A Survey: Attitudes towards fermentation analyzed with Multiple Correspondence Analysis (and supplementary observations)

Hervé Abdi, Dominique Valentin, & Sylvie Chollet

2018-10-23 15:48:23

Contents

1 Prelude

If you want to make sure that you have a clean start, you can execute the following commands:

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

Or, better (see below, preamble), you can use an `Rproject` for this project.

2 Preamble

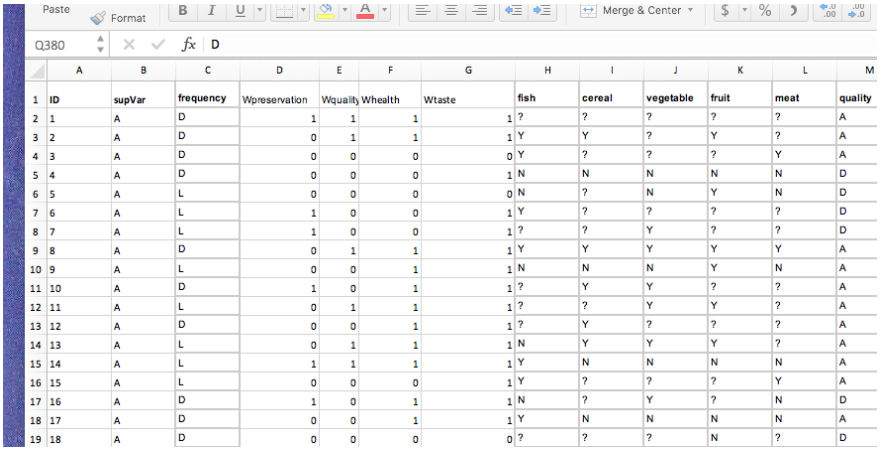
Make sure that your start this analysis a new `Rproject` so that the default directory will be correctly set.

Before we start the analysis, we need to have our standard packages installed from `Github`) and the corresponding libraries loaded:

- `Distatis`
- `ExPosition`
- `InPosition`
- `PTCA4CTA`
- `R4SPISE2018`
- `dplyr` (to recode the data)
- `car` (to recode the data)
- `grid`, `gridExtra`, and `gTable` (to save tables as graphics, useful for power points)
- `stringi` (to recode strings)

and all their extensions:

```
# Decomment all/some these lines if the packages are not installed
#_____________________________________________________________________
# # October 19, 2018. Temporary fix for an Rstudio problem
# # if this error is generated
# build_site()
# Error in eval(substitute(expr), data, enclos = parent.frame()) :
# is.character(repos) is not TRUE
# # from Rstudio team: use these two lines
# repos <- getOption("repos")
# options(repos = setNames(as.character(repos), names(repos)))
#_____________________________________________________________________
#
# devtools::install_github('HerveAbdi/PTCA4CATA')
# devtools::install_github('HerveAbdi/DistatisR')
# devtools::install_github('HerveAbdi/data4PCCAR')
# devtools::install_github('HerveAbdi/R4SPISE2018') # of course!
#  install.packages('prettyGraphs')
#  install.packages('Matrix')
#  install.packages('dplyr')
#  install.packages('gridExtra')
#  install.packages('grid')
#  install.packages('gtable')
#  install.packages('stringi')
#  install.packages('printr')
#  install.packages('kableExtra')
#  load the libraries that we will need
suppressMessages(library(Matrix))
suppressMessages(library(prettyGraphs))
suppressMessages(library(ExPosition))
suppressMessages(library(InPosition))
suppressMessages(library(DistatisR))
suppressMessages(library(PTCA4CATA))
suppressMessages(library(data4PCCAR))
suppressMessages(library(dplyr))
suppressMessages(library(gridExtra))    # to save a table as a graph
suppressMessages(library(grid))         # that will be saved in the
```



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ID	supVar	frequency	Wpreservation	Wquality	Whealth	Wtaste	fish	cereal	vegetable	fruit	meat	quality
2	1	A	D	1	1	1	1	?	?	?	?	?	A
3	2	A	D	0	1	1	1	Y	Y	?	Y	?	A
4	3	A	D	0	0	0	0	Y	?	?	?	Y	A
5	4	A	D	0	0	0	1	N	N	N	N	N	D
6	5	A	L	0	0	0	0	N	?	N	Y	N	D
7	6	A	L	1	0	0	1	Y	?	?	?	?	D
8	7	A	L	1	0	0	1	?	?	Y	?	?	D
9	8	A	D	0	1	1	1	Y	Y	Y	Y	Y	A
10	9	A	L	0	0	1	1	N	N	N	Y	N	A
11	10	A	D	1	0	1	1	?	Y	Y	?	?	A
12	11	A	L	0	1	1	1	?	?	Y	Y	?	A
13	12	A	D	0	0	1	1	?	Y	?	?	?	A
14	13	A	L	0	1	1	1	N	Y	Y	Y	?	A
15	14	A	L	1	1	1	1	Y	N	N	N	N	A
16	15	A	L	0	0	0	1	Y	?	?	?	Y	A
17	16	A	D	1	0	1	1	N	?	Y	?	N	D
18	17	A	D	0	0	1	1	Y	N	N	N	N	A
19	18	A	D	0	0	0	0	?	?	?	N	?	D

Figure 1: The Data Excel File

```
suppressMessages(library(gtable))       # powerpoint with the figures
# suppressMessages(library(printr))     # To pretty print tables
# suppressMessages(library(kableExtra)) # To pretty print tables
```

3 Introduction

3.1 File name for the power point to save the figures

The name for the power point is stored as

```
name4Graphs = 'fermentationFrom2Countries.pptx'
```

3.2 Parameters for the analysis: data file name

The data are stored in an excel file called 'fermentedFoodSurvey.xlsx' that is stored with the package and whose path can be recovered as:

```
file2read.name     <- 'fermentedFoodSurvey.xlsx' # xls data file name
path2file <- system.file("extdata", file2read.name, package = "R4SPISE2018")
```

In the xls-file fermentedFoodSurvey.xlsx, the data are stored in the sheet called `dataMCA` whose name is stored in the variable `sheetName4Data`:

```
sheetName4Data     <- 'dataMCA' # sheet name for the data
```

3.2.1 A peek at the data

The excel data file looks like the figure below:

3.2.2 To replicate with different data

When you record you own data, make sure that you follow the same format, this way the script described in this vignette will apply to your own analysis with minimal change.

3.2.3 The general analytic strategy

We will first compute the results of the analysis, then create the graphics, and finally save everything into a PowerPoint.

4 Run the statistical analysis

4.1 Read the (active and supplementary) data

The excel file name and location (i.e., path) are stored in the variable `path2file`. To read the data we will use the function `DistatisR::read.df.excel()` (based upon the function `readxl::read_excel()`).

```
rawData.tmp <- DistatisR::read.df.excel(path = path2file,
                                sheet = sheetName4Data)$df.data
#
```

The raw data are now read and stored in the data frame `rawData.tmp`. For these data to be used in an MCA, they need to be transformed into `factors` and stored in the data frame called `rawData`.

```
# Transform the data into factors
rawData <- apply(rawData.tmp,2,as.factor)
```

4.2 Look at the data

The first thing in a survey is to look at the data and to select the questions that create enough differences between the respondents. A quick and dirty way top look at the data is to use the function `summary()`.

```
knitr::kable( t( summary(rawData) ) )
```

supVar	A:373	S: 30	NA
frequency	D:186	L:217	NA
Wpreservation	0:299	1:104	NA
Wquality	0:274	1:129	NA
Whealth	0:188	1:215	NA
Wtaste	0:107	1:296	NA
fish	?: 97	N:153	Y:153
cereal	?:104	N:124	Y:175
vegetable	?: 55	N: 85	Y:263
fruit	?: 70	N:108	Y:225
meat	?:107	N:163	Y:133
quality	A:247	D:156	NA
industrial	A:121	D:282	NA
preservation	A:322	D: 81	NA
health	A:345	D: 58	NA
expensive	A: 88	D:315	NA
taste	A:357	D: 46	NA
trust	A:253	D:150	NA
clear	A:275	D:128	NA
innovation	A:182	D:221	NA
sex	M:111	W:292	NA
age	<25:142	>25:261	NA
occupation	FM :130	other:273	NA
nation	F :220	VN:183	NA

From the summary, we see that most questions have two possible answers and that a few of them have three possible answers.

4.2.1 A bit more on the variables

The three variables starting with `W` were responses to the question *What makes you want to buy a fermented product?* These variables were coded as `1` if checked and `0` if not. The variables `fish` to `meat` were answers to the question `Fermentation is well suited for ...:` (3 responses were possible: `Yes`, `No`, and `Do not know`). The variables `quality` to `clear` were Likert scales (answers from 1 to 4) about statements on fermentation; These Likert scales were recoded as `A` (agree) and `D` (disagree). `Sex` is coded as `M` vs `W`. the variable `Age` is coded as less then 25 (`<25`) and more than 25 (`>25`). Occupation was binarized as "Food or microbiology" (`FM`) versus `other`. Nationality (`nation`) had two levels French (`F`) versus Vietnamese (`VN`).

To know more about the questions, you can have a look at the questionnaire `questionnaireFermentation.pdf` that you can find in the directory that you can recover from the variable `path2file`.

4.3 Analysis plan

The data were previously "cleaned" so that we only kept questions that created differences between the respondents.

In addition, we have some information about the respondents: Sex `(sex)`, Age `(age)`, their country of residence `(nation)`.

4.4 Selection and recoding

In MCA, the data need to be recoded as nominal variables with the constraint that the different levels of a given variable are roughly balanced. When all the variables are naturally nominal, the process is straightforward but some categories may need to be fused to have the levels roughly balanced. For ordinal or quantitative data, the procedure is to bin the data so that the bins are roughly balanced.

4.5 From one dataframe to the other

From the dataframe `rawData` we now create the new dataframe `cleanData`. It will have the same rows but the columns are obtained by recoding the columns of `rawData`.

The first step is to create the new data frame with only the variables we want to use, we also re-order the variables to have the respondent variables (i.e., Sex, Age, city) first. This is done with this line of code:

```
temp.df <- dplyr::select(as.data.frame(rawData),
                         supVar,
                         sex , age, occupation, nation, frequency,
                         Wpreservation, Wquality, Whealth, Wtaste,
                         fish, cereal,  vegetable, fruit, meat,
                         quality, industrial, preservation, health,
                         expensive, taste, trust, clear, innovation)
```

Note that here, for a first pass, we keep all variables:

4.5.1 Participants' description

to make the results and graphs more readable, we recode the levels of some variables to shorter names

```
temp.df[,'age'] <- plyr::mapvalues(temp.df[,'age'],
                           from = c("<25", ">25"), to = c("Y", "O"))
temp.df[,'occupation'] <- plyr::mapvalues(temp.df[,'occupation'],
                           from = c("FM", "other"), to = c("F", "O"))
temp.df[,'nation'] <- plyr::mapvalues(temp.df[,'nation'],
                           from = c("F", "VN"), to = c("F", "V"))
```

4.5.2 New cleaned data set(s)

We can now put together the new data set. Note that in this example the supplementary observations were stored in the same excel sheet as the active data but the supplementary observations' names all start the letter `S`. The status of the observations (i.e., active vs. supplementary) is recorded in the variable `supVar` that we will use to subset the observations into the active set (stored in the data frame `cleanData`) and the supplementary set (stored in the data frame `cleanData.sup`).

```
cleanData.tmp <- temp.df
cleanData.tmp <- cleanData.tmp[complete.cases(cleanData.tmp),]
cleanData     <- cleanData.tmp[cleanData.tmp[,1] == 'A', 2:ncol(cleanData.tmp)]
cleanData.sup <- cleanData.tmp[cleanData.tmp[,1] == 'S', 2:ncol(cleanData.tmp)]
```

The dataframe is now ready to be analyzed with multiple correspondence analysis

5 Multiple Correspondence Analysis

5.1 MCA for the active data set

We use the package `ExPosition` to compute the MCA

```
resMCA <- epMCA(cleanData, graphs = FALSE)
```

5.2 Supplementary observations

To compute the supplementary projections, we use the function `ExPosition::supplementaryRows()`. But before using this function, we need to prepare the supplementary data. To do so, we need to transform the supplementary data from a data frame with variables being factors to a new data frame with 0/1 variables (with as many 0/1 variables as there are levels for this variable). For this transformation to be correctly done, it needs to be done on the merged data set (i.e., active + supplementary). All this is done with the code below

```
# recode the factors as set of 0/1 variables
testclean <- makeNominalData(rbind(cleanData,cleanData.sup))
clean.Sup <-  testclean[cleanData.tmp[,1] == 'S',]
# barycentric code for nation
clean.Sup[,(colnames(testclean) %in% 'nation.F')] <- .5
clean.Sup[,(colnames(testclean) %in% 'nation.V')] <- .5
#
resMCA.sup <- supplementaryRows(SUP.DATA = clean.Sup, res = resMCA)
colnames(resMCA.sup$fii) <- paste0('Dimension ',
                                   1:ncol(resMCA.sup$fii))
```

5.3 Inferences

Inferences are performed with the package `InPosition`

```
resMCA.inf <- epMCA.inference.battery(cleanData, graphs = FALSE)
```

6 Graphics

Here we make the standard graphs for the active variables and observations, and then for the supplementary observations

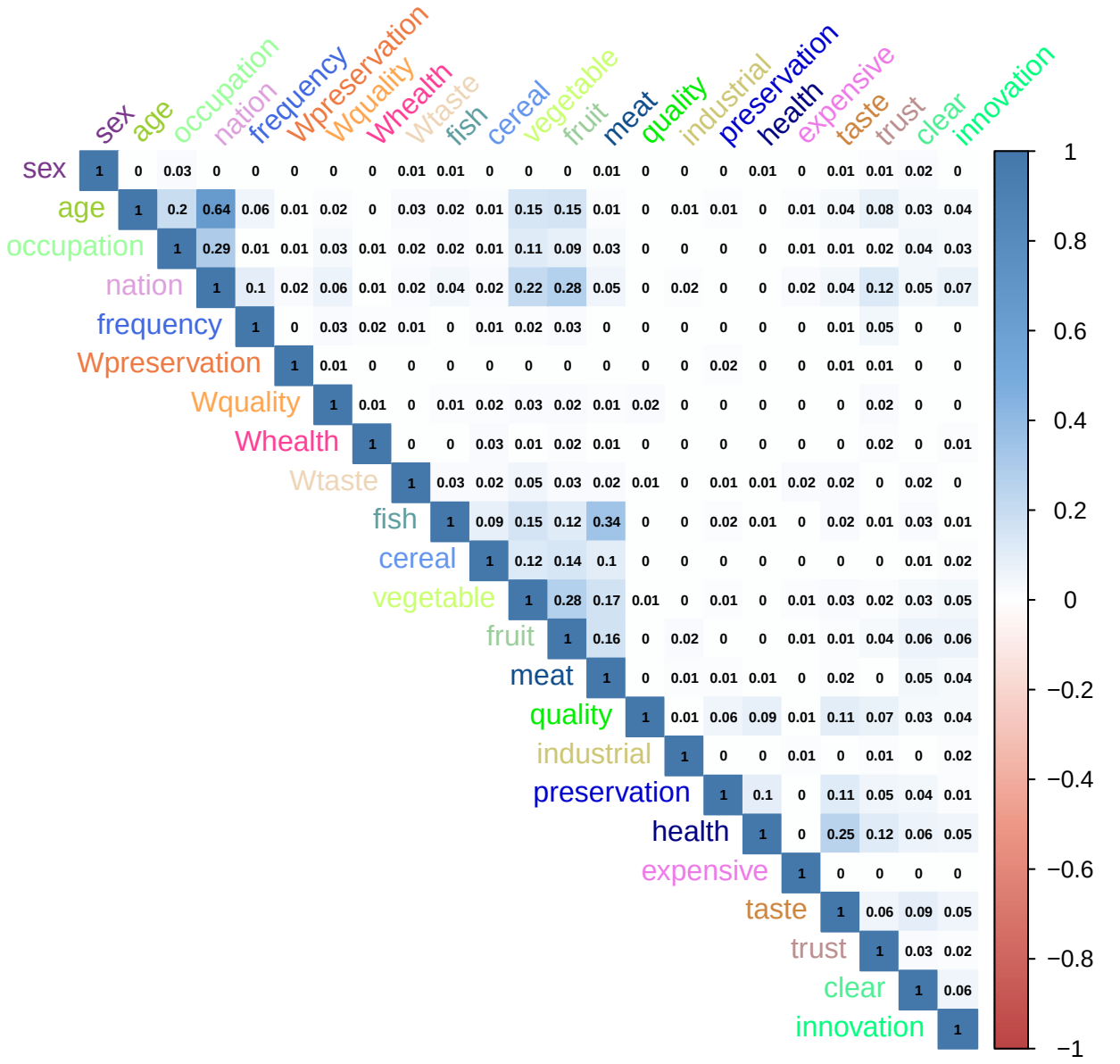
6.1 Get some colors for the variables

```
cJ <- resMCA$ExPosition.Data$cj
color4Var <- prettyGraphs::prettyGraphsColorSelection(ncol(cleanData))
```

6.2 Heatmap for the correlations between qualitative variables

The equivalent of a correlation matrix can be obtained in MCA by computing the ϕ^2 correlation matrix associated to the Burt-table derived from the 0/1 matrix. This is obtained from the following code:

```
# Pseudo Heat Map. Correlation ----
# We need correlation to compare with PCA
corrMatBurt.list <- phi2Mat4BurtTable(cleanData)
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corr4MCA.r <- corrplot::corrplot(
         as.matrix(corrMatBurt.list$phi2.mat),
         method="color", col=col(200),
         type="upper",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col=color4Var,
         tl.srt = 45, #Text label color and rotation
         number.cex = .5,
         diag = TRUE # needed to have the color of variables correct
         )
```




```
# dev.new()
a0000.corMat.phi <- recordPlot()
```

6.3 Variable contributions

The contributions for the variables can be obtained from the function `data4PCCAR::ctr4Variables` as:

```
varCtr <- data4PCCAR::ctr4Variables(cJ)
rownames(color4Var) <- rownames(varCtr)
```

The contribution of a variable is the sum of the contributions of all its levels, this parameter measures the importance of this variable for a given dimension.

```
nFact <- min(5, ncol(cJ) - 1)
#knitr::kable(round( varCtr[,1:nFact]*1000 ) )
# save table as a graph
```

```
ctrTable <- tableGrob(round(varCtr[,1:nFact]*1000))
h <- grobHeight(ctrTable)
w <- grobWidth(ctrTable)
title <- textGrob("Variable Contributions",
                  y = unit(0.5,"npc") + 0.92*h,
                  # fine tune the position of the title
                   just = "centre",
                   gp = gpar(fontsize = 14))
TableWithTitle <- gTree(children = gList(ctrTable, title))
```

```
# Note: Potential problems with grid.draw(). If it does not plot
# recordPlot() will fail and the graph will not be saved in the powerpoint
# and will generate a strange error message
grid.draw(TableWithTitle)
```

```
#a0000.2.ctrTable  <- recordPlot()
```

To create graphics we will color the different labels of a variable with the same color. First we create the vector storing the color with the function `data4PCCAR::coloringLevels`:

```
col4Levels <- data4PCCAR::coloringLevels(
                       rownames(resMCA$ExPosition.Data$fj), color4Var)
col4Labels <- col4Levels$color4Levels
```

As an alternative to the table of contributions, we can display the contributions with some bar plots. Note that these bar plots need some "tweaking" to correctly works.

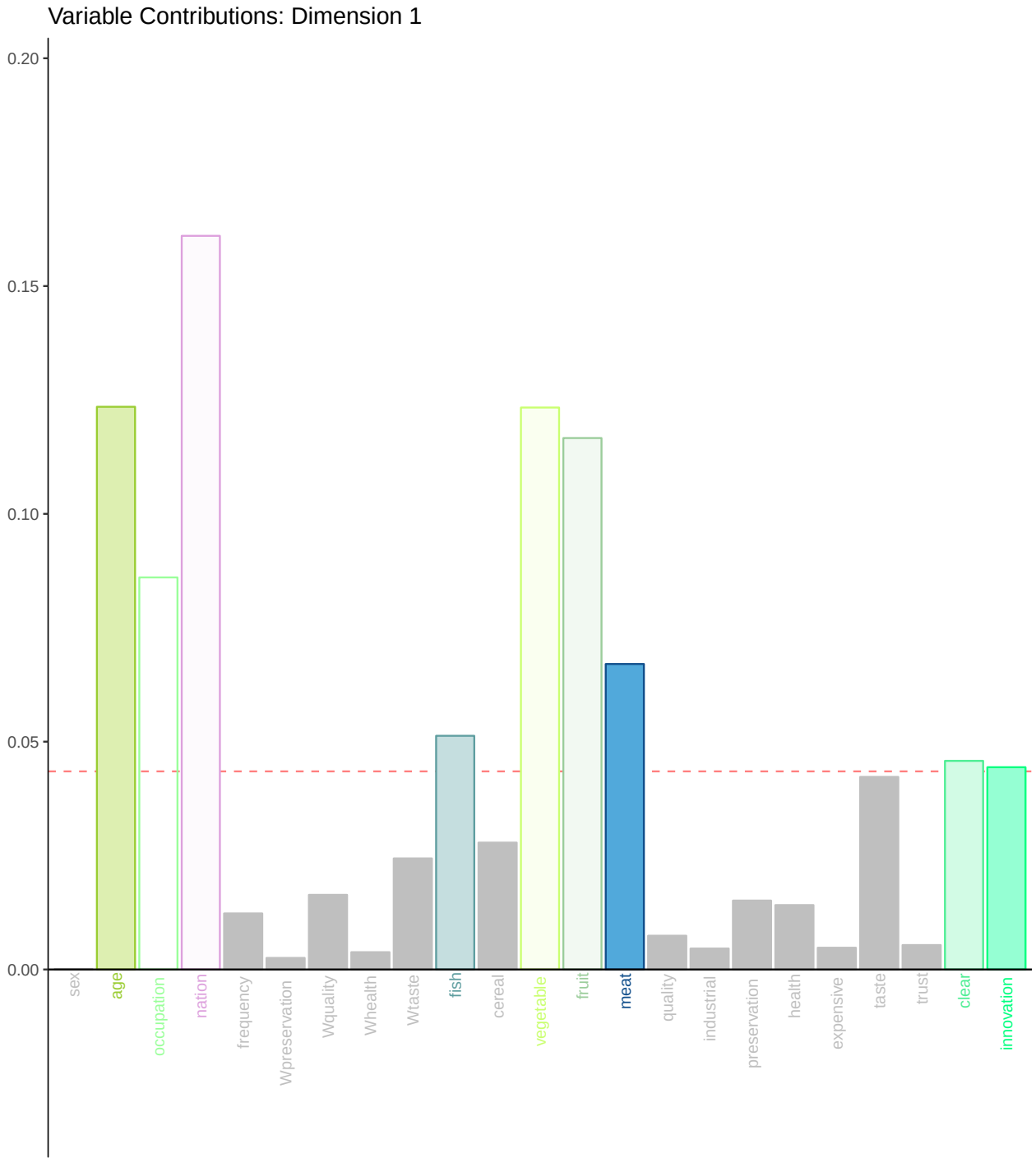
6.3.1 Variable contributions Dimension 1.

```
varCtr1 <- varCtr[,1]
names(varCtr1) <- rownames(varCtr)
a0005.Var.ctr1  <- PrettyBarPlot2(varCtr1,
                     main = 'Variable Contributions: Dimension 1',
                                 ylim = c(-.03, 1.2*max(varCtr1)),
                                 threshold = 1 / nrow(varCtr),
                                 color4bar = gplots::col2hex(color4Var)
)
print(a0005.Var.ctr1)
```

Variable Contributions

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimens
sex	0	1	10	4	189
age	124	1	24	67	0
ccupation	86	0	11	10	25
nation	161	2	43	45	1
requency	12	1	32	145	50
servation	3	0	9	28	28
Wquality	16	6	32	11	1
Whealth	4	3	5	40	5
Wtaste	24	0	5	23	168
fish	51	198	10	154	5
cereal	28	168	0	134	3
vegetable	123	187	5	36	3
fruit	117	210	10	12	11
meat	67	209	3	158	16
quality	7	1	123	14	24
industrial	5	0	2	1	179
servation	15	0	95	15	2
health	14	0	171	27	3
expensive	5	1	6	0	187
taste	42	1	115	67	7
trust	5	0	206	4	0
clear	46	7	53	0	30
nnovation	44	0	29	3	63

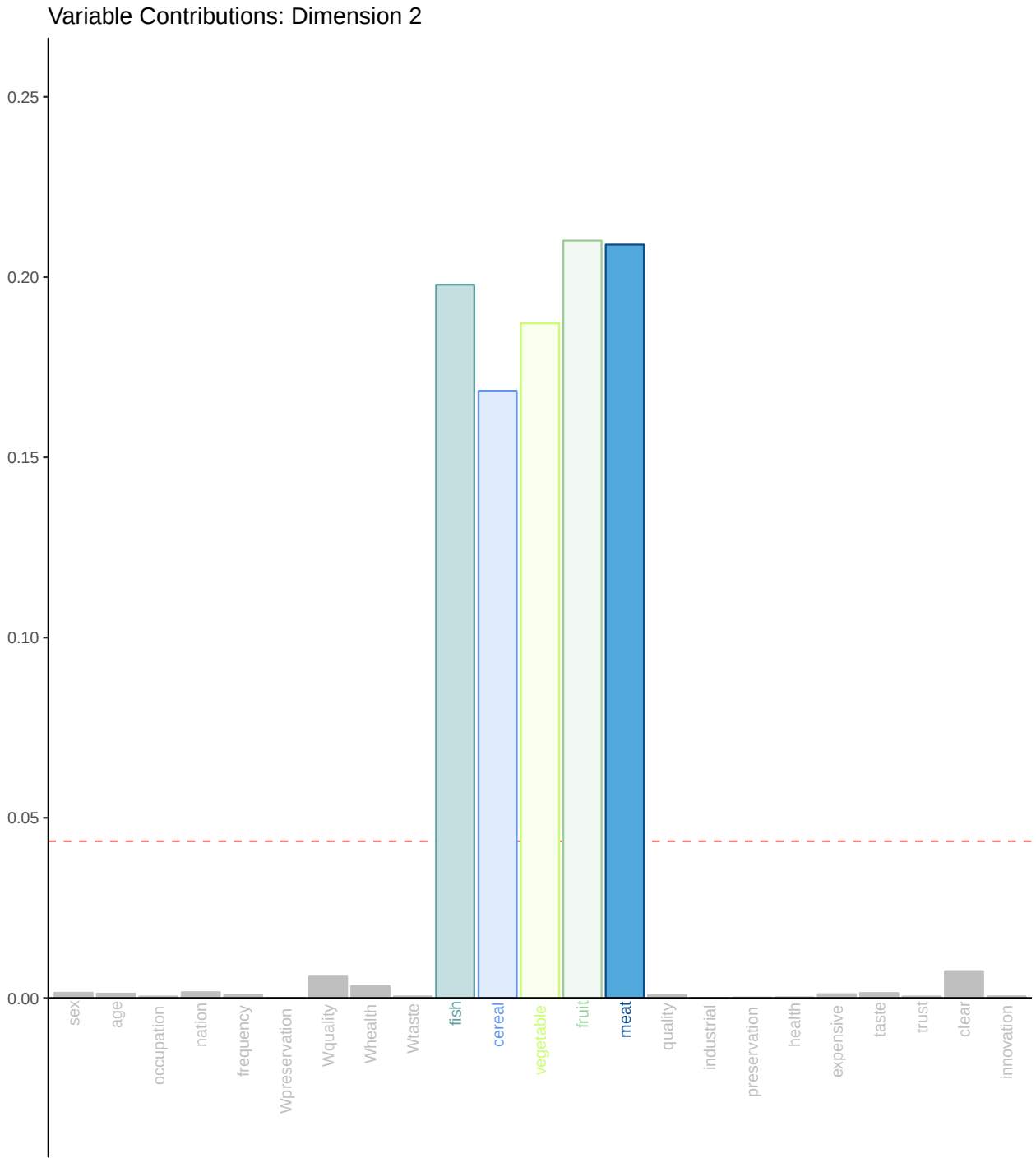
Figure 2: Variable Contributions (per mille).



6.3.2 Variable contributions Dimension 2.

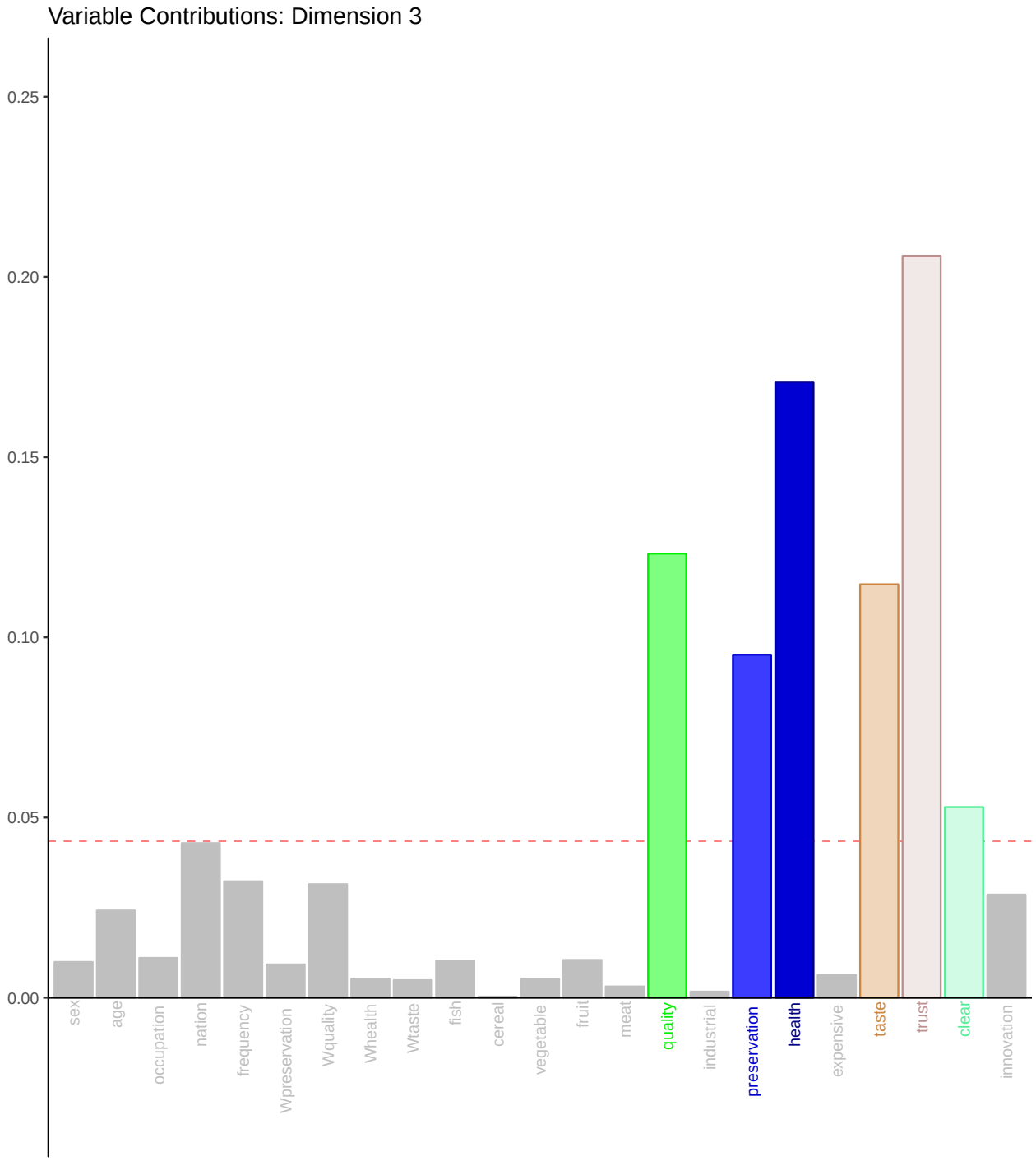
```
varCtr2 <- varCtr[,2]
names(varCtr2) <- rownames(varCtr)
a0006.Var.ctr2  <- PrettyBarPlot2(varCtr2,
                    main = 'Variable Contributions: Dimension 2',
                                ylim = c(-.03, 1.2*max(varCtr2)),
```

```
                                threshold = 1 / nrow(varCtr),
                                color4bar = gplots::col2hex(color4Var)
)
print(a0006.Var.ctr2)
```



6.3.3 Variable contributions Dimension 3.

```
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(color4Var)
)
print(a0006.Var.ctr3)
```

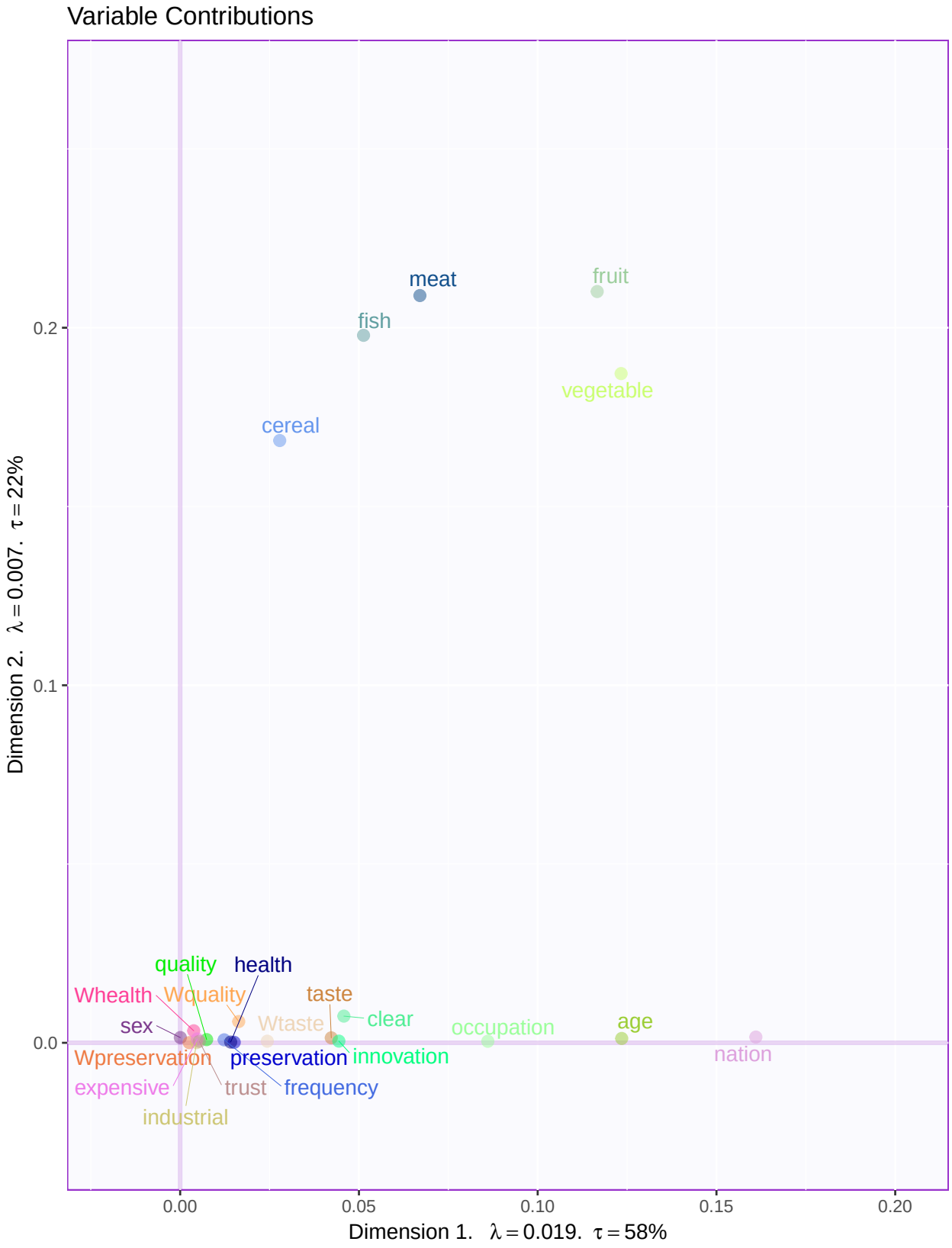


6.3.3.1 Pseudo factor plots with variable contributions

Another way to visualize the variable contributions is to use a pseudo-factor plot with the contributions:

```
ctrV12 <- PTCA4CATA::createFactorMap(X =  varCtr,
                        title = "Variable Contributions",
                        col.points = color4Var,
                        col.labels = color4Var,
                        alpha.points = 0.5,
```

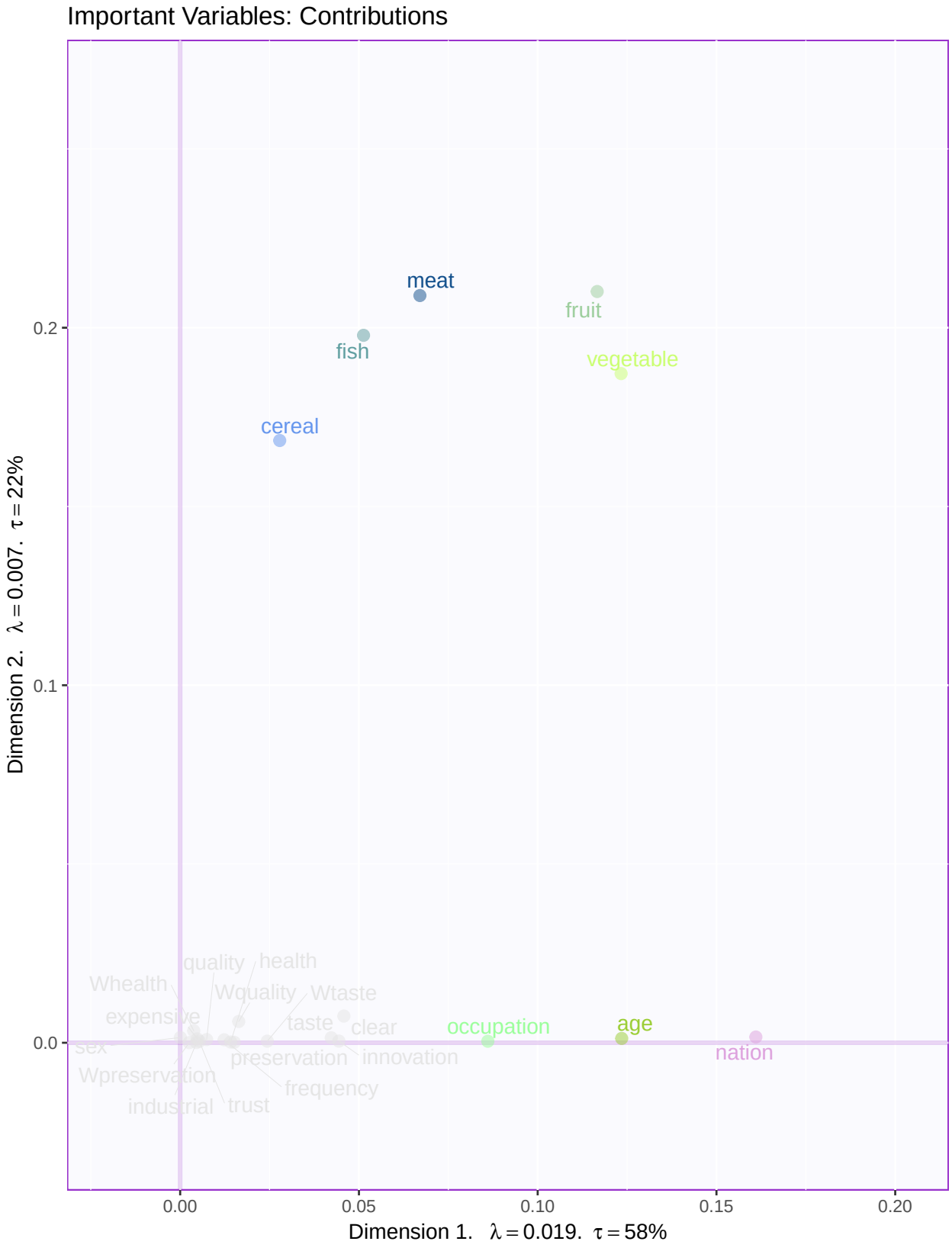
```
                        cex = 2.5,
                        alpha.labels = 1,
                        text.cex = 4,
                        font.face = "plain",
                        font.family = "sans")

ctr.labels <- createxyLabels.gen(
  1,2, lambda = resMCA$ExPosition.Data$eigs,
  tau = resMCA$ExPosition.Data$t
)
a0007.Var.ctr12  <- ctrV12$zeMap  + ctr.labels
#
print(a0007.Var.ctr12)
```



Add a contribution plot with important variables only

```
absCtrVar <- as.matrix(varCtr) %*% diag(resMCA$ExPosition.Data$eigs)
varCtr12  <- (absCtrVar[,1] + absCtrVar[,2]) /
   (resMCA$ExPosition.Data$eigs[1] + resMCA$ExPosition.Data$eigs[2])
importantVar <- (varCtr12 >=  1 / length(varCtr12))
col4ImportantVar <- color4Var
col4NS <- 'gray90'
col4ImportantVar[!importantVar] <- col4NS
```

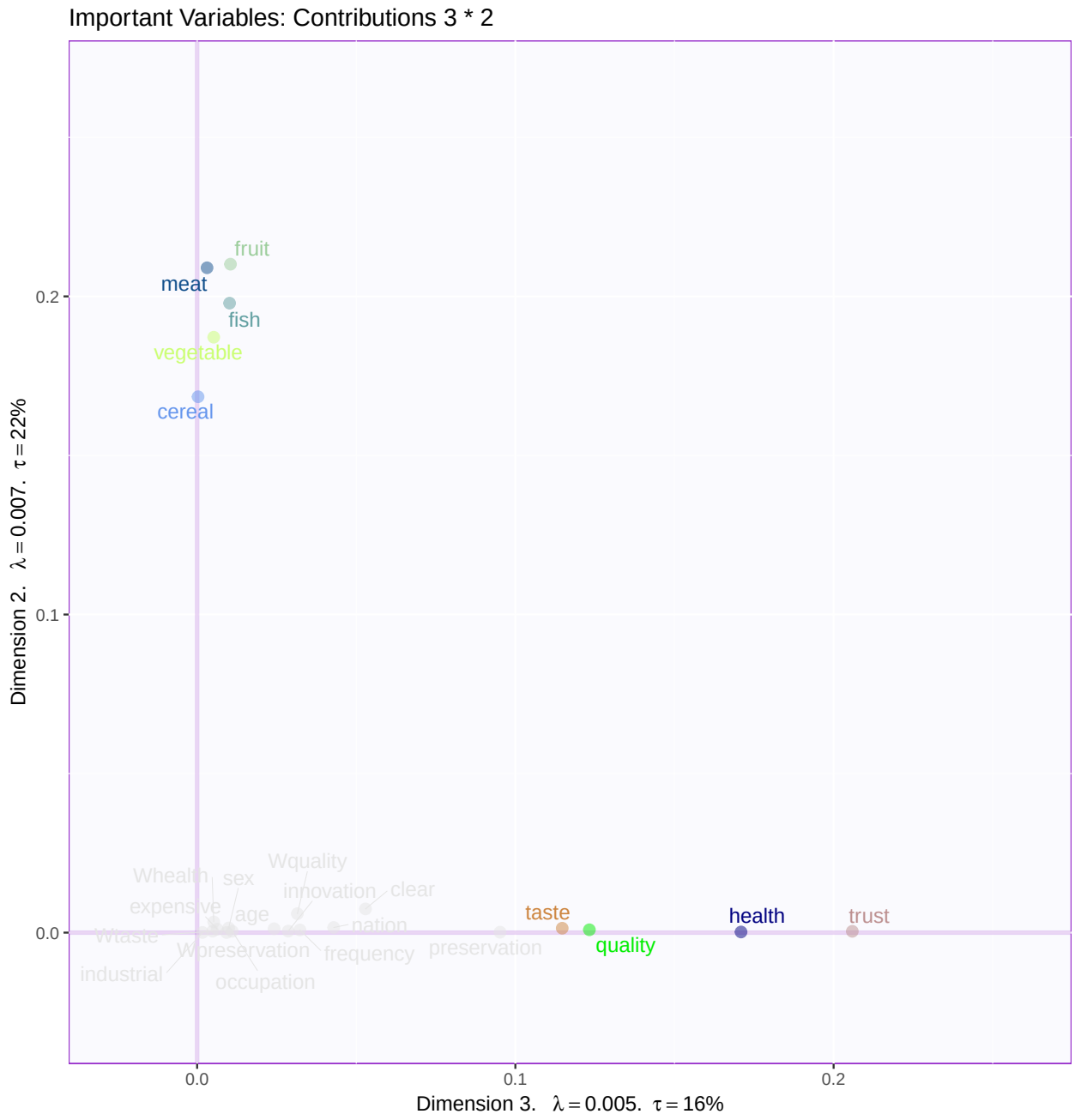
```
ctrV12.imp <- PTCA4CATA::createFactorMap(X =  varCtr,
                         title = "Important Variables: Contributions",
                         col.points = col4ImportantVar,
                         col.labels = col4ImportantVar,
                         alpha.points = 0.5,
                         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)
```



6.3.3.2 A map with Dimension 2 & 3

```r
#absCtrVar <- as.matrix(varCtr) %*% diag(resMCA$ExPosition.Data$eigs)
varCtr23  <- (absCtrVar[,3] + absCtrVar[,2]) /
   (resMCA$ExPosition.Data$eigs[3] + resMCA$ExPosition.Data$eigs[2])
importantVar23 <- (varCtr23 >=  1 / length(varCtr23))
col4ImportantVar23 <- color4Var
col4NS <- 'gray90'
col4ImportantVar23[!importantVar23] <- col4NS
```

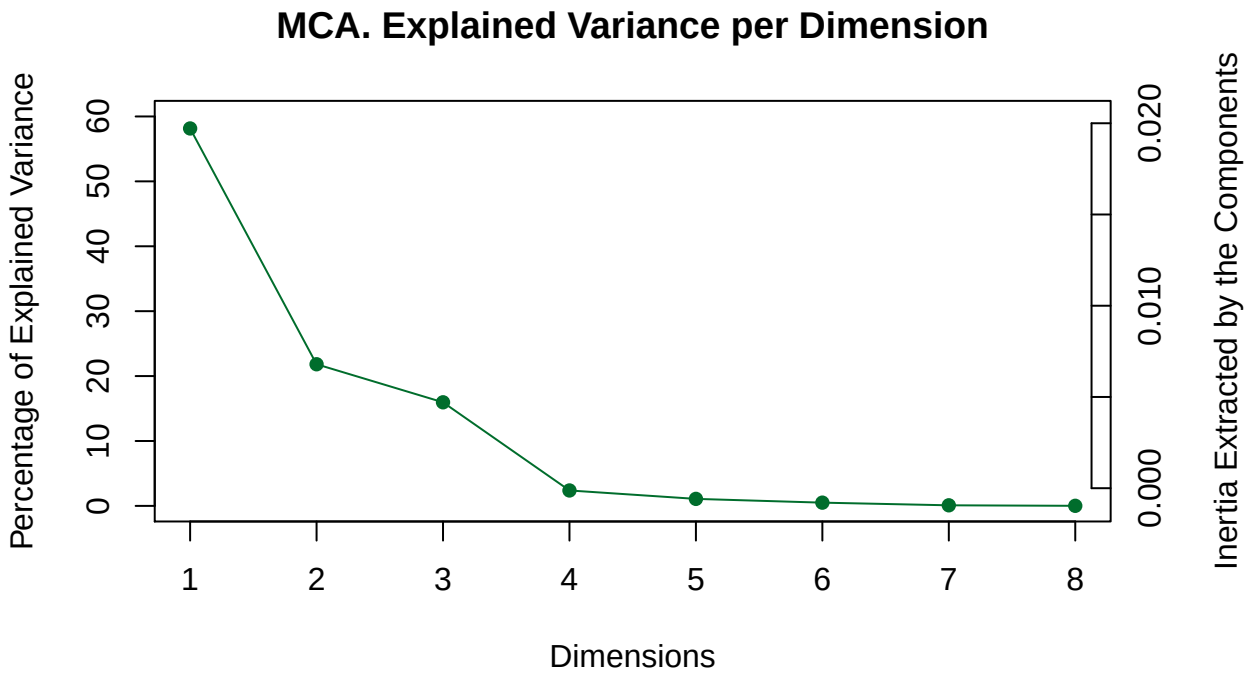
```r
ctrV23.imp <- PTCA4CATA::createFactorMap(X =  varCtr,
                                         axis1 = 3, axis2 = 2,
                         title = "Important Variables: Contributions 3 * 2",
                         col.points = col4ImportantVar23,
                         col.labels = col4ImportantVar23,
                         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$ExPosition.Data$eigs,
  tau = resMCA$ExPosition.Data$t
)
a0009.Var.ctr23.imp  <- ctrV23.imp$zeMap  + ctr.labels23
#
print(a0009.Var.ctr23.imp)
```



6.4 Scree for MCA

6.4.1 Plain Scree

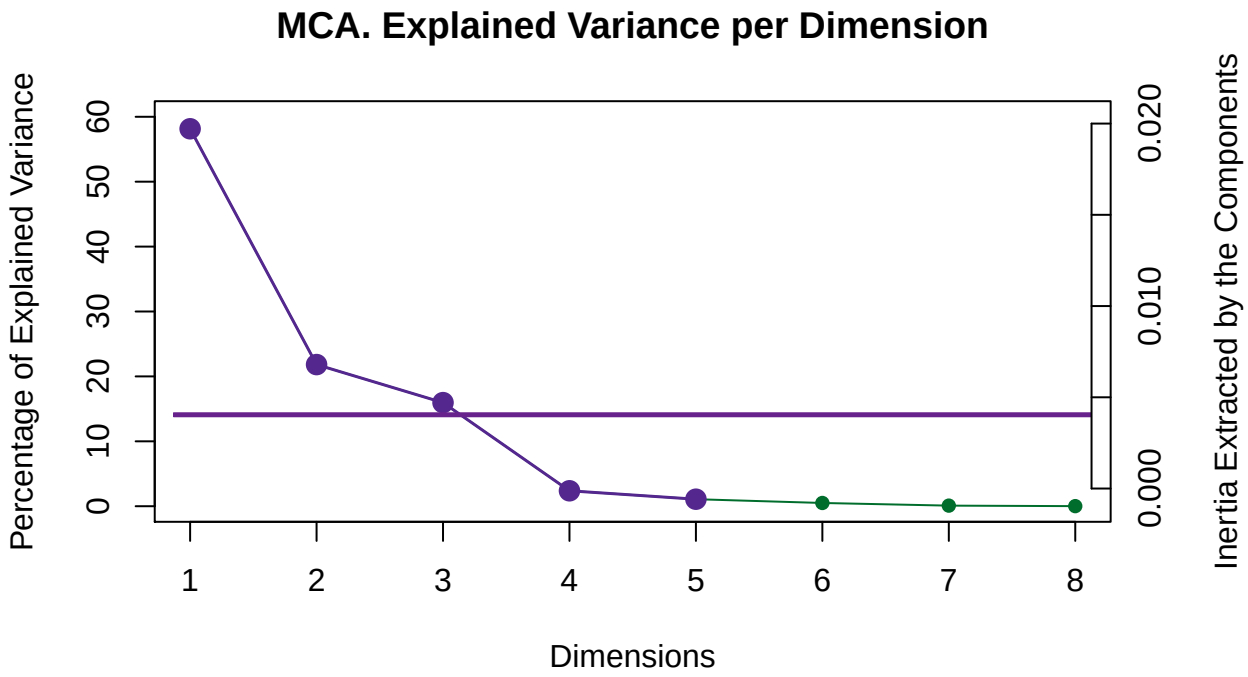
```
scree.mca <- PlotScree(ev = resMCA$ExPosition.Data$eigs,
                       title = "MCA. Explained Variance per Dimension")
```




```
b0001a.Scree <- recordPlot() # Save the plot
```

6.4.2 Scree with inference

```
scree.mca <- PlotScree(ev = resMCA$ExPosition.Data$eigs,
               p.ev = resMCA.inf$Inference.Data$components$p.vals,
               plotKaiser = TRUE,
               title = "MCA. Explained Variance per Dimension")
```

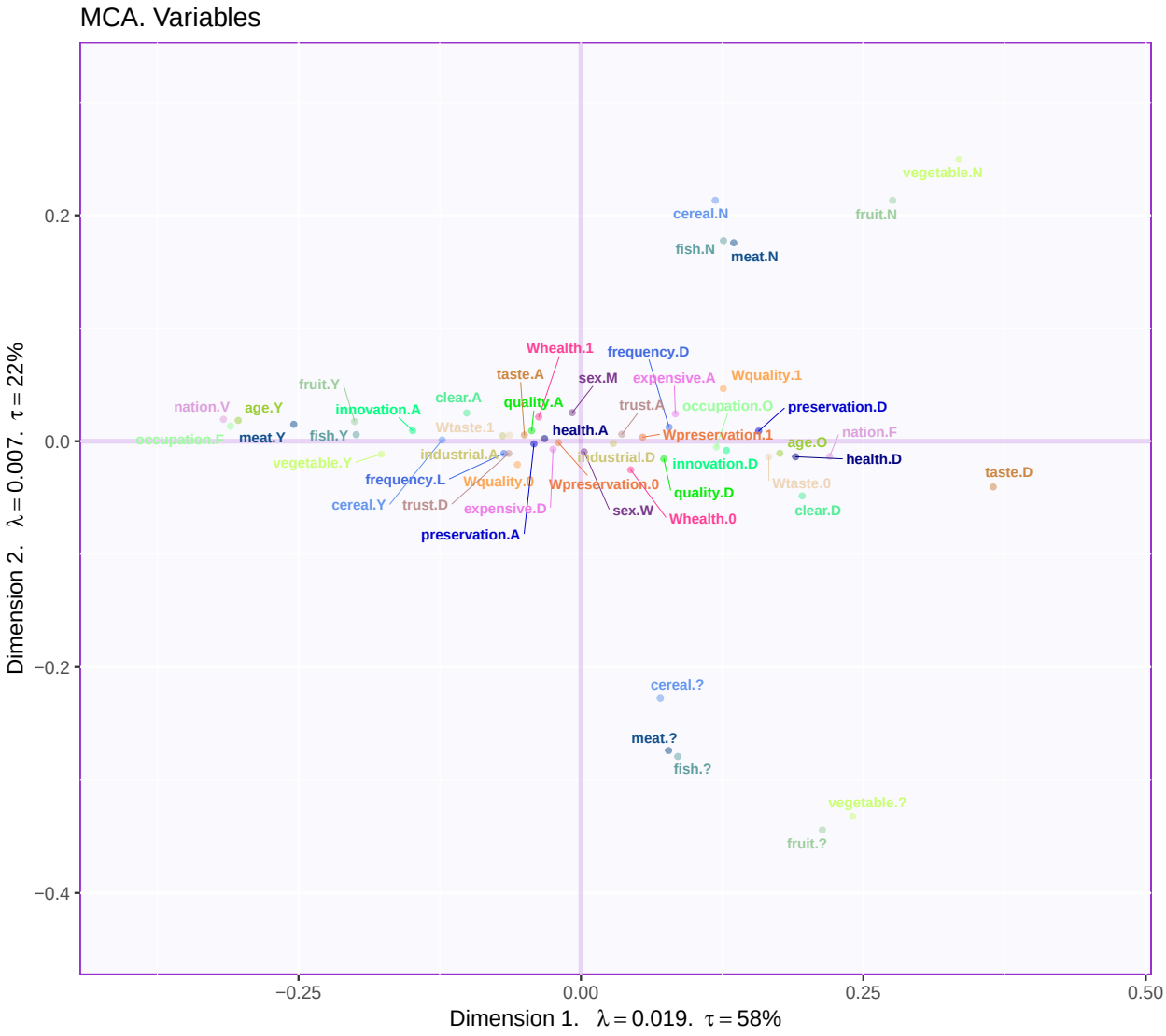


```
b0001b.Scree <- recordPlot() # Save the plot
```

6.5 Variable Map

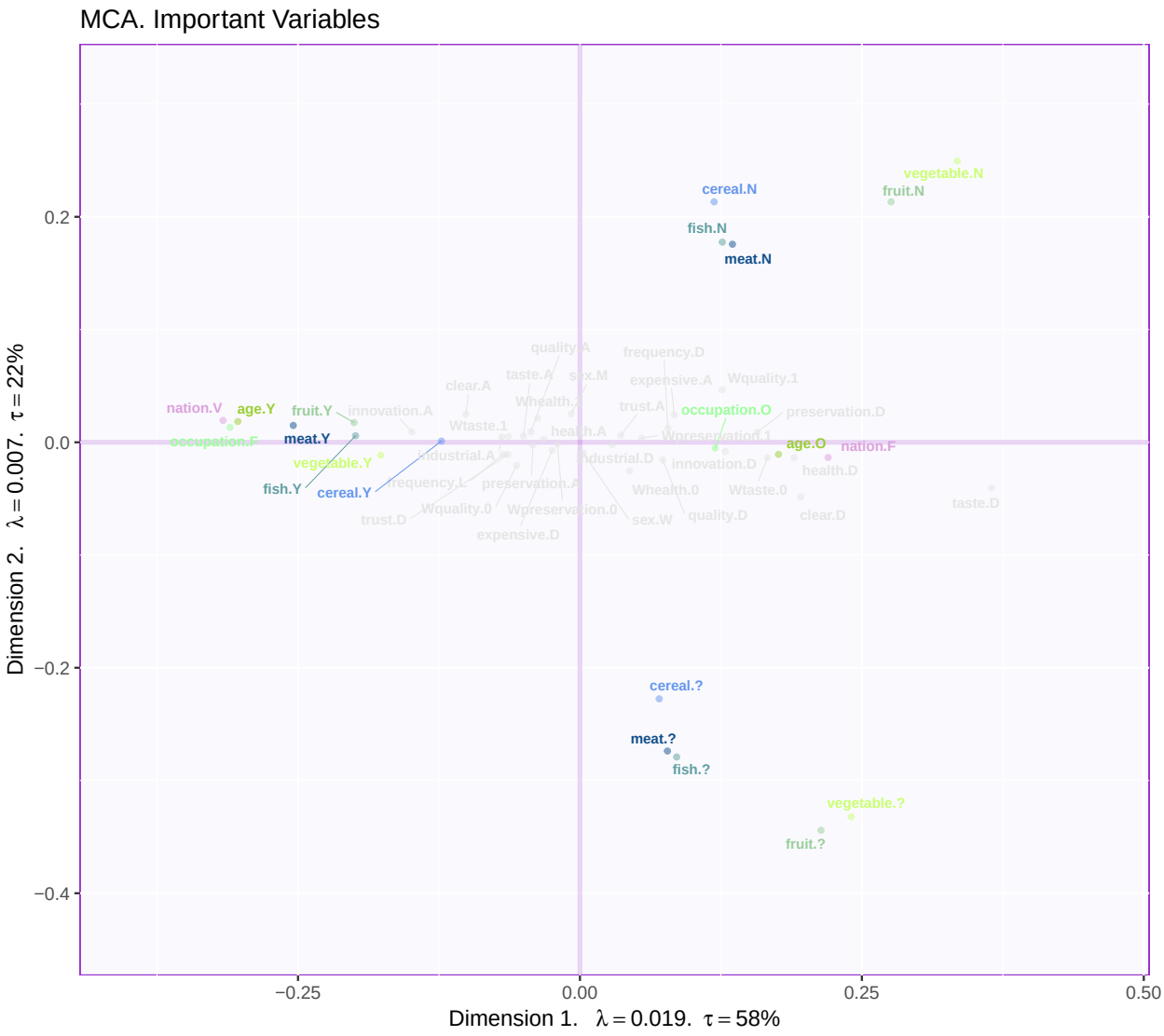
```
axis1 = 1
axis2 = 2
Fj <- resMCA$ExPosition.Data$fj
# generate the set of maps
BaseMap.Fj <- createFactorMap(X = Fj , # resMCA$ExPosition.Data$fj,
                              axis1 = axis1, axis2 = axis2,
                              title = 'MCA. Variables',
                              col.points = col4Labels, cex = 1,
                              col.labels = col4Labels, text.cex = 2.5,
                              force = 2)
# add labels
labels4MCA <- createxyLabels.gen(x_axis = axis1, y_axis = axis2,
               lambda = resMCA$ExPosition.Data$eigs,
               tau = resMCA$ExPosition.Data$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)
```



6.5.1 Variable Map with only important variables

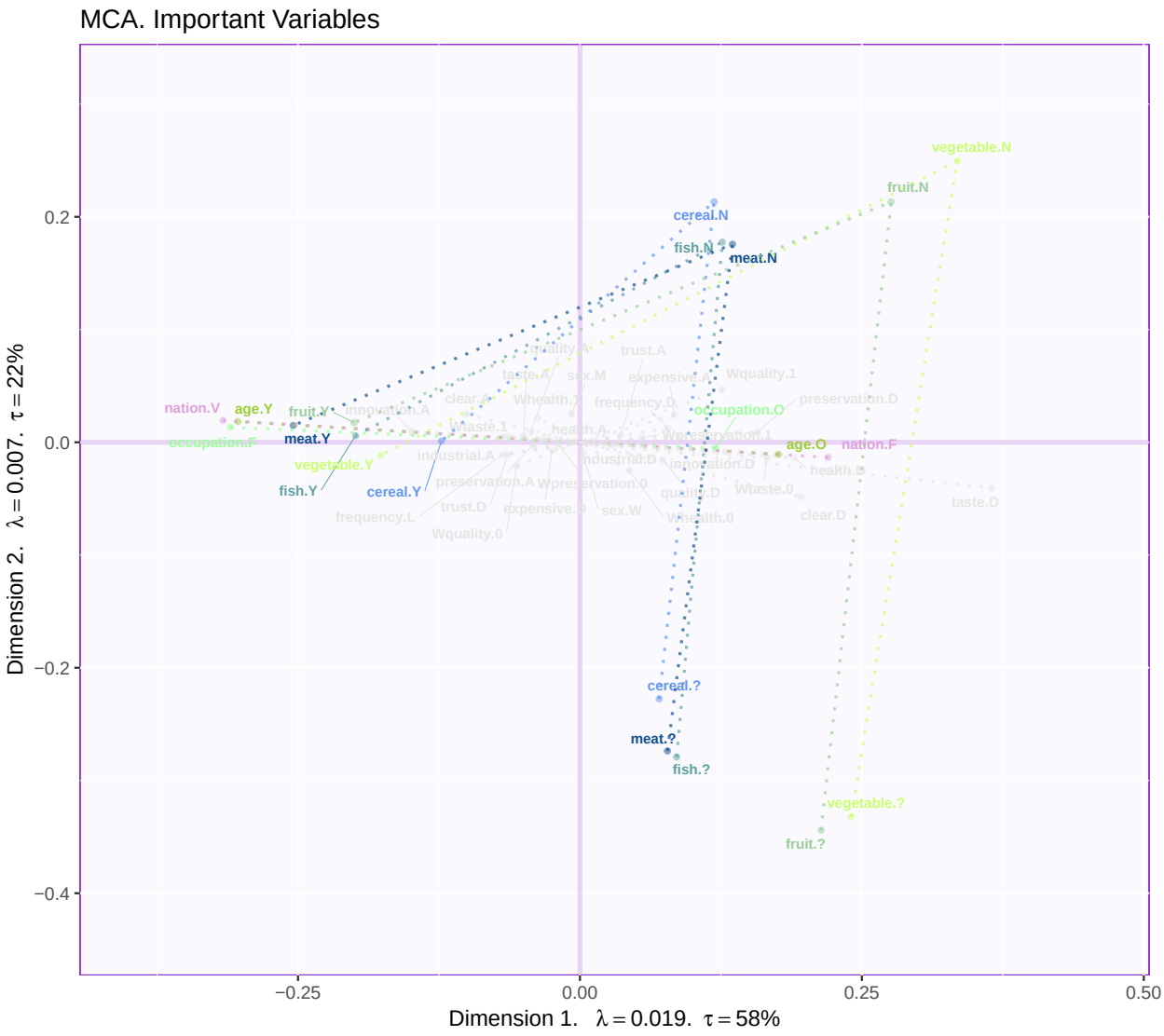
```
col4Levels.imp <- data4PCCAR::coloringLevels(rownames(Fj),
                                             col4ImportantVar)
BaseMap.Fj.imp <- createFactorMap(X = Fj , # resMCA$ExPosition.Data$fj,
                                  axis1 = axis1, axis2 = axis2,
                                  title = '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)
```



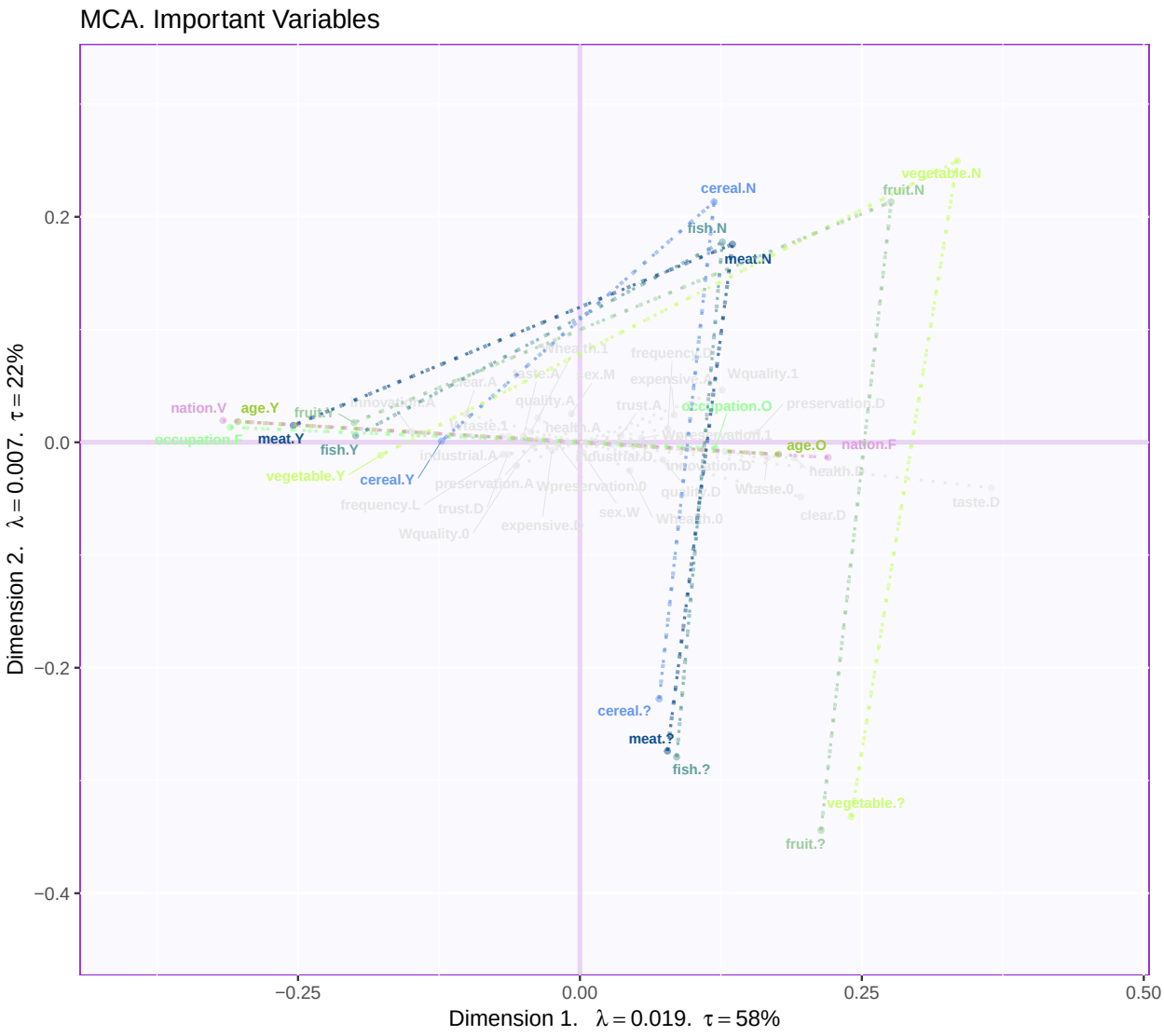
6.5.2 Map with important variables and lines

The MCA map for the variables is easier to read when the levels of the qualitative variables are linked to each other with lines as show below

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




```
zeNames          <- getVarNames(rownames(Fj))
importantsLabels <- zeNames$stripedNames %in% zeNames$variableNames[importantVar]
Fj.imp <- Fj[importantsLabels,]
lines4J.imp <- addLines4MCA(Fj.imp,
                            col4Var = col4Levels$color4Variables[which(importantVar)],
                            size = .9, linetype = 3, alpha = .5)
 b0021.BaseMap.Fj <-  b0020.BaseMap.Fj + lines4J.imp
 print( b0021.BaseMap.Fj)
```



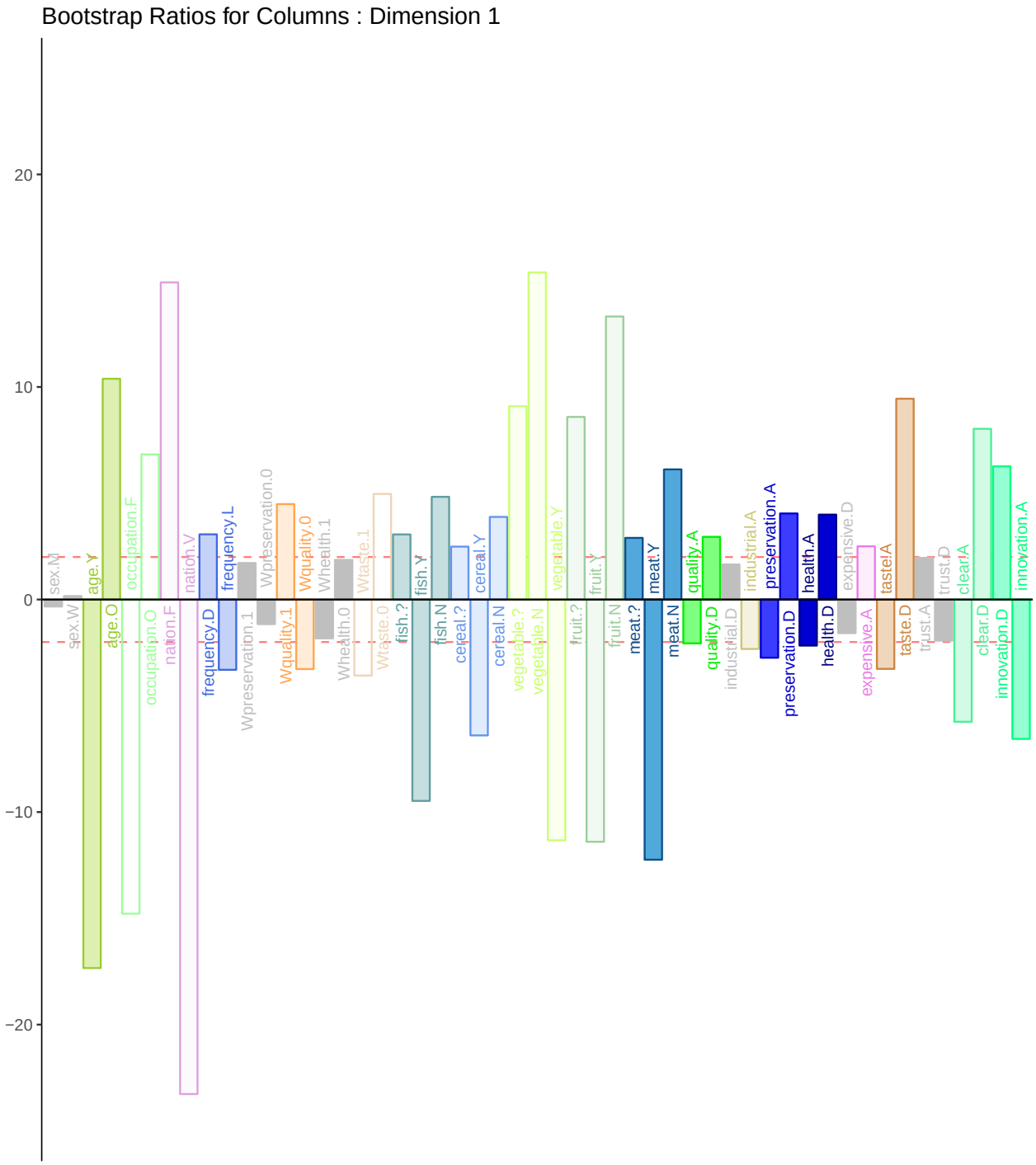
6.5.3 Map with important variables and lines dimension 2 & 3

Have a look at the map for Dimensions 2 and 3.

6.5.4 Bootstrap ratios

Just like in PCA, the significance of each column (i.e., level of a qualitative variable) can be assessed with a bootstrap rations. For example, the bootstrap ratios for all levels of all qualitative variables for Dimension 1 are plotted as

```
c0001.Levels.BR  <- PrettyBarPlot2(
      resMCA.inf$Inference.Data$fj.boots$tests$boot.ratios[,1], # BR
                  main = 'Bootstrap Ratios for Columns : Dimension 1',
                           threshold = 2,
                           color4bar = gplots::col2hex(col4Labels)
)
print(c0001.Levels.BR)
```



6.5.5 Qualitative variables pseudo-F

If we assume that all levels of the variable are balanced Pseudo-F ratios for the K qualitative variables can also be obtained from the bootstrap ratios with the formula:

$$F_j = \frac{J_k}{J_k - 1} \sum_{j}^{J_k} t_k^2$$

where J_k (respectively t_k) is the number of levels (respectively bootstrap ratio) for the k-th qualitative variable.

If we want to take into account the frequency of the different levels of a qualitative variable, and if we denote by $w_{j(k)}$ the frequency of level j of the k-th variable, the pseudo-F ratios can be computed as:

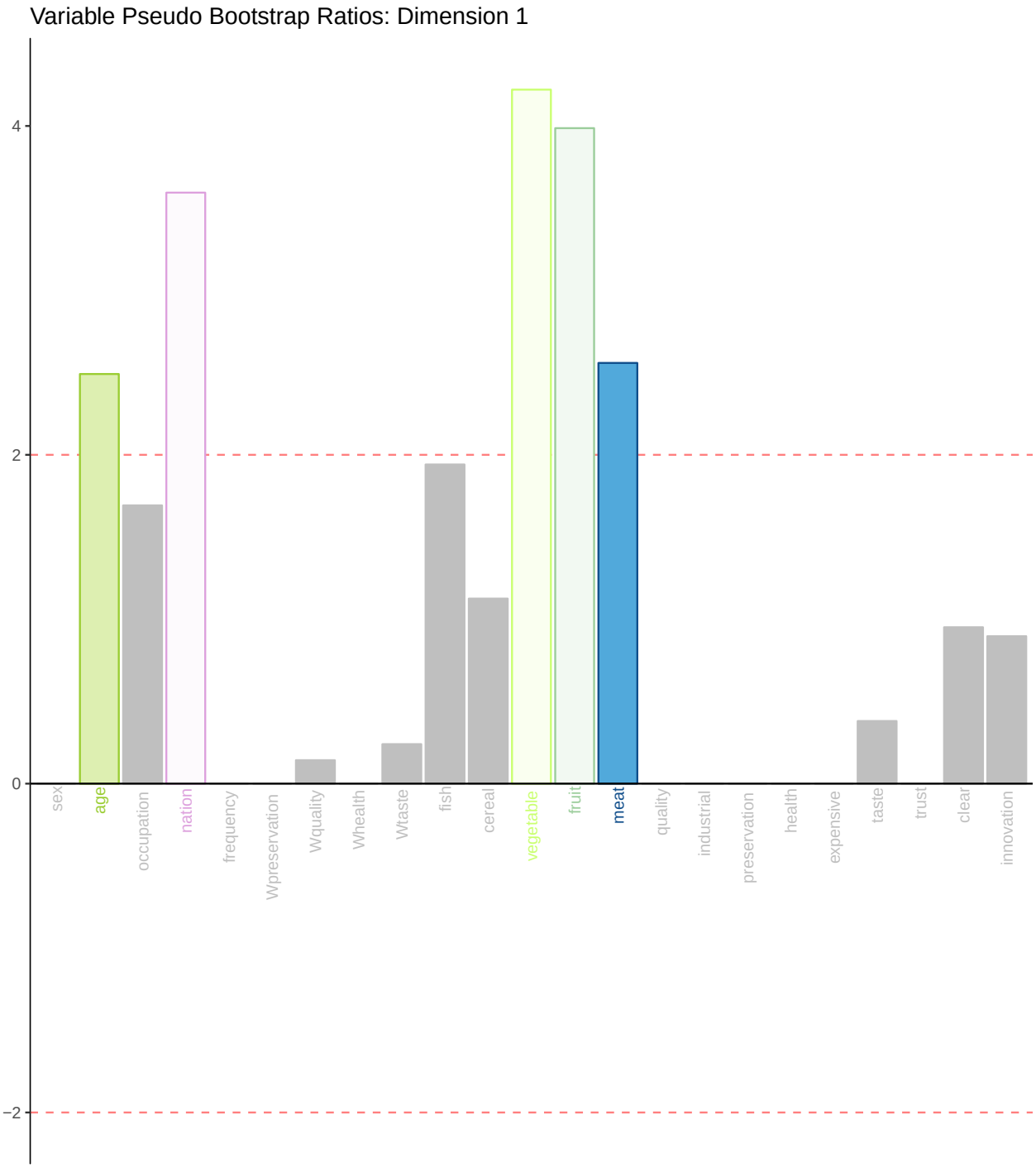
$$F_k = \frac{1}{J_k - 1} \sum_{j}^{J_k} w_{j(k)} t^2.$$

These pseudo-F ratios evaluates if a dimension of the MCA creates a reliable difference between the levels of a qualitative variable; by contrast, a column bootstrap ratio evaluates if this column systematically loads on the same side of a dimension.

```
# Get the pseudo Bootstrqp Rqtios
BrLevels <- resMCA.inf$Inference.Data$fj.boots$tests$boot.ratios
wJ       <- 1 / resMCA.inf$Fixed.Data$ExPosition.Data$W
nIter    <- 1000
Br4Variables <- data4PCCAR::BR4varMCA(BrLevels, wJ, nIter)
```

6.5.5.1 Pseudo-BR Dimension 1

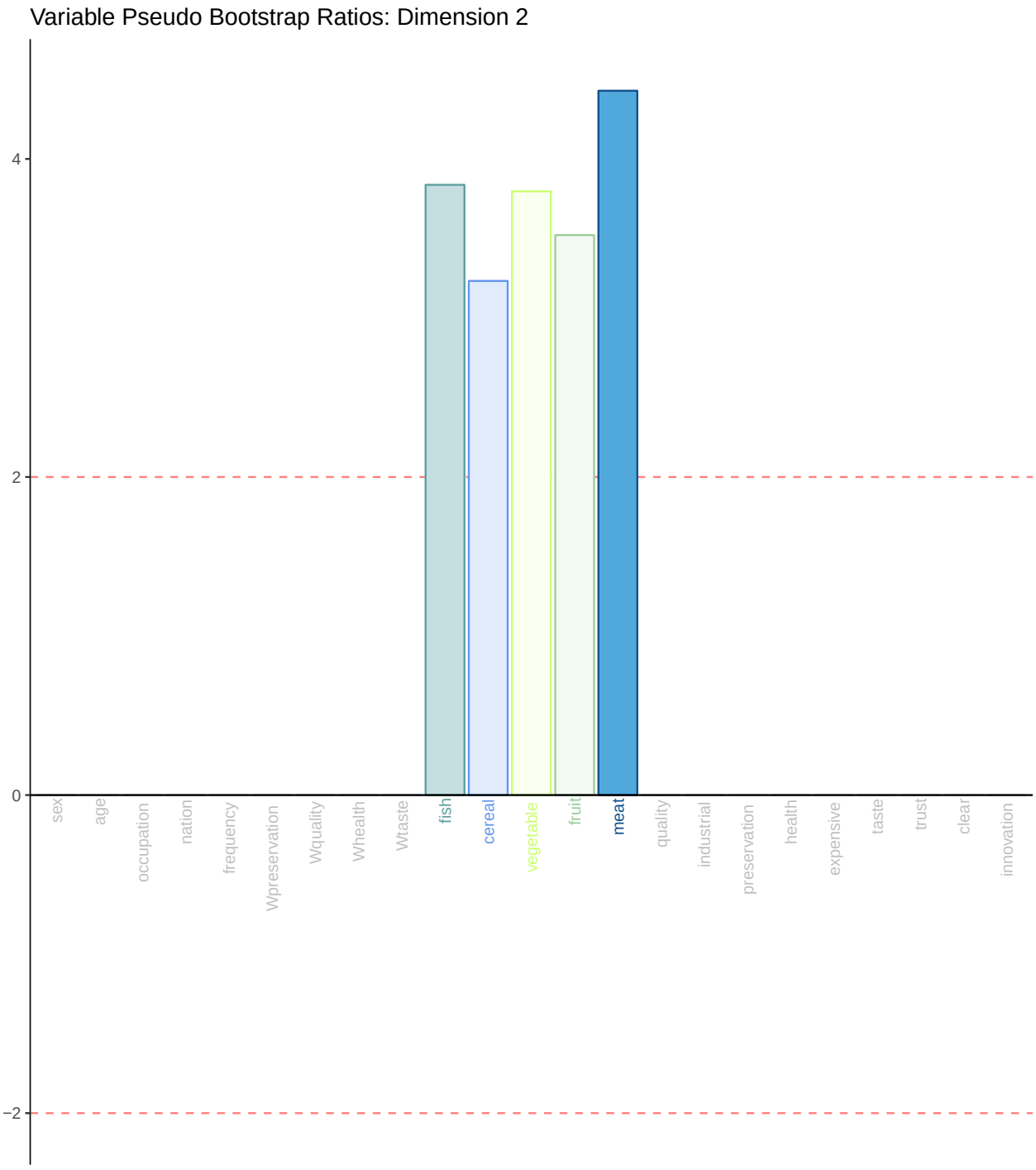
```
VarBR1 <- Br4Variables$pseudoBR.pos[,1]
c0010.Var.br1  <- PrettyBarPlot2(VarBR1,
                    main = 'Variable Pseudo Bootstrap Ratios: Dimension 1',
                              ylim = 2,
                               threshold = 2,
                               color4bar = gplots::col2hex(color4Var)
)
print(c0010.Var.br1)
```



6.5.5.2 Pseudo-BR Dimension 2

```
VarBR2 <- Br4Variables$pseudoBR.pos[,2]
c0011.Var.br2  <- PrettyBarPlot2(VarBR2,
                    main = 'Variable Pseudo Bootstrap Ratios: Dimension 2',
                              ylim = 2,
                               threshold = 2,
                               color4bar = gplots::col2hex(color4Var)
)
```

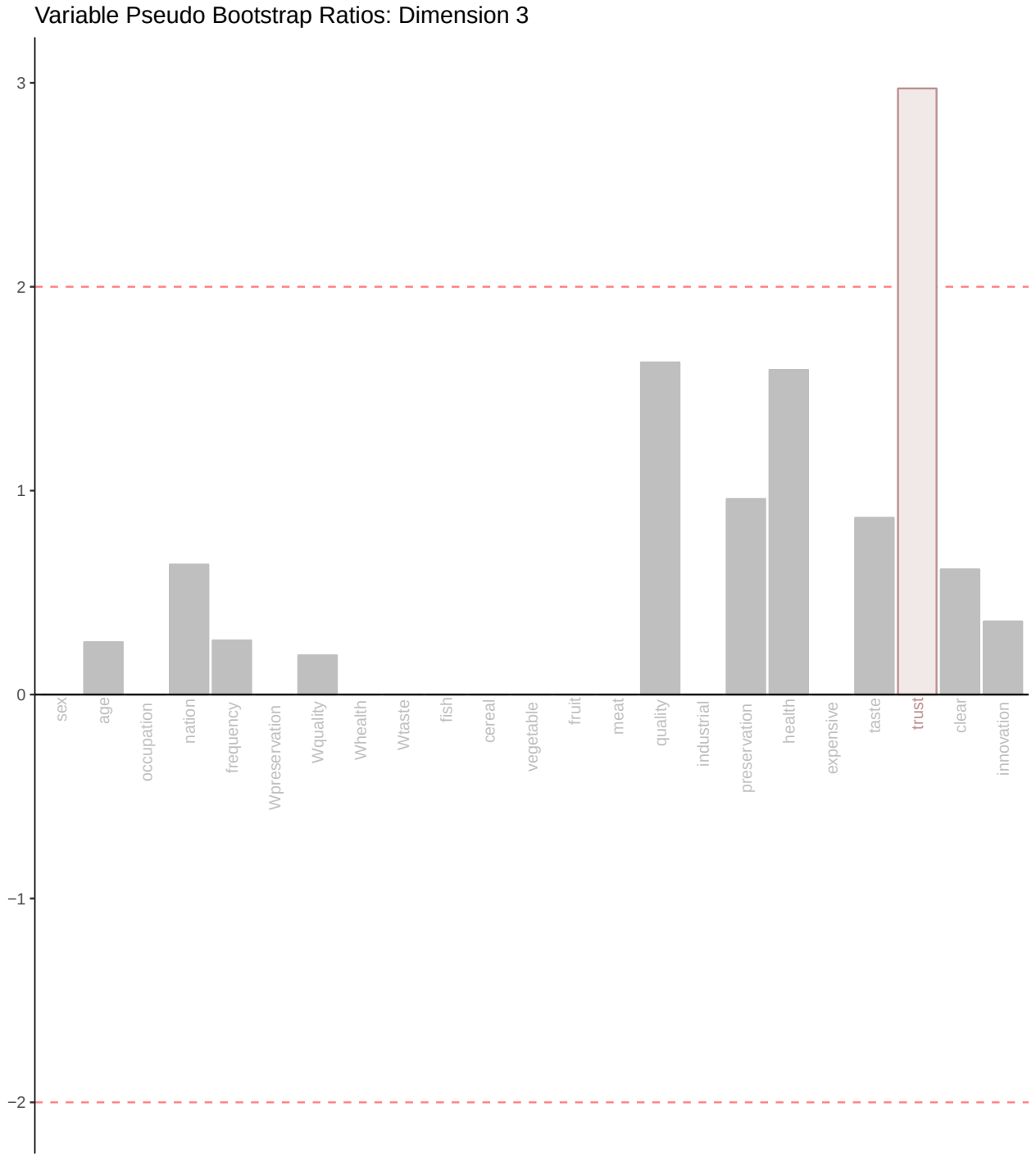
```
print(c0011.Var.br2)
```



6.5.5.3 Pseudo-BR Dimension 3

```
VarBR3 <- Br4Variables$pseudoBR.pos[,3]
c0012.Var.br3  <- PrettyBarPlot2(VarBR3,
                  main = 'Variable Pseudo Bootstrap Ratios: Dimension 3',
                            ylim = 2,
```

```
                                threshold = 2,
                                color4bar = gplots::col2hex(color4Var)
)
print(c0012.Var.br3)
```



6.5.6 Pseudo-Bootstrapped Map for Levels of Variables

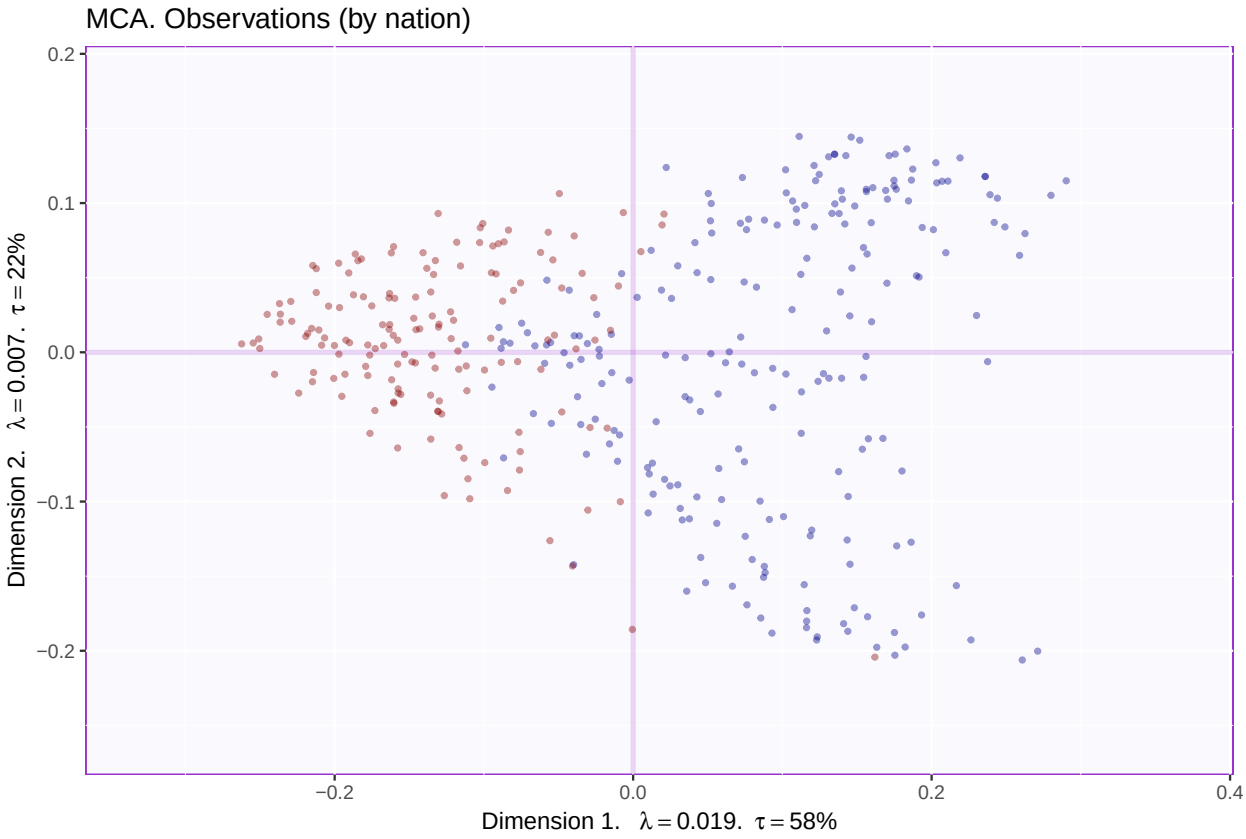
In a similar way to the graphs of the Levels of the qualitative variables based on contributions we can also plot graphs of the Levels of the qualitative variables based on the bootstrap ratios, These will look very similar, with the exception of the variable `occupation`, which has an important contribution but a non-significant pseudo bootstrap ratio.

6.6 Map for the observations

Here we will color the observations by `nation,` to evaluate if this variable is an important source of variation for this data set.

```
Fi <- resMCA$ExPosition.Data$fi
colCity <- c('darkblue', 'red4')
nI <- nrow(Fi)
col4I.City <- rep("",nI)

for (i in 1:length(colCity) ){
  lindex <- cleanData[,'nation'] %in% unique(cleanData[,'nation'])[i]
  col4I.City[lindex] <- colCity[i]
}
# generate the set of maps
BaseMap.Fi <- createFactorMap(X = Fi , # resMCA$ExPosition.Data$fj,
                              axis1 = axis1, axis2 = axis2,
                              title = 'MCA. Observations (by nation)',
                              col.points = col4I.City,
                              alpha.points = .4, cex = .9,
                              col.labels = col4I.City,
                              text.cex = 2.5,
                              force = 2)
# make the maps
d0001.BaseMapNoLabels.Fi  <- BaseMap.Fi$zeMap_background +
                                  BaseMap.Fi$zeMap_dots + labels4MCA
```

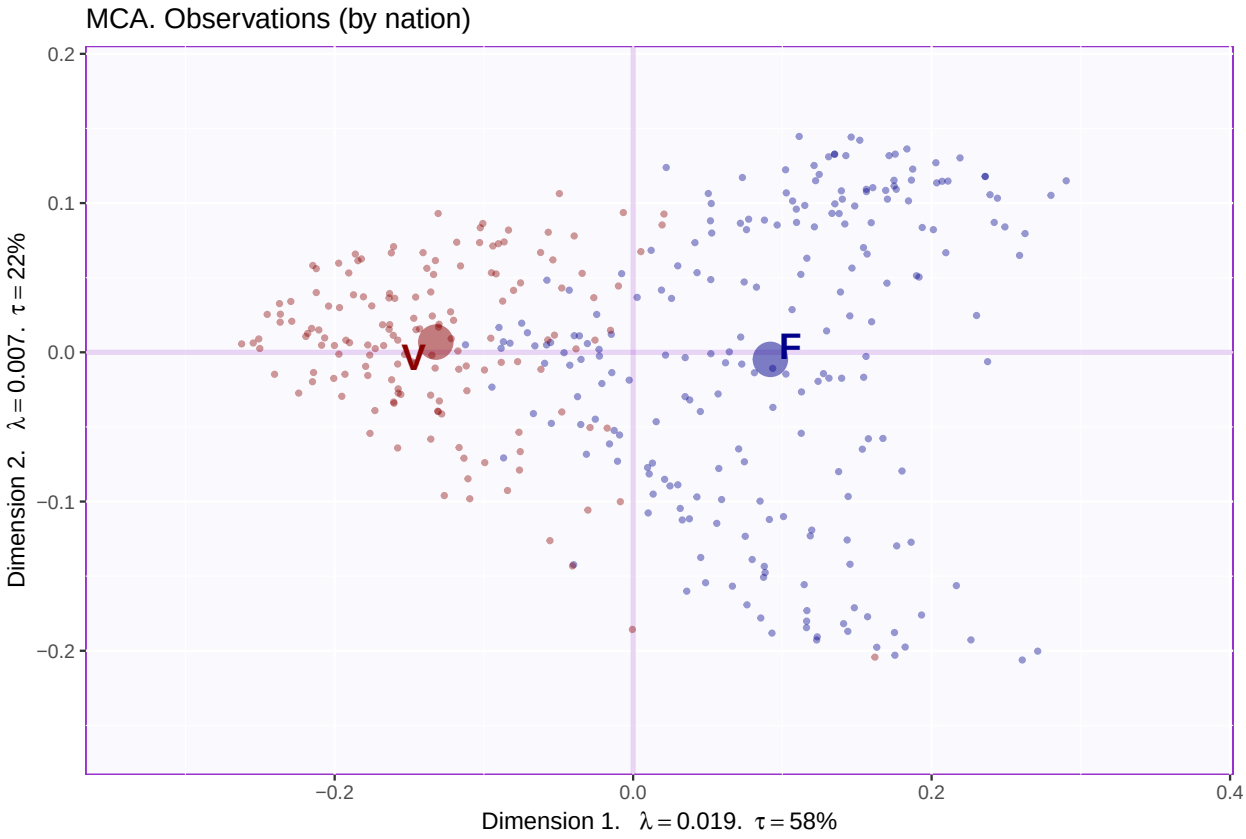
```
print(d0001.BaseMapNoLabels.Fi)
```



6.7 Add Group Means, Confidence, and Tolerance Intervals

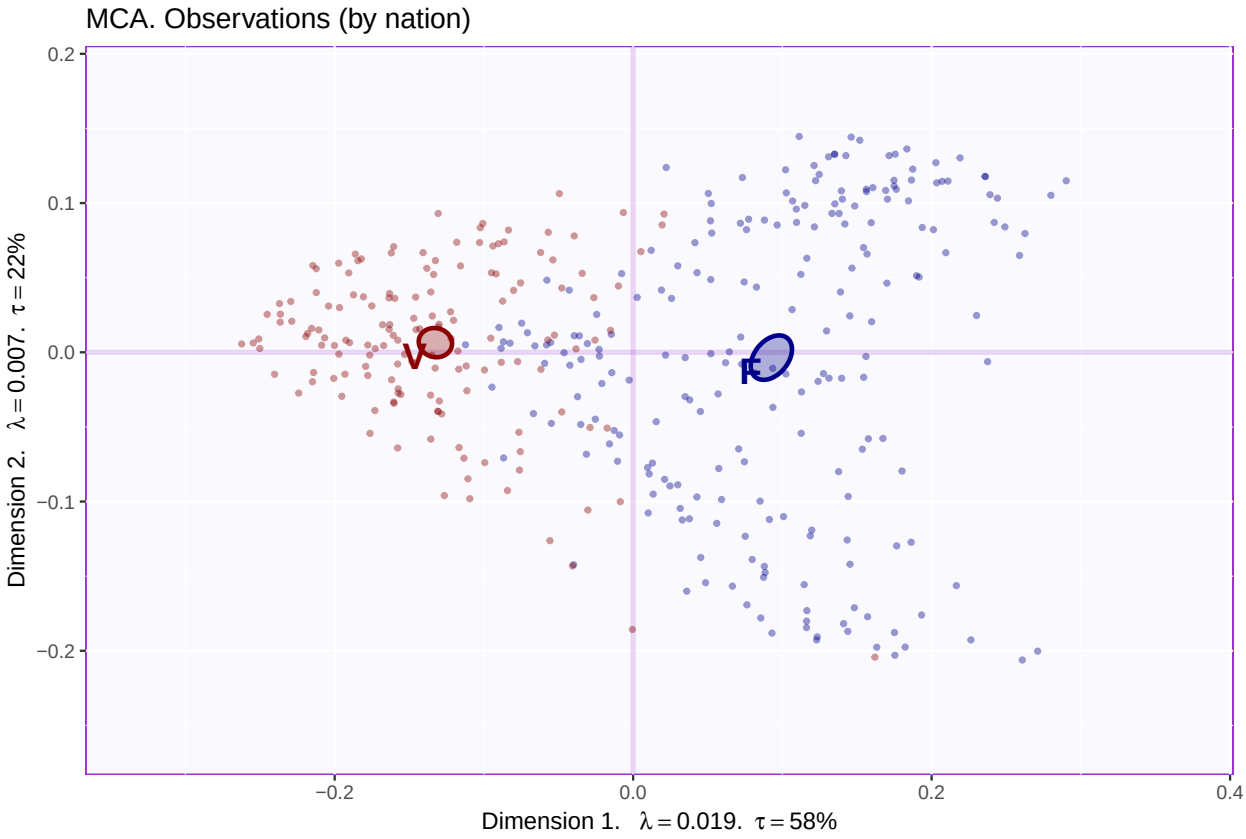
6.7.1 Group Means

```
# Bootstrap for CI:
BootCube.Gr <- PTCA4CATA::Boot4Mean(resMCA$ExPosition.Data$fi,
                                    design = cleanData$nation,
                                    niter = 100,
                                    suppressProgressBar = TRUE)
nationsMeans <- PTCA4CATA::getMeans(resMCA$ExPosition.Data$fi, cleanData$nation)
# colCity <- c('darkblue', 'red4')
MapGroup <- PTCA4CATA::createFactorMap(nationsMeans,
                               # use the constraint from the main map
                               constraints = BaseMap.Fi$constraints,
                               col.points = colCity,
                               cex = 7,  # size of the dot (bigger)
                               col.labels = colCity,
                               text.cex = 6)
d002.Map.I.withMeans <- d0001.BaseMapNoLabels.Fi  +
                           MapGroup$zeMap_dots + MapGroup$zeMap_text
print(d002.Map.I.withMeans)
```



6.7.2 With bootstrapped confidence intervals

```
GraphElli <- PTCA4CATA::MakeCIEllipses(BootCube.Gr$BootCube[,1:2,],
                            names.of.factors = c("Dimension 1","Dimension 2"),
                            col = colCity,
                            p.level = .95)
d003.Map.I.withCI <-  d0001.BaseMapNoLabels.Fi +
                          MapGroup$zeMap_text +  GraphElli
print(d003.Map.I.withCI)
```



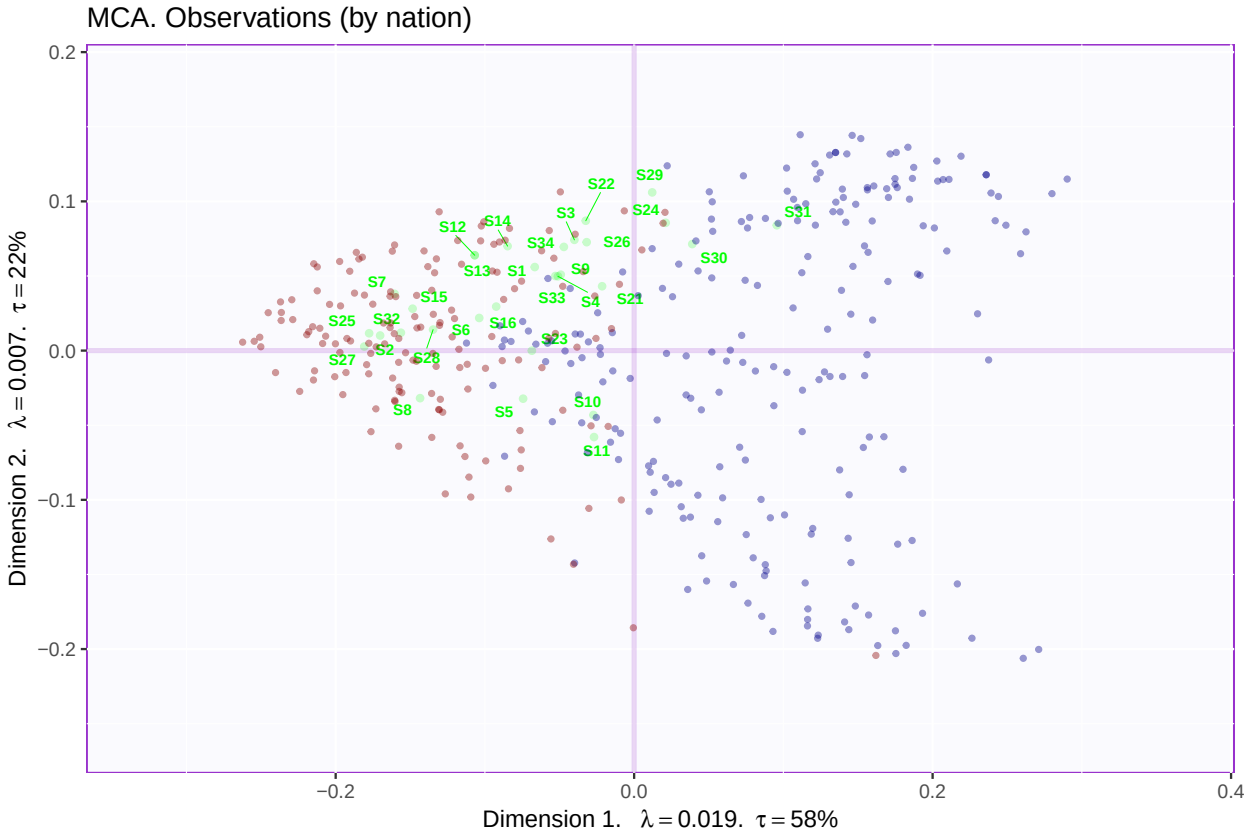
6.7.3 With Tolerance convex hulls

```
GraphTI.Hull <- PTCA4CATA::MakeToleranceIntervals(resMCA$ExPosition.Data$fi,
                            design = as.factor(cleanData$nation),
                            # line below is needed
                            names.of.factors =  c("Dim1","Dim2"), # needed
                            col = colCity,
                            line.size = .50,
                            line.type = 3,
                            alpha.ellipse = .2,
                            alpha.line    = .4,
                            p.level       = .75)
#_____________________________________________________________________
# Create the map:
d005.Map.I.withTIHull <- d002.Map.I.withMeans  +
                           GraphTI.Hull + MapGroup$zeMap_dots +
                           MapGroup$zeMap_text + MapGroup$zeMap_dots
#_____________________________________________________________________
# plot it
dev.new()
print(d005.Map.I.withTIHull)
```

6.8 Projecting a subsample

```
Fi.sup  <- resMCA.sup$fii
col     <- 'green'
nI.sup <- nrow(Fi.sup)
col4I.sup <- rep("",nI.sup)
# generate the set of maps
BaseMap.Fi.sup <- createFactorMap(X = Fi.sup , # resMCA$ExPosition.Data$fj,
                                  axis1 = axis1, axis2 = axis2,
                                  constraints = BaseMap.Fi$constraints,
                                  title = '',
                                  col.points = 'green',
                                  alpha.points = .2, cex = 1.2,
                                  col.labels = 'green',
                                  text.cex = 2.5,
                                  force = 2)
# make the maps
e0001.BaseMapNoLabels.Fi.sup  <- BaseMap.Fi$zeMap_background +
                                      BaseMap.Fi.sup$zeMap_dots +
                                      BaseMap.Fi.sup$zeMap_text +
                                      BaseMap.Fi$zeMap_dots +
                                      labels4MCA
```

```
print(e0001.BaseMapNoLabels.Fi.sup)
```



7 Save the graphics as a PowerPoint

The graphics are saved as a power point called `fermentationFrom2Countries.pptx` with the following command

```
list2Graphs <- PTCA4CATA::saveGraph2pptx(file2Save.pptx = name4Graphs,
                 title = 'Attitudes toward fermented products',
                 addGraphNames = TRUE)
```

Note that we could also have created a power point with `Rmarkdown` by using the following options in the preamble:

```
output:
      powerpoint_presentation:
           slide_level: 4
```

instead of (for example):

```
output:
       rmarkdown::html_vignette:
          toc: true
          number_sections: true
```