R-Project on Metabolomic Dataset

Christian Peralta

18/12/2020

Project Description

The Data

As part of the course *Statistics and Probability* thought by Prof Jose Nunes, who belongs to the department of Anthropology and Population Genetics at the University of Geneva, I have to develop an R-package. To this aim I focused on possible functions of R which might be useful for my internship in the group of Biomedical and Metabolomics Analysis led by Prof. Serge Rudaz, at the School of Pharmaceutical Sciences of the University of Geneva.

I based my functions in a data frame which comes from a toxicology study aiming to characterize the metabolic effects of Trimethyltin on a 3D "mini brain" cell system involving neurons and glial cells. All the details are described in the article *González-Ruiz, V., Schvartz, D., Sandström, J., Pezzatti, J., Jeanneret, F., Tonoli, D., Boccard, J., Monnet-Tschudi, F., Sanchez, J. C., & Rudaz, S. (2019). An integrative multi-omicsworkflow to address multifactorial toxicology experiments. Metabolites, 9(4), 1–15. https://doi.org/10.3390/metabo9040079*.

The experimental factors include two maturation states (immature and mature), two exposure types (24 hours and 10 days) and three doses (0, 0.5 and 1 uM). Three biological replicates were considered for each combination of factors and 189 metabolites were measured.

The Functions

This project is composed by two main parts, the first one has a data-handling purpose where I tried to mimic the most likely scenario which is having several data frames and merge them. Here I need to highlight the astonishing functions **Map()** and **Reduce()** which make the code being easier, hence cleaner, as well as less error-prone as if I would have used a "for loop".

The second part is a fusion of statistical and visualization tools which I found particularly useful in my function **PC.Graph** where I extract the Principal Components which explain a 97% of variability of the data, I generate the combinations between them (325) and then the graphs. See the document All_PC.pdf

Last but not least, just mention that this is not the code that appears in my R-package script, since one has to use the "::" notation, such as "stats::prcomp" or "ggplot2::autoplot".

Packages

```r
library(readxl)
library(reshape2)
library(ggplot2)
library(cowplot)
library(ggfortify)
library(randomForest)
```

Dealing with the Data frame

Reading form an Excel file

```r
MS <- read_excel('Copy of Dataset_TMT.xlsx')
```

Splitting up the original data frame

G.Slice creates slices from an original data frame based on how many columns it has and writes them as '.csv' documents. G.Slice function takes as parameters the data frame (*df*), two values which indicates which columns must be always kept (*fix.c.range1* and *fix.c.range2*). The next two parameters has the role of determining the size (in columns) of each slice; the first column (*init.pos*) to be included, and the specific number of columns to be included (*jump*). Lastly, one more parameter in case one want to modify the name, settle as default as 'Slice' (*base*).

```r
G.Slice <- function(df, fix.c.range1, fix.c.range2, init.pos, jump, base= 'Slice'){
  stop.pos <- ncol(df)
  start.points <- seq(init.pos, stop.pos, jump)
  One.Slice<-function(x=NULL){
    y <- if (x+9 <= stop.pos) x+9 else stop.pos
    modify <- c(fix.c.range1:fix.c.range2, x:y)
    df2 <- df[, modify]
    file_name <- paste(base, '_', x,'-', y, '_.csv', sep ='')
    write.table(df2, file_name, row.names = F, sep='\t', col.names = T)
  }
  Map(One.Slice, start.points)
}

G.Slice(df = MS, fix.c.range1 = 1, fix.c.range2 = 5, init.pos = 6, jump = 10)
```

Merging the slices into one final data frame

Mer.Slice function merges two or more data frames into one taking only one parameter, *name*, which is the pattern to identify the data frames of interest when calling for the list of files in the working directory.

```r
Mer.Slice <- function(name){
  all.files_list <- list.files(pattern = name, full.names = T)
  Read.list.names <- function(x){
    read.table(x, header= T, sep = '\t')
  }
  all.df_list <- Map(Read.list.names, all.files_list)
  Reduce(merge, all.df_list)
```

```
}

Final_DF <- Mer.Slice(name = 'Slice_')
```

Multi-Dimensional Reduction Functions

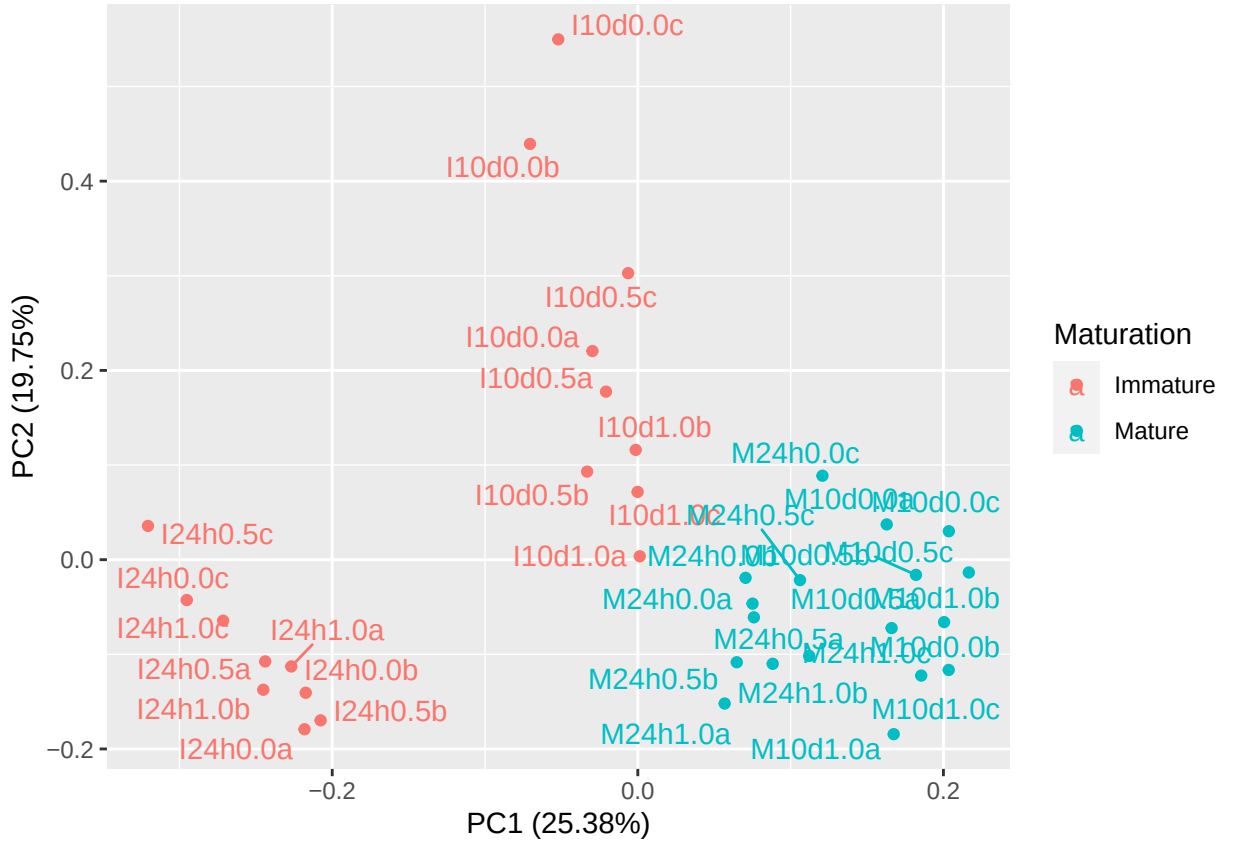
Principal Component Analysis (PCA)

PCA.Function and PC.Graph are designed for data frames where one needs to remove a range of columns.

PCA.Function applies PCA onto a data frame and generates a plot of the desired principal components. The parameters are a data frame (*df*), two values to determine the range of columns to exclude (*col1* and *col2*), a variable as the tag (*tag*), as well as a parameter for the color in the plot (*id*). The last two which indicated the principal components to be displayed in the plot (*pcx* and *pcy*).

```
PCA.Function <- function(df, col1, col2, tag, id, pcx=1, pcy=2){
  df1 <- df[,-c(col1:col2)]
  pcax <- prcomp(df1, scale. = T, center = T)
  df2 <- df
  row.names(df2) <- df[,tag]
  autoplot(pcax, colour = id , data = df2, label = T, label.repel = T, x = pcx,
           y = pcy)
}

PCA.Function(df = Final_DF, col1 = 1, col2 = 5, tag = 'Sample', id =  'Maturation')
```



PC.Graph applies PCA onto a data frame from which obtains how many principal components (PCs) are needed to cover 97% of variability on the data. Moreover, generates the combination between those PCs and generates a plot for each combination which are gathered in a .pdf file. As parameters, a data frame (*df*), two values to determine the range of columns to exclude (*col1* and *col2*), a variable as the tag (*tag*), as well as a parameter for the color in the plot (*id*).

```
PC.Graph <- function(df, col1, col2, tag, id){
  df1 <- df[,-c(col1:col2)]
  pca <- prcomp(df1, scale. = T)
  pc.summ <- summary(pca)
  a <- sum(pc.summ$importance[3,] < 0.975)
  b <- c(1:a)
  pc.pairs <- as.data.frame(t(combn(b, 2)))
  One.plot <- function(pcx, pcy){
    df2 <- df
    row.names(df2) <- df[, tag]
    name = paste("PCA_", pcx, "_vs_", pcy)
    autoplot(pca, data = df2, colour = id, label = T, label.repel = T, main = name,
             x = pcx, y = pcy, frame = T)
  }
  all.plots <- Map(One.plot, pc.pairs$V1, pc.pairs$V2)
  split.plots <- split(all.plots, ceiling(seq_along(all.plots)/20))
  Grid.plots <- function(x){
    out <- plot_grid(plotlist = x, ncol = 4, nrow = 5)
    plot(out)
  }
```

```
  pdf(file = "All_PC.pdf", width = 30, height = 50)
  Map(Grid.plots, split.plots)
  dev.off()
}

PC.Graph(Final_DF, col1 = 1, col2 = 5, tag = "Sample", id = "Maturation")
```

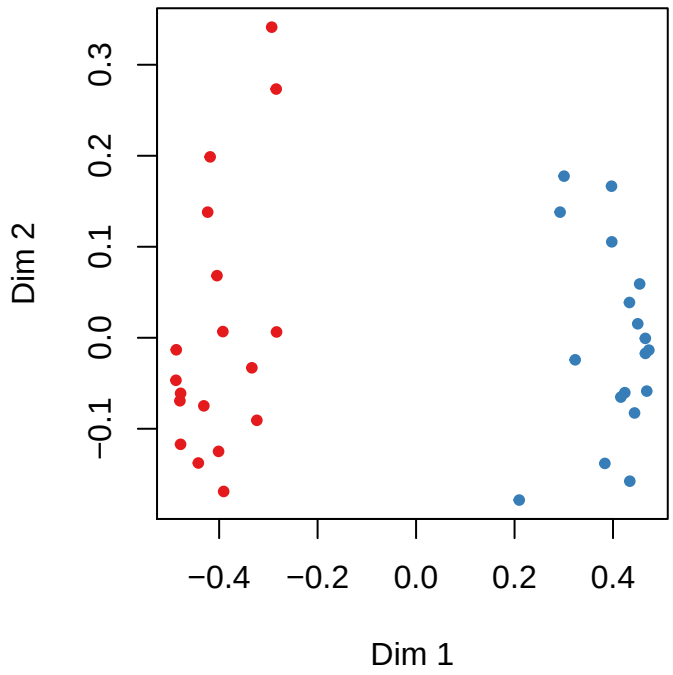
Non-metric MultiDimensional Scaling (NMDS)

Random Forest Model

Make.RF function generates MultiDimensional Scaling plot from a regression formula. As parameter, Make.RF function takes a data frame (*df*), number of variables taken into account each iteration of the analysis (*mtry.p*) and a variable response to settle the formula (*var.response*).

```
Make.RF <- function(df, mtry.p, var.response){
  formula <- formula(paste(var.response, '~.'))
  rfor1 <- randomForest(data = df, formula, proximity = T, mtry = mtry.p, ntree = 1000)
  MDSplot(rfor1, df[[var.response]])
}

Make.RF(df = Final_DF[-1], mtry.p = 5, var.response = "Maturation")
```

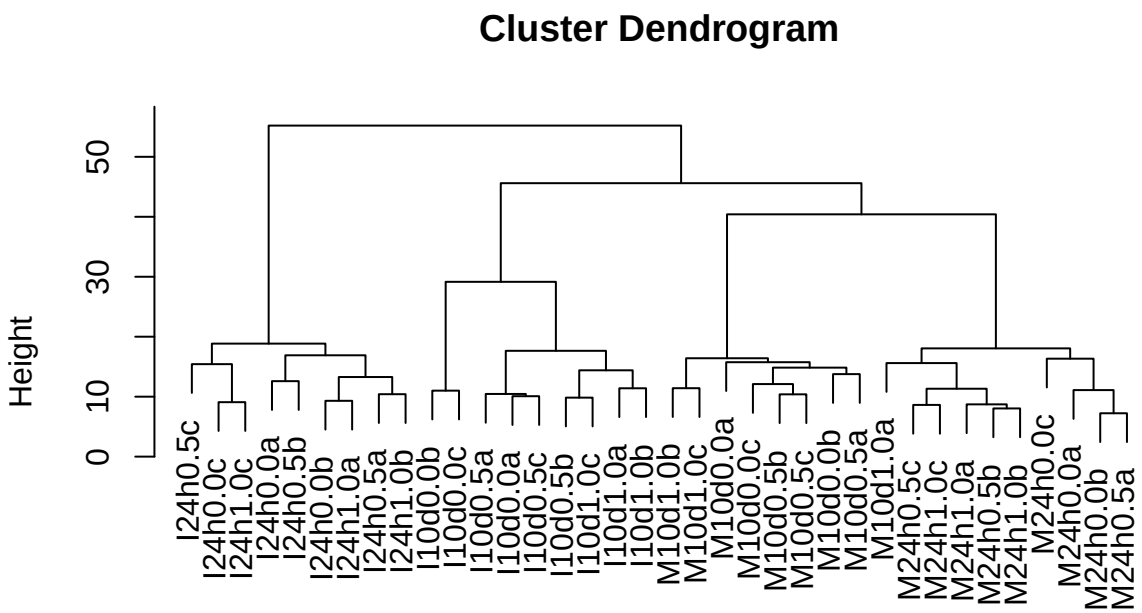


Cluster Analysis

Clust.Function applies a PCA analysis and generates a cluster dendrogram using by default the method 'ward.D2'. For this aim, Clust.Function takes six parameters, (*df*) which is the data frame, (*col1*) and (*col2*) which make up the range of columns to get rid of. The parameter *tag* is the variable used as label, lastly, (*pc1*) and (*pc2*) which indicates how many of principal component are of interest.

```
Clust.Function <- function(df, col1, col2, tag, pc1, pc2){
  df1 <- df[,-c(col1:col2)]
  pcax <- prcomp(df1, scale. = T, center = T)
  df2 <- df
  row.names(df2) <- df[,tag]
  distPCA <- dist(pcax$x[,pc1:pc2])
  clust <- hclust(distPCA, method = "ward.D2")
  plot(clust,labels = row.names(df2), xlab = paste('Scores of PCA', pc1, 'to', pc2),
       sub = 'Method "ward.D2" applied')
}

Clust.Function(df = Final_DF, col1 = 1, col2 = 5, tag = 'Sample',
               pc1 = 1, pc2 = 26)
```



Final Remark

I could not have enjoyed more this course and project, the latter has made me change my mindset which was adapted to Python and has helped me to witnessed how powerful is R and how many features I still need to discover, such as dplyr and the tidyverse universe.

Thus, I am looking forward to learning more about R and statistical analysis.

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] randomForest_4.6-14 ggfortify_0.4.11    cowplot_1.1.1
## [4] ggplot2_3.3.3       reshape2_1.4.4      readxl_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.5         RColorBrewer_1.1-2 cellranger_1.1.0   compiler_4.0.3
##  [5] pillar_1.4.7       plyr_1.8.6         tools_4.0.3        digest_0.6.27
##  [9] evaluate_0.14      lifecycle_0.2.0    tibble_3.0.4       gtable_0.3.0
## [13] pkgconfig_2.0.3    rlang_0.4.10       DBI_1.1.1          ggrepel_0.9.0
## [17] yaml_2.2.1         xfun_0.20          gridExtra_2.3      withr_2.3.0
## [21] stringr_1.4.0      dplyr_1.0.3        knitr_1.30         generics_0.1.0
## [25] vctrs_0.3.6        grid_4.0.3         tidyselect_1.1.0   glue_1.4.2
## [29] R6_2.5.0           rmarkdown_2.6      farver_2.0.3       tidyr_1.1.2
## [33] purrr_0.3.4        magrittr_2.0.1     scales_1.1.1       ellipsis_0.3.1
## [37] htmltools_0.5.0    assertthat_0.2.1   colorspace_2.0-0   labeling_0.4.2
## [41] stringi_1.5.3      munsell_0.5.0      crayon_1.3.4
```