8. Adjuntos: Ejemplos códigos

A continuación se adjuntan ejemplos de los códigos más relevantes que se elaboraron y emplearon durante el TFG para el análisis de datos y la obtención de resultados.

8.1 Imputación de VFs

```
#############################################################
##### Inputacion datos SET03  por mice ####
#############################################################

library(mice)

# Imputacion por MICE ----------------------------------------------------
------
#CB
prot03_CB <-
  read.delim(

"~/Dropbox/denamic/data/set03/00_Set03_CB_ListaParaInputacionNAs.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

sum(apply(prot03_CB, 2, function(colum) {
  sum(is.na(colum))
}))##2964

prot03_CB_Imputada <- mice(t(prot03_CB), method = "norm.predict")
# #despues deberia hacer: complete(prot03_CB_Imputada) para obtener la
matriz imputada

prot03_CB_Imputada <- complete(prot03_CB_Imputada)
sum(apply(prot03_CB_Imputada, 2, function(colum) {
  sum(is.na(colum))
}))##0

prot03_CB_Imputada <- t(prot03_CB_Imputada)

#Guardar la matriz para PCA sin normalizar
write.table(
  prot03_CB_Imputada,
  file =
"~/Dropbox/denamic/data/set03/01_Set03_CB_Imputada_Por_MICE.txt",
  col.names = T,
  row.names = T
)

#HP
prot03_HP <-
  read.delim(

"~/Dropbox/denamic/data/set03/00_Set03_HP_ListaParaInputacionNAs.txt",
    header = T,
    row.names = 1,
    sep = " "
  )
```

```r
prot03_HP_Imputada <- mice(t(prot03_HP), method = "norm.predict")

prot03_HP_Imputada <- complete(prot03_HP_Imputada)
sum(apply(prot03_HP_Imputada, 2, function(colum) {
  sum(is.na(colum))
}))##0

prot03_HP_Imputada <- t(prot03_HP_Imputada)

#Guardar la matriz para PCA sin normalizar
write.table(
  prot03_HP_Imputada,
  file =
"~/Dropbox/denamic/data/set03/01_Set03_HP_Imputada_Por_MICE.txt",
  col.names = T,
  row.names = T
)
```

8.2 PCAs

```
###########################
## PCAs SET01 #############
###########################

#to get the colors linked to pestides and shapes linked to gender
source("~/Dropbox/denamic/scripts/Pesticides_Colors_and_GenderShapes.R
")
library(mixOmics) #to obtain the PCA'S


################PROTEOMICA###########################

#SET01
#CB
prot01_CB <-
  read.delim("~/Dropbox/denamic/data/set01/00_Set01_prot_CB.txt")

#homogenizacion de nombres
noms <- colnames(prot01_CB)
noms <- gsub(".", "_", noms, fixed = T)
colnames(prot01_CB) <- gsub(".", "_", colnames(prot01_CB), fixed = T)
prot01_CB <- t(prot01_CB)
par(mar = c(8, 2, 4, 2))
boxplot(t(prot01_CB),
        las = 2,
        main = "Set 01 - Proteomics- CB",
        col = "grey")

# Design matrix -------------------------------------------------------
-----

##MATRIZ DE DISEÑO
ids <- rownames(prot01_CB)
designProt01_CB <- do.call("rbind", strsplit(ids, "_", fixed = T))
rownames(designProt01_CB) <- ids
colnames(designProt01_CB) <-
  c("Pesticide", "Gender", "Replicate", "Tissue")
ncol(prot01_CB)


# Saving design matrix ------------------------------------------------
-----
write.table(
  designProt01_CB,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designProt01_CB.txt",
  row.names = T,
  col.names = T
)

X <- prot01_CB
boxplot(X)
boxplot(scale(X, center = TRUE, scale = TRUE))

# PCAs and loading plots scaled ---------------------------------------
-----

##con Scale=T
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

par(mar = c(5, 4, 2, 2))

####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingprot01_cb_scaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingprot01_cb_scaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAprot01_cb_scaled")
plotIndiv(
  pcaX,
  group = designProt01_CB[, 1],
  ind.names = FALSE,
  title = "Set 01 - Proteomics - CB-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designProt01_CB[, 2]],
  col.per.group = MYcolors[designProt01_CB[, 1]]
)

#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designProt01_CB[, 1]),
       col = unique(MYcolors[designProt01_CB[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designProt01_CB[, 2]),
  pch = unique(myPCH[designProt01_CB[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

## to check if the simbols and shapes are well assigned
plotIndiv(
  pcaX,
  group = factor(designProt01_CB[, 1], levels = c("VH", "END",
"CYP")),
  ind.names = rownames(designProt01_CB),
  legend = F,
  title = "Set01_Prot_CB_withIDS",
  ellipse = F,
  col.per.group = MYcolors[c("VH", "END", "CYP")]
)


# PCAs and loading plots NON scaled ----------------------------------
-----
```

```
#con Scale=F

pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = F)

par(mar = c(5, 4, 2, 2))


####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingprot01_cb_NONscaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingprot01_cb_NONscaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAprot01_cb_NONscaled")
plotIndiv(
  pcaX,
  group = factor(designProt01_CB[, 1], levels = c("VH", "END",
"CYP")),
  ind.names = FALSE,
  title = "Set 01 - Proteomics - CB-NonScaled",
  size.title = rel(1),
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designProt01_CB[, 2]],
  col.per.group = MYcolors[c("VH", "END", "CYP")]
)

#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designProt01_CB[, 1]),
       col = unique(MYcolors[designProt01_CB[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designProt01_CB[, 2]),
  pch = unique(myPCH[designProt01_CB[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()


# HP -----------------------------------------------------------------
-----
prot01_HP <-
  read.delim("~/Dropbox/denamic/data/set01/00_Set01_prot_HP.txt")

#homogenizacion de nombres
noms <- colnames(prot01_HP)
noms <- gsub(".", "_", noms, fixed = T)
colnames(prot01_HP) <- gsub(".", "_", colnames(prot01_HP), fixed = T)
prot01_HP <- t(prot01_HP)
```

```
par(mar = c(8, 2, 4, 2))
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Prot01_HP")
boxplot(t(prot01_HP),
        las = 2,
        main = "Set 01 - Proteomics- HP",
        col = "grey")
dev.off()

# Design matrix -------------------------------------------------------
-----
ids <- rownames(prot01_HP)
designProt01_HP <- do.call("rbind", strsplit(ids, "_", fixed = T))
rownames(designProt01_HP) <- ids
colnames(designProt01_HP) <-
  c("Pesticide", "Gender", "Replicate", "Tissue")

# Saving design matrix ------------------------------------------------
-----
write.table(
  designProt01_HP,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designProt01_HP.txt",
  row.names = T,
  col.names = T
)

X <- prot01_HP
boxplot(X)
boxplot(scale(X, center = TRUE, scale = TRUE))

# PCAs and loading plots scaled ---------------------------------------
-----

##con Scale=T
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

par(mar = c(5, 4, 2, 2))

####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingprot01_HP_scaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingprot01_HP_scaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAprot01_HP_scaled")
plotIndiv(
  pcaX,
  group = designProt01_HP[, 1],
  ind.names = FALSE,
  title = "Set 01 - Proteomics - HP-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designProt01_HP[, 2]],
  col.per.group = MYcolors[designProt01_HP[, 1]]
)


#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designProt01_HP[, 1]),
       col = unique(MYcolors[designProt01_HP[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designProt01_HP[, 2]),
  pch = unique(myPCH[designProt01_HP[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

# PCAs and loading plots NON scaled ----------------------------------
-----

#con Scale=F

pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = F)

par(mar = c(5, 4, 2, 2))


####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingprot01_HP_NONscaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingprot01_HP_NONscaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAprot01_HP_NONscaled")
plotIndiv(
  pcaX,
  group = designProt01_HP[, 1],
  ind.names = FALSE,
  title = "Set 01 - Proteomics -HP-NonScaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designProt01_HP[, 2]],
  col.per.group = MYcolors[designProt01_HP[, 1]]
)

#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designProt01_HP[, 1]),
       col = unique(MYcolors[designProt01_HP[, 1]]),
       pch = myPCH)
```

```
#to plot the shape according to sex
legend(
  "topleft",
  unique(designProt01_HP[, 2]),
  pch = unique(myPCH[designProt01_HP[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

# Metabolomics -------------------------------------------------------
-----

# CB -----------------------------------------------------------------
-----
met01_CB <-
  read.delim("~/Dropbox/denamic/data/set01/00_Set01_met_CB.txt")

#homogenizacion de nombres
noms <- colnames(met01_CB)
noms <- gsub(".", "_", noms, fixed = T)
colnames(met01_CB) <- gsub(".", "_", colnames(met01_CB), fixed = T)
met01_CB <- t(met01_CB)
par(mar = c(8, 2, 4, 2))
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Met01_CB")
boxplot(
  t(met01_CB),
  las = 2,
  main = "Set 01 -Metabolomics- CB",
  col = "grey",
  cex.axis = 0.7
)
dev.off()


# Design matrix ------------------------------------------------------
-----
ids <- rownames(met01_CB)
designMet01_CB <- do.call("rbind", strsplit(ids, "_", fixed = T))

rownames(designMet01_CB) <- ids
colnames(designMet01_CB) <-
  c("Pesticide", "Gender", "Replicate", "Tissue")


# Saving design matrix -----------------------------------------------
-----
write.table(
  designMet01_CB,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designMet01_CB.txt",
  row.names = T,
  col.names = T
)

##boxplot to see if it is necessary scaling
X <- met01_CB
```

```
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Met01_CB_IsScallingNec
essary")
boxplot(
  X,
  xlab = "Metabolites",
  main = "Set 01 -Scalling?-Metabolomics- CB",
  col = "grey",
  cex.axis = 0.7
)
dev.off()
boxplot(scale(X, center = TRUE, scale = F))

# PCAs and loading plots scaled --------------------------------------
-----

##con Scale=T
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

par(mar = c(5, 4, 2, 2))

####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingMet01_cb_scaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingMet01_cb_scaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAMet01_cb_scaled")
plotIndiv(
  pcaX,
  group = designMet01_CB[, 1],
  ind.names = FALSE,
  title = "Set 01 - Metabolomics - CB-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designMet01_CB[, 2]],
  col.per.group = MYcolors[designMet01_CB[, 1]]
)

#to plot the  colour according to pesticide
# legend("bottomleft", c("VH", "PEST"), col = c(1,2), pch = myPCH)
legend("bottomleft",
       unique(designMet01_CB[, 1]),
       col = unique(MYcolors[designMet01_CB[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designMet01_CB[, 2]),
  pch = unique(myPCH[designMet01_CB[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)
```

```
dev.off()

## to check if the simbols and shapes are well assigned
plotIndiv(
  pcaX,
  group = designMet01_CB[, 1],
  ind.names = rownames(designMet01_CB),
  legend = F,
  title = "Set01_Met_CB_withIDS",
  ellipse = F,
  col.per.group = MYcolors[designMet01_CB[, 1]]
)


# PCAs and loading plots NON scaled ----------------------------------
-----

#con Scale=F

pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = F)

par(mar = c(5, 4, 2, 2))

####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingMet01_cb_NONscaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingMet01_cb_NONscaled")
dev.off()

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAMet01_cb_NONscaled")
plotIndiv(
  pcaX,
  group = designMet01_CB[, 1],
  ind.names = FALSE,
  title = "Set 01 - Metabolomics - CB-NonScaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designMet01_CB[, 2]],
  col.per.group = MYcolors[designMet01_CB[, 1]]
)

#to plot the  colour according to pesticide
# legend("bottomleft", c("VH", "PEST"), col = c(1,2), pch = myPCH)
legend("bottomleft",
       unique(designMet01_CB[, 1]),
       col = unique(MYcolors[designMet01_CB[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designMet01_CB[, 2]),
  pch = unique(myPCH[designMet01_CB[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

# HP ------------------------------------------------------------------
-----
met01_HP <-
  read.delim("~/Dropbox/denamic/data/set01/00_Set01_met_HP.txt")
View(met01_HP)

#homogenizacion de nombres
noms <- colnames(met01_HP)
noms <- gsub(".", "_", noms, fixed = T)
colnames(met01_HP) <- gsub(".", "_", colnames(met01_HP), fixed = T)
met01_HP <- t(met01_HP)
par(mar = c(8, 2, 4, 2))
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Met01_HP")
boxplot(
  t(met01_HP),
  las = 2,
  main = "Set 01 -Metabolomics- HP",
  col = "grey",
  cex.axis = 0.7
)
dev.off()

# Design matrix -------------------------------------------------------
-----
ids <- rownames(met01_HP)
designMet01_HP <- do.call("rbind", strsplit(ids, "_", fixed = T))
rownames(designMet01_HP) <- ids
colnames(designMet01_HP) <-
  c("Pesticide", "Gender", "Replicate", "Tissue")

# Saving design matrix ------------------------------------------------
-----
write.table(
  designMet01_HP,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designMet01_HP.txt",
  row.names = T,
  col.names = T
)

##boxplot to see if it is necessary scaling
X <- met01_HP
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Met01_HP_IsScallingNec
essary")
boxplot(
  X,
  xlab = "Metabolites",
  main = "Set 01 -Scalling?-Metabolomics- HP",
  col = "grey",
  cex.axis = 0.7
)
dev.off()
boxplot(scale(X, center = TRUE, scale = F))
```

```
# PCAs and loading plots scaled --------------------------------------
-----

##con Scale=T

pcaX <-
  pca(X,
      ncomp = 3,
      center = TRUE,
      scale = TRUE)##Error: cannot rescale a constant/zero column to
unit variance.

###REMOVING COLUMNS WITH NO VARIANCE
#calculo de las varianzas para cada metabolito
VarianceMetabolites <- apply(X, 2, var)
#identificacion de los metabolitos cuya varianza==0
a <- 0
MetabolitesWithNoVariance <- character()
for (element in VarianceMetabolites) {
  a <- a + 1
  if (element == 0) {
    cat(element, " ", names(VarianceMetabolites[a]) , "\n")
    MetabolitesWithNoVariance <-
      c(MetabolitesWithNoVariance, names(VarianceMetabolites[a]))
  }
}
#indices de las columnas correspondientes a los metabolitos cuya
varianza=0
indexes <- NULL
for (element in MetabolitesWithNoVariance) {
  indexes <- c(indexes, (which(colnames(X) == element)))
}
####las columnas sin varianza son la 16 y la 19, se corresponden con
"5-HIAA" y "ACETYLCHOLINE"
#eliminacion de las columnas cuya varianza=0
X <- X[, -indexes]

# Saving Metabolomic HP data with columns with no Variance deleted ---
-----
write.table(X ,
            file = "~/Dropbox/denamic/data/set01/01_Set01_met_HP.txt",
            row.names = T,
            col.names = T)

#CONTINUAMOS CON EL PCA
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

###hacer loadings tb para ver donde tienden a acumularse, queremos que
la mayoria esten sobre el 0
par(mar = c(5, 4, 2, 2))

png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingMet01_HP_scaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingMet01_HP_scaled")
dev.off()
```

```r
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/PCAMet01_HP_scaled")
plotIndiv(
  pcaX,
  group = designMet01_HP[, 1],
  ind.names = FALSE,
  title = "Set 01 - Metabolomics -HP-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designMet01_HP[, 2]],
  col.per.group = MYcolors[designMet01_HP[, 1]]
)


#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designMet01_HP[, 1]),
       col = unique(MYcolors[designMet01_HP[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designMet01_HP[, 2]),
  pch = unique(myPCH[designMet01_HP[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

# Clinical data analysis ---------------------------------------------
-----
clinic01 <-
  read.delim("~/Dropbox/denamic/data/set01/00_Set01_clinic.txt", sep =
" ")
a <- clinic01
View(clinic01)
rownames(clinic) == rownames(designClinic01)
rownames(clinic)[21]
sum(rownames(designClinic01) == "END_M_III5")
clinic <- clinic[-21, ]
clinic01 <- clinic
X <- clinic01
write.table(clinic01,
            file = "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
            col.names = T,
            row.names = T)

X <-
  read.delim(
    "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )
```

```
# Design matrix ------------------------------------------------------
-----

##MATRIZ DE DISEÑO
ids <- rownames(clinic01)
designClinic01 <- do.call("rbind", strsplit(ids, "_", fixed = T))
rownames(designClinic01) <- ids
colnames(designClinic01) <- c("Pesticide", "Gender", "Replicate")

# Saving design matrix -----------------------------------------------
-----
write.table(
  designClinic01,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designClinic01.txt",
  row.names = T,
  col.names = T
)

X <-
  read.delim(
    "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

designClinic01 <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designClinic01.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

##boxplot to see if it is necessary scaling
X <- t(clinic01)
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Clinic01_IsScallingNec
essary")
boxplot(
  X,
  main = "Set 01 -Scalling?-Clinic",
  col = "grey",
  cex.axis = 0.7,
  las = 2
)

which(colnames(X) == "END_M_III5")
X <- X[, -13]

dev.off()
boxplot(scale(X, center = TRUE, scale = F))

Xscaled = scale(X, center = TRUE, scale = TRUE)

boxplot(scale(X, center = TRUE, scale = TRUE))

#es necesario escalar
X <-
```

```
  read.delim(
    "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

designClinic01 <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designClinic01.txt",
    header = T,
    row.names = 1,
    sep = " "
  )


# PCAs and loading plots scaled --------------------------------------
-----

##con Scale=T
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

par(mar = c(5, 4, 2, 2))

####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingClinic01_scaled")
plot(pcaX$loadings$X[, 1:2], main = "loadingClinic01_scaled")
dev.off()

png(filename = "~/Dropbox/denamic/plots/Set01/Pcas/Clinic01_scaled")
plotIndiv(
  pcaX,
  group = designClinic01[, 1],
  ind.names = FALSE,
  title = "Set 01 -Clinic01-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designClinic01[, 2]],
  col.per.group = MYcolors[designClinic01[, 1]]
)

#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designClinic01[, 1]),
       col = unique(MYcolors[designClinic01[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designClinic01[, 2]),
  pch = unique(myPCH[designClinic01[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()

##I'M GOING TO DELETE THE "OUTLIERS" VALUE TO PLOT AGAIN THE PCA
x11()
plotIndiv(
  pcaX,
  group = designClinic01[, 1],
  ind.names = rownames(designClinic01),
  legend = F,
  title = "Set01_Clinic01_Scaled_WithIDS",
  ellipse = F,
  col.per.group = MYcolors[designClinic01[, 1]]
)

##OUTLIERS DETECTED: ("END_M_III5", "VH_F_I5"),vamos a eliminarlos y
comenzar de nuevo el PCA Analisis
outliers <- c("END_M_III5", "VH_F_I5")
indexes <- NULL
for (element in outliers) {
  indexes <- c(indexes, which(rownames(X) == element))
}

clinic01 <- clinic01[-indexes, ]
# Design matrix -------------------------------------------------------
-----

##MATRIZ DE DISEÑO
ids <- rownames(clinic01)
designClinic01 <- do.call("rbind", strsplit(ids, "_", fixed = T))
rownames(designClinic01) <- ids
colnames(designClinic01) <- c("Pesticide", "Gender", "Replicate")


# Saving design matrix ------------------------------------------------
-----
write.table(
  designClinic01,
  file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designClinic01.txt",
  row.names = T,
  col.names = T
)

##boxplot to see if it is necessary scaling
X <- clinic01
png(filename =
"~/Dropbox/denamic/plots/Set01/Boxplots/Boxplot_Clinic01_IsScallingNec
essary")
boxplot(
  X,
  main = "Set 01 -Scalling?-Clinic",
  col = "grey",
  cex.axis = 0.7,
  las = 2
)

dev.off()
boxplot(scale(X, center = TRUE, scale = F))
```

```
Xscaled = scale(X, center = TRUE, scale = TRUE)

boxplot(scale(X, center = TRUE, scale = TRUE))

#es necesario escalar

# PCAs and loading plots scaled -------------------------------------
-----
X <-
  read.delim(
    "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

designClinic01 <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set01/DesignMatrixes/designClinic01.txt",
    header = T,
    row.names = 1,
    sep = " "
  )
gsub("...", "_", colnames(X))
colnames(X)

X <-
  read.delim(
    "~/Dropbox/denamic/data/set01/01_Set01_clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )

##con Scale=T
pcaX <- pca(X,
            ncomp = 3,
            center = TRUE,
            scale = TRUE)

par(mar = c(5, 4, 2, 2))


####hacer loadings tb para ver donde tienden a acumularse, queremos
que la mayoria esten sobre el 0
png(filename =
"~/Dropbox/denamic/plots/Set01/Pcas/loadingClinic01_scaled")

plot(
  pcaX$loadings$X[, 1:2],
  main = "loadingClinic01_scaled",
  xlim = c(0.2, 0.9),
  col = "red"
)
text(
  pcaX$loadings$X[, 1:2][, 1],
  y = pcaX$loadings$X[, 1:2][, 2] ,
  labels = rownames(pcaX$loadings$X),
  pos = 4
)
dev.off()

png(filename = "~/Dropbox/denamic/plots/Set01/Pcas/Clinic01_scaled")
plotIndiv(
  pcaX,
  group = designClinic01[, 1],
  ind.names = FALSE,
  title = "Set 01 -Clinic01-Scaled",
  size.title = 1.2,
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designClinic01[, 2]],
  col.per.group = MYcolors[designClinic01[, 1]]
)

#to plot the  colour according to pesticide
legend("bottomleft",
       unique(designClinic01[, 1]),
       col = unique(MYcolors[designClinic01[, 1]]),
       pch = myPCH)
#to plot the shape according to sex
legend(
  "topleft",
  unique(designClinic01[, 2]),
  pch = unique(myPCH[designClinic01[, 2]]) ,
  bty = "o",
  ncol = 2,
  col = "gray45"
)

dev.off()
```

8.3 Correción de ruido, ARSyN

```
######################################################################
#######
################         ARSYNSEQ
################################
######################################################################
#######
library(NOISeq)

#####ADDING TO THE DESIGN MATRIXES THE PEST_GENDER COLUMN ------------
---

#CB
design_CB <-
  read.delim(
    "~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_CB",
    header = T,
    row.names = 1,
    sep = " "
  )
data <- design_CB
data$Pest_Gender <-
  apply(data[, c(1, 2)] , 1 , paste , collapse = "_")

###Saving Design Matrix for Arsyn CB
write.table(
  data,

"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_CB.txt
",
  col.names = T,
  row.names = T,
  sep = " ",
  quote = F
)

#HP
design_HP <-
  read.delim(
    "~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_HP",
    header = T,
    row.names = 1,
    sep = " "
  )
data <- design_HP
data$Pest_Gender <-
  apply(data[, c(1, 2)] , 1 , paste , collapse = "_")

###Saving Design Matrix for Arsyn HP
write.table(
  data,

"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_HP.txt
",
  col.names = T,
  row.names = T,
  sep = " ",
  quote = F
)
```

```
# applying ARSYNSEQ  ------------------------------------------------
-----

# #CB -------------------------------------------------------------
-----
protCB <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set03/01_Set03_CB_Imputada_Por_MICE.txt",
    sep = " ",
    header = T,
    row.names = 1
  )
protCB <- t(protCB)

designCB <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_CB.txt
",
    sep = " ",
    header = T,
    row.names = 1
  )
design <- designCB

prot <- readData(data = protCB, factors = design)

prot_postARSYN <-
  ARSyNseq(
    data = prot,
    factor = "Pest_Gender",
    logtransf = TRUE,
    norm = "n"
  )
prot_postARSYNmatrix <- prot_postARSYN@assayData$exprs

prot_postARSYNmatrix <- t(prot_postARSYNmatrix)

###Saving Design Matrix for Arsyn CB
write.table(
  prot_postARSYNmatrix,
  "~/Dropbox/denamic/data/set03/02_Set03_CB_PostArsyn",
  col.names = T,
  row.names = T,
  sep = " ",
  quote = F
)

# #HP -------------------------------------------------------------
-----
protHP <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set03/01_Set03_HP_Imputada_Por_MICE.txt",
    sep = " ",
    header = T,
    row.names = 1
  )
protHP <- t(protHP)
```

```
designHP <-
  read.delim(
    file =
"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_HP.txt
",
    sep = " ",
    header = T,
    row.names = 1
  )
design <- designHP

prot <- readData(data = protHP, factors = design)

prot_postARSYN <-
  ARSyNseq(
    data = prot,
    factor = "Pest_Gender",
    logtransf = TRUE,
    norm = "n"
  )
prot_postARSYNmatrix <- prot_postARSYN@assayData$exprs

prot_postARSYNmatrix <- t(prot_postARSYNmatrix)

###Saving Design Matrix for Arsyn CB
write.table(
  prot_postARSYNmatrix,
  "~/Dropbox/denamic/data/set03/02_Set03_HP_PostArsyn",
  col.names = T,
  row.names = T,
  sep = " ",
  quote = F
)
```

8.4 PLS

```
###########################################
### Estrategia Seleccion Variables ########
##########################################

library(mixOmics)
library(mice)

source("~/Dropbox/denamic/scripts/Pesticides_Colors_and_GenderShapes.R
")

# CB -----------------------------------------------------------------
-----

#lectura matriz X=proteomics
cb <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_CB_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
cb <- cb[-which(rownames(cb) == "CYPER_M_I6_CB"), ]
rownames(cb) <- gsub("_CB", "", rownames(cb))

#lectura matriz Y=clinical
clinical <-
  read.delim(
    "~/Dropbox/denamic/data/set03/00_Set03_Clinic.txt",
    header = T,
    row.names = 1,
    sep = " "
  )
clinical <- clinical[-which(rownames(clinical) == "CYP_M_I6"), ]

#que observaciones son distintas entre cada matriz
#LOS    que estan en clinical y NO en cb,#2 observaciones
distintas:""VH_M_I4"" "CHLOR01_M_mM"
noestan <- NULL
for (element in rownames(clinical)) {
  if (sum(rownames(cb) == element) == 0) {
    print(element)
    noestan <- c(element, noestan)
  }
}

posNoestan <- NULL

for (element in noestan) {
  posNoestan <- c(which(rownames(clinical) == element), posNoestan)
}
posNoestan#6,23
clinical <- clinical[-posNoestan, ]

#los que estan en cb Y NO en clinical :"CHLOR01_M_I1"
noestan <- NULL
for (element in rownames(cb)) {
  if (sum(rownames(clinical) == element) == 0) {
    print(element)
    noestan <- c(element, noestan)
  }
}
posNoestan <- NULL
for (element in noestan) {
  posNoestan <- c(which(rownames(cb) == element), posNoestan)
}
posNoestan#6
cb <- cb[-posNoestan, ]

#ORDENADO FILAS MATRIZ CLINICAL, SEGUN EL ORDEN DE CB
clinical <- clinical[rownames(cb), , drop = FALSE]

###eliminando Radial Maze de clinical
clinical <- clinical[, 1:3]

###Imputacion valores faltantes de clinical por MICE
sum(apply(clinical, 2, function(colum) {
  sum(is.na(colum))
}))##14

imputed.clinical <- mice(clinical, method = "norm.predict")
# #despues deberia hacer: complete(prot03_CB_Imputada) para obtener la
matriz imputada
imputed.clinical <- complete(imputed.clinical)

clinical.org <- clinical

clinical <- imputed.clinical

#guardado matriz imputada de clinical
write.table(
  clinical,
  file =
"~/Dropbox/denamic/data/set03/02_Set03_ClinicalImputedByMice.txt",
  col.names = T,
  row.names = T,
  sep = "\t"
)

# centrado y escalado matrices ---------------------------------------
-----
#PROTEOM
cb <- scale(cb, center = TRUE, scale = FALSE)
#CLINICAL DATA
clinical <- scale(clinical, center = TRUE, scale = TRUE)

####CB QUEDARME SOLO CON LOS GENES CON VARIABILIDAD ELEVADA:TOTAL
TOTAL
load(

"~/Dropbox/denamic/data/set03/03_Set03_TotalTotalUnionOfProteinsCommin
gFromLimmaAndCv.RData"
)
cb <- cb[, TotalTotalUnionCB]

###voy a coger la design matrix para cb y quitarme las muestras y
proteinas que no me interesan
designCB <-
  read.delim(
"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_CB.txt
",
    header = T,
    row.names = 1,
    sep = " "
  )
designCB <- designCB[-which(rownames(designCB) == "CYPER_M_I6_CB"), ]
rownames(designCB) <- gsub("_CB", "", rownames(designCB))
designCB <- designCB[-(which(rownames(designCB) == noestan)), ]

# Seleccion del numero de componentes para el PLS --------------------
-----
myresult <-
  pls(cb,
      clinical,
      ncomp = 25,
      mode = "regression",
      scale = FALSE)
png(
  filename = "~/Dropbox/denamic/plots/Set03/PLS/PLS_Set03_MICE_NO_CYP_
ChoosingNofComponents_.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)
par(mfrow = c(1, 2))
barplot(myresult$explained_variance$X,
        las = 2,
        main = "X")
barplot(myresult$explained_variance$Y,
        las = 2,
        main = "Y")
dev.off()

#VAMOS A UTILIZAR 2 COMPONENTES

# # Aplicacion PLS  ---------------------------------------------------
-----
# eliminando muestra unica, CYP ----------------- --------
PLS_result <-
  pls(
    X = cb,
    Y = clinical,
    scale = FALSE,
    mode = "regression",
    ncomp = 2,
    logratio = "none",
    near.zero.var = F
  )

#arreglado
par(mar = c(5, 4, 2, 2))
png(
  filename = "~/Dropbox/denamic/plots/Set03/PLS/PLS_TFG.png",
  width = 1000,
  height = 500 ,
  units = "px",
  type = "cairo",
  pointsize = 22
```

```
)

#LIMITES EJES
minimoEjesX <-
  min(c(min(PLS_result$variates[[1]]), min(PLS_result$variates[[1]])))
maximoEjesX <-
  max(c(max(PLS_result$variates[[1]]), max(PLS_result$variates[[1]])))
limitesX <- c(minimoEjesX, maximoEjesX)

minimoEjesY <-
  min(c(min(PLS_result$variates[[2]]), min(PLS_result$variates[[2]])))
maximoEjesY <-
  max(c(max(PLS_result$variates[[2]]), max(PLS_result$variates[[2]])))
limitesY <- c(minimoEjesY, maximoEjesY)

plotIndiv(
  PLS_result,
  group = factor(designCB[, 1], levels = c(
    "CAR", "CHLOR01", "CHLOR03", "CHLOR1", "VH"
  )),
  # Levels: CAR CHLOR01 CHLOR03 CHLOR1 VH
  ind.names = FALSE,
  title = "PLS Set03 ",
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designCB[, 2]],
  pch.levels = designCB[, 2],
  col.per.group = MYcolors[c("CAR", "CHLOR01", "CHLOR03", "CHLOR1",
"VH")],
  #as.character(designCB[,1])],
  size.title = rel(1.4),
  comp = c(1, 2),
  xlim = list(limitesX, limitesY),
  ylim = list(limitesX, limitesY),
  size.xlabel = rel(1.2),
  size.ylabel = rel(1.2)
)

dev.off()

# CORRELATION CIRCLE PLOT, feature selection is needed
png(
  filename =
"~/Dropbox/denamic/plots/Set03/PLS/Set03_Pls_CorrelationCirclePlot.png
",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 25
)
plotVar(object = PLS_result,
        comp = 1:2,
        cex = c(4, 10))
dev.off()

#LOADING without clinical Radial Maze
png(filename =
"~/Dropbox/denamic/plots/Set03/PLS/PLS_LoadingClinicalWithoutRadialMaz
e_Y_IMPUTED_WITH_MICE_AND_NO_CYP.png")
```

```
par(mfrow = c(1, 1))
plot(
  PLS_result$loadings$Y,
  main =
"PLSclinical_WithoutRadialMaze_Y_imputed_with_MICE_AND_NO_CYP",
  col = "red",
  xlim = c(-0.5, 1.3)
)
text(
  x = PLS_result$loadings$Y[, 1],
  y = PLS_result$loadings$Y[, 2] ,
  labels = rownames(PLS_result$loadings$Y),
  pos = 4
)
dev.off()

# CROSS-Validation, as we have a small amount of observations
#POR LOOCV
myperfLoo = perf(
  PLS_result,
  validation = "loo",
  progressBar = TRUE,
  nrepeat = 100
)

#guardado de los objetos generados y utilizados en el pls que van a
ser utilizados posteriormente en el spls...
save(cb, clinical, designCB, myperfLoo, PLS_result, file =
"~/Dropbox/denamic/data/set03/PLS_objects_for_cb_sPLS.RData")

# HP -----------------------------------------------------------------
-----
#lectura matriz X=proteomics
hp <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_HP_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
rownames(hp) <- gsub("_HP", "", rownames(hp))

cborig <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_CB_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
rownames(cborig) <- gsub("_CB", "", rownames(cborig))

#voy a ver si son exactamante las mismas ratas, las de cb que las de
hp, puesto que de esta forma
#puedo mantenter el uso de la matriz de clinical imputada y eliminar
directamente las que no se
#necesitaban en cb
sum(sort(rownames(hp)) == sort(rownames(cborig)))#27, efectivamente
por lo que puedo aplicar lo expuesto

hp <- hp[-which(rownames(hp) == "CYPER_M_I6"), ]
```

```
#puesto que hay las mismas muestras para HP que para CB
#voy a eliminar la que estaba en cb, y por ello en hp, y no en
clinical que es "CHLOR01_M_I1"
hp <- hp[-which(rownames(hp) == "CHLOR01_M_I1"), ]


#voy a ordenar la matriz hp segun el orden de cb de esta forma la
tendremos ordenada tambien igual
#que clinical
# utilizaremos la misma matriz Y imputada
# y la misma matriz de disenyo
# clinical<-clinical[rownames(cb),,drop=FALSE]
hp <- hp[rownames(cb), , drop = FALSE]
designHP <- designCB

# centrado y escalado matrices ---------------------------------------
-----
#PROTEOM
hp <- scale(hp, center = TRUE, scale = FALSE)

####CB QUEDARME SOLO CON LOS GENES CON VARIABILIDAD ELEVADA:TOTAL
TOTAL
load(

"~/Dropbox/denamic/data/set03/03_Set03_TotalTotalUnionOfProteinsCommin
gFromLimmaAndCv.RData"
)
hp <- hp[, TotalTotalUnionHP]

# Seleccion del numero de componentes para el PLS --------------------
-----
myresult <-
  pls(hp,
      clinical,
      ncomp = 24,
      mode = "regression",
      scale = FALSE)
png(
  filename = "~/Dropbox/denamic/plots/Set03/PLS/PLS_Set03_MICE_NO_CYP_
ChoosingNofComponents_HP_.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)
par(mfrow = c(1, 2))
barplot(myresult$explained_variance$X,
        las = 2,
        main = "X")
barplot(myresult$explained_variance$Y,
        las = 2,
        main = "Y")
dev.off()

#VAMOS A UTILIZAR 2 COMPONENTES

# # Aplicacion PLS  ---------------------------------------------------
-----

PLS_result <-
  pls(
    X = hp,
    Y = clinical,
    scale = FALSE,
    mode = "regression",
    ncomp = 2,
    logratio = "none",
    near.zero.var = F
  )
#arreglado
par(mar = c(5, 4, 2, 2))
png(
  filename = "~/Dropbox/denamic/plots/Set03/PLS/PLS_HP_TFG.png",
  width = 1000,
  height = 500 ,
  units = "px",
  type = "cairo",
  pointsize = 22
)

#LIMITES EJES
minimoEjesX <-
  min(c(min(PLS_result$variates[[1]]), min(PLS_result$variates[[1]])))
maximoEjesX <-
  max(c(max(PLS_result$variates[[1]]), max(PLS_result$variates[[1]])))
limitesX <- c(minimoEjesX, maximoEjesX)

minimoEjesY <-
  min(c(min(PLS_result$variates[[2]]), min(PLS_result$variates[[2]])))
maximoEjesY <-
  max(c(max(PLS_result$variates[[2]]), max(PLS_result$variates[[2]])))
limitesY <- c(minimoEjesY, maximoEjesY)


plotIndiv(
  PLS_result,
  group = factor(designHP[, 1], levels = c(
    "CAR", "CHLOR01", "CHLOR03", "CHLOR1", "VH"
  )),
  # Levels: CAR CHLOR01 CHLOR03 CHLOR1 VH
  ind.names = FALSE,
  title = "PLS Set03",
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designHP[, 2]],
  pch.levels = designHP[, 2],
  col.per.group = MYcolors[c("CAR", "CHLOR01", "CHLOR03", "CHLOR1",
"VH")],
  #as.character(designCB[,1])],
  size.title = rel(1.4),
  comp = c(1, 2),
  xlim = list(limitesX, limitesY),
  ylim = list(limitesX, limitesY),
  size.xlabel = rel(1.2),
  size.ylabel = rel(1.2)
)

dev.off()

# CORRELATION CIRCLE PLOT, feature selection is needed
png(
  filename =
"~/Dropbox/denamic/plots/Set03/PLS/Set03_Pls_HP_CorrelationCirclePlot.
png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 25
)
plotVar(object = PLS_result,
        comp = 1:2,
        cex = c(4, 10))
dev.off()

#LOADING without clinical Radial Maze
png(filename =
"~/Dropbox/denamic/plots/Set03/PLS/PLS_HP_LoadingClinicalWithoutRadial
Maze_Y_IMPUTED_WITH_MICE_AND_NO_CYP.png")
par(mfrow = c(1, 1))
plot(
  PLS_result$loadings$Y,
  main =
"PLS_HP_clinical_WithoutRadialMaze_Y_imputed_with_MICE_AND_NO_CYP",
  col = "red",
  xlim = c(-0.5, 1.3)
)
text(
  x = PLS_result$loadings$Y[, 1],
  y = PLS_result$loadings$Y[, 2] ,
  labels = rownames(PLS_result$loadings$Y),
  pos = 4
)
dev.off()

# CROSS-Validation, as we have a small amount of observations
#POR LOOCV
myperfLooHP = perf(
  PLS_result,
  validation = "loo",
  progressBar = TRUE,
  nrepeat = 100
)

PLS_result_HP <- PLS_result
clinicalHP <- clinical

#guardado de los objetos generados y utilizados en el pls que van a
ser utilizados posteriormente en el spls...
save(hp, clinicalHP, designHP, myperfLooHP, PLS_result_HP, file =
"~/Dropbox/denamic/data/set03/PLS_objects_for_HP_sPLS.RData")
```

8.5 Estudio MSEP + sPLS

```r
#####################################################################
#####Estrategia 4.1 minimizacion num variables por componente####
#####################################################################
library(mixOmics)

source("~/Dropbox/denamic/scripts/Pesticides_Colors_and_GenderShapes.R
")

load("~/Dropbox/denamic/data/set03/PLS_objects_for_cb_sPLS.RData")

##############################################
####  VARIABLE SELECTION####
##############################################
Proteins_Number_1 <-
  c(
    seq(from = 5, to = 100, by = 5),
    seq(from = 100, to = 150, by = 10),
    200,
    seq(from = 250, to = 650, by = 100),
    658
  )
Proteins_Number_2 <- c(seq(from = 200, to = 400, by = 20), 658)
#X=cb, Y=clinical

matrizParaMSEP_4 <-
  matrix(ncol = length(Proteins_Number_2),
         nrow = length(Proteins_Number_1))
colnames(matrizParaMSEP_4) <- as.character(Proteins_Number_2)
rownames(matrizParaMSEP_4) <- as.character(Proteins_Number_1)

MSEP_perVariable_perProteinNumber_Matrixes_4 <-
  list(
    "MWM...Escape.Latency.Day.3" = matrizParaMSEP_4,
    "Rotarod...Time" = matrizParaMSEP_4,
    "Beam.Walking...Faults" = matrizParaMSEP_4
  )

#rows make reference to the number of variables for the C1 and columns
to the C2
for (nombre in names(MSEP_perVariable_perProteinNumber_Matrixes_4)) {
  for (numberfila in Proteins_Number_1) {
    for (numbercolumna in Proteins_Number_2) {
      SPLSresult <-
        mixOmics::spls(
          X = cb,
          Y = clinical,
          ncomp = 2,
          mode = 'regression',
          scale = FALSE,
          keepX = c(numberfila, numbercolumna),
          keepY = c(3, 3)
        )

      # LOO CV
      spls.loo <-
        perf(
          SPLSresult,
          ncomp = 2,
          mode = 'regression',
          keepX = c(numberfila, numbercolumna),
          validation = 'loo'
        )
      valorMSEP <-
        spls.loo$MSEP[which(rownames(spls.loo$MSEP) == nombre), 2]


MSEP_perVariable_perProteinNumber_Matrixes_4[[which(names(MSEP_perVari
able_perProteinNumber_Matrixes_4) ==

nombre)]][which(rownames(matrizParaMSEP_4) == numberfila),
which(colnames(matrizParaMSEP_4) ==

numbercolumna)] <- valorMSEP
    }

  }

}
MSEP_CB_perVariable_perProteinNumber_Matrixes_4 <-
  MSEP_perVariable_perProteinNumber_Matrixes_4
MSEP_CB_perVariable_perProteinNumber_Matrixes_4


#############
###PLOT 4 MWM
#############
load(

"~/Dropbox/denamic/data/set03/33MSEP_CB_perVariable_perProteinNumber_M
atrixes_4_1.RData"
)

MwmMSEP <- MSEP_CB_perVariable_perProteinNumber_Matrixes_4[[1]]
colores <-
  c(
    "orange",
    "deepskyblue1",
    "blue",
    "blue",
    "darkgreen",
    "green",
    "yellow",
    "yellow",
    "blueviolet",
    "blueviolet",
    "black",
    "red"
  )
png(
  filename =
"~/Dropbox/denamic/plots/Set03/EstrategiaSeleccionVariablesPLS/2aEstra
tegia/4_1_Intento/4_1_MSEP_CB_MWM_DependingOnTheNumberOfVariables.png"
,
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

par(mar = c(5.1, 5.1, 4.1, 2.1))
```

```
X <- as.integer(rownames(MwmMSEP))
minY <- min(MwmMSEP)
maxY <- max(MwmMSEP)
count <- 0
legend_content <- NULL

for (element in colnames(MwmMSEP)) {
  count <- count + 1
  if (count == 1) {
    plot(
      x = X,
      y = MwmMSEP[, count],
      xlab = "Number of Proteins in the C1",
      ylab = "MSEP",
      type = "l",
      col = colores[count],
      xlim = c(5, 658),
      ylim = c(minY, maxY),
      main = "MSEP_CB_MWM_DependingOnTheNumberOfVariables",
      cex.lab = 2,
      cex.axis = 1.5,
      cex.main = 2,
      lwd = 2
    )

    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
  }
  else{
    par(new = TRUE)
    plot(
      x = X,
      y = MwmMSEP[, count],
      axes = FALSE,
      xlab = "",
      ylab = "",
      type = "l",
      col = colores[count],
      ylim = c(minY, maxY),
      xlim = c(5, 658),
      lwd = 2
    )
    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
  }
}

# legend_content
legend(
  "topright",
  legend = legend_content,
  col = names(legend_content),
  fill = names(legend_content) ,
  ncol = 6,
  title = "Number of proteins in the Component 2",
  cex = 1.6,
  pt.cex = 1.3
)
abline(h = MwmMSEP[33, 12], col = "red", lwd = 2)
```

```
dev.off()

#############################
#####PLOT BEAM WALKING#######
#############################
BeamWalking <-
  MSEP_CB_perVariable_perProteinNumber_Matrixes_4[[which(
    names(MSEP_CB_perVariable_perProteinNumber_Matrixes_4) ==
"Beam.Walking...Faults"
  )]]
colores <-
  c(
    "orange",
    "deepskyblue1",
    "blue",
    "blue",
    "darkgreen",
    "green",
    "yellow",
    "yellow",
    "blueviolet",
    "blueviolet",
    "black",
    "red"
  )
png(
  filename =
"~/Dropbox/denamic/plots/Set03/EstrategiaSeleccionVariablesPLS/2aEstra
tegia/4_1_Intento/MSEP_CB_Beam_Walking_DependingOnTheNumberOfVariables
4.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

par(mar = c(5.1, 5.1, 4.1, 2.1))
X <- as.integer(rownames(BeamWalking))
minY <- min(BeamWalking)
maxY <- max(BeamWalking)
count <- 0
legend_content <- NULL

for (element in colnames(BeamWalking)) {
  count <- count + 1
  if (count == 1) {
    plot(
      x = X,
      y = BeamWalking[, count],
      xlab = "Number of Proteins in the C1",
      ylab = "MSEP",
      type = "l",
      col = colores[count],
      ylim = c(minY, maxY),
      xlim = c(5, 658),
      main = "MSEP_CB_Beam_Walking_DependingOnTheNumberOfVariables",
      cex.lab = 2,
      cex.axis = 1.5,
      cex.main = 2,
      lwd = 2
    )
    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
  }
  else{
    par(new = TRUE)
    plot(
      x = X,
      y = BeamWalking[, count],
      axes = FALSE,
      xlab = "",
      ylab = "",
      type = "l",
      col  = colores[count],
      ylim = c(minY, maxY),
      xlim = c(5, 658),
      lwd = 2
    )
    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
  }
}

# legend_content
legend(
  "topright",
  legend = legend_content,
  col = names(legend_content),
  fill = names(legend_content) ,
  ncol = 6,
  title = "Number of proteins in the Component 2",
  cex = 1.6,
  pt.cex = 1.3
)

abline(h = BeamWalking[33, 12], col = "red", lwd = 2)

dev.off()

#################
###PLOT 4 ROTAROD
#################
RotarodMSEP <-

MSEP_CB_perVariable_perProteinNumber_Matrixes_4[[which(names(MSEP_CB_p
erVariable_perProteinNumber_Matrixes_4) ==

"Rotarod...Time")]]
colores <-
  c(
    "orange",
    "deepskyblue1",
    "blue",
    "blue",
    "darkgreen",
    "green",
    "yellow",
    "yellow",
    "blueviolet",
    "blueviolet",
    "black",
    "red"
  )
png(
  filename =
"~/Dropbox/denamic/plots/Set03/EstrategiaSeleccionVariablesPLS/2aEstra
tegia/4_1_Intento/MSEP_CB_Rotarod_DependingOnTheNumberOfVariables4_Ver
tical_ABLINE.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

par(mar = c(5.1, 5.1, 4.1, 2.1))
X <- as.integer(rownames(RotarodMSEP))
minY <- min(RotarodMSEP)
maxY <- max(RotarodMSEP)
count <- 0
legend_content <- NULL
for (element in colnames(RotarodMSEP)) {
  count <- count + 1
  if (count == 1) {
    plot(
      x = X,
      y = RotarodMSEP[, count],
      xlab = "Number of Proteins in the C1",
      ylab = "MSEP",
      type = "l",
      col = colores[count],
      ylim = c(minY, maxY),
      xlim = c(5, 658),
      main = "MSEP_CB_Rotarod_DependingOnTheNumberOfVariables",
      cex.lab = 2,
      cex.axis = 1.5,
      cex.main = 2,
      lwd = 2
    )

    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
  }
  else{
    par(new = TRUE)
    plot(
      x = X,
      y = RotarodMSEP[, count],
      axes = FALSE,
      xlab = "",
      ylab = "",
      type = "l",
      col = colores[count],
      ylim = c(minY, maxY),
      xlim = c(5, 658),
      lwd = 2
    )
    incorporacionLeyenda <- element
    names(incorporacionLeyenda) <- colores[count]
    legend_content <- c(legend_content, incorporacionLeyenda)
```

```r
  }
}

# legend_content
legend(
  "topright",
  legend = legend_content,
  col = names(legend_content),
  fill = names(legend_content) ,
  ncol = 6,
  title = "Number of proteins in the Component 2",
  cex = 1.6,
  pt.cex = 1.3
)

abline(
  h = RotarodMSEP[33, 12],
  col = c("red", "green"),
  lwd = 2,
  v = 65
)

dev.off()


#LIMITES EJES

####Bar plot to compare  MSEP of SPLS vs MSEP of PLS
mySPresult <-
  mixOmics::spls(
    X = cb,
    Y = clinical,
    ncomp = 2,
    mode = 'regression',
    scale = FALSE,
    keepX = c(65, 300),
    keepY = c(3, 3)
  )

SPLS_65_300_myperfLoo <-
  perf(
    mySPresult,
    ncomp = 2,
    mode = 'regression',
    keepX = c(65, 300),
    validation = 'loo'
  )

MSEP_PLS_perClinicalVariable <- myperfLoo$MSEP[, 2]
MSEP_SPLS_perClinicalVariable <- SPLS_65_300_myperfLoo$MSEP[, 2]

NombresFilas <- names(MSEP_CB_perVariable_perProteinNumber_Matrixes_4)

BarplotContent <- matrix(ncol = 2, nrow = 3)
rownames(BarplotContent) <- NombresFilas
BarplotContent <- as.data.frame(BarplotContent)
BarplotContent$MSEP_PLS <- MSEP_PLS_perClinicalVariable
BarplotContent$MSEP_SPLS_65_300 <- MSEP_SPLS_perClinicalVariable

BarplotContent <- BarplotContent[, -c(1, 2)]
BarplotContent <- as.data.frame(BarplotContent)
BarplotContent <- t(BarplotContent)
#Plots from the data in BarplotContent
# Grouped Bar Plot
png(
  filename =
"~/Dropbox/denamic/plots/Set03/EstrategiaSeleccionVariablesPLS/Set03_M
SEP_PLSvsSPLS_65_300AllClinicalVariables.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 20
)
par(mfrow = c(1, 1))
barplot(
  BarplotContent,
  main = "MSEP for each Test",
  xlab = "",
  col = c("red", "blue"),
  ylab = "MSEP 2C",
  legend = rownames(BarplotContent),
  beside = TRUE
)
dev.off()

########################
###spls application  C1=65, C2=300
########################
minEjesX <- min(mySPresult$variates[[1]])
maxEjesX <- max(mySPresult$variates[[1]])
limitEjesX <- c(minEjesX, maxEjesX)

minEjesY <- min(mySPresult$variates[[2]])
maxEjesY <- max(mySPresult$variates[[2]])
limitEjesY <- c(minEjesY, maxEjesY)


png(
  filename = "~/Dropbox/denamic/plots/Set03/SPLS/sPLS_TFG.png",
  width = 1000,
  height = 500 ,
  units = "px",
  type = "cairo",
  pointsize = 22
)
par(mfrow = c(1, 2))
plotIndiv(
  mySPresult,
  group = factor(designCB[, 1], levels = c(
    "CAR", "CHLOR01", "CHLOR03", "CHLOR1", "VH"
  )),
  # Levels: CAR CHLOR01 CHLOR03 CHLOR1 VH
  ind.names = F,
  title = "sPLS Set03 ",
  legend = F,
  style = "graphics",
  #"graphics"
  ellipse = F,
  pch = myPCH[designCB[, 2]],
  col.per.group = MYcolors[c("CAR", "CHLOR01", "CHLOR03", "CHLOR1",
"VH")],
  #as.character(designCB[,1])],
  size.title = rel(1.4),
  pch.levels = designCB[, 2],
  comp = c(1, 2),
  xlim = list(limitEjesX, limitEjesY),
  ylim = list(limitEjesX, limitEjesY),
  size.xlabel = rel(1.2),
  size.ylabel = rel(1.2)

)
dev.off()

# CORRELATION CIRCLE PLOT, feature selection is needed
png(
  filename =
"~/Dropbox/denamic/plots/Set03/SPLS/Set03_sPls_65_300_CorrelationCircl
ePlot.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 25
)
plotVar(object = mySPresult,
        comp = 1:2,
        cex = c(4, 10))
dev.off()


png(
  filename =
"~/Dropbox/denamic/plots/Set03/SPLS/NoNameSet03_sPls_CorrelationCircle
PlotNoNames.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 25
)
plotVar(
  object = mySPresult,
  comp = 1:2,
  cex = c(4, 10),
  var.names = c(TRUE, FALSE)
)
dev.off()

#correlation0.5
png(
  filename =
"~/Dropbox/denamic/plots/Set03/SPLS/03_sPls_CB_0_3_CorrelationCirclePl
otNoNames.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo",
  pointsize = 25
)
plotVar(
  object = mySPresult,
  comp = 1:2,
  cex = c(4, 10),
  var.names = c(TRUE, F),
  cutoff = 0.3,
  rad.in = 0.4
)
dev.off()

CorrelationPlotData_0_3_CB_spls <-
  plotVar(
    object = mySPresult,
    comp = 1:2,
    cex = c(4, 10),
    var.names = c(TRUE, F),
    cutoff = 0.3,
    rad.in = 0.4
  )
#
# Guardado de objeto , que contenga las N prot seleccionadas, y  que
esten separadas tambien por componente--------


TotalSelectedSPLS <- abs(mySPresult$loadings$X) > 0
# # TotalSelectedSPLS
#
table(TotalSelectedSPLS[, 1]) #65  para el primer componente
table(TotalSelectedSPLS[, 2]) #300 para el segundo componente

Componente1_Selected <-
  rownames(TotalSelectedSPLS)[TotalSelectedSPLS[, 1]]
Componente2_Selected <-
  rownames(TotalSelectedSPLS)[TotalSelectedSPLS[, 2]]
interseccion <- character(0)
for (element in Componente1_Selected) {
  if (element %in% Componente2_Selected == TRUE) {
    interseccion <- c(element, interseccion)
  }
}

CB_prot_loadings_sign <- mySPresult$loadings

save(
  Componente1_Selected,
  Componente2_Selected,
  interseccion,
  CB_prot_loadings_sign,
  file =
"~/Dropbox/denamic/data/set03/SelectedVariablesPerComponenent_and_sign
_CB_AND_Intersection4_1.RData"
)

save(CorrelationPlotData_0_3_CB_spls, file =
"~/Dropbox/denamic/data/set03/correlationPlotData_0_3_CB_spls.RData")
```

8.6 Multi-block PLS

```
################################
## multiblock  set01 #
################################
library(mixOmics)
library(ggplot2)

source("~/Dropbox/denamic/scripts/Pesticides_Colors_and_GenderShapes.R
")

# cargado matrices para multiblock
load(file =
"~/Dropbox/denamic/data/set01/MatricesParaTestsMultiBlockPLS.RData")

data = list(met = met, prot = prot)

#1.0 relacion matrices omicas
design = matrix(
  1,
  ncol = length(data),
  nrow = length(data),
  dimnames = list(names(data), names(data))
)
diag(design) = 0

# set number of component per data set
ncomp = c(2)

TCGA.block.pls = block.pls(
  X = data,
  Y = clinic,
  design = design,
  scale = F
)

# in plotindiv we color the samples per breast subtype group but the
method is unsupervised!
# here Y is the protein data set
plotIndiv(
  TCGA.block.pls,
  group = designMatrix[, 1],
  ind.names = FALSE,
  legend = T
) #funciona

#LIMITES EJES
minimoMet <-
  min(c(
    min(TCGA.block.pls$variates[[1]]),
    min(TCGA.block.pls$variates[[1]])
  ))
maximoMet <-
  max(c(
    max(TCGA.block.pls$variates[[1]]),
    max(TCGA.block.pls$variates[[1]])
  ))
limitesMet <- c(minimoMet, maximoMet)

minimoProt <-
  min(c(
    min(TCGA.block.pls$variates[[2]]),
    min(TCGA.block.pls$variates[[2]])
  ))
maximoProt <-
  max(c(
    max(TCGA.block.pls$variates[[2]]),
    max(TCGA.block.pls$variates[[2]])
  ))
limitesProt <- c(minimoProt, maximoProt)

minimoClinic <-
  min(c(
    min(TCGA.block.pls$variates[[3]]),
    min(TCGA.block.pls$variates[[3]])
  ))
maximoClinic <-
  max(c(
    max(TCGA.block.pls$variates[[3]]),
    max(TCGA.block.pls$variates[[3]])
  ))
limitesClinic <- c(minimoClinic, maximoClinic)

png(
  filename =
"~/Dropbox/denamic/plots/Set01/MultiBlockPLS/MultiBlockTFG.png",
  width = 900,
  height = 300 ,
  units = "px",
  type = "cairo",
  pointsize = 22
)
par(mfrow = c(1, 3))

plotIndiv(
  TCGA.block.pls,
  group = factor(designMatrix[, 1], levels = c("VH", "END", "CYP")),
  # Levels: CAR CHLOR01 CHLOR03 CHLOR1 VH
  ind.names = FALSE,
  title = "Multi-block PLS ",
  legend = F,
  style = "graphics",
  ellipse = F,
  pch = myPCH[designMatrix[, 2]],
  pch.levels = designMatrix[, 2],
  col.per.group = MYcolors[c("VH", "END", "CYP")],
  #as.character(designCB[,1])],
  size.title = rel(1.5),
  comp = c(1, 2),
  xlim = list(limitesMet, limitesProt, limitesClinic),
  ylim = list(limitesMet, limitesProt, limitesClinic),
  size.xlabel = rel(1.3),
  size.ylabel = rel(1.3)
)

dev.off()

save(TCGA.block.pls, file =
"~/Dropbox/denamic/data/set01/Multiblock_Result.RData")

#####################
### Loadings analysis
```

```r
#####################

#metabolites data
#################
LoadingsMet <- TCGA.block.pls$loadings$met
LoadingsMet <- as.data.frame(LoadingsMet)

# COMP 1
png(
  filename =
"~/Dropbox/denamic/plots/Set01/AnalisisLoadings/MetC1.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

ggplot(data = LoadingsMet, aes(abs(LoadingsMet$`comp 1`))) +
  geom_histogram(
    breaks = seq(min(abs(
      LoadingsMet$`comp 1`
    )), max(abs(
      LoadingsMet$`comp 1`
    )), by = 0.003),
    col = "red",
    fill = "blue",
    alpha = .5
  ) +
  labs(title = "Loadings Metabolomics C1") +
  labs(x = "Loading values", y = "Count") +
  theme(
    axis.text = element_text(size = 40),
    axis.title = element_text(size = 40, face = "bold"),
    title = element_text(size = 60, face = "bold")
  )

dev.off()

#Comp2
png(
  filename =
"~/Dropbox/denamic/plots/Set01/AnalisisLoadings/MetC2.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

ggplot(data = LoadingsMet, aes(abs(LoadingsMet$`comp 2`))) +
  geom_histogram(
    breaks = seq(min(abs(
      LoadingsMet$`comp 2`
    )), max(abs(
      LoadingsMet$`comp 2`
    )), by = 0.003),
    col = "red",
    fill = "blue",
    alpha = .5
  ) +
  labs(title = "Loadings Metabolomics C2") +
  labs(x = "Loading values", y = "Count") +
  theme(
    axis.text = element_text(size = 40),
    axis.title = element_text(size = 40, face = "bold"),
    title = element_text(size = 60, face = "bold")
  )

dev.off()

###############
## PROTS
###############
LoadingsProt <- TCGA.block.pls$loadings$prot
LoadingsProt <- as.data.frame(LoadingsProt)

# COMP 1
png(
  filename =
"~/Dropbox/denamic/plots/Set01/AnalisisLoadings/ProtC1.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

ggplot(data = LoadingsProt, aes(abs(LoadingsProt$`comp 1`))) +
  geom_histogram(
    breaks = seq(min(abs(
      LoadingsProt$`comp 1`
    )), max(abs(
      LoadingsProt$`comp 1`
    )), by = 0.003),
    col = "red",
    fill = "blue",
    alpha = .5
  ) +
  labs(title = "Loadings Proteomics C1") +
  labs(x = "Loading values", y = "Count") +
  theme(
    axis.text = element_text(size = 40),
    axis.title = element_text(size = 40, face = "bold"),
    title = element_text(size = 60, face = "bold")
  )

dev.off()

#Comp2
png(
  filename =
"~/Dropbox/denamic/plots/Set01/AnalisisLoadings/ProtC2.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

ggplot(data = LoadingsProt, aes(abs(LoadingsProt$`comp 2`))) +
  geom_histogram(
    breaks = seq(min(abs(
      LoadingsProt$`comp 2`
    )), max(abs(
      LoadingsProt$`comp 2`
    )), by = 0.003),
    col = "red",
    fill = "blue",
    alpha = .5
  ) +
  labs(title = "Loadings Proteomics C2") +
  labs(x = "Loading values", y = "Count") +
  theme(
    axis.text = element_text(size = 40),
    axis.title = element_text(size = 40, face = "bold"),
    title = element_text(size = 60, face = "bold")
  )

dev.off()

LoadingsClinic <- TCGA.block.pls$loadings$Y
```

8.7 Selección por VIP

```r
###########################
## VIP study ##############
###########################

library(mixOmics)

load(file = "~/Dropbox/denamic/data/set01/Multiblock_Result.RData")


##VIP function
Victor_VIP <- function(object, omic) {
  W = object$loadings[[omic]]
  H = object$ncomp
  q = ncol(object$X$Y)
  p = ncol(object$X[[omic]])
  VIP = matrix(0, nrow = p, ncol = H)
  cor2 = cor(object$X$Y, object$variates[[omic]], use = "pairwise") ^
2
  cor2 = as.matrix(cor2, nrow = q)
  VIP[, 1] = W[, 1] ^ 2
  if (H > 1) {
    for (h in 2:H) {
      if (q == 1) {
        Rd = cor2[, 1:h]
        VIP[, h] = Rd %*% t(W[, 1:h] ^ 2) / sum(Rd)
      }
      else {
        Rd = apply(cor2[, 1:h], 2, sum)
        VIP[, h] = Rd %*% t(W[, 1:h] ^ 2) / sum(Rd)
      }
    }
  }
  VIP = sqrt(p * VIP)
  rownames(VIP) = rownames(W)
  colnames(VIP) = paste("comp", 1:H)
  return(invisible(VIP))
}
prot_vip <- Victor_VIP(object = TCGA.block.pls, omic = "prot")
met_vip <- Victor_VIP(object = TCGA.block.pls, omic = "met")

# selection of metabolites with vip>1
#per component

Met_comp1 <- names(which(met_vip[, "comp 1"] > 1))
Met_comp2 <- names(which(met_vip[, "comp 2"] > 1))

#all
Met_all <- union(Met_comp1, Met_comp2)

#selection of proteins with vip>1
#per component
Prot_comp1 <- names(which(prot_vip[, "comp 1"] > 1))
Prot_comp2 <- names(which(prot_vip[, "comp 2"] > 1))

#all
Prot_all <- union(Prot_comp1, Prot_comp2)

save(Met_comp1,
  Met_comp2,
  Met_all,
  Prot_comp1,
  Prot_comp2,
  Prot_all,
  file = "~/Dropbox/denamic/data/set01/VipResultsPerOmic.RData"
)
```

8.8 Correlación variables con tests

```
######################################################################
#####
## set01, correlation variables with motor and cognitie skills per
gender #
######################################################################
#####
library(dplyr)

load(file = "~/Dropbox/denamic/data/set01/VipResultsPerOmic.RData")

# to get the cb and clinical matrixes already used for the pls and
spls analysis, which are centered and scaled ( just clinical)
load("~/Dropbox/denamic/data/set01/MatricesParaTestsMultiBlockPLS.RDat
a")

##Proteins with Tests
ProteinsUnionComponents <- Prot_all
FeaturedCB <- prot[, ProteinsUnionComponents]

FeaturedCBSorted <- FeaturedCB
clinicalSorted <- clinic
designCBSorted <- designMatrix

##Separating Matrixes prot and clinical per gender
FeaturedCBSorted <- as.data.frame(FeaturedCBSorted)
clinicalSorted <- as.data.frame(clinicalSorted)
designCBSorted <- as.data.frame(designCBSorted)

FeaturedCBSorted$Gender <- designCBSorted$Gender
clinicalSorted$Gender <- designCBSorted$Gender

### seleccion por sexo

##seleccion   hembras
FeaturedCBSorted$ids <- rownames(FeaturedCBSorted)
clinicalSorted$ids <- rownames(clinicalSorted)

FeaturedCBSortedFemales <- FeaturedCBSorted %>% filter(Gender == "F")
rownames(FeaturedCBSortedFemales) <- FeaturedCBSortedFemales$ids
FeaturedCBSortedFemales$ids <- NULL
FeaturedCBSortedFemales$Gender <- NULL

clinicalSortedFemales <- clinicalSorted %>% filter(Gender == "F")
rownames(clinicalSortedFemales) <- clinicalSortedFemales$ids
clinicalSortedFemales$ids <- NULL
clinicalSortedFemales$Gender <- NULL

###seleccion machos
FeaturedCBSortedMales <- FeaturedCBSorted %>% filter(Gender == "M")
rownames(FeaturedCBSortedMales) <- FeaturedCBSortedMales$ids
FeaturedCBSortedMales$ids <- NULL
FeaturedCBSortedMales$Gender <- NULL

clinicalSortedMales <- clinicalSorted %>% filter(Gender == "M")
rownames(clinicalSortedMales) <- clinicalSortedMales$ids
clinicalSortedMales$ids <- NULL
clinicalSortedMales$Gender <- NULL

##correlation study
```

```r
nombrescolumnas <- c(colnames(clinic), "Most_Correlated", "Value")

CorrelationDataFrame <- matrix(data = rep(0),
                               ncol = 7,
                               nrow = 265)
colnames(CorrelationDataFrame) <- nombrescolumnas
rownames(CorrelationDataFrame) <- ProteinsUnionComponents

CorrelationDataFrame <- as.data.frame(CorrelationDataFrame)

#################
#Males
#FOR PEARSON
#################
CorrelationDataFramePearsonMales <- CorrelationDataFrame

##filling the matrix,3 first columns
for (protein in ProteinsUnionComponents) {
  for (ClinicalTest in colnames(clinicalSortedMales)) {
    Corr <-
      cor(x = FeaturedCBSortedMales[, protein],
          y = clinicalSortedMales[, ClinicalTest],
          method = "pearson")
    CorrelationDataFramePearsonMales[protein, ClinicalTest] <- Corr
  }
}

#filling the last two columns
CorrelationDataFramePearsonMales <-
  as.data.frame(CorrelationDataFramePearsonMales)

for (Line in 1:nrow(CorrelationDataFramePearsonMales)) {
  TestPos <-
    which(abs(CorrelationDataFramePearsonMales[Line, ][1:5]) ==
max(abs(CorrelationDataFramePearsonMales[Line, ][1:5])))
  Test <- colnames(CorrelationDataFramePearsonMales)[TestPos]
  Value <- CorrelationDataFramePearsonMales[Line, TestPos]
  CorrelationDataFramePearsonMales[Line, "Most_Correlated"] <- Test
  CorrelationDataFramePearsonMales[Line, "Value"] <- Value
}

# Grouping the proteins by most correlated tests
PearsonM <- CorrelationDataFramePearsonMales
PearsonM <- PearsonM[order(PearsonM$Most_Correlated), ]

##### MOTOR TESTS
#BEAM WALKING
PearsonBeamWalkingM <- PearsonM
PearsonBeamWalkingM$Prots <- rownames(PearsonM)
PearsonBeamWalkingM <-
  PearsonBeamWalkingM %>% filter(Most_Correlated ==
"Beam.Walking...Faults")
rownames(PearsonBeamWalkingM) <- PearsonBeamWalkingM$Prots
PearsonBeamWalkingM$Prots <- NULL
#ROTAROD
PearsonRotarodM <- PearsonM
PearsonRotarodM$Prots <- rownames(PearsonM)
PearsonRotarodM <-
  PearsonRotarodM %>% filter(Most_Correlated == "Rotarod...Time")
rownames(PearsonRotarodM) <- PearsonRotarodM$Prots
PearsonRotarodM$Prots <- NULL
```

```
#### COGNITIVE TESTS
#MWM
PearsonMWMM <- PearsonM
PearsonMWMM$Prots <- rownames(PearsonM)
PearsonMWMM <-
  PearsonMWMM %>% filter(Most_Correlated ==
"MWM...Escape.Latency.Day.3")
rownames(PearsonMWMM) <- PearsonMWMM$Prots
PearsonMWMM$Prots <- NULL

save(PearsonM,
     PearsonBeamWalkingM,
     PearsonRotarodM,
     PearsonMWMM,
     file =
"~/Dropbox/denamic/data/set03/06_DonePerGender_Males_CorrelatedProtein
sWithTests_ForFunctionalEnrichment.RData")

#################
#Females
#FOR PEARSON
#################
CorrelationDataFramePearsonFemales <- CorrelationDataFrame

##filling the matrix,3 first columns
for (protein in ProteinsUnionComponents) {
  for (ClinicalTest in colnames(clinicalSortedFemales)) {
    Corr <-
      cor(x = FeaturedCBSortedFemales[, protein],
          y = clinicalSortedFemales[, ClinicalTest],
          method = "pearson")
    CorrelationDataFramePearsonFemales[protein, ClinicalTest] <- Corr
  }
}

#filling the last two columns
CorrelationDataFramePearsonFemales <-
  as.data.frame(CorrelationDataFramePearsonFemales)

for (Line in 1:nrow(CorrelationDataFramePearsonFemales)) {
  TestPos <-
    which(abs(CorrelationDataFramePearsonFemales[Line, ][1:3]) ==
max(abs(CorrelationDataFramePearsonFemales[Line, ][1:3])))
  Test <- colnames(CorrelationDataFramePearsonFemales)[TestPos]
  Value <- CorrelationDataFramePearsonFemales[Line, TestPos]
  CorrelationDataFramePearsonFemales[Line, "Most_Correlated"] <- Test
  CorrelationDataFramePearsonFemales[Line, "Value"] <- Value

}

# Grouping the proteins by most correlated tests
PearsonF <- CorrelationDataFramePearsonFemales
PearsonF <- PearsonF[order(PearsonF$Most_Correlated), ]

##### MOTOR TESTS
#BEAM WALKING
PearsonBeamWalkingF <- PearsonF
PearsonBeamWalkingF$Prots <- rownames(PearsonF)
PearsonBeamWalkingF <-
```

```
  PearsonBeamWalkingF %>% filter(Most_Correlated ==
"Beam.Walking...Faults")
rownames(PearsonBeamWalkingF) <- PearsonBeamWalkingF$Prots
PearsonBeamWalkingF$Prots <- NULL

#ROTAROD
PearsonRotarodF <- PearsonF
PearsonRotarodF$Prots <- rownames(PearsonF)
PearsonRotarodF <-
  PearsonRotarodF %>% filter(Most_Correlated == "Rotarod...Time")
rownames(PearsonRotarodF) <- PearsonRotarodF$Prots
PearsonRotarodF$Prots <- NULL

#### COGNITIVE TESTS
#MWM
PearsonMWMF <- PearsonF
PearsonMWMF$Prots <- rownames(PearsonF)
PearsonMWMF <-
  PearsonMWMF %>% filter(Most_Correlated ==
"MWM...Escape.Latency.Day.3")
rownames(PearsonMWMF) <- PearsonMWMF$Prots
PearsonMWMF$Prots <- NULL

save(PearsonF,
     PearsonBeamWalkingF,
     PearsonRotarodF,
     PearsonMWMF,
     file =
"~/Dropbox/denamic/data/set03/06_DonePerGender_Females_CorrelatedProte
insWithTests_ForFunctionalEnrichment.RData")
###metab with Tests
```

8.9 Enriquecimiento funcional

```
######################################################################
##############################################
## Per Gender Functional Enrichment Proteins correlated positvively
and negatively with Motor and Cogntivie skills #
######################################################################
##############################################
library(biomaRt)
library(ggplot2)

#Functions For the Functional Enrichment Analysis
EnrichALLterms = function (test,
                           notTest,
                           annotation,
                           p.adjust.method = "fdr") {
  annot2test = unique(annotation[, 2])
  #IN TEST THE DIFFERENTIALLY expressed genes, in not test the rest of
genes
  resultat = t(
    sapply(
      annot2test,
      Enrich1term,
      test = test,
      notTest = notTest,
      annotation = annotation
    )
  )

  return (data.frame(
    resultat,
    "adjPval" = p.adjust(as.numeric(resultat[, "pval"]), method =
p.adjust.method),
    stringsAsFactors = F
  ))

}

Enrich1term = function (term, test, notTest, annotation) {
  annotTest = length(intersect(test, annotation[annotation[, 2] ==
term, 1]))

  if ((annotTest) > 0) {
    annotNOTtest = length(intersect(notTest, annotation[annotation[,
2] == term, 1]))
    mytest = matrix(c(
      annotTest,
      length(test) - annotTest,
      annotNOTtest,
      length(notTest) - annotNOTtest
    ),
    ncol = 2)
    resultat = c(
      term,
      annotTest,
      length(test),
      annotNOTtest,
      length(notTest),
      fisher.test(mytest, alternative = "greater")$p.value
    )
    names(resultat) = c("term",
                        "annotTest",
                        "test",
                        "annotNotTest",
                        "notTest",
                        "pval")
  } else {
    resultat = c(term, 0, 0, 0, 0, 100)
    names(resultat) = c("term",
                        "annotTest",
                        "test",
                        "annotNotTest",
                        "notTest",
                        "pval")
  }

  return(resultat)

}
##Preparacion fichero anotacion
biomartRnorvegicus = useMart(biomart = "ENSEMBL_MART_ENSEMBL",
                             dataset = "rnorvegicus_gene_ensembl",
                             host = "dec2016.archive.ensembl.org")
atributos = listAttributes(biomartRnorvegicus)

#FULL PROT = 1223 PROT DEL ANALISIS POST NA SELECCION
FULLPROTCB <-
  read.delim(
    "~/Dropbox/denamic/data/set03/01_Set03_CB_Imputada_Por_MICE.txt",
    header = TRUE,
    row.names = 1,
    sep = " "
  )

totesprotCB <- colnames(FULLPROTCB)

myannotTOTALCBbiomart = getBM(
  attributes = c("uniprot_swissprot", "uniprot_genename",
"name_1006"),
  filters = "uniprot_swissprot" ,
  values = totesprotCB,
  mart = biomartRnorvegicus
)

#####################
#MALES
#####################

load(file =
"~/Dropbox/denamic/data/set03/06_DonePerGender_Males_CorrelatedProtein
sWithTests_ForFunctionalEnrichment.RData")

###FUNCTIONAL ENRICHMENT CON LAS PROTEINAS CORRELACIONADAS
POSITIVAMENTE CON LA MEJORA MOTORA

# Las correlacionadas positivamente con Rotarod y Negativamente con
Beam Walking
MejoraMotoraProtsMales <-
  c(rownames(PearsonRotarodM)[PearsonRotarodM$Value > 0],
rownames(PearsonBeamWalkingM)[PearsonBeamWalkingM$Value < 0])

myEnrichResultsMejoraMotoraMales = EnrichALLterms(
  test = MejoraMotoraProtsMales,
  notTest = setdiff(totesprotCB, MejoraMotoraProtsMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

### CORR POSITIVA ROTAROD
CorrPosRotarodMales <-
  rownames(PearsonRotarodM)[PearsonRotarodM$Value > 0]

FECorrPosRotarodMales = EnrichALLterms(
  test = CorrPosRotarodMales,
  notTest = setdiff(totesprotCB, CorrPosRotarodMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

##CORR NEGATIVA ROTAROD
CorrNegRotarodMales <-
  rownames(PearsonRotarodM)[PearsonRotarodM$Value < 0]

FECorrNegRotarodMales = EnrichALLterms(
  test = CorrNegRotarodMales,
  notTest = setdiff(totesprotCB, CorrNegRotarodMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

##CORR POSITIVA BEAM WALKING
CorrPosBWMales <-
  rownames(PearsonBeamWalkingM)[PearsonBeamWalkingM$Value > 0]

FECorrPosBWMales = EnrichALLterms(
  test = CorrPosBWMales,
  notTest = setdiff(totesprotCB, CorrPosBWMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)
View(FECorrPosBWMales)

## CORR NEGATIVA BEAM WALKING

CorrNegBWMales <-
  rownames(PearsonBeamWalkingM)[PearsonBeamWalkingM$Value < 0]

FECorrNegBWMales = EnrichALLterms(
  test = CorrNegBWMales,
  notTest = setdiff(totesprotCB, CorrNegBWMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

######################################
##Mejora motora corr values histogram machos
# Beam Walking
PearsonMejoraMotoraMales <- PearsonM[MejoraMotoraProtsMales, ]

png(
  filename =
"~/Dropbox/denamic/plots/Set03/AnalisisFuncionalCB/HistogramsCorrelati
on/MejoraMotoraMachos.png",
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

ggplot(data = PearsonMejoraMotoraMales,
aes(PearsonMejoraMotoraMales$Value)) +
  geom_histogram(
    breaks = seq(
      min(PearsonMejoraMotoraMales$Value),
      max(PearsonMejoraMotoraMales$Value),
      by = 0.05
    ),
    col = "red",
    fill = "blue",
    alpha = .5
  ) +
  labs(title = "Mejora Motora Machos") +
  labs(x = "Correlation Values", y = "Count") +
  theme(
    axis.text = element_text(size = 40),
    axis.title = element_text(size = 40, face = "bold"),
    title = element_text(size = 60, face = "bold")
  )

dev.off()

###FUNCTIONAL ENRICHMENT CON LAS PROTEINAS CORRELACIONADAS
NEGATIVAMENTE CON LA MEJORA MOTORA

# Las correlacionadas negativamente con Rotarod y positivamente con
Beam Walking
EmpeoranMotoraProtsMales <-
  c(rownames(PearsonRotarodM)[PearsonRotarodM$Value < 0],
rownames(PearsonBeamWalkingM)[PearsonBeamWalkingM$Value > 0])


myEnrichResultsEmpeoranMotoraMales = EnrichALLterms(
  test = EmpeoranMotoraProtsMales,
  notTest = setdiff(totesprotCB, EmpeoranMotoraProtsMales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)


# MWM, FE ENRICHMENT CON LAS CORRELACIONADAS NEGATIVAMENTE CON LA
MEJORA COGNITIVA
## COR positiva con MWM
MWMPosM <- rownames(PearsonMWMM)[PearsonMWMM$Value > 0]

myEnrichResultsMWMEmpeoranCognitivoMales = EnrichALLterms(
  test = MWMPosM,
  notTest = setdiff(totesprotCB, MWMPosM),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

##FE ENRICHMENT CON LAS CORRELACIONADAS POSITIVAMENTE CON LA MEJORA
COGNITIVA
#cor negativa con MWM
```

```
MWMNegM <- rownames(PearsonMWMM)[PearsonMWMM$Value < 0]

myEnrichResultsMWMMejoraCognitivaMales = EnrichALLterms(
  test = MWMNegM,
  notTest = setdiff(totesprotCB, MWMNegM),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

save(
  myEnrichResultsEmpeoranMotoraMales,
  myEnrichResultsMejoraMotoraMales,
  myEnrichResultsMWMEmpeoranCognitivoMales,
  myEnrichResultsMWMMejoraCognitivaMales,
  file =
"~/Dropbox/denamic/data/set03/FEnrichment_PerGender_Males_Results_Mejo
raYEmpeoraCognitivoYMotora.RData"
)


##########
## FEMALES
##########

load(file =
"~/Dropbox/denamic/data/set03/06_DonePerGender_Females_CorrelatedProte
insWithTests_ForFunctionalEnrichment.RData")

### CORR POSITIVA ROTAROD

CorrPosRotarodFemales <-
  rownames(PearsonRotarodF)[PearsonRotarodF$Value > 0]

FECorrPosRotarodFemales = EnrichALLterms(
  test = CorrPosRotarodFemales,
  notTest = setdiff(totesprotCB, CorrPosRotarodFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

##CORR NEGATIVA ROTAROD
CorrNegRotarodFemales <-
  rownames(PearsonRotarodF)[PearsonRotarodF$Value < 0]

FECorrNegRotarodFemales = EnrichALLterms(
  test = CorrNegRotarodFemales,
  notTest = setdiff(totesprotCB, CorrNegRotarodFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)


##CORR POSITIVA BEAM WALKING
CorrPosBWFemales <-
  rownames(PearsonBeamWalkingF)[PearsonBeamWalkingF$Value > 0]

FECorrPosBWFemales = EnrichALLterms(
  test = CorrPosBWFemales,
  notTest = setdiff(totesprotCB, CorrPosBWFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
```

```
)

## CORR NEGATIVA BEAM WALKING

CorrNegBWFemales <-
  rownames(PearsonBeamWalkingF)[PearsonBeamWalkingF$Value < 0]

FECorrNegBWFemales = EnrichALLterms(
  test = CorrNegBWFemales,
  notTest = setdiff(totesprotCB, CorrNegBWFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

###FUNCTIONAL ENRICHMENT CON LAS PROTEINAS CORRELACIONADAS
POSITIVAMENTE CON LA MEJORA MOTORA

# Las correlacionadas positivamente con Rotarod y Negativamente con
Beam Walking

MejoraMotoraProtsFemales <-
  c(rownames(PearsonRotarodF)[PearsonRotarodF$Value > 0],
rownames(PearsonBeamWalkingF)[PearsonBeamWalkingF$Value <

0])

myEnrichResultsMejoraMotoraFemales = EnrichALLterms(
  test = MejoraMotoraProtsFemales,
  notTest = setdiff(totesprotCB, MejoraMotoraProtsFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

###FUNCTIONAL ENRICHMENT CON LAS PROTEINAS CORRELACIONADAS
NEGATIVAMENTE CON LA MEJORA MOTORA

# Las correlacionadas negativamente con Rotarod y positivamente con
Beam Walking

EmpeoranMotoraProtsFemales <-
  c(rownames(PearsonRotarodF)[PearsonRotarodF$Value < 0],
rownames(PearsonBeamWalkingF)[PearsonBeamWalkingF$Value >

0])

myEnrichResultsEmpeoranMotoraFemales = EnrichALLterms(
  test = EmpeoranMotoraProtsFemales,
  notTest = setdiff(totesprotCB, EmpeoranMotoraProtsFemales),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)


# MWM, FE ENRICHMENT CON LAS CORRELACIONADAS NEGATIVAMENTE CON LA
MEJORA COGNITIVA
## COR positiva con MWM

MWMPosF <- rownames(PearsonMWMF)[PearsonMWMF$Value > 0]

myEnrichResultsMWMEmpeoranCognitivoFemales = EnrichALLterms(
  test = MWMPosF,
  notTest = setdiff(totesprotCB, MWMPosF),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

##FE ENRICHMENT CON LAS CORRELACIONADAS POSITIVAMENTE CON LA MEJORA
COGNITIVA
#cor negativa con MWM
MWMNegF <- rownames(PearsonMWMF)[PearsonMWMF$Value < 0]

myEnrichResultsMWMMejoraCognitivaFemales = EnrichALLterms(
  test = MWMNegF,
  notTest = setdiff(totesprotCB, MWMNegF),
  annotation = myannotTOTALCBbiomart[, c(1, 3)],
  p.adjust.method = "fdr"
)

save(
  myEnrichResultsEmpeoranMotoraFemales,
  myEnrichResultsMejoraMotoraFemales,
  myEnrichResultsMWMEmpeoranCognitivoFemales,
  myEnrichResultsMWMMejoraCognitivaFemales,
  file =
"~/Dropbox/denamic/data/set03/FEnrichment_PerGender_Females_Results_Me
joraYEmpeoraCognitivoYMotora.RData"
)
```

8.10 Estudio del comportamiento, promediado, de las variables

```
#####################################
## Set01 Function Protein Behaviour #
#####################################
library(dplyr)

##seguir del load matriu promediada
load(file =
"~/Dropbox/denamic/data/set01/Set01_ProtPromediada_sinCentradoNiEscala
do.RData")

clinicalTodasImputada <-
  read.delim(file =
"~/Dropbox/denamic/data/set01/00_Set01_prot_CB.txt",
             header = T,
             row.names = 1)

clinicalTodasImputada <- t(clinicalTodasImputada)
rownames(clinicalTodasImputada) <-
  gsub(
    pattern = ".",
    replacement = "_",
    rownames(clinicalTodasImputada),
    fixed = T
  )
rownames(clinicalTodasImputada) <-
  gsub(
    pattern = "_CB",
    replacement = "",
    rownames(clinicalTodasImputada),
    fixed = T
  )
clinicalTodasImputada <- as.data.frame(clinicalTodasImputada)

clinicalPromediada <- ProtPromediada
rownames(clinicalPromediada) <- clinicalPromediada$names
clinicalPromediada$names <- NULL
clinicalPromediada <- t(clinicalPromediada)
clinicalPromediada <- as.data.frame(clinicalPromediada)

TestBehaviour <- function(protein) {
  MatrizPromediadaCB <- clinicalPromediada
  CB <- clinicalTodasImputada
  cbALL <- CB

  protinteresPROMEDIADA <- MatrizPromediadaCB[protein, , drop = F]

  Xvalues <- 1:(length(colnames(protinteresPROMEDIADA)) / 2)

  protintFemales <-
    protinteresPROMEDIADA[, c("VH_F", "CYP_F", "END_F"), drop = F]
  protintMales <-
    protinteresPROMEDIADA[, c("VH_M", "CYP_M", "END_M"), drop = F]

  ####To plot SE, groups
  ##afegir el Standard Error per punt
  CB <- CB[sort(rownames(CB)), protein, drop = F]

  designCB <- do.call("rbind", strsplit(row.names(CB), "_", fixed =
T))
  designCB <- designCB[, -c(3, 4)]
  designCB <- as.data.frame(designCB)
  colnames(designCB) <- c("Treatment", "Gender")

  CB$Gender <- designCB$Gender
  CB$Treatment <- designCB$Treatment

  ###TO PLOT ALL LINE
  cbALL <- cbALL[sort(rownames(cbALL)), protein, drop = F]

  designCBALL <-
    do.call("rbind", strsplit(row.names(cbALL), "_", fixed = T))
  designCBALL <- designCBALL[, 1]
  designCBALL <- as.data.frame(designCBALL)
  colnames(designCBALL) <- "Treatment"

  cbALL$Treatment <- designCBALL$Treatment

  ##Selection by gender and Treatment, groups

  #For genders line
  CB$names <- rownames(CB)

  #Males VH
  MalesVH <- CB %>% filter(Gender == "M", Treatment == "VH")
  #Females vH
  FemalesVH <- CB %>% filter(Gender == "F", Treatment == "VH")

  #Males CAR
  MalesEnd <- CB %>% filter(Gender == "M", Treatment == "END")

  #Females CAR
  FemalesEnd <- CB %>% filter(Gender == "F", Treatment == "END")

  #Males CHLOR01
  MalesCyp <- CB %>% filter(Gender == "M", Treatment == "CYP")

  #Females CHLORO1
  FemalesCyp <- CB %>% filter(Gender == "F", Treatment == "CYP")

  PestsGenderF <-
    list(
      "FemalesVH" = FemalesVH,
      "FemalesCyp" = FemalesCyp,
      "FemalesEnd" = FemalesEnd
    )

  PestsGenderM <-
    list("MalesVH" = MalesVH,
         "MalesCyp" = MalesCyp,
         "MalesEnd" = MalesEnd)

  ###For ALL line
  cbALL$names <- rownames(cbALL)

  # VH
```

```r
VH <- cbALL %>% filter(Treatment == "VH")

# CAR
End <- cbALL %>% filter(Treatment == "END")

# CHLOR01
Cyp <- cbALL %>% filter(Treatment == "CYP")

Pests <- list("VH" = VH,
              "Cyp" = Cyp,
              "End" = End)

###PLOT PARA LOS PROMEDIOS
minY <- min(CB[, protein])
maxY <- max(CB[, protein])

route <-
  paste("~/Dropbox/denamic/plots/Set01/AnalisisFuncionalCB/",
        protein,
        ".png",
        sep = "")
png(
  filename = route,
  width = 1280,
  height = 720 ,
  units = "px",
  type = "cairo"
)

par(mar = c(5.1, 5.1, 4.1, 2.1))

# for All line, Y values
YvaluesAll <- NULL

for (group in Pests) {
  YvaluesAll <- c(YvaluesAll, (mean(group[, protein])))
}

#Females
plot(
  x = Xvalues,
  y = protintFemales[1, ],
  type = "l",
  col = "black",
  main = protein,
  cex.lab = 2,
  cex.axis = 1.2,
  cex.main = 2,
  lwd = 2,
  xlim = c(1, 3),
  ylim = c(minY, maxY),
  xaxt = "n",
  ylab = "Averaged Expression",
  xlab = "Condition"
)
axis(
  1,
  at = 1:3,
  labels = c("VH", "CYP", "END"),
  cex.axis = 1.5
)
```

```
#Standard Errors
pos <- 0
for (Treatment in PestsGenderF) {
  pos <- pos + 1
  Barra <- Treatment[, protein]
  MeanBarra <- mean(Barra)
  SEBarra <- sd(Barra) / length(Barra)
  minBarra <- MeanBarra - SEBarra
  maxBarra <- MeanBarra + SEBarra
  par(new = TRUE)
  plot(
    x = c(pos, pos),
    y = c(minBarra, maxBarra),
    axes = FALSE,
    xlab = "",
    ylab = "",
    type = "l",
    col = "black",
    ylim = c(minY, maxY),
    xlim = c(1, 3),
    lwd = 1
  )
}

#Males
par(new = TRUE)
plot(
  x = Xvalues,
  y = protintMales[1, ],
  axes = FALSE,
  xlab = "",
  ylab = "",
  type = "l",
  col = "blue",
  ylim = c(minY, maxY),
  xlim = c(1, 3),
  lwd = 2
)

pos <- 0
for (Treatment in PestsGenderM) {
  pos <- pos + 1
  #SEdata<-
  Barra <- Treatment[, protein]
  MeanBarra <- mean(Barra)
  SEBarra <- sd(Barra) / length(Barra)
  minBarra <- MeanBarra - SEBarra
  maxBarra <- MeanBarra + SEBarra
  par(new = TRUE)
  plot(
    x = c(pos, pos),
    y = c(minBarra, maxBarra),
    axes = FALSE,
    xlab = "",
    ylab = "",
    type = "l",
    col = "blue",
    ylim = c(minY, maxY),
    xlim = c(1, 3),
    lwd = 1
    )
  }

  #ALL line
  par(new = TRUE)

  plot(
    x = Xvalues,
    y = YvaluesAll,
    axes = FALSE,
    lty = 2,
    xlab = "",
    ylab = "",
    type = "l",
    col = "red",
    xlim = c(1, 3),
    ylim = c(minY, maxY),
    lwd = 2
  )

  #ALL Standard Error
  pos <- 0
  for (group in Pests) {
    pos <- pos + 1
    Barra <- group[, protein]
    MeanBarra <- mean(Barra)
    SEBarra <- sd(Barra) / length(Barra)
    minBarra <- MeanBarra - SEBarra
    maxBarra <- MeanBarra + SEBarra
    par(new = TRUE)
    plot(
      x = c(pos, pos),
      y = c(minBarra, maxBarra),
      axes = FALSE,
      xlab = "",
      ylab = "",
      type = "l",
      col = "red",
      ylim = c(minY, maxY),
      xlim = c(1, 3),
      lwd = 1
    )
  }

  legend(
    "topright",
    legend = c("Females", "Males", "All"),
    col = c("black", "blue", "red"),
    fill = c("black", "blue", "red"),
    ncol = 1,
    title = "Gender",
    cex = 1.6
  )

  dev.off()
}

# Example with P21707
TestBehaviour("P21707")
which(rownames(clinicalPromediada) == "P21707")
```

8.11 Filtrado por *limma y CV*

```
###########################################
#### SET03: filtering low variability ####
###########################################
library(dplyr)
library(limma)

# CB ---------------------------------------------------------------
-----

# #Preparacion de matriz con una columna de nombres que incluya la
combinación del  pesticida y el sexo--------

CB <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_CB_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminacion rata tratada con CYP ya que solo hay una
CB <- CB[-which(rownames(CB) == "CYPER_M_I6_CB"), ]
CB2 <- CB

designCB <- do.call("rbind", strsplit(row.names(CB), "_", fixed = T))
designCB <- designCB[, -c(3, 4)]
data <- designCB
data <- as.data.frame(data)
data$Combination <-
  apply(data[, c(1, 2)] , 1 , paste , collapse = "_")

CB2$names <- data$Combination

new_CB <- CB2 %>% select(names, everything())

# Obtencion de la matriz que tenga promediada la expresion proteica
por pesticida y sexo --------

matriz_promediada_CB <- aggregate(. ~ names, FUN = mean, data =
new_CB)

#comprobacion de que esta bien promediado
#para CAR_F la tabla me dice que la media es de 0.52877666 ( para la
prot A1L1J8)
#manualmente: para CAR_F me sale (0.113760115 + 0.943793213)/2=
0.5287767 , perfecto pues.

# Guardado de matriz promediada ---------------------------------------
-----

matriz_promediada_CB <- t(matriz_promediada_CB)
colnames(matriz_promediada_CB) <- matriz_promediada_CB[1, ]
matriz_promediada_CB <- matriz_promediada_CB[-1, ]

write.table(
  matriz_promediada_CB,
  file =
"~/Dropbox/denamic/data/set03/02_Set03_CB_Matriz_Promediada_Por_Pestic
ida_Y_Sexo.txt",
  sep = "\t",
  quote = F,
  col.names = T,
  row.names = T
)

# HP -----------------------------------------------------------------------
-----
# #Preparacion de matriz con una columna de nombres que incluya la
combinación del  pesticida y el sexo--------
HP <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_HP_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminacion rata tratada con CYP ya que solo hay una
HP <- HP[-which(rownames(HP) == "CYPER_M_I6_HP"), ]
HP2 <- HP

designHP <- do.call("rbind", strsplit(row.names(HP), "_", fixed = T))
designHP <- designHP[, -c(3, 4)]
data <- designHP
data <- as.data.frame(data)
data$Combination <-
  apply(data[, c(1, 2)] , 1 , paste , collapse = "_")

HP2$names <- data$Combination

new_HP <- HP2 %>% select(names, everything())

# Obtencion de la matriz que tenga promediada la expresion proteica
por pesticida y sexo --------

matriz_promediada_HP <- aggregate(. ~ names, FUN = mean, data =
new_HP)

#comprobacion de que esta bien promediado
#para CAR_F la tabla me dice que la media es de 0.52877666 ( para la
prot A1L1J8)
#manualmente: para CAR_F me sale (0.113760115 + 0.943793213)/2=
0.5287767 , perfecto pues.

# Guardado de matriz promediada --------------------------------------
-----

matriz_promediada_HP <- t(matriz_promediada_HP)
colnames(matriz_promediada_HP) <- matriz_promediada_HP[1, ]
matriz_promediada_HP <- matriz_promediada_HP[-1, ]

write.table(
  matriz_promediada_HP,
  file =
"~/Dropbox/denamic/data/set03/02_Set03_HP_Matriz_Promediada_Por_Pestic
ida_Y_Sexo.txt",
  sep = "\t",
  quote = F,
  col.names = T,
  row.names = T
)
```

```
# Deteccion de proteinas con baja variabilidad -----------------------
-----
matriz_promediada_HP <-
  read.delim(

"~/Dropbox/denamic/data/set03/02_Set03_HP_Matriz_Promediada_Por_Pestic
ida_Y_Sexo.txt",
    header = T,
    sep = "\t",
    row.names = 1
  )
matriz_promediada_CB <-
  read.delim(

"~/Dropbox/denamic/data/set03/02_Set03_CB_Matriz_Promediada_Por_Pestic
ida_Y_Sexo.txt",
    header = T,
    sep = "\t",
    row.names = 1
  )

# CB -----------------------------------------------------------------
-----

sdCB <- apply(matriz_promediada_CB, 1, sd)
# boxplot(sdCB)

meanCB = rowMeans(matriz_promediada_CB)

cvCB = abs(sdCB / meanCB)

cvCB_3 <- names(which(cvCB > 3))#587

# HP -----------------------------------------------------------------
-----

sdHP <- apply(matriz_promediada_HP, 1, sd)

meanHP = rowMeans(matriz_promediada_HP)

cvHP = abs(sdHP / meanHP)

sum(cvHP > 3)#477 prot

cvHP_3 <- names(which(cvHP > 3))


# Limma analysis -----------------------------------------------------
-----

designHP <-
  read.delim(

"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_HP.txt
",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminamos CYP
designHP <- designHP[-which(rownames(designHP) == "CYPER_M_I6_HP"), ]
```

```
designCB <-
  read.delim(

"~/Dropbox/denamic/data/set03/DesignMatrixes/designProt03_ARSYN_CB.txt
",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminamos CYP
designCB <- designCB[-which(rownames(designCB) == "CYPER_M_I6_CB"), ]


# CB ------------------------------------------------------------------
-----

#Expresion matrixes
protCB <-
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_CB_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminacion CYP
protCB <- protCB[-which(rownames(protCB) == "CYPER_M_I6_CB"), ]

# #voom must be used if we need to log transf the data

sex_cb <- factor(designCB[rownames(protCB), "Gender"])

pest_cb = factor(designCB[rownames(protCB), "Pesticide"], levels =
c("VH", "CHLOR01", "CHLOR03", "CHLOR1", "CAR"))

prot_matrix_CB = model.matrix( ~ sex_cb + pest_cb + sex_cb * pest_cb)

fit_CB = lmFit(t(protCB), prot_matrix_CB)

fit_p <- eBayes(fit_CB)

fit_p$p.value

# TopRankedCB ---------------------------------------------------------
-----

betas <-
  colnames(fit_p$p.value)[-c(1, 2)] #quitamos intercept y  la que solo
estudia el efecto del sexo no nos interesan
TopRankedCBLists <- list()

for (element in betas) {
  TopRankedCBLists[[element]] <-
    topTable(fit_p, coef = element, number = 1223)
}


pvaluesNumber_CB_Signif <- list()
padjNumber_CB_Signif <- list()

pvaluesNames_CB_Signif <- list()
```

```
padjNames_CB_Signif <- list()

for (position in 1:length(TopRankedCBLists)) {
  pvaluesNames_CB_Signif[[names(TopRankedCBLists[position])]] <-
    character()

  padjNames_CB_Signif[[names(TopRankedCBLists[position])]] <-
    character()

  for (row in 1:nrow(TopRankedCBLists[[position]])) {
    if (TopRankedCBLists[[position]][row, 4] < 0.05) {
      pvaluesNames_CB_Signif[[names(TopRankedCBLists[position])]] <-
        c(pvaluesNames_CB_Signif[[names(TopRankedCBLists[position])]],
rownames(TopRankedCBLists[[position]][row, ]))
    }
  }

  pvaluesNumber_CB_Signif[[names(TopRankedCBLists[position])]] <-

length(pvaluesNames_CB_Signif[[names(TopRankedCBLists[position])]])

  for (row in 1:nrow(TopRankedCBLists[[position]])) {
    if (TopRankedCBLists[[position]][row, 5] < 0.05) {
      padjNames_CB_Signif[[names(TopRankedCBLists[position])]] <-
        c(padjNames_CB_Signif[[names(TopRankedCBLists[position])]],
rownames(TopRankedCBLists[[position]][row, ]))
    }
  }

  padjNumber_CB_Signif[[names(TopRankedCBLists[position])]] <-
    length(padjNames_CB_Signif[[names(TopRankedCBLists[position])]])
}
pvaluesNumber_CB_Signif
padjNumber_CB_Signif
# Unions-Unique analysis ---------------------------------------------
------------

PCBlist <-
  list("pvaluesNames_CB_Signif" = pvaluesNames_CB_Signif,
       "padjNames_CB_Signif" = padjNames_CB_Signif)

######GLOBALS######
TotalPvalUnionCB <- character()
TotalPadjUnionCB <- character()

for (i in 1:2) {
  if (names(PCBlist[i]) == "pvaluesNames_CB_Signif") {
    for (position in 1:length(PCBlist[[i]])) {
      TotalPvalUnionCB <-
        base::unique(c(TotalPvalUnionCB,
pvaluesNames_CB_Signif[[position]]))
    }
  }
  else{
    for (position in 1:length(PCBlist[[i]])) {
      TotalPadjUnionCB <-
        base::unique(c(TotalPadjUnionCB,
padjNames_CB_Signif[[position]]))
    }
  }
}
length(TotalPvalUnionCB)#736
length(TotalPadjUnionCB)#248


#####Per Pesticide#########
PestPvalUnionCB <- character()
PestPadjUnionCB <- character()

for (i in 1:2) {
  if (names(PCBlist[i]) == "pvaluesNames_CB_Signif") {
    for (position in 1:4) {
      PestPvalUnionCB <-
        base::unique(c(PestPvalUnionCB,
pvaluesNames_CB_Signif[[position]]))
    }
  }
  else{
    for (position in 1:4) {
      PestPadjUnionCB <-
        base::unique(c(PestPadjUnionCB,
padjNames_CB_Signif[[position]]))
    }
  }

}
length(PestPvalUnionCB)#487
length(PestPadjUnionCB)#71


#####Por interaccion Pesticida y Sexo#########

InteraccPvalUnionCB <- character()
InteraccPadjUnionCB <- character()
names(pvaluesNames_CB_Signif)
for (i in 1:2) {
  if (names(PCBlist[i]) == "pvaluesNames_CB_Signif") {
    for (position in 5:8) {
      InteraccPvalUnionCB <-
        base::unique(c(InteraccPvalUnionCB,
pvaluesNames_CB_Signif[[position]]))
    }
  }
  else{
    for (position in 5:8) {
      InteraccPadjUnionCB <-
        base::unique(c(InteraccPadjUnionCB,
padjNames_CB_Signif[[position]]))
    }
  }

}
length(InteraccPvalUnionCB)#638
length(InteraccPadjUnionCB)#233


# HP -----------------------------------------------------------------
-----

#Expresion matrixes
protHP <-
```

```
  read.delim(
    "~/Dropbox/denamic/data/set03/02_Set03_HP_PostArsyn",
    header = T,
    row.names = 1,
    sep = " "
  )
#eliminacion CYP
protHP <- protHP[-which(rownames(protHP) == "CYPER_M_I6_HP"), ]

##voom must be used if we need to log transf the data

sex_hp <- factor(designHP[rownames(protHP), "Gender"])

pest_hp = factor(designHP[rownames(protHP), "Pesticide"], levels =
c("VH", "CHLOR01", "CHLOR03", "CHLOR1", "CAR"))


prot_matrix_HP = model.matrix( ~ sex_hp + pest_hp + sex_hp * pest_hp)

fit_HP = lmFit(t(protHP), prot_matrix_HP)

fit_p <- eBayes(fit_HP)


# TopRankedHP ---------------------------------------------------------
-----

betas <-
  colnames(fit_p$p.value)[-c(1, 2)] #quitamos intercept y la que solo
estudia el efecto del sexo ya que no nos interesan
TopRankedHPLists <- list()

for (element in betas) {
  TopRankedHPLists[[element]] <-
    topTable(fit_p, coef = element, number = 1223)
}


pvaluesNumber_HP_Signif <- list()
padjNumber_HP_Signif <- list()

pvaluesNames_HP_Signif <- list()
padjNames_HP_Signif <- list()

for (position in 1:length(TopRankedHPLists)) {
  pvaluesNames_HP_Signif[[names(TopRankedHPLists[position])]] <-
    character()

  padjNames_HP_Signif[[names(TopRankedHPLists[position])]] <-
    character()

  for (row in 1:nrow(TopRankedHPLists[[position]])) {
    if (TopRankedHPLists[[position]][row, 4] < 0.05) {
      pvaluesNames_HP_Signif[[names(TopRankedHPLists[position])]] <-
        c(pvaluesNames_HP_Signif[[names(TopRankedHPLists[position])]],
rownames(TopRankedHPLists[[position]][row, ]))
    }
  }

  pvaluesNumber_HP_Signif[[names(TopRankedHPLists[position])]] <-
```

```
length(pvaluesNames_HP_Signif[[names(TopRankedHPLists[position])]])

  for (row in 1:nrow(TopRankedHPLists[[position]])) {
    if (TopRankedHPLists[[position]][row, 5] < 0.05) {
      padjNames_HP_Signif[[names(TopRankedHPLists[position])]] <-
        c(padjNames_HP_Signif[[names(TopRankedHPLists[position])]],
rownames(TopRankedHPLists[[position]][row, ]))
    }
  }

  padjNumber_HP_Signif[[names(TopRankedHPLists[position])]] <-
    length(padjNames_HP_Signif[[names(TopRankedHPLists[position])]])
}

# Unions-Unique analysis --------------------------------------------
------------
PHPlist <-
  list("pvaluesNames_HP_Signif" = pvaluesNames_HP_Signif,
       "padjNames_HP_Signif" = padjNames_HP_Signif)

######GLOBALS######
TotalPvalUnionHP <- character()
TotalPadjUnionHP <- character()

for (i in 1:2) {
  if (names(PHPlist[i]) == "pvaluesNames_HP_Signif") {
    for (position in 1:length(PHPlist[[i]])) {
      TotalPvalUnionHP <-
        base::unique(c(TotalPvalUnionHP,
pvaluesNames_HP_Signif[[position]]))
    }
  }
  else{
    for (position in 1:length(PHPlist[[i]])) {
      TotalPadjUnionHP <-
        base::unique(c(TotalPadjUnionHP,
padjNames_HP_Signif[[position]]))
    }
  }

}

#####Per Pesticide#########

PestPvalUnionHP <- character()
PestPadjUnionHP <- character()

for (i in 1:2) {
  if (names(PHPlist[i]) == "pvaluesNames_HP_Signif") {
    for (position in 1:4) {
      PestPvalUnionHP <-
        base::unique(c(PestPvalUnionHP,
pvaluesNames_HP_Signif[[position]]))
    }
  }
  else{
    for (position in 1:4) {
      PestPadjUnionHP <-
        base::unique(c(PestPadjUnionHP,
padjNames_HP_Signif[[position]]))
```

```
    }
  }

}
length(PestPvalUnionHP)#377
length(PestPadjUnionHP)#40


#####Por interaccion Pesticida y Sexo#########

InteraccPvalUnionHP <- character()
InteraccPadjUnionHP <- character()

for (i in 1:2) {
  if (names(PHPlist[i]) == "pvaluesNames_HP_Signif") {
    for (position in 5:8) {
      InteraccPvalUnionHP <-
        base::unique(c(InteraccPvalUnionHP,
pvaluesNames_HP_Signif[[position]]))
    }
  }
  else{
    for (position in 5:8) {
      InteraccPadjUnionHP <-
        base::unique(c(InteraccPadjUnionHP,
padjNames_HP_Signif[[position]]))
    }
  }

}
length(InteraccPvalUnionHP)#508
length(InteraccPadjUnionHP)#103


# todos los objetos a guardar ----------------------------------------
-----

save(
  cvCB_3,
  cvHP_3,
  padjNames_CB_Signif,
  TotalPadjUnionCB,
  PestPadjUnionCB,
  InteraccPadjUnionCB,
  padjNames_HP_Signif,
  TotalPadjUnionHP,
  PestPadjUnionHP,
  InteraccPadjUnionHP,
  file =
"~/Dropbox/denamic/data/set03/03_Set03_ProteinsSelectedBycv_3_ANDlimma
.RData"
)

# Union del total de proteinas de Expresion diferencial y de cv ------
-----

#CB
length(TotalPadjUnionCB)
length(cvCB_3)
TotalTotalUnionCB <- base::union(TotalPadjUnionCB, cvCB_3)
length(TotalTotalUnionCB)#685
```

```
#HP
length(TotalPadjUnionHP)
length(cvHP_3)
TotalTotalUnionHP <- base::union(TotalPadjUnionHP, cvHP_3)
length(TotalTotalUnionHP)#529

save(TotalTotalUnionCB, TotalTotalUnionHP, file =
"~/Dropbox/denamic/data/set03/03_Set03_TotalTotalUnionOfProteinsCommin
gFromLimmaAndCv.RData")
```