aggMap.j2 <- baseMap.j2$zeMap_background + # background layer
  baseMap.j2$zeMap_dots + baseMap.j2$zeMap_text +  # dots & labels
  label4Map + zeArrows2
# We print this Map with the following code
print(b000.aggMap.j2)
```



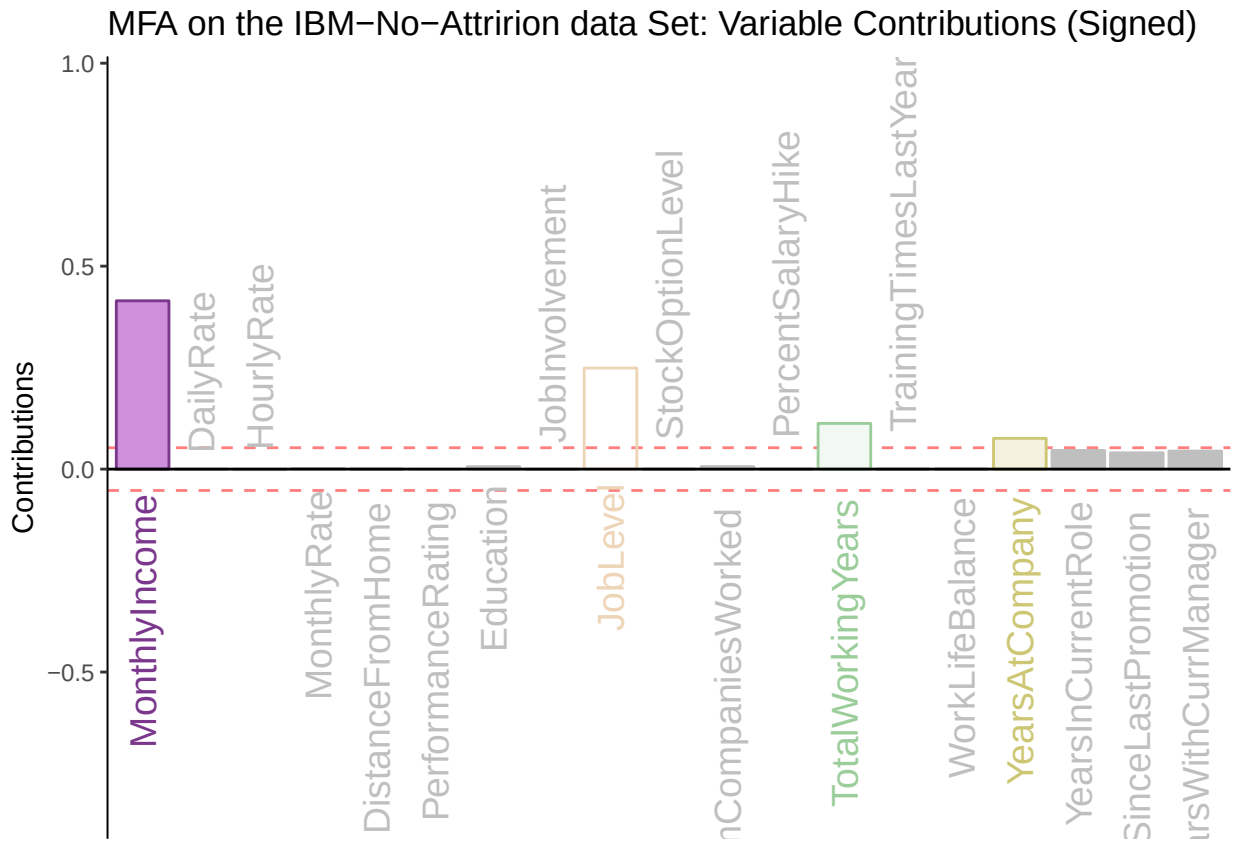

```
baseMap.j21 <- PTCA4CATA::createFactorMap(resMFA$mexPosition.Data$Table$Q,col.points = resMFA$Plotting.
                                         col.labels =resMFA$Plotting.Data$fj.col ,
                                         alpha.points =  .3,display.labels = TRUE,axis1 = 2,axis2 = 3)
# arrows
label4Map3 <- createxyLabels.gen(2,3,
                                 lambda = resMFA$mexPosition.Data$InnerProduct$eigs,
                                 tau = resMFA$mexPosition.Data$InnerProduct$t )

zeArrows3 <- addArrows(resMFA$mexPosition.Data$Table$Q , col = resMFA$Plotting.Data$fj.col,axis1 = 2,ax
# A graph for the J-set
b000.aggMap.j21 <- baseMap.j21$zeMap_background + # background layer
  baseMap.j21$zeMap_dots + baseMap.j21$zeMap_text +  # dots & labels
```

```
  label4Map3 + zeArrows3
# We print this Map with the following code
print(b000.aggMap.j21)
```

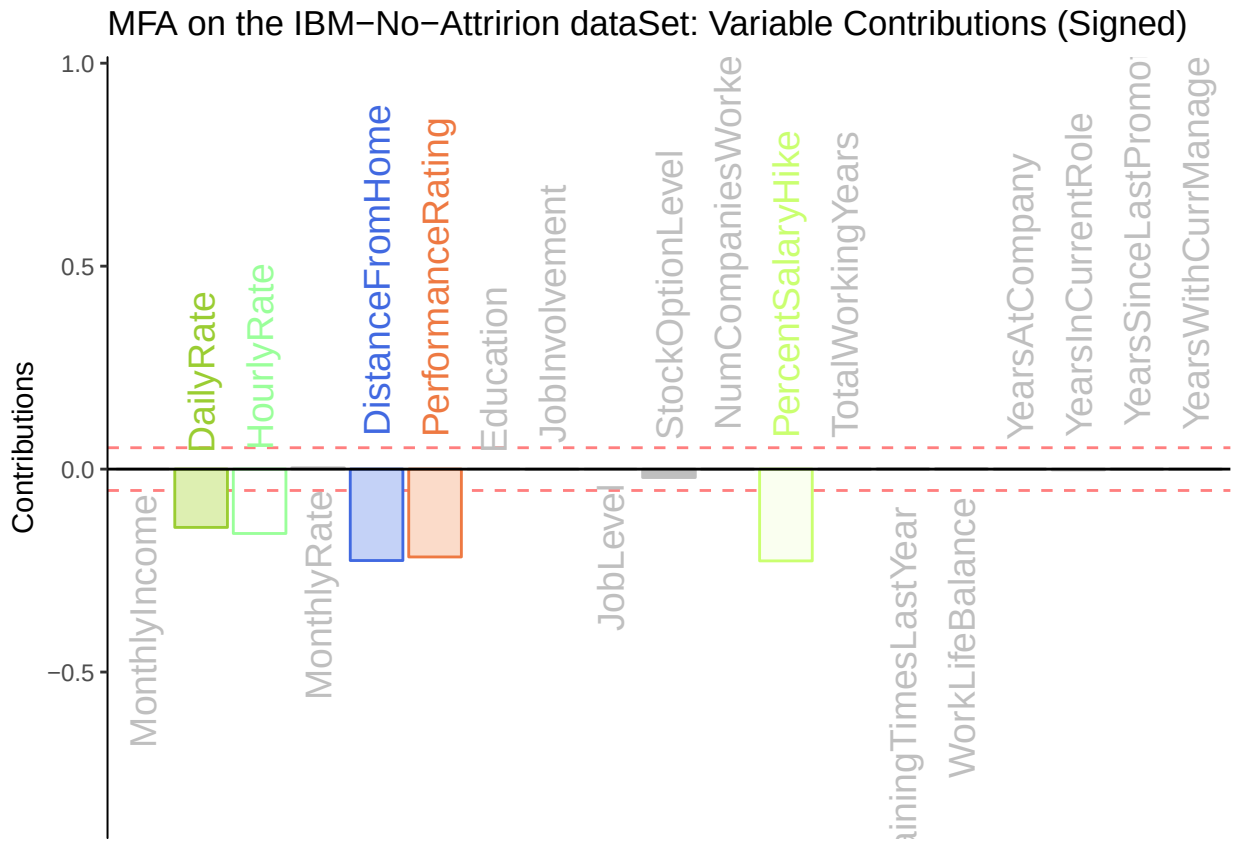


10.5 Contribution

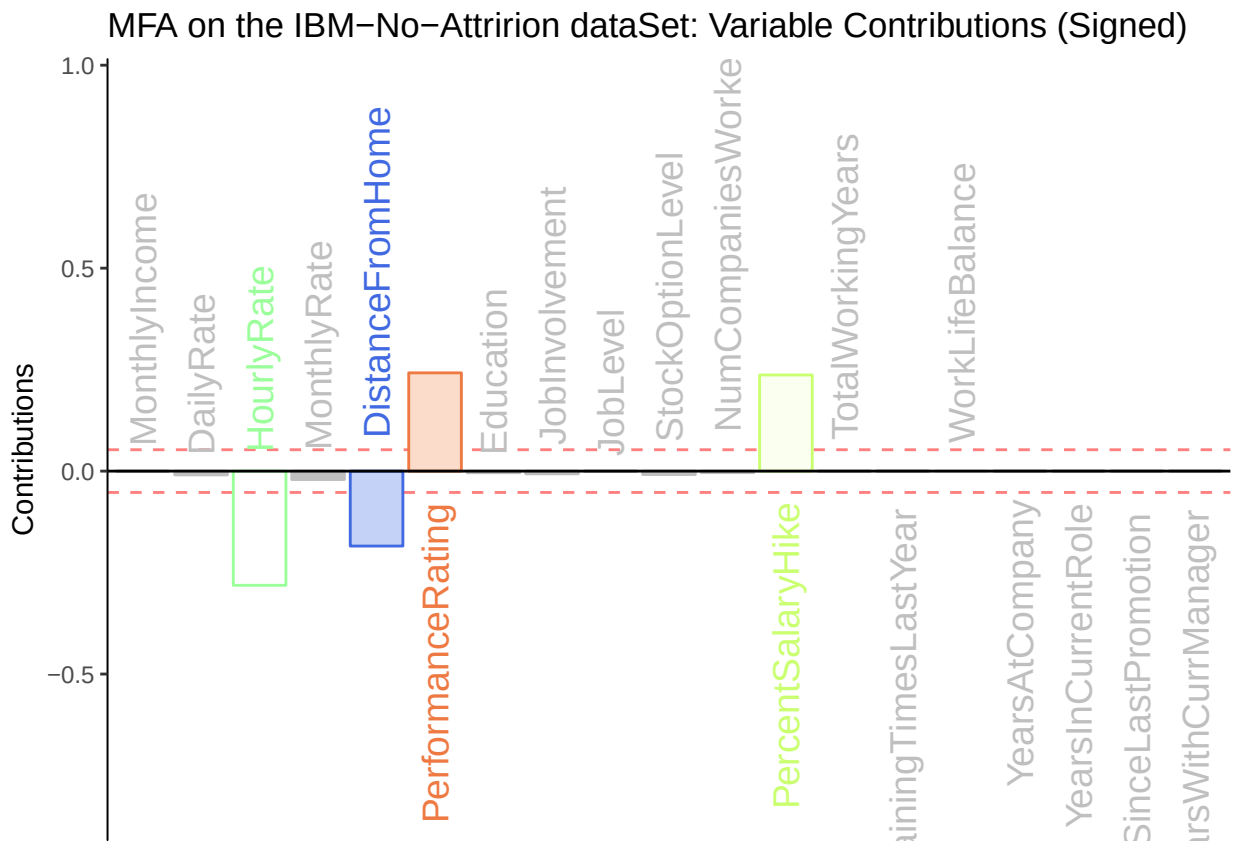
```
col4J.ibm <- prettyGraphsColorSelection(NCOL(data1))
signed.ctrJ <- resMFA$mexPosition.Data$Table$cj * sign(resMFA$mexPosition.Data$Table$Q)
b003.ctrJ.s.1 <- PrettyBarPlot2(signed.ctrJ[,1],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'MFA on the IBM-No-Attririon data Set: Variable Contributions (S
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b003.ctrJ.s.1)
```




```
b004.ctrJ.s.2 <- PrettyBarPlot2(signed.ctrJ[,2],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'MFA on the IBM-No-Attririon dataSet: Variable Contributions (Si
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b004.ctrJ.s.2)
```

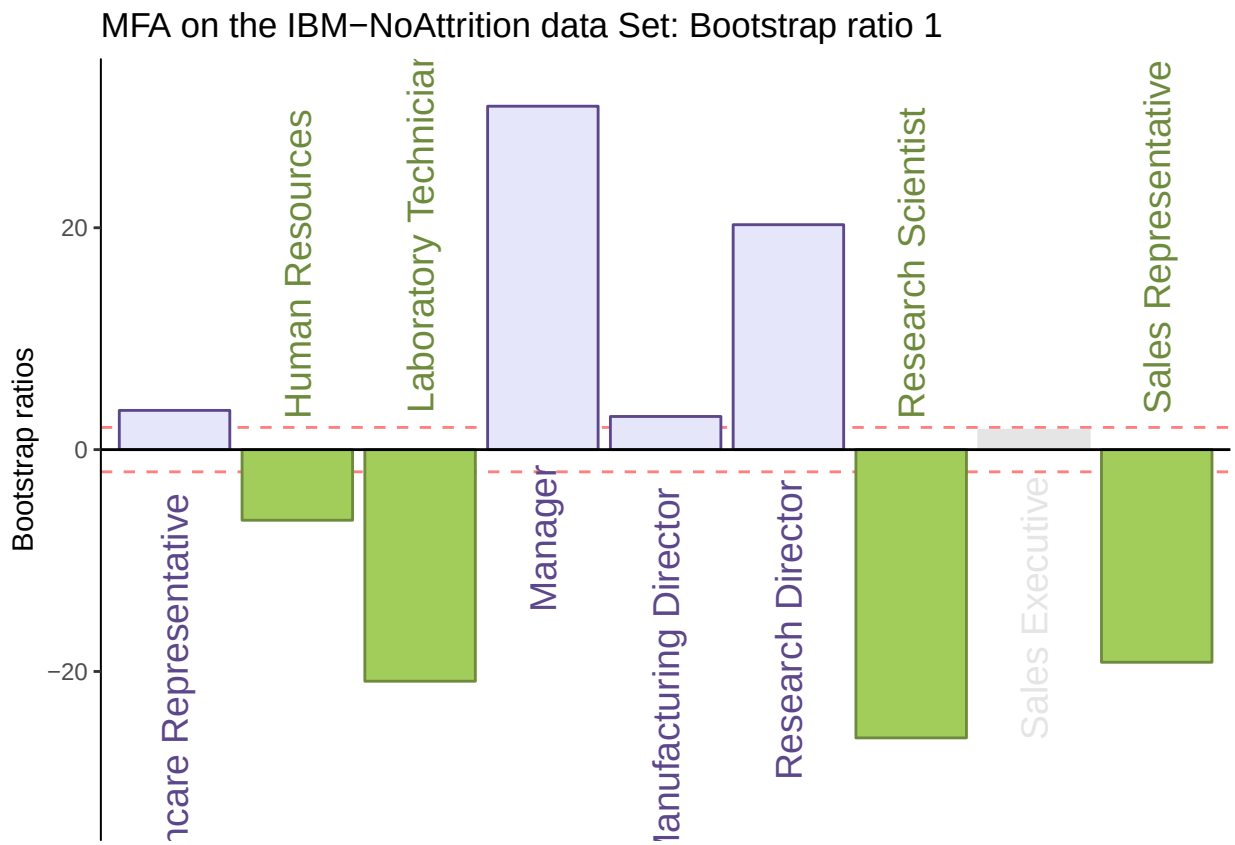



```
b004.ctrJ.s.3 <- PrettyBarPlot2(signed.ctrJ[,3],
                                threshold = 1 / NROW(signed.ctrJ),
                                font.size = 5,
                                color4bar = gplots::col2hex(col4J.ibm), # we need hex code
                                main = 'MFA on the IBM-No-Attririon dataSet: Variable Contributions (Sig
                                ylab = 'Contributions',
                                ylim = c(1.2*min(signed.ctrJ), 1.2*max(signed.ctrJ))
)
print(b004.ctrJ.s.3)
```

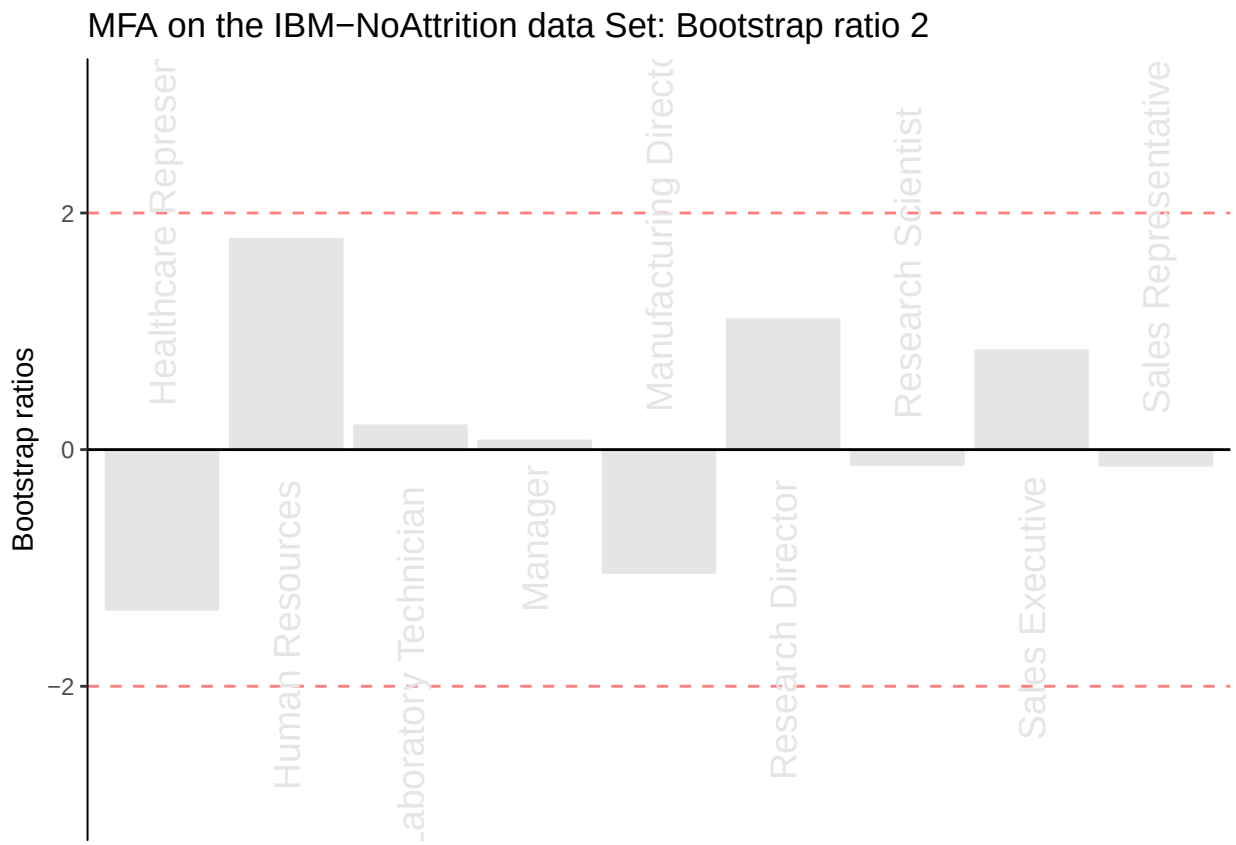


10.6 Bootstrap Ratios for Variables

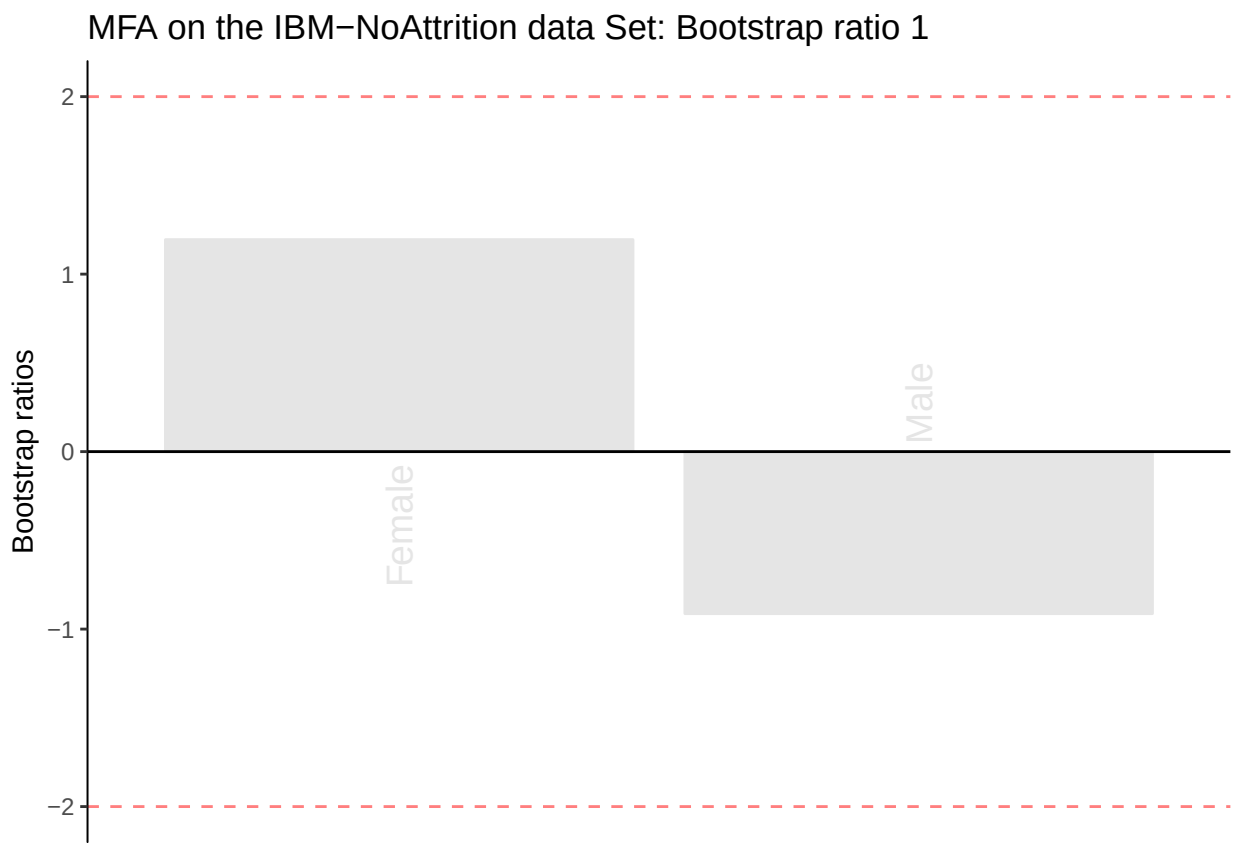
```
BR2 <- bootRatios.Gr$boot.ratios
ba001.BR111 <- PrettyBarPlot2(BR2[,1],
                              threshold = 2,
                              font.size = 5,
                              #color4bar = gplots::col2hex(col4J.ibm),
                              main = paste0( 'MFA on the IBM-NoAttrition data Set: Bootstrap ratio ',1),
                              ylab = 'Bootstrap ratios'
                              #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR111)
```



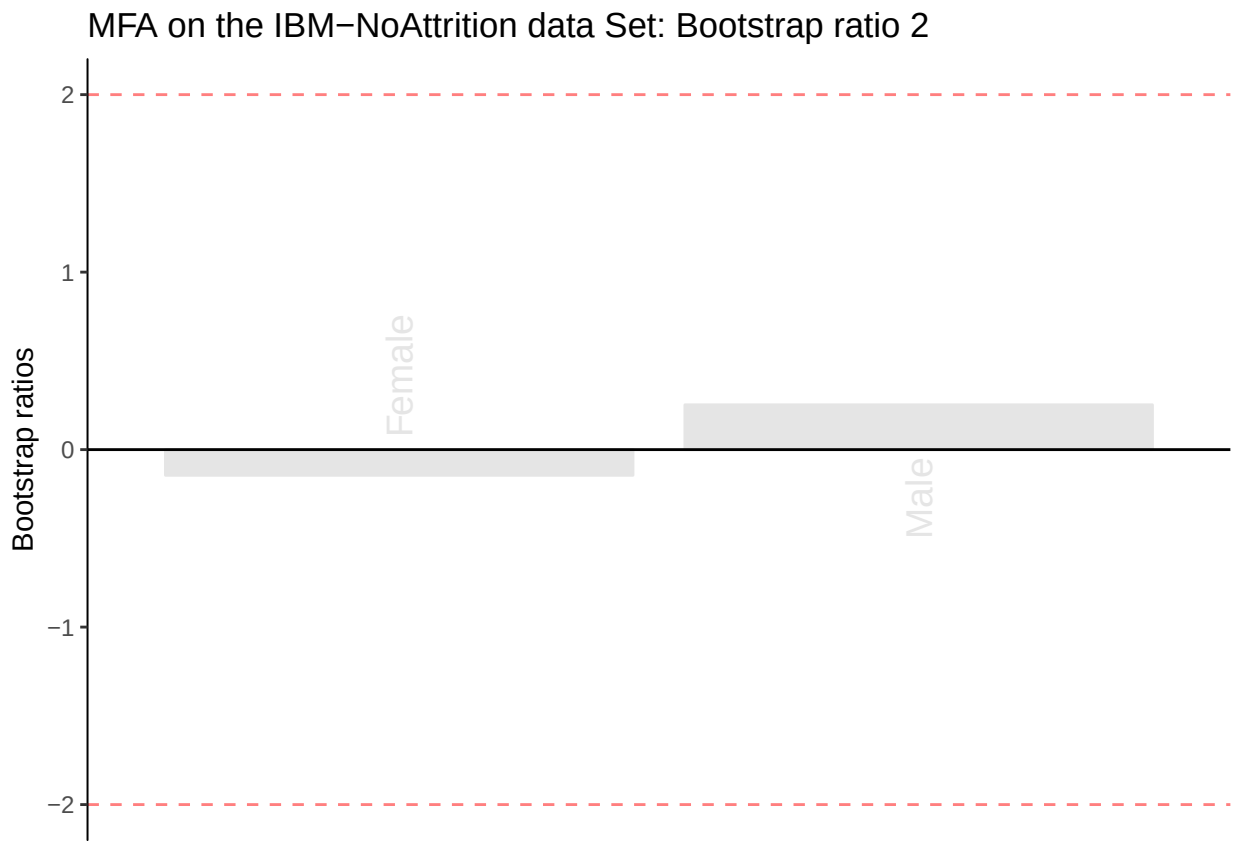

```
#
ba002.BR211 <- PrettyBarPlot2(BR2[,2],
                              threshold = 2,
                              font.size = 5,
                              #color4bar = gplots::col2hex(col4J.ibm),
                              main = paste0(
                                'MFA on the IBM-NoAttrition data Set: Bootstrap ratio ',2),
                              ylab = 'Bootstrap ratios'
)
print(ba002.BR211)
```




```
BR3 <- bootRatios.Gr0$boot.ratios
ba001.BR1113 <- PrettyBarPlot2(BR3[,1],
                               threshold = 2,
                               font.size = 5,
                               #color4bar = gplots::col2hex(col4J.ibm),
                               main = paste0( 'MFA on the IBM-NoAttrition data Set: Bootstrap ratio ',1),
                               ylab = 'Bootstrap ratios'
                               #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)
print(ba001.BR1113)
```




```
#
ba002.BR2114 <- PrettyBarPlot2(BR3[,2],
                               threshold = 2,
                               font.size = 5,
                               #color4bar = gplots::col2hex(col4J.ibm),
                               main = paste0(
                                 'MFA on the IBM-NoAttrition data Set: Bootstrap ratio ',2),
                               ylab = 'Bootstrap ratios'
)
print(ba002.BR2114)
```



10.7 Summary

- People who are Research Director and Manager have high Monthly Income and Job-level with more number of experience
- No significant inference can be concluded with the gender type affecting monthly income or Job level
- Sales Representative group has the lowest Income with less years of experience