7. Attachments

7.1. Scripts

7.1.1. Attachment I: Master data set creation script

```
### IMPORTACIONES ###
library("XLConnect")
library("stats")
library("devtools")
library("FactoMineR")
library("RColorBrewer")
library("NOISeq")
library("openxlsx")

### DIRECTORIOS ###
dir_data_met <- "/Users/Elena/Dropbox/TFG/Analisis_WD/met/data"
dir_data_prot <- "/Users/Elena/Dropbox/TFG/Analisis_WD/prot/data"
dir_data_clinical <- "/Users/Elena/Dropbox/TFG/Analisis_WD/clinical"
dir_output <- "/Users/Elena/Dropbox/TFG/Analisis_WD"

### IMPORT METABOLOMIC DATA ###
setwd(dir_data_met)

## CYP ##
data_CYP <- readWorksheetFromFile("Individual data CYP.xlsx", sheet = 1)
rownames(data_CYP) <- data_CYP[,1] #Add names
data_CYP <- data_CYP[,4:length(colnames(data_CYP))] #Quitamos las columnas con los codigos de KEGG y
otros

## END_ST ##
data_END_ST <- readWorksheetFromFile("Individual data Endosulfan males_females_final.xlsx", sheet = 1,
startRow = 3)
rownames(data_END_ST) <- data_END_ST[,1] #Add names
data_END_ST <- data_END_ST[,2:length(colnames(data_END_ST))]

## END_HP ##
data_END_HP <- readWorksheetFromFile("Individual data Endosulfan males_females_final.xlsx", sheet = 2,
startRow = 3)
rownames(data_END_HP) <- data_END_HP[,1] #Add names
data_END_HP <- data_END_HP[,2:length(colnames(data_END_HP))]

## END_CB ##
data_END_CB <- readWorksheetFromFile("Individual data Endosulfan males_females_final.xlsx", sheet = 3,
startRow = 3)
rownames(data_END_CB) <- data_END_CB[,1] #Add names
data_END_CB <- data_END_CB[,2:length(colnames(data_END_CB))]

## END_CX ##
data_END_CX <- readWorksheetFromFile("Individual data Endosulfan males_females_final.xlsx", sheet = 4,
startRow = 3)
rownames(data_END_CX) <- data_END_CX[,1] #Add names
data_END_CX <- data_END_CX[,2:length(colnames(data_END_CX))]


### IMPORT PROTEOMIC DATA ###
setwd(dir_data_prot)
data_prot <- read.xlsx("DataMat_Denamic_Set1.xlsx", 1, startRow = 5, cols = c(c(1), c(5:103)), colNames
= TRUE, rowNames = TRUE)
data_prot <- as.data.frame(sapply(data_prot, as.numeric), row.names = rownames(data_prot))


### IMPORT CLINICAL DATA ###
```

```
setwd(dir_data_clinical)
data_clinical <- readWorksheetFromFile("Table Analysis Behaviour-Proteomics April 2015 (Ratas
analizadas proteomicamente).xlsx", sheet = 2)

### HOMOGENIZACION DE LOS NOMBRES ###
list_data_met <- list(data_CYP, data_END_CB, data_END_CX, data_END_HP, data_END_ST)
for (i in 1:length(list_data_met)) {
  names <- c()
  for (j in 1:length(colnames(list_data_met[[i]]))) {
    split <- unlist(strsplit(colnames(list_data_met[[i]])[j], split = "\\."))
    t <- split[1]
    s <- split[2]
    n <- split[3]
    tis <- substr(split[4], 1, 2)
    name <- paste(t, s, n, tis, sep = " ")
    names <- c(names, name)
  }
  colnames(list_data_met[[i]]) <- names
}

names_prot <- c()
treatment <- c()
sex <- c()
tissue <- c()
numb <- c()
for (i in 1:length(colnames(data_prot))) {
  split <- unlist(strsplit(colnames(data_prot)[i], split = "\\."))
  t <- split[1]
  s <- split[2]
  n <- paste(split[3], split[4], sep = "")
  tis <- split[6]
  treatment <- c(treatment, t)
  sex <- c(sex, s)
  numb <- c(numb, n)
  tissue <- c(tissue, tis)
  name <- paste(t, s, n, tis, sep = " ")
  names_prot <- c(names_prot, name)
}
colnames(data_prot) <- names_prot

names_clinical <- c()
for (i in 1:length(rownames(data_clinical))){
  s <- substr(data_clinical[i, 1], 1, 1)
  t <- data_clinical[i, 8]
  n <- data_clinical[i, 2]
  name <- paste(t, s, n, sep = " ")
  names_clinical <- c(names_clinical, name)
}
rownames(data_clinical) <- names_clinical
data_clinical <- data_clinical[,c(3:7)]


### NEW DATA FRAME ###
master_frame <- as.data.frame(t(data_prot)) #Proteomics added to master frame
#Now we add the metabolic contents to the new master frame
for (i in 1:length(list_data_met)) {
  for (j in 1:length(rownames(list_data_met[[i]]))) {
    metabolite <- rownames(list_data_met[[i]])[j]
    for (z in 1:length(colnames(list_data_met[[i]]))) {
      individual <- colnames(list_data_met[[i]])[z]
      master_frame[individual, metabolite] <- list_data_met[[i]][j, z]
    }
  }
}

# INFO REGARDING INDIVIDUALS
master_frame[, "Treatment"] <- treatment
master_frame[, "Sex"] <- sex
master_frame[, "Tissue"] <- tissue
master_frame[, "Number"] <- numb
```

```
# INFO CLINICAL DATA
for (i in 1:length(rownames(master_frame))) {
  for (j in 1:length(rownames(data_clinical))) {
    full_name <- rownames(master_frame)[i]
    name <- rownames(data_clinical)[j]
    if (grepl(name, full_name)) {
      master_frame[i,colnames(data_clinical)] <- data_clinical[j,]
    }
  }
}

### OUTPUT ###
setwd(dir_output)
write.table(master_frame, "master_frame.txt", sep="\t")
```

7.1.2. Attachment II: Pre-processing and exploration of Set 01 script

```
#Script para realizar PCA sobre datos pre-procesados
library(NOISeq)

### DIRECTORIOS ###
dir_analisis <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01"
dir_data <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/data"
dir_output <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_exp/"

### IMPORT DATA ###
setwd(dir_data)
metab = read.delim("metabolomics01.txt", header = TRUE, as.is = TRUE, row.names = 1, check.names =
FALSE, dec = ",")
metab2 = metab[-grep("END F I2 CX", rownames(metab)),]
proteom = read.delim("proteomics01.txt", header = TRUE, as.is = TRUE, row.names = 1,
                     check.names = FALSE, dec = ",")

metabTissue = lapply(c(" CX", " HP", " ST", " CB"), function (x) metab2[grep(x, rownames(metab2)),])
names(metabTissue) = c("CX", "HP", "ST", "CB")
proteomTissue = lapply(c(" CX", " HP", " ST", " CB"), function (x) proteom[grep(x,
rownames(proteom)),])
names(proteomTissue) = c("CX", "HP", "ST", "CB")

# Count and remove missing values ----------------------------------------

NAmetab = apply(metab2, 2, function(x) sum(is.na(x)))
NAproteom = apply(proteom, 2, function(x) sum(is.na(x)))

metab25 = metab2[,-which(NAmetab > 0.25*nrow(metab2))]; dim(metab25)  # 98 98
proteom25 = proteom[,-which(NAproteom > 0.25*nrow(proteom))]; dim(proteom25)  # 99 722

## Per tissue
NAmetabTissue = lapply(metabTissue, function (y) apply(y, 2, function(x) sum(is.na(x))))
NAproteomTissue = lapply(proteomTissue, function (y) apply(y, 2, function(x) sum(is.na(x))))

metabTissue25 = lapply(1:length(metabTissue),
                       function (i) metabTissue[[i]][,-which(NAmetabTissue[[i]] >
0.25*nrow(metabTissue[[i]]))])
proteomTissue25 = lapply(1:length(proteomTissue),
                         function (i) proteomTissue[[i]][,-which(NAproteomTissue[[i]] >
0.25*nrow(proteomTissue[[i]]))])
names(metabTissue25) = names(proteomTissue25) = names(metabTissue)

# Missing value imputation -----------------------------------------------
library(impute)

metabKNN = impute.knn(t(metab25), k = 2, rowmax = 0.5, colmax = 0.8, maxp = 1500,
rng.seed=362436069)$data
proteomKNN = impute.knn(t(proteom25), k = 2, rowmax = 0.5, colmax = 0.8, maxp = 1500,
rng.seed=362436069)$data

metabTissueKNN = lapply(metabTissue25,
                        function (x) impute.knn(t(x), k = 2, rowmax = 0.5, colmax = 0.8, maxp = 1500,
rng.seed=362436069)$data)
proteomTissueKNN = lapply(proteomTissue25,
                          function (x) impute.knn(t(x), k = 2, rowmax = 0.5, colmax = 0.8, maxp = 1500,
rng.seed=362436069)$data)

# Quantile normalization of metabolomics ---------------------------------

metabTissueKNN = lapply(metabTissueKNN, function(x) normalizeBetweenArrays(x, method="quantile"))

# Arsynseq ---------------------------------------------------------------

setwd(dir_data)
charac = read.delim("characteristics01.txt", header = TRUE, as.is = TRUE, row.names = 1, check.names =
FALSE)
charac[,"Number"] = sapply(charac[,"Number"], function (x) substr(x, start = 1, stop = nchar(x)-1))
```

```
charac[,"TSTN"] = apply(charac[,c("Treatment", "Sex", "Tissue", "Number")], 1, paste, collapse = "_")

charac_l_m = list(1:4)
for (i in 1:length(metabTissueKNN)) {
  charac_l_m[[i]] = charac[colnames(metabTissueKNN[[i]]),]
}
metabTissueNOISeq = list(1:length(metabTissueKNN))
for (i in 1:length(metabTissueKNN)) {
  metabTissueNOISeq[[i]] = readData(metabTissueKNN[[i]], charac_l_m[[i]])
}
metabTissueNoNoise = lapply(metabTissueNOISeq, function (x) ARSyNseq(x, factor = "TSTN", norm = "n",
logtransf = FALSE))
names(metabTissueNoNoise) = names(metabTissueKNN)
metabTissueNoNoise = lapply(metabTissueNoNoise, function(x) x@assayData$exprs)


charac_l_p = list(1:4)
for (i in 1:length(proteomTissueKNN)) {
  charac_l_p[[i]] = charac[colnames(proteomTissueKNN[[i]]),]
}
proteomTissueNOISeq = list(1:length(proteomTissueKNN))
for (i in 1:length(proteomTissueKNN)) {
  proteomTissueNOISeq[[i]] = readData(proteomTissueKNN[[i]], charac_l_p[[i]])
}
proteomTissueNoNoise = lapply(proteomTissueNOISeq, function (x) ARSyNseq(x, factor = "TSTN", norm =
"n", logtransf = TRUE))
names(proteomTissueNoNoise) = names(proteomTissueKNN)
proteomTissueNoNoise = lapply(proteomTissueNoNoise, function(x) x@assayData$exprs)


setwd(dir_data)
for (i in 1:length(proteomTissueNoNoise)) {
  write.table(proteomTissueNoNoise[[i]], paste(names(proteomTissueNoNoise)[i], "prot_NoNoiseData.txt",
sep = "_"), sep = "\t")
  write.table(metabTissueNoNoise[[i]], paste(names(metabTissueNoNoise)[i], "met_NoNoiseData.txt", sep =
"_"), sep = "\t")
}

# Graphical representations ------------------------------------------------
setwd(dir_output)
pdf(file = "heatmaps_norm.pdf", width = 3.5*4, height = 3.5*3)
par(mfcol = c(1,1))
for (i in 1:length(metabTissueKNN)) {
  heatmap(metabTissueKNN[[i]], main = paste("Metabolomics (", names(metabTissueKNN)[i], ")", sep =""),
          margins = c(7,3))
  heatmap(proteomTissueKNN[[i]], main = paste("Proteomics (", names(proteomTissueKNN)[i], ")", sep
=""),
          margins = c(7,3))
}
dev.off()

miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
pdf(file = "boxplots_norm.pdf", width = 3.5*6, height = 3.5*3)
par(mfcol = c(2,4), mar = c(10,5,5,5))
for (i in 1:length(metabTissueKNN)) {
  sorted_m = metabTissueKNN[[i]][,order(colnames(metabTissueKNN[[i]]))]
  sorted_p = proteomTissueKNN[[i]][,order(colnames(proteomTissueKNN[[i]]))]
  if (i == 1) { #cortex has one less sample in metabolomics
    boxplot(sorted_m, main = paste("Metabolomics (", names(metabTissueKNN)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 10, 1, 1, 1, 1, 11, 11, 11, 11, 11, 2, 2, 2, 2, 9, 9, 9, 9,
7, 7, 7)])
  } else {
    boxplot(sorted_m, main = paste("Metabolomics (", names(metabTissueKNN)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 10, 1, 1, 1, 1, 11, 11, 11, 11, 11, 11, 2, 2, 2, 2, 9, 9, 9,
9, 7, 7, 7)])
  }
  boxplot(sorted_p, main = paste("Proteomics (", names(proteomTissueKNN)[i], ")", sep =""), las = 2,
          col = miscolores[c(10, 10, 10, 10, 1, 1, 1, 1, 11, 11, 11, 11, 11, 11, 2, 2, 2, 2, 9, 9, 9,
9, 7, 7, 7)])
```

```
}
dev.off()

# PCA ----------------------------------------------------------------------

### LOG + CENTER ###
#Per tissue
protTissue_data = lapply(proteomTissueNoNoise, function (x) scale(x, center = TRUE, scale = FALSE))
metabTissue_data = lapply(metabTissueNoNoise, function (x) scale(log(x), center = TRUE, scale = TRUE))
protTissue_data = lapply(protTissue_data, function (x) t(x))
metabTissue_data = lapply(metabTissue_data, function (x) t(x))

### APLICAMOS PCA ###
data2pca = list("Metabolomics (Cortex)" = metabTissue_data[[1]],
                "Metabolomics (Hippocampus)" = metabTissue_data[[2]],
                "Metabolomics (Striatum)" = metabTissue_data[[3]],
                "Metabolomics (Cerebellum)" = metabTissue_data[[4]],
                "Proteomics (Cortex)" = protTissue_data[[1]],
                "Proteomics (Hippocampus)" = protTissue_data[[2]],
                "Proteomics (Striatum)" = protTissue_data[[3]],
                "Proteomics (Cerebellum)" = protTissue_data[[4]])
pca.results = lapply(data2pca, PCA.GENES)


### EXPLAINED VARIANCE ###
setwd(dir_output)
samp <- c()
for (i in 1:length(data2pca)) {
  samp <- c(samp, length(rownames(data2pca[[i]])))
}

pdf(file = "explainedvariance_mother_arsynseq_norm_log.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(2,4))
for (i in 1:length(pca.results)) {
  barplot(pca.results[[i]]$var.exp[,1], names = 1:samp[i],
  xlab = "PC", ylab = "explained variance", ylim = c(0,0.7),
  main = names(pca.results)[i])
}
dev.off()

#### COLORS AND SHAPES ###
miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
charac_l = list(1:length(data2pca))
for (i in 1:length(data2pca)) {
  charac_l[[i]] = charac[rownames(data2pca[[i]]),]
}

#Pesticides as colors
col.pest <- miscolores[1:3]
pesticides <- charac[,"Treatment"]
names(pesticides) <- rownames(charac)
names(col.pest) <- unique(pesticides)
mycol = col.pest[pesticides]

#Sex as shapes and tissues as filled-in shapes or empty shapes
myshapes = c(0, 15, 2, 17, 1, 16, 5, 18)
group_l <- list()
for (j in 1:length(data2pca)) {
  group <- c()
  for (i in 1:length(rownames(charac_l[[j]]))) {
    group <- c(group, paste(charac_l[[j]][i,"Sex"], charac_l[[j]][i, "Number"], sep = "-"))
  }
  group_l[[j]] <- group
}
mypch_l = list()
for (j in 1:length(data2pca)) {
  pch.group = myshapes[1:length(unique(group_l[[j]]))]
  names(pch.group) = unique(group_l[[j]])
  mypch = pch.group[group_l[[j]]]
  mypch_l[[j]] = mypch
}
```

```
setwd(dir_output)

### LOADINGS PLOT ###
pdf(file = "PCAloadings12_mother_arsynseq_norm_log.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(2,4))
for (i in 1:length(pca.results)) {
  plot(pca.results[[i]]$loadings[,1:2], col="white", cex = 0.5,
       xlab = paste("PCA 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0), "%", sep=""),
       ylab = paste("PCA 2 ", round(pca.results[[i]]$var.exp[2,1]*100,0), "%", sep=""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$loadings[,1:2]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1),
       ylim = range(pca.results[[i]]$loadings[,1:2]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1))

  points(pca.results[[i]]$loadings[,1], pca.results[[i]]$loadings[,2], pch = 0)
}
dev.off()

pdf(file = "PCAloadings13_mother_arsynseq_norm_log.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(2,4))
for (i in 1:length(pca.results)) {
  plot(pca.results[[i]]$loadings[,1:3], col="white", cex = 0.5,
       xlab = paste("PCA 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0), "%", sep=""),
       ylab = paste("PCA 3 ", round(pca.results[[i]]$var.exp[2,1]*100,0), "%", sep=""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$loadings[,1:3]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1),
       ylim = range(pca.results[[i]]$loadings[,1:3]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1))
  points(pca.results[[i]]$loadings[,1], pca.results[[i]]$loadings[,2], pch = 0)
}
dev.off()

### PCA SCORES PLOT (WITH COLORS AND SHAPES) ###
pdf("PCAscores12_mother_arsynseq_norm_log.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(2,4))
for (i in 1:length(pca.results)) {
  rango = diff(range(pca.results[[i]]$scores[,1:2]))

  plot(pca.results[[i]]$scores[,1:2], col = "white",
       xlab = paste("PC 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0),
                    "%", sep = ""),
       ylab = paste("PC 2 ", round(pca.results[[i]]$var.exp[2,1]*100,0),
                    "%", sep = ""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$scores[,1:2]) + 0.02*rango*c(-1,1),
       ylim = range(pca.results[[i]]$scores[,1:2]) + 0.02*rango*c(-1,1))

  points(pca.results[[i]]$scores[,1], pca.results[[i]]$scores[,2],
         pch = mypch_l[[i]], col = mycol, cex = 1.5)
  legend("topright", c("END", "CYP", "VH"), col = col.pest, pch = 19, bty = "o", ncol = 2, box.col =
"black")
  legend("right", c("M", "F"), pch = c(0, 15), bty = "o", ncol = 2)
  if (i == 1 | i== 2 | i == 5 | i == 6) {
    legend("bottomright", c("II", "I", "III"), pch = c(0, 2, 1), bty = "o", ncol = 2)
  } else {
    legend("bottomright", c("I", "II", "III"), pch = c(0, 2, 1), bty = "o", ncol = 2)
  }
}
dev.off()

pdf("PCAscores13_mother_arsynseq_norm_log.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(2,4))
for (i in 1:length(pca.results)) {
  rango2 = diff(range(pca.results[[i]]$scores[,c(1,3)]))
  plot(pca.results[[i]]$scores[,c(1,3)], col = "white",
       xlab = paste("PC 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0),
                    "%", sep = ""),
       ylab = paste("PC 3 ", round(pca.results[[i]]$var.exp[3,1]*100,0),
```

```
                    "%", sep = ""),
        main = names(data2pca)[i],
        xlim = range(pca.results[[i]]$scores[,c(1,3)]) + 0.02*rango2*c(-1,1),
        ylim = range(pca.results[[i]]$scores[,c(1,3)]) + 0.02*rango2*c(-1,1))

  points(pca.results[[i]]$scores[,1], pca.results[[i]]$scores[,3],
         pch = mypch_l[[i]], col = mycol, cex = 1.5)
  legend("topright", c("END", "CYP", "VH"), col = col.pest, pch = 19, bty = "o", ncol = 2, box.col =
"black")
  legend("right", c("M", "F"), pch = c(0, 15), bty = "o", ncol = 2)
  if (i == 1 | i== 2 | i == 5 | i == 6) {
    legend("bottomright", c("II", "I", "III"), pch = c(0, 2, 1), bty = "o", ncol = 2)
  } else {
    legend("bottomright", c("I", "II", "III"), pch = c(0, 2, 1), bty = "o", ncol = 2)
  }
}
dev.off()
```

7.1.3. Attachment III: Differential expression analysis for Set 01 script

```
# Differential expression analysis for metabolomics and proteomics data of DENAMIC Set 01.
# Analysis by tissues.
# Only VH and END data will be considered, to compare with transcriptomics data.

library(plyr)

dir_analisis <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01"
dir_data <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/data"
dir_output <- "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/"

# Data import ------------------------------------------------------------

#Import data as rows for prots/metabolites and columns for samples
setwd(dir_data)
met_CB = read.delim("CB_met_NoNoiseData_norm.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
met_HP = read.delim("HP_met_NoNoiseData_norm.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
prot_CB = read.delim("CB_prot_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1, check.names
= FALSE)
prot_HP = read.delim("HP_prot_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1, check.names
= FALSE)

met = list("CB" = met_CB, "HP" = met_HP)
prot = list("CB" = prot_CB, "HP" = prot_HP)

# Leave only END and VH ------------------------------------------------

met = lapply(met, function(x) x[,-grep("CYP", colnames(x))])
prot = lapply(prot, function(x) x[,-grep("CYP", colnames(x))])

# Model design -----------------------------------------------------------

charac = read.delim("characteristics01.txt", header = TRUE, as.is = TRUE, row.names = 1, check.names =
FALSE)
charac[,"Number"] = sapply(charac[,"Number"], function (x) substr(x, start = 1, stop = nchar(x)-1))

sex_m_l = list()
pest_m_l = list()
mother_m_l = list()
sex_p_l = list()
pest_p_l = list()
mother_p_l = list()
for (i in 1:2) {
  sex_m_l[[i]] = factor(charac[colnames(met[[i]]), "Sex"])
  pest_m_l[[i]] = factor(charac[colnames(met[[i]]), "Treatment"], levels = c("VH","END"))
  mother_m_l[[i]] = factor(charac[colnames(met[[i]]), "Number"])
  sex_p_l[[i]] = factor(charac[colnames(prot[[i]]), "Sex"])
  pest_p_l[[i]] = factor(charac[colnames(prot[[i]]), "Treatment"], levels = c("VH","END"))
  mother_p_l[[i]] = factor(charac[colnames(prot[[i]]), "Number"])
}

met_matrix = list()
prot_matrix = list()
for (i in 1:2) {
  met_matrix[[i]] = model.matrix(~ sex_m_l[[i]] + pest_m_l[[i]] + sex_m_l[[i]] * pest_m_l[[i]])
  prot_matrix[[i]] = model.matrix(~ sex_p_l[[i]] + pest_p_l[[i]] + sex_p_l[[i]] * pest_p_l[[i]])
}
names(met_matrix) = c("CB", "HP")
names(prot_matrix) = c("CB", "HP")

# Voom transformation ----------------------------------------------------
m_trans_l = list()
p_trans_l = prot
for (i in 1:2) {
  m_trans_l[[i]] <- voom(met[[i]], met_matrix[[i]],plot=TRUE)
}
```

```
# Limma pipeline ------------------------------------------------------------
fit_m = list()
fit_p = list()
for (i in 1:2) {
  fit_m[[i]] = lmFit(m_trans_l[[i]], met_matrix[[i]])
  fit_m[[i]] = eBayes(fit_m[[i]])
  fit_p[[i]] = lmFit(p_trans_l[[i]], prot_matrix[[i]])
  fit_p[[i]] = eBayes(fit_p[[i]])
}

# Venn diagrams -------------------------------------------------------------

###Adjusted p-value
setwd(dir_output)
miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]

for (i in 1:2) {
  results_m = decideTests(fit_m[[i]])[,c(3,4)]
  results_p = decideTests(fit_p[[i]])[,c(3,4)]

  pdf(paste(names(met_matrix)[i], "metab_Venn_adj.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_m, names = c("Intercept", "Sex", "Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()

  pdf(paste(names(met_matrix)[i], "proteom_Venn_adj.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_p, names = c("Intercept", "Sex", "Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()
}

###P-value without adjusting
for (i in 1:2) {
  results_m = decideTests(fit_m[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]
  results_p = decideTests(fit_p[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]

  pdf(paste(names(met_matrix)[i], "metab_Venn.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_m, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()

  pdf(paste(names(met_matrix)[i], "proteom_Venn.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_p, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()
}

### PDFs with everything
pdf("all_Venn_0.05.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2))
for (i in 1:2) {
  results_m = decideTests(fit_m[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]
  results_p = decideTests(fit_p[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]

  vennDiagram(results_m, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(met_matrix)[i], "(Metabolomics)", sep = " "), outer = FALSE)

  vennDiagram(results_p, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(met_matrix)[i], "(Proteomics)", sep = " "), outer = FALSE)
}
dev.off()

pdf("all_Venn_adj.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2))
for (i in 1:2) {
  results_m = decideTests(fit_m[[i]])[,c(3,4)]
  results_p = decideTests(fit_p[[i]])[,c(3,4)]
```

```
  vennDiagram(results_m, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(met_matrix)[i], "(Metabolomics)", sep = " "), outer = FALSE)

  vennDiagram(results_p, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(met_matrix)[i], "(Proteomics)", sep = " "), outer = FALSE)
}
dev.off()

# Create tables for future use ------------------------------------------

met_p_CB = list()
met_p_HP = list()
prot_p_CB = list()
prot_p_HP = list()

for (j in 1:4) {
    met_p_CB[[j]] = topTable(fit_m[[1]], coef = j, number = nrow(fit_m[[1]]))[,c("P.Value",
"adj.P.Val")]
    met_p_HP[[j]] = topTable(fit_m[[2]], coef = j, number = nrow(fit_m[[2]]))[,c("P.Value",
"adj.P.Val")]
    prot_p_CB[[j]] = topTable(fit_p[[1]], coef = j, number = nrow(fit_p[[1]]))[,c("P.Value",
"adj.P.Val")]
    prot_p_HP[[j]] = topTable(fit_p[[2]], coef = j, number = nrow(fit_p[[2]]))[,c("P.Value",
"adj.P.Val")]
}

names(met_p_CB) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(met_p_HP) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(prot_p_CB) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(prot_p_HP) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")


setwd(dir_output)
for (i in 1:4) {
  write.table(met_p_CB[[i]], paste(names(met_p_CB)[i], "met_CB.txt", sep ="_"), sep = "\t", quote =
FALSE)
  write.table(met_p_HP[[i]], paste(names(met_p_HP)[i], "met_HP.txt", sep ="_"), sep = "\t", quote =
FALSE)
  write.table(prot_p_CB[[i]], paste(names(prot_p_CB)[i], "prot_CB.txt", sep ="_"), sep = "\t", quote =
FALSE)
  write.table(prot_p_HP[[i]], paste(names(prot_p_HP)[i], "prot_HP.txt", sep ="_"), sep = "\t", quote =
FALSE)
}

# Creation of average tables ------------------------------------------

met_averages = list()
prot_averages = list()

for (i in 1:2) { #1 is CB, 2 is HP
  VH_F_m = rowMeans(met[[i]][,grep("VH F ", colnames(met[[i]]))])
  END_F_m = rowMeans(met[[i]][,grep("END F ", colnames(met[[i]]))])
  VH_M_m = rowMeans(met[[i]][,grep("VH M ", colnames(met[[i]]))])
  END_M_m = rowMeans(met[[i]][,grep("END M ", colnames(met[[i]]))])
  met_averages[[i]] = data.frame(VH_F_m, END_F_m, VH_M_m, END_M_m)

  VH_F_p = rowMeans(prot[[i]][,grep("VH F ", colnames(prot[[i]]))])
  END_F_p = rowMeans(prot[[i]][,grep("END F ", colnames(prot[[i]]))])
  VH_M_p = rowMeans(prot[[i]][,grep("VH M ", colnames(prot[[i]]))])
  END_M_p = rowMeans(prot[[i]][,grep("END M ", colnames(prot[[i]]))])
  prot_averages[[i]] = data.frame(VH_F_p, END_F_p, VH_M_p, END_M_p)
}

colnames(met_averages[[1]]) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M")
colnames(met_averages[[2]]) = c("HP_VH_F", "HP_END_F", "HP_VH_M", "HP_END_M")
colnames(prot_averages[[1]]) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M")
colnames(prot_averages[[2]]) = c("HP_VH_F", "HP_END_F", "HP_VH_M", "HP_END_M")
```

```
#Export mean tables (individual tissues)
setwd(dir_output)
write.table(met_averages[[1]], "metabolomics_averages_CB.txt", sep = "\t")
write.table(met_averages[[2]], "metabolomics_averages_HP.txt", sep = "\t")
write.table(prot_averages[[1]], "proteomics_averages_CB.txt", sep = "\t")
write.table(prot_averages[[2]], "proteomics_averages_HP.txt", sep = "\t")

met_means = t(rbind.fill(as.data.frame(t(met_averages[[1]])), as.data.frame(t(met_averages[[2]]))))
prot_means = t(rbind.fill(as.data.frame(t(prot_averages[[1]])), as.data.frame(t(prot_averages[[2]]))))
colnames(met_means) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M", "HP_VH_F", "HP_END_F", "HP_VH_M",
"HP_END_M")
colnames(prot_means) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M", "HP_VH_F", "HP_END_F",
"HP_VH_M", "HP_END_M")

#Export mean tables (both tissues)
write.table(met_means, "metabolomics_averages.txt", sep = "\t", quote = FALSE)
write.table(prot_means, "proteomics_averages.txt", sep = "\t", quote = FALSE)

# Log2Ratio tables ----------------------------------------------------------


CB_met = transform(met_averages[[1]], log2_CB_F = log2(met_averages[[1]][,2]/met_averages[[1]][,1]),
                   log2_CB_M = log2(met_averages[[1]][,4]/met_averages[[1]][,3]))[,c(5,6)]
HP_met = transform(met_averages[[2]], log2_CB_F = log2(met_averages[[2]][,2]/met_averages[[2]][,1]),
                   log2_CB_M = log2(met_averages[[2]][,4]/met_averages[[2]][,3]))[,c(5,6)]

CB_prot = transform(prot_averages[[1]], log2_CB_F = prot_averages[[1]][,2]-prot_averages[[1]][,1],
                   log2_CB_M = prot_averages[[1]][,4]-prot_averages[[1]][,3])[,c(5,6)]
HP_prot = transform(prot_averages[[2]], log2_CB_F = prot_averages[[2]][,2]-prot_averages[[2]][,1],
                   log2_CB_M = prot_averages[[2]][,4]-prot_averages[[2]][,3])[,c(5,6)]

#Export for each individual tissue
write.table(CB_met, "metabolomics_log2_CB.txt", sep = "\t", quote = FALSE)
write.table(HP_met, "metabolomics_log2_HP.txt", sep = "\t", quote = FALSE)
write.table(CB_prot, "proteomics_log2_CB.txt", sep = "\t", quote = FALSE)
write.table(HP_prot, "proteomics_log2_HP.txt", sep = "\t", quote = FALSE)

met_log2 = t(rbind.fill(as.data.frame(t(CB_met)), as.data.frame(t(HP_met))))
prot_log2 = t(rbind.fill(as.data.frame(t(CB_prot)), as.data.frame(t(HP_prot))))
colnames(met_log2) = c("log2_CB_F", "log2_CB_M", "log2_HP_F", "log2_HP_M")
colnames(prot_log2) = c("log2_CB_F", "log2_CB_M", "log2_HP_F", "log2_HP_M")

write.table(met_log2, "metabolomics_log2.txt", sep = "\t", quote = FALSE)
write.table(prot_log2, "proteomics_log2.txt", sep = "\t", quote = FALSE)
```

7.1.4. Attachment IV: Initial formatting for transcriptomics script

```
dir_trans = "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Transcriptomics"
dir_data = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_exp"

library(NOISeq)

# RNA-seq: genes? isoforms? ------------------------------------------------------------------
setwd(dir_trans)
rnaseqGenes = read.delim("Cuffdiff_RNA-seq/genes.read_group_tracking", header = TRUE, as.is = TRUE)  #
654408 rows
head(rnaseqGenes)
length(unique(rnaseqGenes$tracking_id))  # 18178

rnaseqIsoforms = read.delim("Cuffdiff_RNA-seq/isoforms.read_group_tracking", header = TRUE, as.is =
TRUE)  # 670572 rows
head(rnaseqIsoforms)
length(unique(rnaseqIsoforms$tracking_id))  # 18627

length(intersect(rnaseqGenes$tracking_id, rnaseqIsoforms$tracking_id)) # 18172

setdiff(rnaseqGenes$tracking_id, rnaseqIsoforms$tracking_id)

tail(setdiff(rnaseqIsoforms$tracking_id, rnaseqGenes$tracking_id), 50)

unique(rnaseqIsoforms$tracking_id[grep("NM_173300", rnaseqIsoforms$tracking_id)])
unique(rnaseqGenes$tracking_id[grep("NM_173300", rnaseqGenes$tracking_id)])
unique(rnaseqGenes$tracking_id[grep("_dup", rnaseqGenes$tracking_id)])


mirnaGenes = read.delim("Cuffdiff_miRNA/genes.read_group_tracking", header = TRUE, as.is = TRUE)
head(mirnaGenes)
length(unique(mirnaGenes$tracking_id))

mirnaIsoforms = read.delim("Cuffdiff_miRNA/isoforms.read_group_tracking", header = TRUE, as.is = TRUE)
head(mirnaIsoforms)
length(unique(mirnaIsoforms$tracking_id))
length(intersect(mirnaGenes$tracking_id, mirnaIsoforms$tracking_id))  # 0

# Long format --> Wide format ----------------------------------------------

####### RNA-Seq Genes
rnaseqGenes2 = rnaseqGenes[,c(1:3,7)]
pesti = sapply(rnaseqGenes2$condition, function (x) substr(x, start = 1, stop = nchar(x)-3))
sexo = sapply(rnaseqGenes2$condition, function (x) substr(x, start = nchar(x)-1, stop = nchar(x)-1))
tejido = sapply(rnaseqGenes2$condition, function (x) substr(x, start = nchar(x), stop = nchar(x)))
rnaseqGenes2 = data.frame(rnaseqGenes2, "pesti" = pesti, "sex" = sexo, "tissue" = tejido)
idSample = apply(rnaseqGenes2[,c("pesti", "sex", "tissue", "replicate")], 1, paste, collapse = "_")
rnaseqGenes2 = data.frame(rnaseqGenes2,"id" = idSample)
rnaseqGenes4reshape = rnaseqGenes2[,c("tracking_id", "id", "FPKM")]

#Genes en filas y FPKMs correspondientes a cada muestra en columnas
rnaseqGenesWide = reshape(data = rnaseqGenes4reshape, direction = "wide", timevar = c("id"),
                          idvar = c("tracking_id"))
rownames(rnaseqGenesWide) = rnaseqGenesWide[,"tracking_id"]
rnaseqGenesWide = rnaseqGenesWide[,2:length(colnames(rnaseqGenesWide))]


####### miRNA-Seq Genes
mirnaGenes2 = mirnaGenes[,c(1:3,7)]
pesti = sapply(mirnaGenes2$condition, function (x) substr(x, start = 1, stop = nchar(x)-3))
sexo = sapply(mirnaGenes2$condition, function (x) substr(x, start = nchar(x)-1, stop = nchar(x)-1))
tejido = sapply(mirnaGenes2$condition, function (x) substr(x, start = nchar(x), stop = nchar(x)))
mirnaGenes2 = data.frame(mirnaGenes2, "pesti" = pesti, "sex" = sexo, "tissue" = tejido)
idSample = apply(mirnaGenes2[,c("pesti", "sex", "tissue", "replicate")], 1, paste, collapse = "_")
mirnaGenes2 = data.frame(mirnaGenes2,"id" = idSample)
mirnaGenes4reshape = mirnaGenes2[,c("tracking_id", "id", "FPKM")]
```

```
#Genes en filas y FPKMs correspondientes a cada muestra en columnas
mirnaGenesWide = reshape(data = mirnaGenes4reshape, direction = "wide", timevar = c("id"), idvar =
c("tracking_id"))
rownames(mirnaGenesWide) = mirnaGenesWide[,"tracking_id"]
mirnaGenesWide = mirnaGenesWide[,2:length(colnames(mirnaGenesWide))]

# Export wide format ------------------------------------------------------
setwd(dir_output)
write.table(rnaseqGenesWide, "rnaseqGenesWide.txt", sep="\t")
write.table(mirnaGenesWide, "mirnaseqGenesWide.txt", sep = "\t")

# Imegen equivalents and sample characteristics ---------------------------

setwd(dir_trans)
rna_eq = read.delim("Cuffdiff_RNA-seq/read_groups.info", header = TRUE, as.is = TRUE)
mirna_eq = read.delim("Cuffdiff_miRNA/read_groups.info", header = TRUE, as.is = TRUE)

#RNA-Seq (que tambien vale para miRNA-Seq)
rna_s = sapply(rna_eq$file, function (x) strsplit(x, split = "/"))
rna_samp = sapply(rna_s, function(x) x[8])
rownames(rna_eq) = rna_samp
rna_eq = rna_eq[, 2:3]

pesti = sapply(rna_eq$condition, function (x) substr(x, start = 1, stop = nchar(x)-3))
sexo = sapply(rna_eq$condition, function (x) substr(x, start = nchar(x)-1, stop = nchar(x)-1))
tejido = sapply(rna_eq$condition, function (x) substr(x, start = nchar(x), stop = nchar(x)))

rna_eq = data.frame(rna_eq, "tissue" = tejido, "pesti" = pesti, "sex" = sexo)
id_rna = apply(rna_eq[,c("pesti", "sex", "tissue", "replicate_num")], 1, paste, collapse = "_")
rna_eq = data.frame(rna_eq, "id" = id_rna)

setwd(dir_output)
write.table(rna_eq, "imegen_equivalencies.txt", sep = "\t")

# Data preparation --------------------------------------------------------

library(NOISeq)
mirnaseq_all = as.data.frame(t(mirnaGenesWide))
mirna_all_names = sapply(rownames(mirnaseq_all), function (x) strsplit(x, split = "\\."))
mirna_all_names = sapply(mirna_all_names, function(x) x[2])
rownames(mirnaseq_all) = mirna_all_names

rnaseq_all = as.data.frame(t(rnaseqGenesWide))
rna_all_names = sapply(rownames(rnaseq_all), function (x) strsplit(x, split = "\\."))
rna_all_names = sapply(rna_all_names, function(x) x[2])
rownames(rnaseq_all) = rna_all_names

rnaseqTissue = lapply(c("_H", "_C"), function (x) rnaseq_all[grep(x, rownames(rnaseq_all)),])
names(rnaseqTissue) = c("Hippocampus", "Cerebellum")

mirnaseqTissue = lapply(c("_H", "_C"), function (x) mirnaseq_all[grep(x, rownames(mirnaseq_all)),])
names(mirnaseqTissue) = c("Hippocampus", "Cerebellum")

#Export data with no filter or arsynseq or norm
setwd(dir_data)
write.table(t(mirnaseq_all), "mirna_NonProcessed.txt", sep = "\t")
write.table(t(rnaseq_all), "rna_NonProcessed.txt", sep = "\t")
for (i in 1:2) {
  write.table(t(mirnaseqTissue[[i]]), paste(names(mirnaseqTissue)[i], "_mirna_NonProcessed.txt", sep =
"", sep = "\t")
  write.table(t(rnaseqTissue[[i]]), paste(names(mirnaseqTissue)[i], "_rna_NonProcessed.txt", sep = ""),
sep = "\t")
}
```

7.1.5. Attachment V: Transcriptomics initial formatting script

```
dir_trans = "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Transcriptomics"
dir_data = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"
dir_miRNA = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/miRNA/"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"

library(NOISeq)
library(plyr)


# Import data to format ------------------------------------------------------

setwd(dir_data)

rnaseq_all = read.delim("rna_NonProcessed.txt", header = TRUE, as.is = TRUE)

rnaseqTissue = list("Hippocampus" = read.delim("Hippocampus_rna_NonProcessed.txt", header = TRUE, as.is
= TRUE),
                    "Cerebellum" = read.delim("Cerebellum_rna_NonProcessed.txt", header = TRUE, as.is =
TRUE))


# Import biomart -------------------------------------------------------------

biomart = read.delim("mart_export.txt")
colnames(biomart) = c("ENSEMBLEGeneID", "ENSEMBLTranscriptID", "RefSeqmRNAID",
                      "RefSeqPredictedmRNAID", "miRDBAccesionID")


# Import miRNA predictions ---------------------------------------------------

setwd(dir_miRNA)
mirna_predictions = read.delim("rat_predictions_80.txt")
mirna_predictions = mirna_predictions[which(mirna_predictions[,"score"] > 85),]


# Import miRBase -------------------------------------------------------------

setwd(dir_miRNA)
miRBase = read.delim("rno.gff3")
miRBase = miRBase[,9]
miRBase = sapply(as.character(miRBase), function (x) strsplit(as.character(x), split= "[;=]"))
miRBase = sapply(miRBase, function (x) x[c(4,6)])
miRBase = data.frame(t(miRBase))
colnames(miRBase) = c("Alias", "Name")
rownames(miRBase) = c()


# Change prediction table ----------------------------------------------------

# First, the miRNAs using miRBase

mirna_predictions = mirna_predictions[which(mirna_predictions[,"miRNA"] %in% miRBase[,"Name"]),]
mirna_predictions = data.frame("ID" = miRBase[mirna_predictions[,"miRNA"], "Alias"], mirna_predictions)
# 68952

# Second, the targets using Biomart

mirna_predictions3 = mirna_predictions[which(as.character(mirna_predictions[,"target"]) %in%
biomart[,3]),]
mirna_predictions4 = mirna_predictions[which(as.character(mirna_predictions[,"target"]) %in%
biomart[,4]),]

mirna_predictions3 = data.frame("Gene" = biomart[mirna_predictions3[,"target"], 1], mirna_predictions3)
mirna_predictions4 = data.frame("Gene" = biomart[mirna_predictions4[,"target"], 1], mirna_predictions4)

mirna_predictions = rbind.fill(mirna_predictions3, mirna_predictions4) # 41363
```

```
# Change RNA-seq IDs ----------------------------------------------------

# Using biomart

rnaseq_all = rnaseq_all[which(rownames(rnaseq_all) %in% biomart[,3]),] #Vamos de 18178 genes a 16619
biomart2 = biomart[which(biomart[,3] %in% rownames(rnaseq_all)),]
biomart2 = biomart2[!duplicated(biomart2[,3]),]
biomart2 = biomart2[order(biomart2[,3]),]
rnaseq_all = rnaseq_all[order(rownames(rnaseq_all)),]
rnaseq_all = data.frame("ENSEMBL" = as.character(biomart2[,1]), rnaseq_all)


# How many ENSEMBL IDs are unique?
ensembl = table(table(rnaseq_all[,"ENSEMBL"])) #Hay repeticiones, por tanto para ello sacamos mediana
rnaseq_all = aggregate(rnaseq_all[,2:ncol(rnaseq_all)], by = list("ENSEMBL" =
as.character(rnaseq_all[,"ENSEMBL"])), median)
rownames(rnaseq_all) = rnaseq_all[,"ENSEMBL"]
rnaseq_all = rnaseq_all[,-c(1)]


# Export same elements that were imported with the same name --------------

rnaseqTissue = lapply(c("_H", "_C"), function (x) rnaseq_all[,grep(x, colnames(rnaseq_all))])
names(rnaseqTissue) = c("Hippocampus", "Cerebellum")

setwd(dir_output)
write.table(rnaseq_all, "rna_NonProcessed.txt", sep = "\t")
for (i in 1:2) {
  write.table(rnaseqTissue[[i]], paste(names(rnaseqTissue)[i], "_rna_NonProcessed.txt", sep = ""), sep
= "\t")
}

# Export re-formatted miRNA predictions for later use ---------------------------------
setwd(dir_output)
write.table(mirna_predictions, "mirna_predictions.txt", sep = "\t")
```

7.1.6. Attachment VI: Pre-processing and exploration of transcriptomics script

```
dir_trans = "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Transcriptomics"
dir_data = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_exp"

library(NOISeq)

# Import data
setwd(dir_data)

mirnaseq_all = read.delim("mirna_NonProcessed.txt", header = TRUE, as.is = TRUE)
rnaseq_all = read.delim("rna_NonProcessed.txt", header = TRUE, as.is = TRUE)
mirnaseqTissue = list("Hippocampus" = read.delim("Hippocampus_mirna_NonProcessed.txt", header = TRUE,
as.is = TRUE),
                      "Cerebellum" = read.delim("Cerebellum_mirna_NonProcessed.txt", header = TRUE,
as.is = TRUE))
rnaseqTissue = list("Hippocampus" = read.delim("Hippocampus_rna_NonProcessed.txt", header = TRUE, as.is
= TRUE),
                    "Cerebellum" = read.delim("Cerebellum_rna_NonProcessed.txt", header = TRUE, as.is =
TRUE))

rna_eq = read.delim("characteristics_transcriptomics.txt", header = TRUE, as.is = TRUE)


# First look at data ------------------------------------------------------

zero_mirna = apply(mirnaseq_all, 1, function(x) all(x == 0))
number_zero_mirna = sum(zero_mirna) #210 de 1133

zero_rna = apply(rnaseq_all, 1, function(x) all(x == 0))
number_zero_rna = sum(zero_rna) # 1657 de 15717


## Per tissue
zero_mirnaTissue = lapply(mirnaseqTissue, function (y) apply(y, 1, function(x) all(x==0)))
zero_rnaTissue = lapply(rnaseqTissue, function (y) apply(y, 1, function(x) all(x==0)))

names(zero_mirnaTissue) = c("Hippocampus", "Cerebellum")
names(zero_rnaTissue) = c("Hippocampus", "Cerebellum")

number_zero_mirnaTissue = lapply(zero_mirnaTissue, sum)
# HP: 240 de 1133
# CB: 241 de 1133
number_zero_rnaTissue = lapply(zero_rnaTissue, sum)
# HP: 1875 de 15717
# CB: 1931 de 15717

# Graphical representations (before arsynseq and filter and normalization) ----------------

setwd(dir_output)

# Demasiados datos para heatmap

miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
pdf(file = "boxplots_noarsynseq_nofilter_nonorm.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2), mar = c(10,5,5,5))
for (i in 1:length(rnaseqTissue)) {
  sorted_mirna = mirnaseqTissue[[i]][,order(colnames(mirnaseqTissue[[i]]))]
  sorted_rna = rnaseqTissue[[i]][,order(colnames(rnaseqTissue[[i]]))]
  if (i == 1) {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7)])
  } else {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
```

```
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 7, 7, 7)])
  }
  boxplot(sorted_rna + 1, main = paste("RNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2, log =
"y",
          col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7, 7)])
}
dev.off()

# Filtros ----------------------------------------------------------------

#Quitamos todos los genes que tienen todo 0s y otras cosas con filtered.data de NOISeq

for (i in 1:2) {
  factor_rna = as.vector(sapply(colnames(rnaseqTissue[[i]]), function(x) substr(x, start = 1, stop =
3)))
  rnaseqTissue[[i]] = filtered.data(rnaseqTissue[[i]], factor_rna, norm = TRUE, method = 1, cv.cutoff =
500, cpm = 1)
  factor_mirna = as.vector(sapply(colnames(mirnaseqTissue[[i]]), function(x) substr(x, start = 1, stop
= 3)))
  mirnaseqTissue[[i]] = filtered.data(mirnaseqTissue[[i]], factor_mirna, norm = TRUE, method = 1,
cv.cutoff = 500, cpm = 1)
}


setwd(dir_data)
for (i in 1:2) {
  write.table(mirnaseqTissue[[i]], paste(names(mirnaseqTissue)[i], "_mirna_Filtered.txt", sep = ""),
sep = "\t")
  write.table(rnaseqTissue[[i]], paste(names(mirnaseqTissue)[i], "_rna_Filtered.txt", sep = ""), sep =
"\t")
}

# Graphical representations (before arsynseq, with filter, no norm) -----------------------

setwd(dir_output)

# Demasiados datos para heatmap

miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
pdf(file = "boxplots_noarsynseq_filter_nonorm.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2), mar = c(10,5,5,5))
for (i in 1:length(rnaseqTissue)) {
  sorted_mirna = mirnaseqTissue[[i]][,order(colnames(mirnaseqTissue[[i]]))]
  sorted_rna = rnaseqTissue[[i]][,order(colnames(rnaseqTissue[[i]]))]
  if (i == 1) {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7)])
  } else {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 7, 7, 7)])
  }
  boxplot(sorted_rna + 1, main = paste("RNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2, log =
"y",
          col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7, 7)])
}
dev.off()

# Apply norm -------------------------------------------------------------

library(limma)
for (i in 1:2) {
  mirnaseqTissue[[i]] = normalizeBetweenArrays(as.matrix(mirnaseqTissue[[i]]), method="quantile")
  rnaseqTissue[[i]] = normalizeBetweenArrays(as.matrix(rnaseqTissue[[i]]), method="quantile")
}


# Graphical representation (with filter and norm, no Arsynseq) -------------

miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
```

```
pdf(file = "boxplots_noarsynseq_filter_norm.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2), mar = c(10,5,5,5))
for (i in 1:length(rnaseqTissue)) {
  sorted_mirna = mirnaseqTissue[[i]][,order(colnames(mirnaseqTissue[[i]]))]
  sorted_rna = rnaseqTissue[[i]][,order(colnames(rnaseqTissue[[i]]))]
  if (i == 1) {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7)])
  } else {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 7, 7, 7)])
  }
  boxplot(sorted_rna + 1, main = paste("RNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2, log =
"y",
          col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7, 7)])
}
dev.off()

# Arsynseq ------------------------------------------------------------------

rna_eq_a = data.frame(rna_eq[,c("tissue", "pesti", "sex")])
rna_eq[,"id"] = sapply(rna_eq[,"id"], as.character)
rownames(rna_eq_a) = rna_eq[,"id"]
rna_eq_a[,"TPS"] = sapply(rna_eq[,"id"], function (x) substr(x, start = 1, stop = nchar(x)-2))
mirna_eq = rna_eq_a[-c(nrow(rna_eq_a), nrow(rna_eq_a)-3),]


mirna_eq_Tissue = lapply(c("_H", "_C"), function (x) mirna_eq[grep(x, rownames(mirna_eq)),])
rna_eq_Tissue = lapply(c("_H", "_C"), function (x) rna_eq_a[grep(x, rownames(rna_eq_a)),])
names(rna_eq_Tissue) = c("H", "C")
names(mirna_eq_Tissue) = c("H", "C")


mirnaseqTissueNOISeq = list()
for (i in 1:length(mirnaseqTissue)) {
  mirnaseqTissueNOISeq[[i]] = readData(mirnaseqTissue[[i]], mirna_eq_Tissue[[i]])
}
mirnaseqTissueNoNoise = lapply(mirnaseqTissueNOISeq, function (x) ARSyNseq(x, factor = "TPS", norm =
"n", logtransf = FALSE))
names(mirnaseqTissueNoNoise) = names(mirnaseqTissue)
mirnaseqTissueNoNoise = lapply(mirnaseqTissueNoNoise, function(x) x@assayData$exprs)


rnaseqTissueNOISeq = list()
for (i in 1:length(rnaseqTissue)) {
  rnaseqTissueNOISeq[[i]] = readData(rnaseqTissue[[i]], rna_eq_Tissue[[i]])
}
rnaseqTissueNoNoise = lapply(rnaseqTissueNOISeq, function (x) ARSyNseq(x, factor = "TPS", norm = "n",
logtransf = FALSE))
names(rnaseqTissueNoNoise) = names(rnaseqTissue)
rnaseqTissueNoNoise = lapply(rnaseqTissueNoNoise, function(x) x@assayData$exprs)


setwd(dir_data)
for (i in 1:2) {
  write.table(mirnaseqTissue[[i]], paste(names(mirnaseqTissue)[i], "_mirna_NoNoiseData.txt", sep = ""),
sep = "\t")
  write.table(rnaseqTissue[[i]], paste(names(mirnaseqTissue)[i], "_rna_NoNoiseData.txt", sep = ""), sep
= "\t")
}

# Graphical representations (after arsynseq) ------------------------------------------------

setwd(dir_output)

miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
pdf(file = "boxplots_arsynseq_filter_norm.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2), mar = c(10,5,5,5))
for (i in 1:length(rnaseqTissueNoNoise)) {
```

```
  sorted_mirna = mirnaseqTissueNoNoise[[i]][,order(colnames(mirnaseqTissueNoNoise[[i]]))]
  sorted_rna = rnaseqTissueNoNoise[[i]][,order(colnames(rnaseqTissueNoNoise[[i]]))]
  if (i == 1) {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7)])
  } else {
    boxplot(sorted_mirna + 1, main = paste("miRNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2,
log = "y",
            col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 7, 7, 7)])
  }
  boxplot(sorted_rna + 1, main = paste("RNA (", names(mirnaseqTissue)[i], ")", sep =""), las = 2, log =
"y",
          col = miscolores[c(10, 10, 10, 1, 1, 1, 11, 11, 11, 2, 2, 2, 9, 9, 9, 7, 7, 7)])
}
dev.off()


# PCA ---------------------------------------------------------------------

data2pca = list("miRNAseq (Hippocampus)" = log(t(mirnaseqTissueNoNoise[[1]])),
                "RNAseq (Hippocampus)" = log(t(rnaseqTissueNoNoise[[1]])),
                "miRNAseq (Cerebellum)" = log(t(mirnaseqTissueNoNoise[[2]])),
                "RNAseq (Cerebellum)" = log(t(rnaseqTissueNoNoise[[2]])))

pca.results = lapply(data2pca, PCA.GENES)


setwd(dir_output)
### Explained variance ###
samp <- c()
for (i in 1:length(data2pca)) {
  samp <- c(samp, length(rownames(data2pca[[i]])))
}
pdf(file = "explainedvariance_arsynseq.pdf", width = 3.5*2, height = 3.5*2)
par(mfcol = c(2,2))
for (i in 1:length(pca.results)) {
  barplot(pca.results[[i]]$var.exp[,1], names = 1:samp[i],
          xlab = "PC", ylab = "explained variance", ylim = c(0,0.7),
          main = names(pca.results)[i])
}
dev.off()

#### COLORS AND SHAPES ###
miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]
#Pesticides as colors
col.pest <- miscolores[1:3]
pesticides <- rna_eq[,"pesti"]
names(pesticides) <- rna_eq[,"id"]
names(col.pest) <- unique(pesticides)
mycol = col.pest[pesticides]

#Sex as shapes and tissues as filled-in shapes or empty shapes
myshapes = c(0, 15, 2, 17, 1, 16, 5, 18)
group_l <- list()
for (j in 1:length(data2pca)) {
  group <- c()
  for (i in 1:length(rownames(data2pca[[j]]))) {
    group <- c(group, paste(rna_eq[i,"sex"], rna_eq[i, "tissue"], sep = "-"))
  }
  group_l[[j]] <- group
}
mypch_l = list()
for (j in 1:length(data2pca)) {
  pch.group = myshapes[1:length(unique(group_l[[j]]))]
  names(pch.group) = unique(group_l[[j]])
  mypch = pch.group[group_l[[j]]]
  mypch_l[[j]] = mypch
}
```

```
### LOADINGS PLOT ###
pdf(file = "PCAloadings12_arsynseq.pdf", width = 3.5*3, height = 3.5*3)
par(mfcol = c(2,2))
for (i in 1:length(pca.results)) {
  plot(pca.results[[i]]$loadings[,1:2], col="white", cex = 0.5,
       xlab = paste("PCA 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0), "%", sep=""),
       ylab = paste("PCA 2 ", round(pca.results[[i]]$var.exp[2,1]*100,0), "%", sep=""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$loadings[,1:2]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1),
       ylim = range(pca.results[[i]]$loadings[,1:2]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1))

  points(pca.results[[i]]$loadings[,1], pca.results[[i]]$loadings[,2], pch = 0)
}
dev.off()

pdf(file = "PCAloadings13_arsynseq.pdf", width = 3.5*3, height = 3.5*3)
par(mfcol = c(2,2))
for (i in 1:length(pca.results)) {
  plot(pca.results[[i]]$loadings[,1:3], col="white", cex = 0.5,
       xlab = paste("PCA 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0), "%", sep=""),
       ylab = paste("PCA 3 ", round(pca.results[[i]]$var.exp[2,1]*100,0), "%", sep=""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$loadings[,1:3]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1),
       ylim = range(pca.results[[i]]$loadings[,1:3]) +
0.02*diff(range(pca.results[[i]]$loadings[,1:2]))*c(-1,1))
  points(pca.results[[i]]$loadings[,1], pca.results[[i]]$loadings[,2], pch = 0)
}
dev.off()


### PCA SCORES PLOT (WITH COLORS AND SHAPES) ###

pdf("PCAscores12_arsynseq.pdf", width = 3.5*3, height = 3.5*3)
par(mfcol = c(2,2))
for (i in 1:length(pca.results)) {
  rango = diff(range(pca.results[[i]]$scores[,1:2]))

  plot(pca.results[[i]]$scores[,1:2], col = "white",
       xlab = paste("PC 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0),
                    "%", sep = ""),
       ylab = paste("PC 2 ", round(pca.results[[i]]$var.exp[2,1]*100,0),
                    "%", sep = ""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$scores[,1:2]) + 0.02*rango*c(-1,1),
       ylim = range(pca.results[[i]]$scores[,1:2]) + 0.02*rango*c(-1,1))

  points(pca.results[[i]]$scores[,1], pca.results[[i]]$scores[,2],
         pch = mypch_l[[i]], col = mycol, cex = 1.5)
  legend("topright", c("Endosulfan", "CYP+END", "Control"), col = col.pest, pch = 19, bty = "o", ncol =
2, box.col = "black")
  legend("right", c("M", "F"), pch = c(0, 15), bty = "o", ncol = 2)
}
dev.off()


pdf("PCAscores13_arsynseq.pdf", width = 3.5*3, height = 3.5*3)
par(mfcol = c(2,2))
for (i in 1:length(pca.results)) {
  rango2 = diff(range(pca.results[[i]]$scores[,c(1,3)]))
  plot(pca.results[[i]]$scores[,c(1,3)], col = "white",
       xlab = paste("PC 1 ", round(pca.results[[i]]$var.exp[1,1]*100,0),
                    "%", sep = ""),
       ylab = paste("PC 3 ", round(pca.results[[i]]$var.exp[3,1]*100,0),
                    "%", sep = ""),
       main = names(data2pca)[i],
       xlim = range(pca.results[[i]]$scores[,c(1,3)]) + 0.02*rango2*c(-1,1),
       ylim = range(pca.results[[i]]$scores[,c(1,3)]) + 0.02*rango2*c(-1,1))
```

```
  points(pca.results[[i]]$scores[,1], pca.results[[i]]$scores[,3],
         pch = mypch_l[[i]], col = mycol, cex = 1.5)
  legend("topright", c("Endosulfan", "CYP+END", "Control"), col = col.pest, pch = 19, bty = "o", ncol =
2, box.col = "black")
  legend("right", c("M", "F"), pch = c(0, 15), bty = "o", ncol = 2)
}
dev.off()
```

7.1.7. Attachment VII: miRDB bibliography check script

```
# Manual bibliography check ------------------------------------------------

#miRNA Bibliography check

setwd("~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/miRNA/")
predictions = read.delim("miRDB_v5.0_prediction_result.txt")

BACE1 = predictions[grep("NM_001207049", predictions[,"target"]),]
t_BACE1 = BACE1[grep("miR-339", BACE1[,"miRNA"]),]

NLRP3 = predictions[grep("NM_001243133", predictions[,"target"]),]
t_NLRP3 = NLRP3[grep("miR-223", NLRP3[,"miRNA"]),]

GRM4 = predictions[grep("NM_00125681", predictions[,"target"]),]
t_GRM4 = GRM4[grep("miR-1202", GRM4[,"miRNA"]),] #No matches for any splice variant

BDNF = predictions[grep("NM_001285422", predictions[,"target"]),]
t_BDNF = BDNF[grep("miR-1", BDNF[,"miRNA"]),]

RHOC = predictions[grep("NM_001106461", predictions[,"target"]),]
t_RHOC = RHOC[grep("miR-509", RHOC[,"miRNA"]),]

ACHE = predictions[grep("NM_001302621", predictions[,"target"]),]
t_ACHE = ACHE[grep("miR-608", RHOC[,"miRNA"]),] #No matches

mTOR = predictions[grep("NM_004958", predictions[,"target"]),]
t_mTOR = mTOR[grep("miR-199", mTOR[,"miRNA"]),]

APAF1 = predictions[grep("NM_013229", predictions[,"target"]),]
t_APAF1 = APAF1[grep("miR-23", APAF1[,"miRNA"]),]

# mirTarBase comparison ----------------------------------------------------

mirdb_rat = read.delim("mirna_predictions_80.txt")
mirtarbase = read.delim("mirTarBase_rat.txt")[,c(1:6)] #miRNA targets experimentally proven
biomart = read.delim("mart_export.txt")

#First of all, we add RefSeq IDs to the mirtarbase frame
names = sapply(mirtarbase[,5], function(x) as.character(biomart[grep(x, biomart[,4]),3]))
mirtarbase[,"RefSeq"] = sapply(names, function(x) max(x))
mirtarbase[,"ID"] = apply(mirtarbase[,c("miRNA","RefSeq")], 1, paste, collapse = "_")


#After that, we can compare it with the rat predictions and check scores
mirdb_rat[,"ID"] = apply(mirdb_rat[,c("miRNA","target")], 1, paste, collapse = "_")

rat_predictions = mirdb_rat[which(mirdb_rat[,"ID"] %in% mirtarbase[,"ID"]),]

write.table(rat_predictions, "mirna_mirtarbase_check.txt", sep = "\t", dec = ",")

# Analysis of miRDB ------------------------------------------------------

#Histograms
pdf("miRNAhistograms.pdf", width = 3.5*4, height = 3.5*2)
par(mfcol = c(1,2))
hist(table(mirdb_rat[,"miRNA"]), plot = TRUE,
     main = "Genes per miRNA",
     xlab = "Genes per miRNA",
     col = colors()[111],
     xlim = c(0,1700))
hist_gene = hist(table(mirdb_rat[,"target"]), plot = TRUE,
                 main = "miRNAs per gene",
                 xlab = "miRNAs per gene",
                 col = colors()[111],
                 xlim = c(0,150))
dev.off()
```

```
#Density plots
miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]

predictions_80 = mirdb_rat[which(mirdb_rat[,"score"] > 80),]
predictions_85 = mirdb_rat[which(mirdb_rat[,"score"] > 85),]
predictions_90 = mirdb_rat[which(mirdb_rat[,"score"] > 90),]
predictions_95 = mirdb_rat[which(mirdb_rat[,"score"] > 95),]

TT80_m = table(as.character(predictions_80[,"miRNA"]))
TT85_m = table(as.character(predictions_85[,"miRNA"]))
TT90_m = table(as.character(predictions_90[,"miRNA"]))
TT95_m = table(as.character(predictions_95[,"miRNA"]))

TT80_t = table(as.character(predictions_80[,"target"]))
TT85_t = table(as.character(predictions_85[,"target"]))
TT90_t = table(as.character(predictions_90[,"target"]))
TT95_t = table(as.character(predictions_95[,"target"]))

pdf("density_plots.pdf", width = 3.5*3, height = 3.5*1.5)
par(mfcol = c(1,2))
plot(density(TT80_m), lwd = 2, col = miscolores[1], main = "Density Plot", xlab = "Genes per miRNA",
     ylim = c(0, 0.03), xlim = c(0,600))
lines(density(TT85_m), lwd = 2, col = miscolores[2])
lines(density(TT90_m), lwd = 2, col = miscolores[3])
lines(density(TT95_m), lwd = 2, col = miscolores[4])

plot(density(TT80_t), lwd = 2, col = miscolores[1], main = "Density Plot", xlab = "miRNAs per gene",
     xlim = c(0,20), ylim = c(0,3))
lines(density(TT85_t), lwd = 2, col = miscolores[2])
lines(density(TT90_t), lwd = 2, col = miscolores[3])
lines(density(TT95_t), lwd = 2, col = miscolores[4])

legend("topright", c("80", "85", "90", "95"), col = miscolores[1:4], pch = 19,
       bty = "o", ncol = 2, box.col = "black")
dev.off()

# Median plot
x = c(80, 85, 90, 95)
y = c(median(predictions_80[,"score"]), median(predictions_85[,"score"]),
      median(predictions_90[,"score"]), median(predictions_95[,"score"]))
pdf("medians_score.pdf", width = 3.5*1.5, height = 3.5*1.5)
plot(x,y, pch = 19, main = "Median evolution with score cut-off")
dev.off()

# Summary table
breaks = c(80, 85, 90, 95)
medians_m = c(median(TT80_m), median(TT85_m), median(TT90_m), median(TT95_m))
medians_t = c(median(TT80_t), median(TT85_t), median(TT90_t), median(TT95_t))
miRNAs = c(length(unique(predictions_80[,"miRNA"])), length(unique(predictions_85[,"miRNA"])),
           length(unique(predictions_90[,"miRNA"])), length(unique(predictions_95[,"miRNA"])))
genes = c(length(unique(predictions_80[,"target"])), length(unique(predictions_85[,"target"])),
          length(unique(predictions_90[,"target"])), length(unique(predictions_95[,"target"])))

# We want to compare with our data, so we import it
setwd("~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data")
mirna_CB = read.delim("Cerebellum_mirna_NoNoiseData.txt")
rna_CB = read.delim("Cerebellum_rna_NoNoiseData.txt")
mirna_HP = read.delim("Hippocampus_mirna_NoNoiseData.txt")
rna_HP = read.delim("Hippocampus_rna_NoNoiseData.txt")


### POR AQUI VOY!

intersections_RNA_CB = c(nrow(predictions_80[which(unique(predictions_80[,"Gene"]) %in%
rownames(rna_CB)),]),
                         nrow(predictions_85[which(unique(predictions_85[,"Gene"]) %in%
rownames(rna_CB)),]),
                         nrow(predictions_90[which(unique(predictions_90[,"Gene"]) %in%
rownames(rna_CB)),]),
                         nrow(predictions_95[which(unique(predictions_95[,"Gene"]) %in%
rownames(rna_CB)),]))
```

```
intersections_RNA_HP = c(nrow(predictions_80[which(unique(predictions_80[,"Gene"]) %in%
rownames(rna_HP)),]),
                         nrow(predictions_85[which(unique(predictions_85[,"Gene"]) %in%
rownames(rna_HP)),]),
                         nrow(predictions_90[which(unique(predictions_90[,"Gene"]) %in%
rownames(rna_HP)),]),
                         nrow(predictions_95[which(unique(predictions_95[,"Gene"]) %in%
rownames(rna_HP)),]))

# We need to change miRNA names to MI format
setwd("~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/miRNA")
miRBase_rat = read.delim("miRDB_rat.txt")

miRBase_rat2 = read.delim("rno.gff3")
miRBase_rat2 = miRBase_rat2[,9]
miRBase_rat2 = sapply(as.character(miRBase_rat2), function (x) strsplit(as.character(x), split=
"[;=]"))
miRBase_rat2 = sapply(miRBase_rat2, function (x) x[c(4,6)])
miRBase_rat2_data = data.frame(t(miRBase_rat2))
colnames(miRBase_rat2_data) = c("Alias", "Name")

nice_names2 = miRBase_rat2_data[which(miRBase_rat2_data[,"Name"] %in% mirdb_rat[,"miRNA"]),]
nice_names2 = nice_names2[order(nice_names2[,2]),]
nice_names2 = nice_names2[!duplicated(nice_names2[,2]),] #752

mirna2 = mirdb_rat
mirna2 = mirna2[order(mirna2[,"miRNA"]),]
mirna2 = mirna2[which(mirna2[,"miRNA"] %in% nice_names2[,2]),]
mirna2 = mirna2[!duplicated(mirna2[,"miRNA"]),] #752 unique miRNAs

#Cambiamos nombres de mirna2 con equivalentes de nice_names2
rownames(mirna2) = nice_names2[,"Alias"]

#Hacemos nuevas tablas de predictions con mirna2
predictions_80_2 = mirna2[which(mirna2[,"score"] > 80),]
predictions_85_2 = mirna2[which(mirna2[,"score"] > 85),]
predictions_90_2 = mirna2[which(mirna2[,"score"] > 90),]
predictions_95_2 = mirna2[which(mirna2[,"score"] > 95),]

intersections_miRNA_CB = c(nrow(predictions_80[which(unique(predictions_80[,"ID"]) %in%
rownames(mirna_CB)),]),
                           nrow(predictions_85[which(unique(predictions_85[,"ID"]) %in%
rownames(mirna_CB)),]),
                           nrow(predictions_90[which(unique(predictions_90[,"ID"]) %in%
rownames(mirna_CB)),]),
                           nrow(predictions_95[which(unique(predictions_95[,"ID"]) %in%
rownames(mirna_CB)),]))
intersections_miRNA_HP = c(nrow(predictions_80[which(unique(predictions_80[,"ID"]) %in%
rownames(mirna_HP)),]),
                           nrow(predictions_85[which(unique(predictions_85[,"ID"]) %in%
rownames(mirna_HP)),]),
                           nrow(predictions_90[which(unique(predictions_90[,"ID"]) %in%
rownames(mirna_HP)),]),
                           nrow(predictions_95[which(unique(predictions_95[,"ID"]) %in%
rownames(mirna_HP)),]))

#Bringing it all together and creating the table
the_table = data.frame("Medians (genes per miRNA)" = medians_m,
                       "Medians (miRNAs per gene)" = medians_t,
                       "Num. miRNAs" = miRNAs,
                       "Num. genes" = genes,
                       "Intersect. miRNAs CB" = intersections_miRNA_CB,
                       "Intersect. miRNAs HP" = intersections_miRNA_HP,
                       "Intersect. Genes CB" = intersections_RNA_CB,
                       "Intersect. Genes HP" = intersections_RNA_HP)
rownames(the_table) = breaks
colnames(the_table) = c("Medians (genes per miRNA)", "Medians (miRNAs per gene)",
                        "Num. miRNAs", "Num. genes", "Intersect miRNAs CB",
                        "Intersect miRNAs HP", "Intersect Genes CB", "Intersect Genes HP")
```

7.1.8. Attachment VIII: Differential expression analysis for transcriptomics script

```
dir_trans = "/Users/Elena/Dropbox/Felipo_OmicsDENAMICdata/Transcriptomics"
dir_data = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE"

library(plyr)

# Data import ------------------------------------------------------------

#Import data as rows for prots/metabolites and columns for samples
setwd(dir_data)
rna_HP = read.delim("Hippocampus_rna_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
mirna_HP = read.delim("Hippocampus_mirna_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
rna_CB = read.delim("Cerebellum_rna_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
mirna_CB = read.delim("Cerebellum_mirna_NoNoiseData.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)


miRNA = list("CB" = mirna_CB, "HP" = mirna_HP)
RNA = list("CB" = rna_CB, "HP" = rna_HP)


# Only keep END and VH ---------------------------------------------------

miRNA = lapply(miRNA, function(x) x[,-grep("CYP_END", colnames(x))])
RNA = lapply(RNA, function(x) x[,-grep("CYP_END", colnames(x))])


# Model design -----------------------------------------------------------

charac = read.delim("characteristics_transcriptomics.txt", header = TRUE, as.is = TRUE, row.names = 1,
check.names = FALSE)
rna_eq = data.frame(charac, row.names = charac[,"id"])
mirna_eq = rna_eq[-c(3),]


sex_mirna_l = list()
pest_mirna_l = list()
sex_rna_l = list()
pest_rna_l = list()

for (i in 1:2) {
  sex_mirna_l[[i]] = factor(mirna_eq[colnames(miRNA[[i]]), "sex"])
  pest_mirna_l[[i]] = factor(mirna_eq[colnames(miRNA[[i]]), "pesti"], levels =
c("Control","Endosulfan"))
  sex_rna_l[[i]] = factor(rna_eq[colnames(RNA[[i]]), "sex"])
  pest_rna_l[[i]] = factor(rna_eq[colnames(RNA[[i]]), "pesti"], levels = c("Control","Endosulfan"))
}

mirna_matrix = list()
rna_matrix = list()
for (i in 1:2) {
  mirna_matrix[[i]] = model.matrix(~ sex_mirna_l[[i]] + pest_mirna_l[[i]] + sex_mirna_l[[i]] *
pest_mirna_l[[i]])
  rna_matrix[[i]] = model.matrix(~ sex_rna_l[[i]] + pest_rna_l[[i]] + sex_rna_l[[i]] * pest_rna_l[[i]])
}
names(mirna_matrix) = c("CB", "HP")
names(rna_matrix) = c("CB", "HP")

# Voom transformation ----------------------------------------------------
mirna_trans_l = list()
rna_trans_l = list()
for (i in 1:2) {
```

```
  mirna_trans_l[[i]] <- voom(miRNA[[i]], mirna_matrix[[i]],plot=TRUE)
  rna_trans_l[[i]] <- voom(RNA[[i]], rna_matrix[[i]],plot=TRUE)
}

# Limma pipeline ------------------------------------------------------------
fit_mirna = list()
fit_rna = list()
for (i in 1:2) {
  fit_mirna[[i]] = lmFit(mirna_trans_l[[i]], mirna_matrix[[i]])
  fit_mirna[[i]] = eBayes(fit_mirna[[i]])
  fit_rna[[i]] = lmFit(rna_trans_l[[i]], rna_matrix[[i]])
  fit_rna[[i]] = eBayes(fit_rna[[i]])
}

# Venn diagrams -------------------------------------------------------------

setwd(dir_output)
miscolores <- colors()[c(554, 89, 111, 512, 17, 586, 132, 428, 601, 568, 86, 390)]

###Adjusted p-values (BH)
for (i in 1:2) {
  results_mirna = decideTests(fit_mirna[[i]])[,c(3,4)]
  results_rna = decideTests(fit_rna[[i]])[,c(3,4)]

  pdf(paste(names(mirna_matrix)[i], "miRNA_Venn_adj.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_mirna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()

  pdf(paste(names(mirna_matrix)[i], "RNA_Venn_adj.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_rna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()
}

###P-values without adjusting
for (i in 1:2) {
  results_mirna = decideTests(fit_mirna[[i]], adjust.method = "none", p.value=0.01)[,c(3,4)]
  results_rna = decideTests(fit_rna[[i]], adjust.method = "none", p.value=0.01)[,c(3,4)]

  pdf(paste(names(mirna_matrix)[i], "miRNA_Venn.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_mirna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()

  pdf(paste(names(mirna_matrix)[i], "RNA_Venn.pdf", sep = "_"), width = 3.5*3, height = 3.5*3)
  vennDiagram(results_rna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)])
  dev.off()
}

pdf("all_Venn_0.05.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2))
for (i in 1:2) {
  results_mirna = decideTests(fit_mirna[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]
  results_rna = decideTests(fit_rna[[i]], adjust.method = "none", p.value=0.05)[,c(3,4)]

  vennDiagram(results_mirna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(mirna_matrix)[i], "(miRNA)", sep = " "), outer = FALSE)

  vennDiagram(results_rna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(mirna_matrix)[i], "(RNA)", sep = " "), outer = FALSE)
}
dev.off()

pdf("all_Venn_adj.pdf", width = 3.5*4, height = 3.5*4)
par(mfcol = c(2,2))
for (i in 1:2) {
  results_mirna = decideTests(fit_mirna[[i]])[,c(3,4)]
```

```
  results_rna = decideTests(fit_rna[[i]])[,c(3,4)]

  vennDiagram(results_mirna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(mirna_matrix)[i], "(miRNA)", sep = " "), outer = FALSE)

  vennDiagram(results_rna, names = c("Pesticide", "Sex∑Pesticide"),
              circle.col = miscolores[c(1,5,9,3)], mar = rep(0,4))
  title(main = paste(names(mirna_matrix)[i], "(RNA)", sep = " "), outer = FALSE)
}
dev.off()

# Create tables for future use ------------------------------------------


mirna_p_CB = list()
mirna_p_HP = list()
rna_p_CB = list()
rna_p_HP = list()

for (j in 1:4) {
  mirna_p_CB[[j]] = topTable(fit_mirna[[1]], coef = j, number = nrow(fit_mirna[[1]]))[,c("P.Value",
"adj.P.Val")]
  mirna_p_HP[[j]] = topTable(fit_mirna[[2]], coef = j, number = nrow(fit_mirna[[2]]))[,c("P.Value",
"adj.P.Val")]
  rna_p_CB[[j]] = topTable(fit_rna[[1]], coef = j, number = nrow(fit_rna[[1]]))[,c("P.Value",
"adj.P.Val")]
  rna_p_HP[[j]] = topTable(fit_rna[[2]], coef = j, number = nrow(fit_rna[[2]]))[,c("P.Value",
"adj.P.Val")]
}

names(mirna_p_CB) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(mirna_p_HP) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(rna_p_CB) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")
names(rna_p_HP) = c("Intercept", "Sex", "Pesti", "Pesti∑Sex")


setwd(dir_output)
for (i in 1:4) {
  write.table(mirna_p_CB[[i]], paste(names(mirna_p_CB)[i], "mirna_CB.txt", sep ="_"), sep = "\t")
  write.table(mirna_p_HP[[i]], paste(names(mirna_p_HP)[i], "mirna_HP.txt", sep ="_"), sep = "\t")
  write.table(rna_p_CB[[i]], paste(names(rna_p_CB)[i], "rna_CB.txt", sep ="_"), sep = "\t")
  write.table(rna_p_HP[[i]], paste(names(rna_p_HP)[i], "rna_HP.txt", sep ="_"), sep = "\t")
}

# Creation of average tables ------------------------------------------

mirna_averages = list()
rna_averages = list()

for (i in 1:2) { #1 is CB, 2 is HP
  VH_F_m = rowMeans(miRNA[[i]][,grep("Control_F", colnames(miRNA[[i]]))])
  END_F_m = rowMeans(miRNA[[i]][,grep("Endosulfan_F", colnames(miRNA[[i]]))])
  VH_M_m = rowMeans(miRNA[[i]][,grep("Control_M", colnames(miRNA[[i]]))])
  END_M_m = rowMeans(miRNA[[i]][,grep("Endosulfan_M", colnames(miRNA[[i]]))])
  mirna_averages[[i]] = data.frame(VH_F_m, END_F_m, VH_M_m, END_M_m)

  VH_F_r = rowMeans(RNA[[i]][,grep("Control_F", colnames(RNA[[i]]))])
  END_F_r = rowMeans(RNA[[i]][,grep("Endosulfan_F", colnames(RNA[[i]]))])
  VH_M_r = rowMeans(RNA[[i]][,grep("Control_M", colnames(RNA[[i]]))])
  END_M_r = rowMeans(RNA[[i]][,grep("Endosulfan_M", colnames(RNA[[i]]))])
  rna_averages[[i]] = data.frame(VH_F_r, END_F_r, VH_M_r, END_M_r)
}

colnames(mirna_averages[[1]]) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M")
colnames(mirna_averages[[2]]) = c("HP_VH_F", "HP_END_F", "HP_VH_M", "HP_END_M")
colnames(rna_averages[[1]]) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M")
colnames(rna_averages[[2]]) = c("HP_VH_F", "HP_END_F", "HP_VH_M", "HP_END_M")


#Export mean tables (individual tissues)
```

```
setwd(dir_output)
write.table(mirna_averages[[1]], "transcriptomics_mirna_averages_CB.txt", sep = "\t")
write.table(mirna_averages[[2]], "transcriptomics_mirna_averages_HP.txt", sep = "\t")
write.table(rna_averages[[1]], "transcriptomics_rna_averages_CB.txt", sep = "\t")
write.table(rna_averages[[2]], "transcriptomics_rna_averages_HP.txt", sep = "\t")

mirna_means = t(rbind.fill(as.data.frame(t(mirna_averages[[1]])),
as.data.frame(t(mirna_averages[[2]]))))
rna_means = t(rbind.fill(as.data.frame(t(rna_averages[[1]])), as.data.frame(t(rna_averages[[2]]))))
colnames(mirna_means) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M", "HP_VH_F", "HP_END_F",
"HP_VH_M", "HP_END_M")
colnames(rna_means) = c("CB_VH_F", "CB_END_F", "CB_VH_M", "CB_END_M", "HP_VH_F", "HP_END_F", "HP_VH_M",
"HP_END_M")

#Export mean tables (both tissues)
write.table(mirna_means, "transcriptomics_mirna_averages.txt", sep = "\t")
write.table(rna_means, "transcriptomics_rna_averages.txt", sep = "\t")


# Log2Ratio tables ------------------------------------------------------

CB_mirna = transform(mirna_averages[[1]], log2_CB_F =
log2((mirna_averages[[1]][,2]+1)/(mirna_averages[[1]][,1]+1)),
                     log2_CB_M = log2((mirna_averages[[1]][,4]+1)/(mirna_averages[[1]][,3]+1)))[,c(5,6)]
HP_mirna = transform(mirna_averages[[2]], log2_CB_F =
log2((mirna_averages[[2]][,2]+1)/(mirna_averages[[2]][,1]+1)),
                     log2_CB_M = log2((mirna_averages[[2]][,4]+1)/(mirna_averages[[2]][,3]+1)))[,c(5,6)]

CB_rna = transform(rna_averages[[1]], log2_CB_F = log2(rna_averages[[1]][,2]/rna_averages[[1]][,1]),
                       log2_CB_M = log2(rna_averages[[1]][,4]/rna_averages[[1]][,3]))[,c(5,6)]
HP_rna = transform(rna_averages[[2]], log2_CB_F = log2(rna_averages[[2]][,2]/rna_averages[[2]][,1]),
                       log2_CB_M = log2(rna_averages[[2]][,4]/rna_averages[[2]][,3]))[,c(5,6)]


#Export for each individual tissue
write.table(CB_mirna, "transcriptomics_mirna_log2_CB.txt", sep = "\t")
write.table(HP_mirna, "transcriptomics_mirna_log2_HP.txt", sep = "\t")
write.table(CB_rna, "transcriptomics_rna_log2_CB.txt", sep = "\t")
write.table(HP_rna, "transcriptomics_rna_log2_HP.txt", sep = "\t")

mirna_log2 = t(rbind.fill(as.data.frame(t(CB_mirna)), as.data.frame(t(HP_mirna))))
rna_log2 = t(rbind.fill(as.data.frame(t(CB_rna)), as.data.frame(t(HP_rna))))
colnames(mirna_log2) = c("log2_CB_F", "log2_CB_M", "log2_HP_F", "log2_HP_M")
colnames(rna_log2) = c("log2_CB_F", "log2_CB_M", "log2_HP_F", "log2_HP_M")

write.table(mirna_log2, "transcriptomics_mirna_log2.txt", sep = "\t")
write.table(rna_log2, "transcriptomics_rna_log2.txt", sep = "\t")
```

7.1.9. Attachment IX: Clinical variable analysis script

```
dir_data = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/data/"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_clinical"
dir_data_met = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/average_tables/"
dir_data_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/average_tables/"
dir_sig = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/paintomics/output"
dir_pval_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/pvalue_tables/"

library(corrplot)

# Import data ------------------------------------------------------------

setwd(dir_data)
data = read.delim("clinical01.txt", header = TRUE, check.names = FALSE, row.names = 1, dec = ",")
data = data[grep(" ST", rownames(data)),] # Solo nos interesa un tejido (las ratas son las mismas)

data_END = data[grep("END", rownames(data)),]
data_VH = data[grep("VH", rownames(data)),]
data_END_M = data_END[grep("M", rownames(data_END)),]
data_END_F = data_END[grep("F", rownames(data_END)),]
data_VH_M = data_VH[grep("M", rownames(data_VH)),]
data_VH_F = data_VH[grep("F", rownames(data_VH)),]

# Wilcoxon ---------------------------------------------------------------

MWM_ENDvsVH = wilcox.test(data_END[,1], data_VH[,1])
MWM_ENDvsVH_M = wilcox.test(data_END_M[,1], data_VH_M[,1])
MWM_ENDvsVH_F = wilcox.test(data_END_F[,1], data_VH_F[,1])

Rotarod_ENDvsVH = wilcox.test(data_END[,2], data_VH[,2])
Rotarod_ENDvsVH_M = wilcox.test(data_END_M[,2], data_VH_M[,2])
Rotarod_ENDvsVH_F = wilcox.test(data_END_F[,2], data_VH_F[,2])

Beam_ENDvsVH = wilcox.test(data_END[,3], data_VH[,3])
Beam_ENDvsVH_M = wilcox.test(data_END_M[,3], data_VH_M[,3])
Beam_ENDvsVH_F = wilcox.test(data_END_F[,3], data_VH_F[,3])

RM_ENDvsVH = wilcox.test(data_END[,4], data_VH[,4])
RM_ENDvsVH_M = wilcox.test(data_END_M[,4], data_VH_M[,4])
RM_ENDvsVH_F = wilcox.test(data_END_F[,4], data_VH_F[,4])

RMT_ENDvsVH = wilcox.test(data_END[,5], data_VH[,5])
RMT_ENDvsVH_M = wilcox.test(data_END_M[,5], data_VH_M[,5])
RMT_ENDvsVH_F = wilcox.test(data_END_F[,5], data_VH_F[,5])

# Correlations -----------------------------------------------------------

mean_END_F = apply(data_END_F, 2, mean, na.rm = TRUE)
mean_END_M = apply(data_END_M, 2, mean, na.rm = TRUE)
mean_VH_F = apply(data_VH_F, 2, mean, na.rm = TRUE)
mean_VH_M = apply(data_VH_M, 2, mean, na.rm = TRUE)
mean_CV = data.frame("VH_F" = mean_VH_F, "END_F" = mean_END_F, "VH_M" = mean_VH_M, "END_M" =
mean_END_M)

setwd(dir_data_met)
met_av_CB = read.delim("metabolomics_averages_CB.txt")
met_av_HP = read.delim("metabolomics_averages_HP.txt")
prot_av_CB = read.delim("proteomics_averages_CB.txt")
prot_av_HP = read.delim("proteomics_averages_HP.txt")

setwd(dir_data_trans)
rna_av_CB = read.delim("transcriptomics_rna_averages_CB.txt")
rna_av_HP = read.delim("transcriptomics_rna_averages_HP.txt")
mirna_av_CB = read.delim("transcriptomics_mirna_averages_CB.txt")
mirna_av_HP = read.delim("transcriptomics_mirna_averages_HP.txt")
```

```
setwd(dir_sig)
sig_rna_HP = as.vector(read.delim("significant_features_rna_HP.txt")[,1])
sig_rna_CB = as.vector(read.delim("significant_features_rna_CB.txt")[,1])
sig_metab_HP = as.vector(read.delim("significant_features_metab_HP.txt")[,1])
sig_metab_CB = as.vector(read.delim("significant_features_metab_CB.txt")[,1])
sig_proteom_CB = as.vector(read.delim("significant_features_prot_CB.txt")[,1])
sig_proteom_HP = as.vector(read.delim("significant_features_prot_HP.txt")[,1])

setwd(dir_pval_trans)
p_val_mirna_pesti = read.delim("Pesti_mirna.txt")
P_val_mirna_sp = read.delim("Pesti∑Sex_mirna.txt")
test1 = rownames(p_val_mirna_pesti[which(p_val_mirna_pesti[,1] < 0.05),])
test2 = rownames(p_val_mirna_pesti[which(p_val_mirna_pesti[,3] < 0.05),])

setwd(dir_output)
pdf("correlation_plots_num.pdf", width = 3.5*8, height = 3.5*2)
cor_m_CB = t(cor(t(met_av_CB[sig_metab_CB,]), t(mean_CV)))
cor_m_HP = t(cor(t(na.omit(met_av_HP[sig_metab_HP,])), t(mean_CV)))
cor_p_CB = t(cor(t(prot_av_CB[sig_proteom_CB,]), t(mean_CV)))
cor_p_HP = t(cor(t(na.omit(prot_av_HP[sig_proteom_HP,])), t(mean_CV)))
cor_r_CB = t(cor(t(rna_av_CB[sig_rna_CB,]), t(mean_CV)))
cor_r_HP = t(cor(t(na.omit(rna_av_HP[sig_rna_HP,])), t(mean_CV)))
cor_mir_CB = t(cor(t(mirna_av_CB[test1,]), t(mean_CV)))
cor_mir_HP = t(cor(t(mirna_av_CB[test2,]), t(mean_CV)))

corrplot(cor_m_CB, method="number")
corrplot(cor_m_HP, method="number")
corrplot(cor_p_CB, method="number")
corrplot(cor_p_HP, method="number")
corrplot(cor_r_CB, method="number")
corrplot(cor_r_HP, method="number")
corrplot(cor_mir_CB, method="number")
corrplot(cor_mir_HP, method="number")
dev.off()
```

7.1.10. Attachment X: Data formatting for Paintomics script

```
# Script to prepare data for Paintomics.
# Metabolites are detected just with their name. No need to change IDs.
# Proteins must have Uniprot ID. They already have them, so again, no need for change.
# Transcriptomics must have ENSEMBL Gene ID. They come with RefSeq IDs.

dir_data_log2_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/log2_tables/"
dir_data_log2_set01 = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/log2_tables/"
dir_data_p_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/pvalue_tables/"
dir_data_p_set01 = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/pvalue_tables/"
dir_data_set01 = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/data/"
dir_data_trans = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data"
dir_miRNA = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/miRNA/"
dir_output = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/paintomics/output"

library(plyr)

# Import miRNA targets ----------------------------------------------------

setwd(dir_miRNA)
mirna_predictions = read.delim("mirna_predictions.txt")

# Import all data we want to re-format ------------------------------------

# Log2 tables
setwd(dir_data_log2_trans)
mirna_log2 = read.delim("transcriptomics_mirna_log2.txt")
rna_log2 = read.delim("transcriptomics_rna_log2.txt")

setwd(dir_data_log2_set01)
metab_log2 = read.delim("metabolomics_log2.txt")
proteom_log2 = read.delim("proteomics_log2.txt")

# P-value tables
setwd(dir_data_p_trans)
mirna_p_pesti_CB = read.delim("Pesti_mirna_CB.txt")
mirna_p_pesti_HP = read.delim("Pesti_mirna_HP.txt")
rna_p_pesti_CB = read.delim("Pesti_rna_CB.txt")
rna_p_pesti_HP = read.delim("Pesti_rna_HP.txt")
mirna_p_sp_CB = read.delim("Pesti∑Sex_mirna_CB.txt")
mirna_p_sp_HP = read.delim("Pesti∑Sex_mirna_HP.txt")
rna_p_sp_CB = read.delim("Pesti∑Sex_rna_CB.txt")
rna_p_sp_HP = read.delim("Pesti∑Sex_rna_HP.txt")

setwd(dir_data_p_set01)
metab_p_pesti_CB = read.delim("Pesti_met_CB.txt")
metab_p_pesti_HP = read.delim("Pesti_met_HP.txt")
proteom_p_pesti_CB = read.delim("Pesti_prot_CB.txt")
proteom_p_pesti_HP = read.delim("Pesti_prot_HP.txt")
metab_p_sp_CB = read.delim("Pesti∑Sex_met_CB.txt")
metab_p_sp_HP = read.delim("Pesti∑Sex_met_HP.txt")
proteom_p_sp_CB = read.delim("Pesti∑Sex_prot_CB.txt")
proteom_p_sp_HP = read.delim("Pesti∑Sex_prot_HP.txt")

# Bind p-values for CB and HP
mirna_p_pesti = as.data.frame(t(rbind.fill(as.data.frame(t(mirna_p_pesti_CB)),
as.data.frame(t(mirna_p_pesti_HP)))))
colnames(mirna_p_pesti) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")
rna_p_pesti = as.data.frame(t(rbind.fill(as.data.frame(t(rna_p_pesti_CB)),
as.data.frame(t(rna_p_pesti_HP)))))
colnames(rna_p_pesti) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")

mirna_p_sp = as.data.frame(t(rbind.fill(as.data.frame(t(mirna_p_sp_CB)),
as.data.frame(t(mirna_p_sp_HP)))))
colnames(mirna_p_sp) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")
rna_p_sp = as.data.frame(t(rbind.fill(as.data.frame(t(rna_p_sp_CB)), as.data.frame(t(rna_p_sp_HP)))))
```

```
colnames(rna_p_sp) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")

metab_p_pesti = as.data.frame(t(rbind.fill(as.data.frame(t(metab_p_pesti_CB)),
as.data.frame(t(metab_p_pesti_HP)))))
colnames(metab_p_pesti) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")
proteom_p_pesti = as.data.frame(t(rbind.fill(as.data.frame(t(proteom_p_pesti_CB)),
as.data.frame(t(proteom_p_pesti_HP)))))
colnames(proteom_p_pesti) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")

metab_p_sp = as.data.frame(t(rbind.fill(as.data.frame(t(metab_p_sp_CB)),
as.data.frame(t(metab_p_sp_HP)))))
colnames(metab_p_sp) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")
proteom_p_sp = as.data.frame(t(rbind.fill(as.data.frame(t(proteom_p_sp_CB)),
as.data.frame(t(proteom_p_sp_HP)))))
colnames(metab_p_sp) = c("P.val_CB", "Adj.P.val_CB", "P.val_HP", "Adj.P.val_HP")

# Export for future use
setwd(dir_data_p_set01)
write.table(metab_p_pesti, "Pesti_met.txt", sep = "\t")
write.table(proteom_p_pesti, "Pesti_prot.txt", sep = "\t")
write.table(metab_p_sp, "Pesti∑Sex_met.txt", sep = "\t")
write.table(proteom_p_sp, "Pesti∑Sex_prot.txt", sep = "\t")

setwd(dir_data_p_trans)
write.table(mirna_p_pesti, "Pesti_mirna.txt", sep = "\t")
write.table(rna_p_pesti, "Pesti_rna.txt", sep = "\t")
write.table(mirna_p_sp, "Pesti∑Sex_mirna.txt", sep = "\t")
write.table(rna_p_sp, "Pesti∑Sex_rna.txt", sep = "\t")


# Import non-processed data ------------------------------------------------

setwd(dir_data_trans)
rna_nonp = read.delim("rna_NonProcessed.txt")
mirna_nonp = read.delim("mirna_NonProcessed.txt")

setwd(dir_data_set01)
metab_nonp = as.data.frame(t(read.delim("metabolomics01.txt", header = TRUE, as.is = TRUE, row.names =
1,
             check.names = FALSE, dec = ",")))
proteom_nonp = as.data.frame(t(read.delim("proteomics01.txt", header = TRUE, as.is = TRUE, row.names =
1,
               check.names = FALSE, dec = ",")))


# Prepare log2 RNA-seq/metab/proteom table ---------------------------------------------

# Replace NAs with zeroes
rna_log2[is.na(rna_log2)] = 0
metab_log2[is.na(metab_log2)] = 0
proteom_log2[is.na(proteom_log2)] = 0

# We import original data and add those filtered genes/metabolites/prots to tables
elim_rna = setdiff(rownames(rna_nonp), rownames(rna_log2))
rna = data.frame(rna_log2)
rna[elim_rna,] = rep(0, 4)

elim_metab = setdiff(rownames(metab_nonp), rownames(metab_log2))
metab = data.frame(metab_log2)
metab[elim_metab,] = rep(0, 4)

elim_proteom = setdiff(rownames(proteom_nonp), rownames(proteom_log2))
proteom = data.frame(proteom_log2)
proteom[elim_proteom,] = rep(0, 4)

setwd(dir_output)
write.table(rna, "rnaseqlog2.txt", sep = "\t", quote = FALSE)
write.table(metab, "metabolomicslog2.txt", sep = "\t", quote = FALSE)
write.table(proteom, "proteomicslog2.txt", sep = "\t", quote = FALSE)
```

```
# Prepare log2 miRNA-seq table ----------------------------------------------------

list_target_frames = list()
for (i in 1:nrow(mirna_log2)) {
  Genes = as.character(mirna_predictions[which(mirna_predictions["ID"] == rownames(mirna_log2)[i]),
"Gene"])
  log2_CB_F = rep(mirna_log2[i, 1], length(Genes))
  log2_CB_M = rep(mirna_log2[i, 2], length(Genes))
  log2_HP_F = rep(mirna_log2[i, 3], length(Genes))
  log2_HP_M = rep(mirna_log2[i, 4], length(Genes))
  list_target_frames[[i]] = data.frame("log2_CB_F" = log2_CB_F,
                                       "log2_CB_M" = log2_CB_M,
                                       "log2_HP_F" = log2_HP_F,
                                       "log2_HP_M" = log2_HP_M)
  list_target_frames[[i]] = data.frame("Genes" = Genes , list_target_frames[[i]])
}
names(list_target_frames) = rownames(mirna_log2)
target_log2 = rbind.fill(list_target_frames)
target_log2 = aggregate(target_log2[,c(2:5)], by = list("Genes" = target_log2[,"Genes"]), sum)
rownames(target_log2) = target_log2[,1]
target_log2 = target_log2[,-c(1)]

# Now we add 0s where there's NAs

target_log2[is.na(target_log2)] = 0

# Export

setwd(dir_output)
write.table(target_log2, "mirnaseq_targets_log2.txt", sep = "\t", quote = FALSE)


# Prepare significant features from p-value tables for RNAseq/metab/prot and for each tissue ----------

rna_sig_CB_pesti = rna_p_pesti[which(rna_p_pesti[,1] < 0.05),]
rna_sig_HP_pesti = rna_p_pesti[which(rna_p_pesti[,3] < 0.05),]
rna_sig_CB_sp = rna_p_sp[which(rna_p_pesti[,1] < 0.05),]
rna_sig_HP_sp = rna_p_sp[which(rna_p_pesti[,3] < 0.05),]

rna_sig_CB = unique(c(rownames(rna_sig_CB_pesti), rownames(rna_sig_CB_sp)))
rna_sig_HP = unique(c(rownames(rna_sig_HP_pesti), rownames(rna_sig_HP_sp)))

metab_sig_CB_pesti = metab_p_pesti[which(metab_p_pesti[,1] < 0.05),]
metab_sig_HP_pesti = metab_p_pesti[which(metab_p_pesti[,3] < 0.05),]
metab_sig_CB_sp = metab_p_sp[which(metab_p_pesti[,1] < 0.05),]
metab_sig_HP_sp = metab_p_sp[which(metab_p_pesti[,3] < 0.05),]

metab_sig_CB = unique(c(rownames(metab_sig_CB_pesti), rownames(metab_sig_CB_sp)))
metab_sig_HP = unique(c(rownames(metab_sig_HP_pesti), rownames(metab_sig_HP_sp)))

proteom_sig_CB_pesti = proteom_p_pesti[which(proteom_p_pesti[,1] < 0.05),]
proteom_sig_HP_pesti = proteom_p_pesti[which(proteom_p_pesti[,3] < 0.05),]
proteom_sig_CB_sp = proteom_p_sp[which(proteom_p_pesti[,1] < 0.05),]
proteom_sig_HP_sp = proteom_p_sp[which(proteom_p_pesti[,3] < 0.05),]

proteom_sig_CB = unique(c(rownames(proteom_sig_CB_pesti), rownames(proteom_sig_CB_sp)))
proteom_sig_HP = unique(c(rownames(proteom_sig_HP_pesti), rownames(proteom_sig_HP_sp)))

setwd(dir_output)
write.table(rna_sig_CB, "significant_features_rna_CB.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(rna_sig_HP, "significant_features_rna_HP.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(metab_sig_CB, "significant_features_metab_CB.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(metab_sig_HP, "significant_features_metab_HP.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(proteom_sig_CB, "significant_features_prot_CB.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(proteom_sig_HP, "significant_features_prot_HP.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
```

```
# Significant features for miRNAseq ------------------------------------

# Pesticide
list_target_frames_pesti = list()
for (i in 1:nrow(mirna_p_pesti)) {
  Genes = as.character(mirna_predictions[which(mirna_predictions["ID"] == rownames(mirna_p_pesti)[i]),
"Gene"])
  Pval_CB = rep(mirna_p_pesti[i, 1], length(Genes))
  Pval_HP = rep(mirna_p_pesti[i, 3], length(Genes))
  list_target_frames_pesti[[i]] = data.frame("Pval_CB" = Pval_CB, "Pval_HP" = Pval_HP)
  list_target_frames_pesti[[i]] = data.frame("Genes" = Genes , list_target_frames_pesti[[i]])
}
names(list_target_frames_pesti) = rownames(mirna_p_pesti)
target_p = rbind.fill(list_target_frames_pesti)
target_p[is.na(target_p)] = 0
target_p = aggregate(target_p[,c(2:3)], by = list("Genes" = target_p[,"Genes"]), sum)
rownames(target_p) = target_p[,1]
target_p = target_p[,-c(1)]

mirna_sig_CB_Pesti = target_p[which(target_p[,1] < 0.05),]
mirna_sig_HP_Pesti = target_p[which(target_p[,2] < 0.05),]

# Pesticide ∑ Sex
list_target_frames_sp = list()
for (i in 1:nrow(mirna_p_sp)) {
  Genes = as.character(mirna_predictions[which(mirna_predictions["ID"] == rownames(mirna_p_sp)[i]),
"Gene"])
  Pval_CB = rep(mirna_p_sp[i, 1], length(Genes))
  Pval_HP = rep(mirna_p_sp[i, 3], length(Genes))
  list_target_frames_sp[[i]] = data.frame("Pval_CB" = Pval_CB, "Pval_HP" = Pval_HP)
  list_target_frames_sp[[i]] = data.frame("Genes" = Genes , list_target_frames_sp[[i]])
}
names(list_target_frames_sp) = rownames(mirna_p_sp)
target_sp = rbind.fill(list_target_frames_sp)
target_sp[is.na(target_sp)] = 0
target_sp = aggregate(target_sp[,c(2:3)], by = list("Genes" = target_sp[,"Genes"]), sum)
rownames(target_sp) = target_sp[,1]
target_sp = target_sp[,-c(1)]

mirna_sig_CB_sp = target_sp[which(target_sp[,1] < 0.05),]
mirna_sig_HP_sp = target_sp[which(target_sp[,2] < 0.05),]

mirna_sig_CB = unique(c(rownames(mirna_sig_CB_Pesti), rownames(mirna_sig_CB_sp)))
mirna_sig_HP = unique(c(rownames(mirna_sig_HP_Pesti), rownames(mirna_sig_HP_sp)))


setwd(dir_output)
write.table(mirna_sig_CB, "significant_features_mirna_CB.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
write.table(mirna_sig_HP, "significant_features_mirna_HP.txt", sep = "\n", quote = FALSE, row.names =
FALSE, col.names = FALSE)
```

7.1.11. Attachment XI: Expression profile script

```
dir_sig = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/paintomics/output"
dir_averages_01 = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/average_tables/"
dir_averages_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/average_tables/"
dir_output_01 = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/set01/output_DE/expression_profiles/"
dir_output_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/expression_profiles/"
dir_pval_trans =
"~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/output_DE/pvalue_tables/"
dir_data_trans = "~/Dropbox/Felipo_OmicsDENAMICdata/Integration/trancriptomics/data/"

# Import data ------------------------------------------------------------

setwd(dir_sig)
sig_rna_HP = as.vector(read.delim("significant_features_rna_HP.txt")[,1])
sig_rna_CB = as.vector(read.delim("significant_features_rna_CB.txt")[,1])
sig_metab_HP = as.vector(read.delim("significant_features_metab_HP.txt")[,1])
sig_metab_CB = as.vector(read.delim("significant_features_metab_CB.txt")[,1])
sig_proteom_CB = as.vector(read.delim("significant_features_prot_CB.txt")[,1])
sig_proteom_HP = as.vector(read.delim("significant_features_prot_HP.txt")[,1])

setwd(dir_pval_trans)
p_val_mirna_pesti = read.delim("Pesti_mirna.txt")
P_val_mirna_sp = read.delim("Pesti∑Sex_mirna.txt")

setwd(dir_averages_01)
metab_av = read.delim("metabolomics_averages.txt")
proteom_av = read.delim("proteomics_averages.txt")

setwd(dir_averages_trans)
rna_av = read.delim("transcriptomics_rna_averages.txt")
mirna_av = read.delim("transcriptomics_mirna_averages.txt")

# Create expression profiles  ---------------------------------------------

expression_profile <- function(gene, matrix, type) {
  ylabs = c("Gene expression", "miRNA expression", "Metabolite amount", "Protein amount")
  plot(x = c(1,2,4,7), rnorm(4), col = "white", xlab = "Female | Male", ylab = ylabs[type],
       main = gene, xaxt = "n", ylim = c(min(matrix[gene,]),max(matrix[gene,])))
  axis(side = 1, at = c(2,3,5,6), labels = rep(c("VH", "END"),2))
  abline(v = 4, col = "grey", lty = 2)
  # lines(c(1,2), c(media vehiculo, media endosulfan))
  lines(c(2:3), c(matrix[gene,1], matrix[gene,2]), type = "b", lwd = 2, col = "coral1", pch = 19) #CB F
  lines(c(5:6), c(matrix[gene,3], matrix[gene,4]), type = "b", lwd = 2, col = "coral1", pch = 19) #CB M
  lines(c(2:3), c(matrix[gene,5], matrix[gene,6]), type = "b", lwd = 2, col = "darkcyan") #HP F
  lines(c(5:6), c(matrix[gene,7], matrix[gene,8]), type = "b", lwd = 2, col = "darkcyan") #HP M
}

# RNA
setwd(dir_output_trans)
rna_matrix = na.omit(rna_av[unique(c(sig_rna_CB, sig_rna_HP)),])
rna_type = 1
pdf("RNA_expression_profile.pdf", width = 3.5*4, height = 3.5*5)
par(mfcol = c(5, 4))
for (i in 1:nrow(rna_matrix)) {
  gene = rownames(rna_matrix)[i]
  expression_profile(gene, rna_matrix, rna_type)
}
dev.off()


# Metab
setwd(dir_output_01)
metab_matrix = na.omit(metab_av[unique(c(sig_metab_CB, sig_metab_HP)),])
metab_type = 3
pdf("metab_expression_profile.pdf", width = 3.5*4, height = 3.5*5)
par(mfcol = c(5, 4))
```

```
for (i in 1:nrow(metab_matrix)) {
  metab = rownames(metab_matrix)[i]
  expression_profile(metab, metab_matrix, metab_type)
}
dev.off()

# Proteom
setwd(dir_output_01)
proteom_matrix = na.omit(proteom_av[unique(c(sig_proteom_CB, sig_proteom_HP)),])
proteom_type = 4
pdf("proteom_expression_profile.pdf", width = 3.5*4, height = 3.5*5)
par(mfcol = c(5, 4))
for (i in 1:nrow(proteom_matrix)) {
  prot = rownames(proteom_matrix)[i]
  expression_profile(prot, proteom_matrix, proteom_type)
}
dev.off()

# miRNA
setwd(dir_output_trans)
test1 = rownames(p_val_mirna_pesti[which(p_val_mirna_pesti[,1] < 0.05),])
test2 = rownames(p_val_mirna_pesti[which(p_val_mirna_pesti[,3] < 0.05),])
sig_mirna = unique(c(test1, test2))
mirna_matrix = na.omit(mirna_av[sig_mirna,])
mirna_type = 2
pdf("mirna_expression_profile.pdf", width = 3.5*4, height = 3.5*5)
par(mfcol = c(5, 4))
for (i in 1:nrow(mirna_matrix)) {
  mirna = rownames(mirna_matrix)[i]
  expression_profile(mirna, mirna_matrix, mirna_type)
}
dev.off()


# Heatmaps for trans ------------------------------------------------------------

setwd(dir_data_trans)
rna_data_CB = read.delim("Cerebellum_rna_NoNoiseData.txt")
rna_data_HP = read.delim("Hippocampus_rna_NoNoiseData.txt")
rna_DE_data_CB = rna_data_CB[sig_rna_CB,]
rna_DE_data_HP = na.omit(rna_data_HP[sig_rna_HP,])

mirna_data_CB = read.delim("Cerebellum_mirna_NoNoiseData.txt")
mirna_data_HP = read.delim("Hippocampus_mirna_NoNoiseData.txt")
mirna_DE_data_CB = mirna_data_CB[test1,]
mirna_DE_data_HP = mirna_data_HP[test2,]

setwd(dir_output_trans)
pdf("heatmaps_DE.pdf", width = 3.5*3, height = 3.5*2)
heatmap(as.matrix(rna_DE_data_CB), main = "Differentially expressed RNAseq CB", margins = c(10,5))
heatmap(as.matrix(rna_DE_data_HP), main = "Differentially expressed RNAseq HP", margins = c(10,5))
heatmap(as.matrix(mirna_DE_data_CB), main = "Differentially expressed miRNAseq CB", margins = c(10,5))
heatmap(as.matrix(mirna_DE_data_HP), main = "Differentially expressed miRNAseq HP", margins = c(10,5))
dev.off()
```

7.2. Other supplementary material

7.2.1. Attachment XII: Set 03 and Set 10 details

Set 03

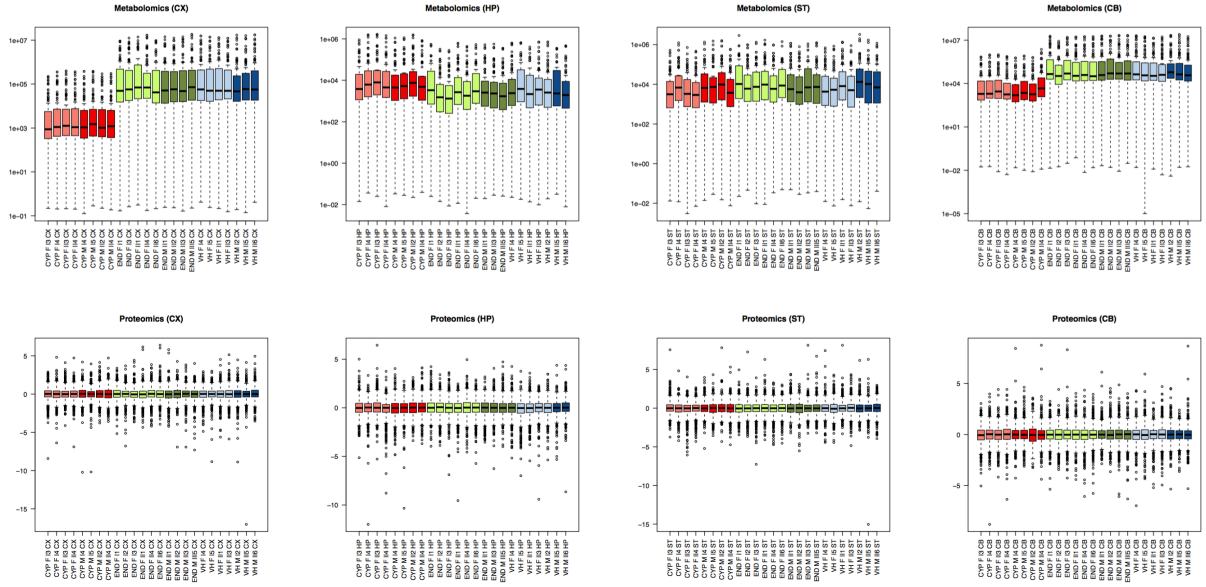
Treatment	Tissues: CB		HP		CX		ST		TOTAL / TISSUE
	F	M	F	M	F	M	F	M	
VH	3	2	3	2	3	2	3	2	5
CYP	1	0	1	0	1	0	1	0	1
CHLOR 0.1	4	1	4	1	4	1	4	1	5
CHLOR 0.3	4	4	4	4	4	4	4	4	8
CHLOR 1	2	2	2	2	2	2	2	2	4
CAR	3	2	3	2	3	2	3	2	5
							TOTAL RATS:		28

Set 10

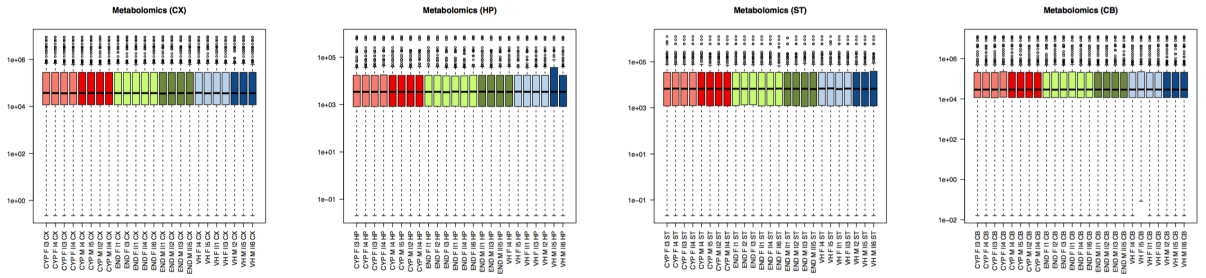
Treatment	Tissues: CB		HP		CX		ST		TOTAL / TISSUE
	F	M	F	M	F	M	F	M	
VH	3	3	3	3	3	3	3	3	6
CYP	2	2	2	2	2	2	2	2	4
CPF	3	3	3	3	3	3	3	3	6
CYP+END	3	3	3	3	3	3	3	3	6
END	3	3	3	3	3	3	3	3	6
							TOTAL RATS:		28

7.2.2. Attachment XIII: Set 01 Boxplots

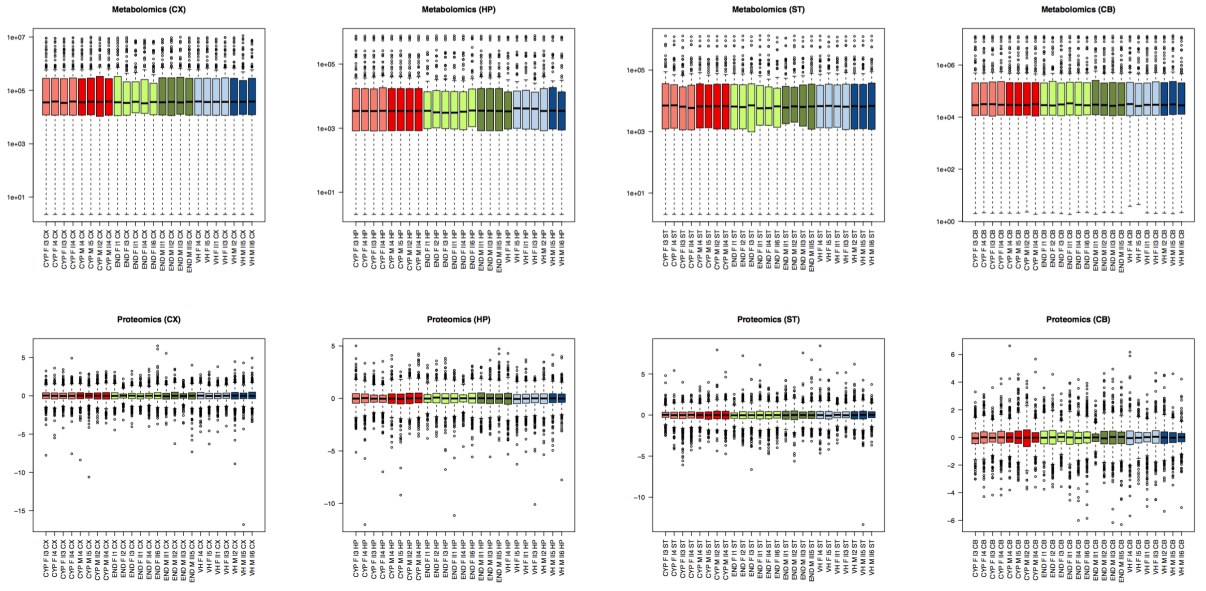
Before any transformation:



After normalization of metabolomics data:

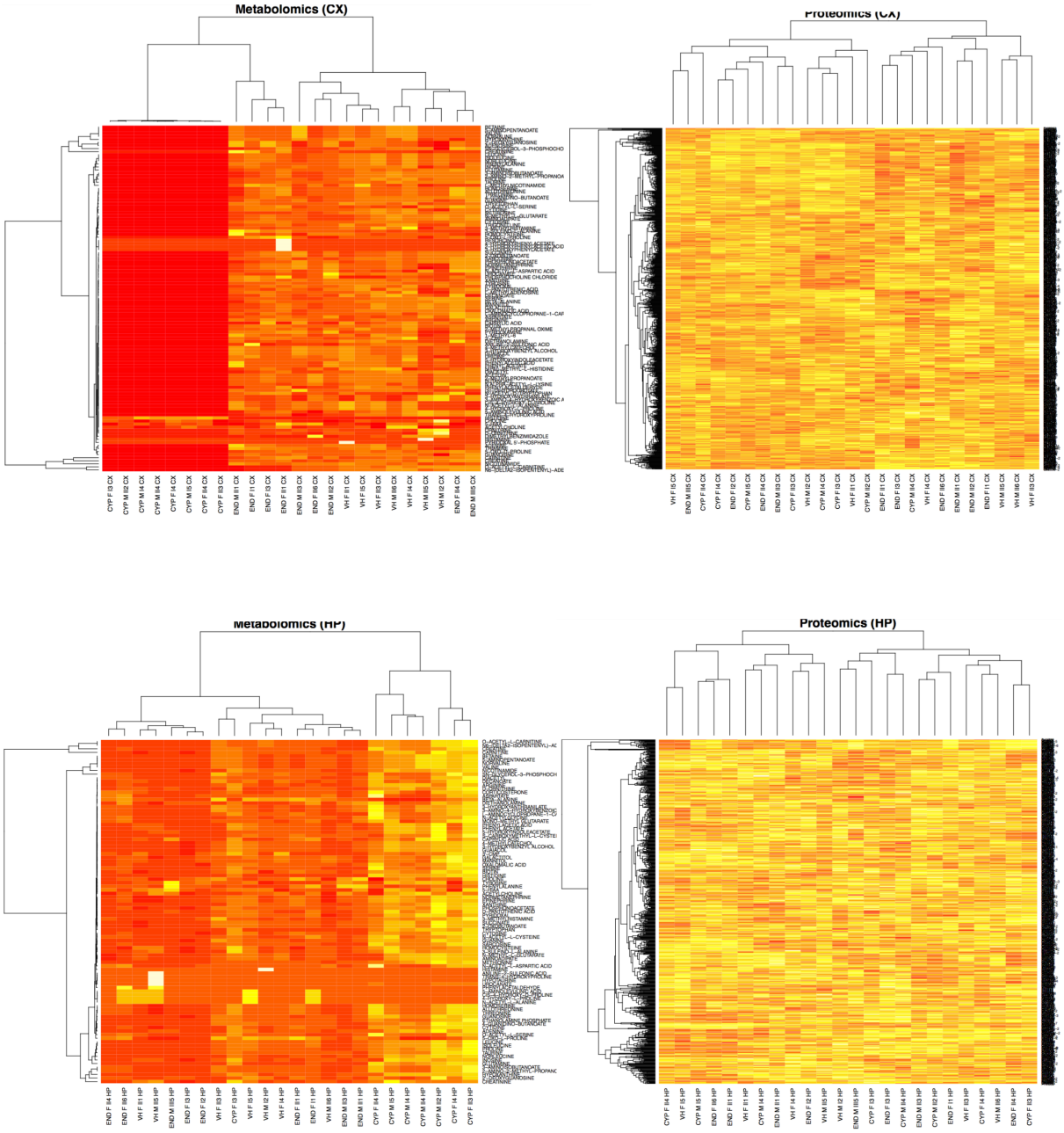


After normalization and arsynseq on metabolomics data and arsynseq on proteomics:

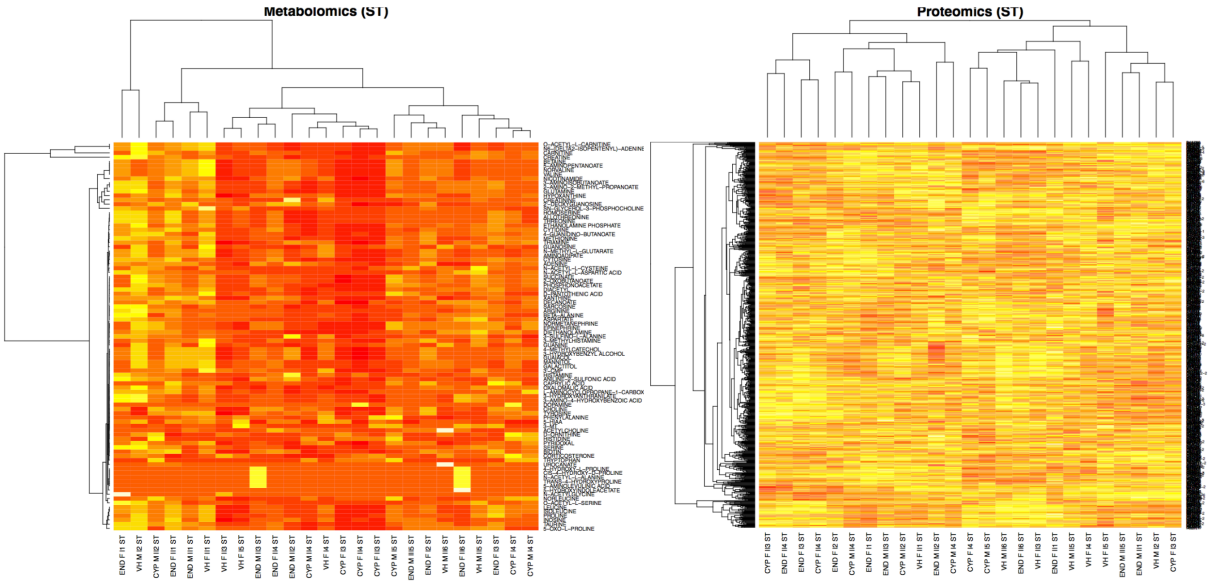


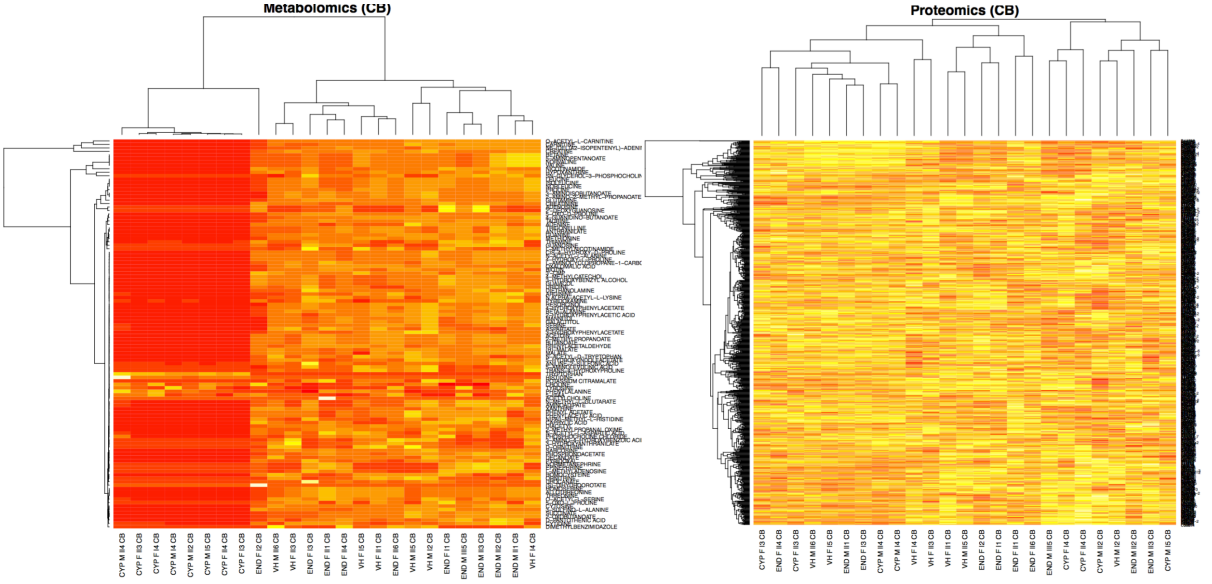
7.2.3. Attachment XIV: Set 01 Heatmaps

Before pre-processing:

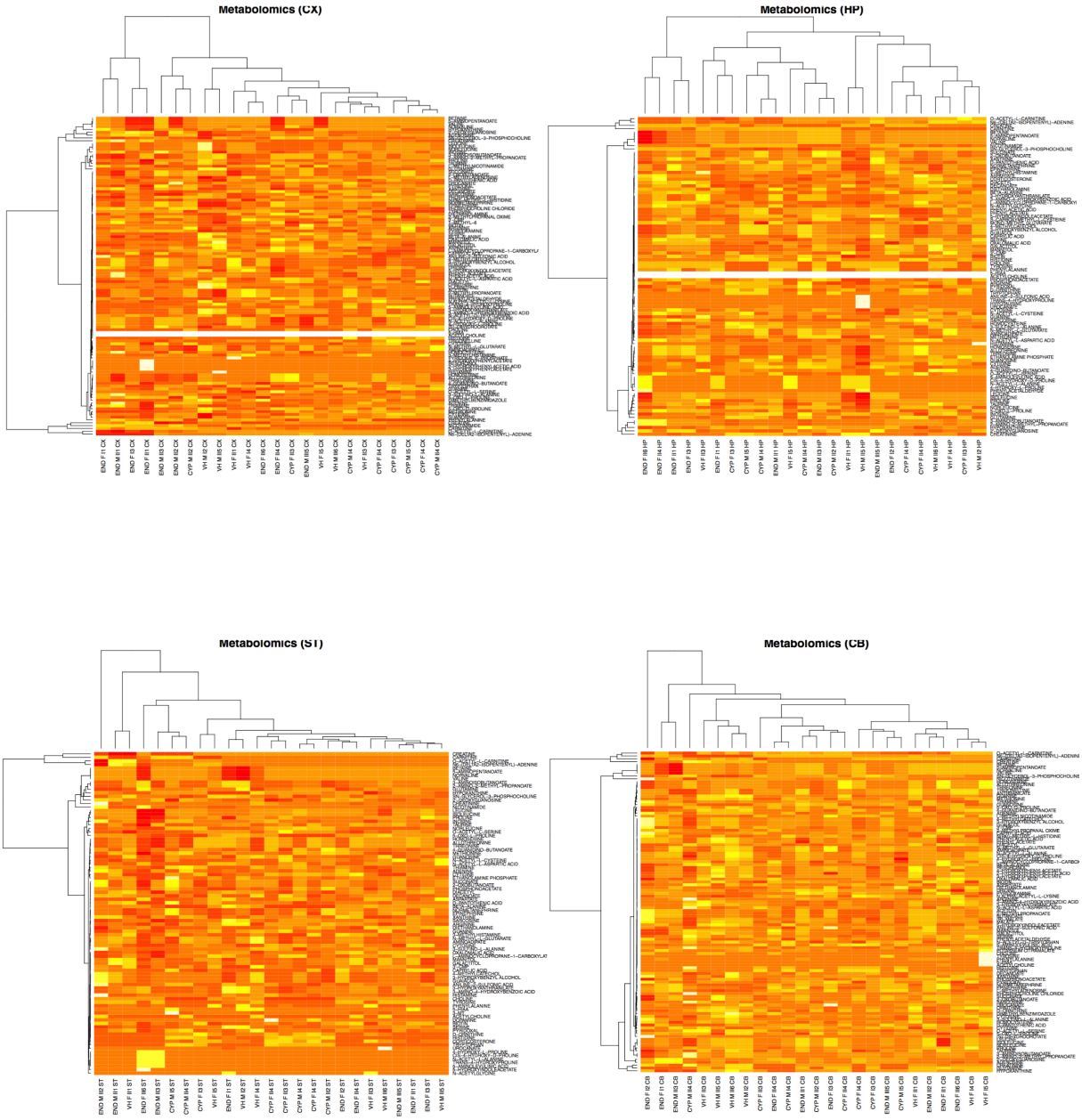


Before pre-processing:

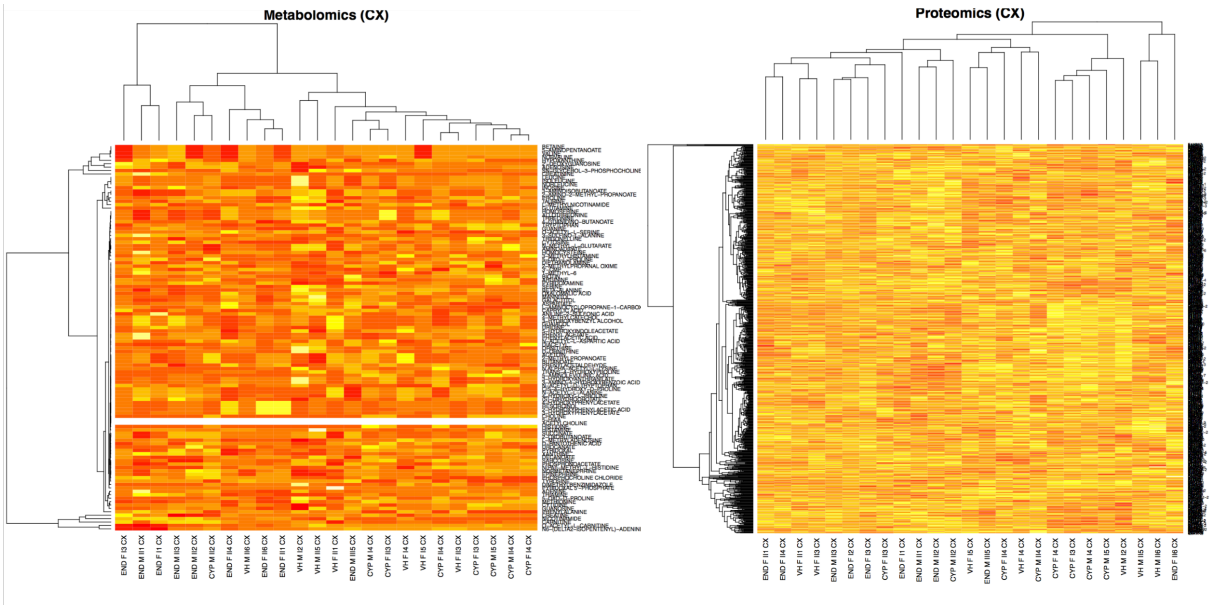


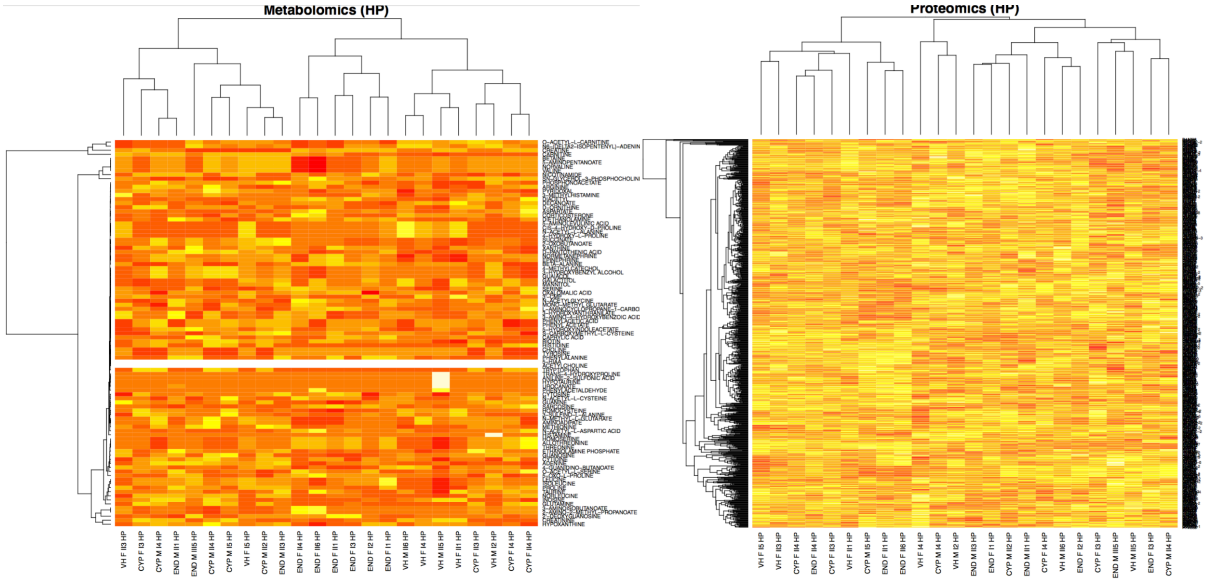


After normalization of metabolomics data:

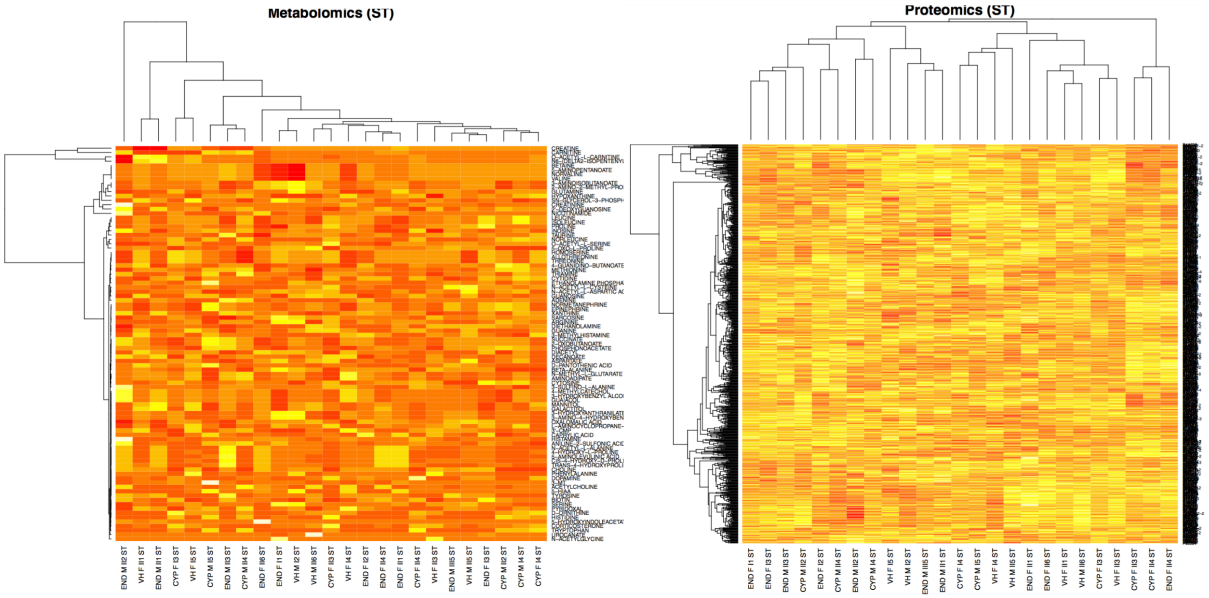


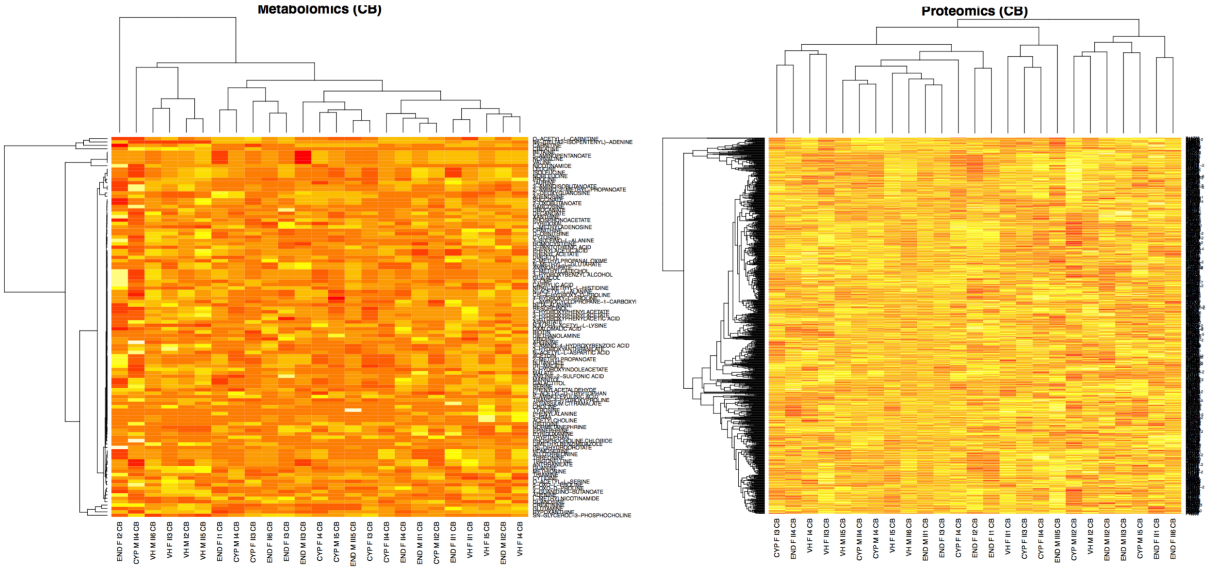
After normalization and arsynseq on of metabolomics data and arsynseq on proteomics:





After normalization and arsynseq on of metabolomics data and arsynseq on proteomics:

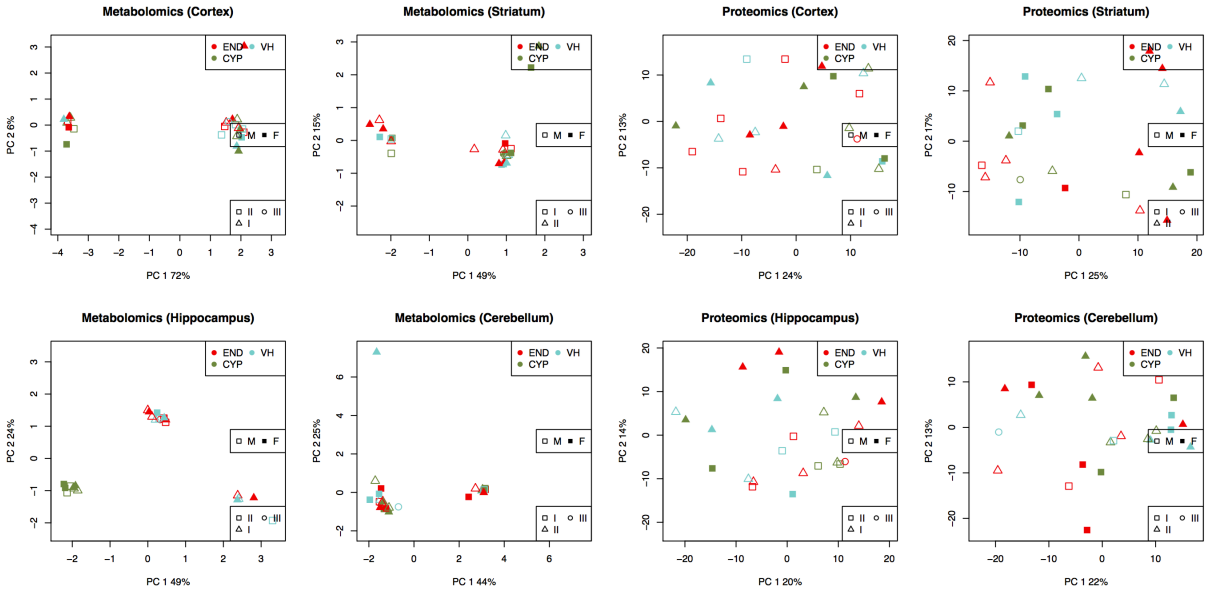




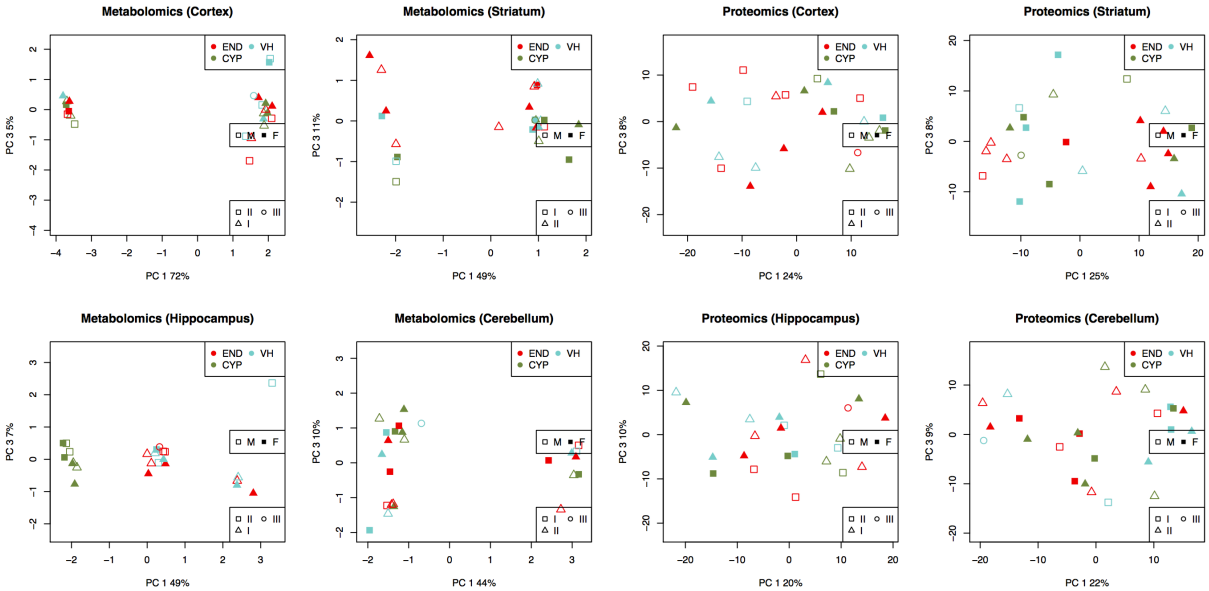
7.2.4. Attachment XV: PCA Set 01

Before pre-processing:

PCA 1 & 2

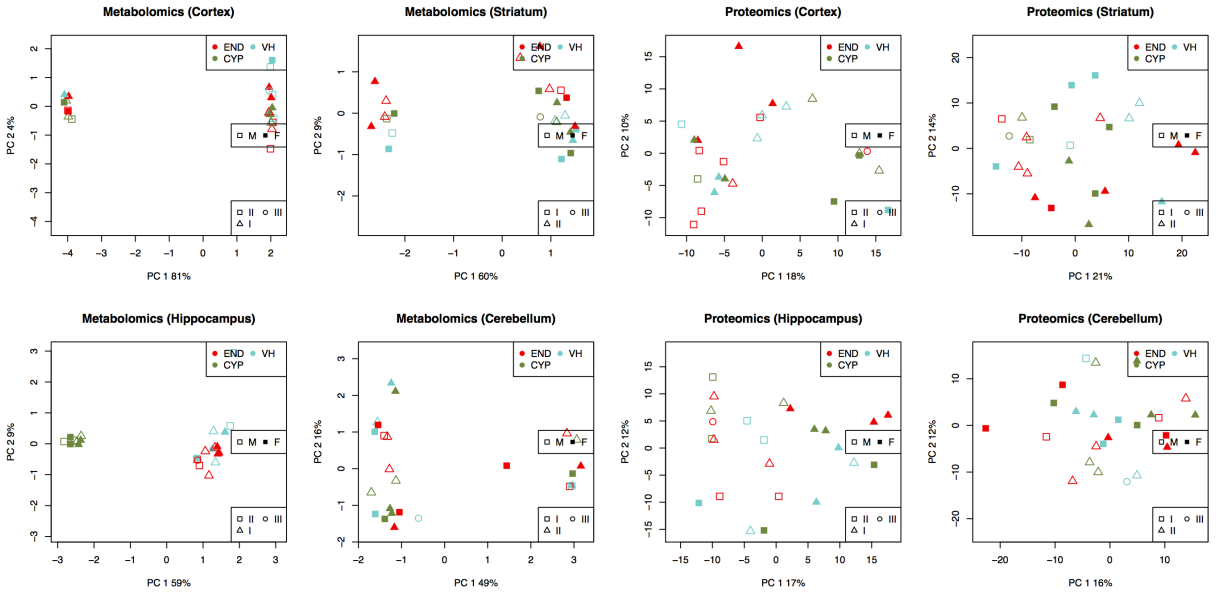


PCA 1 & 3

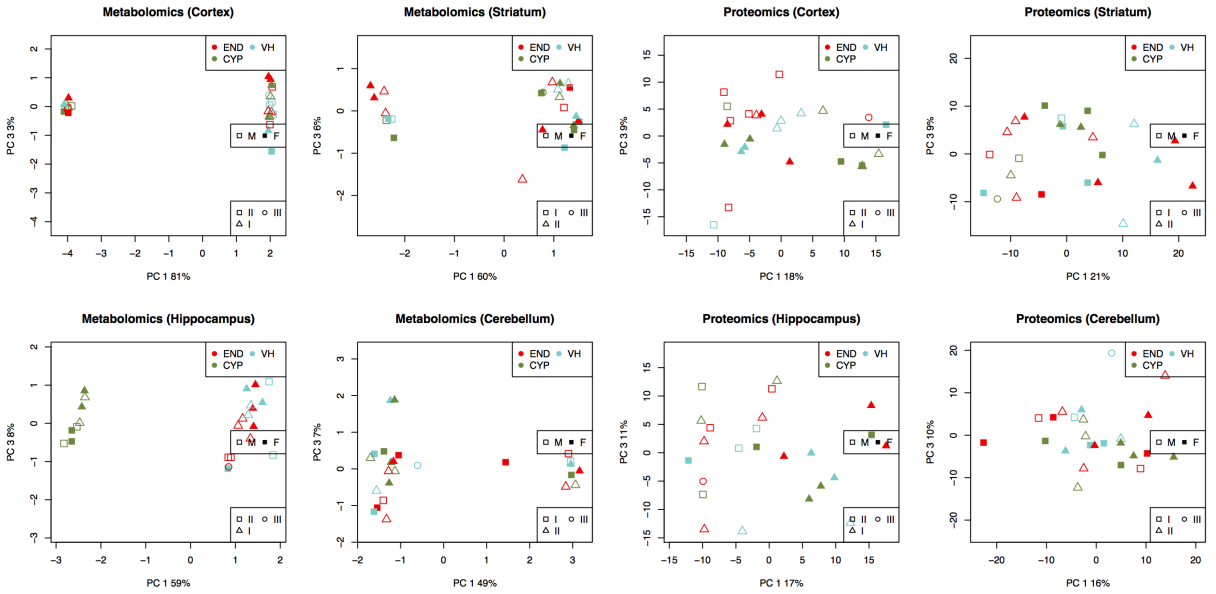


After pre-processing:

PCA 1 & 2

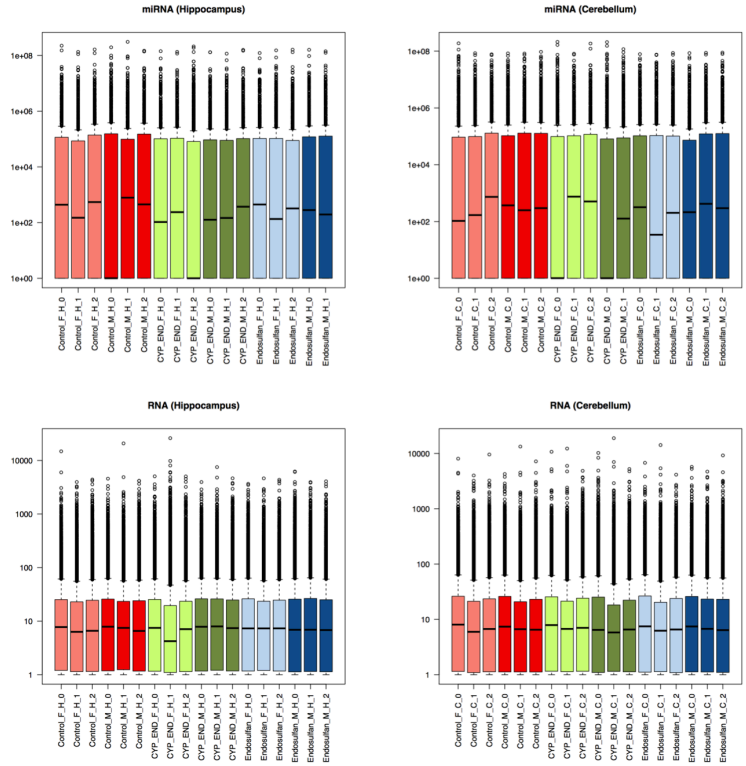


PCA 1 & 3

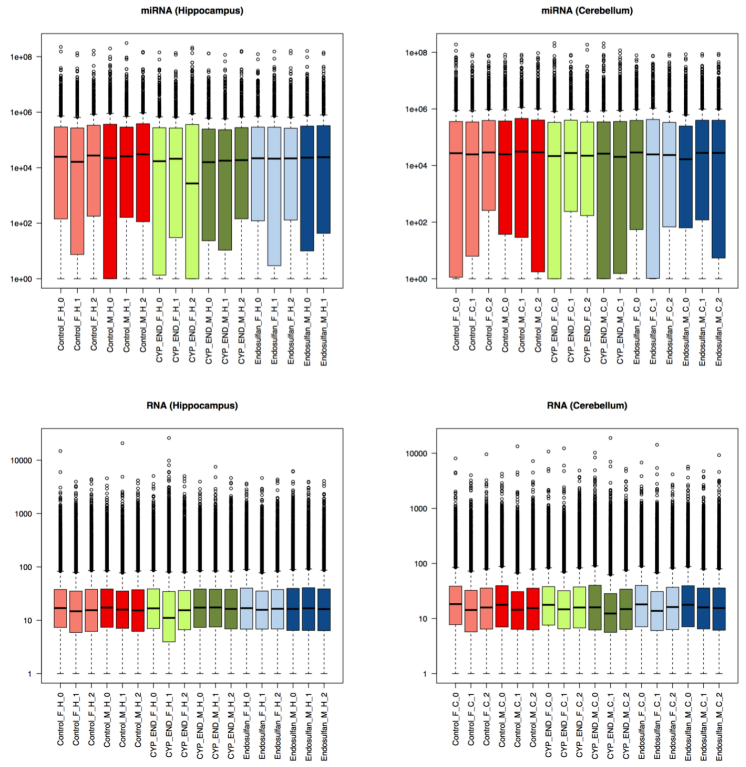


7.2.5. Attachment XVI: Transcriptomics boxplots

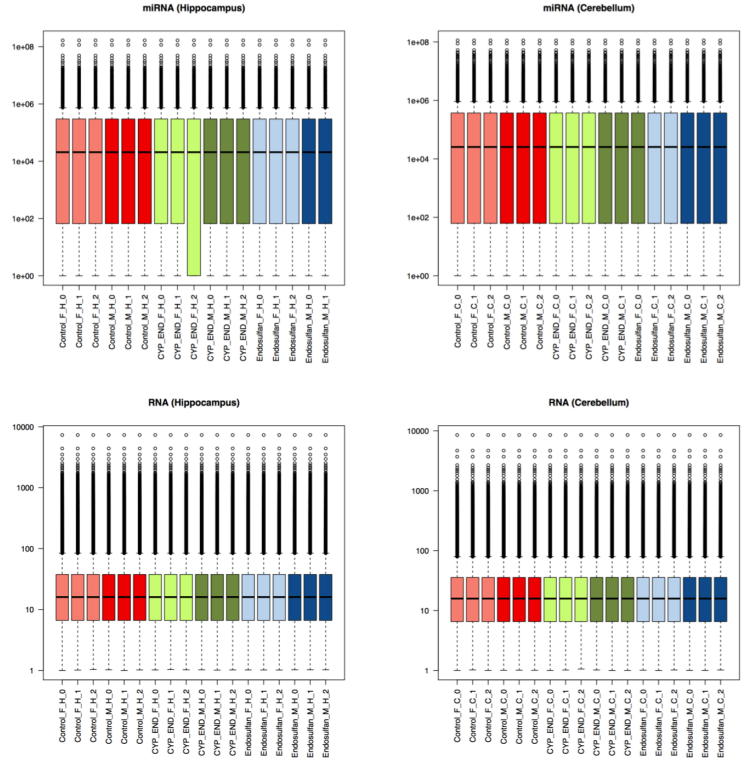
Before pre-processing:



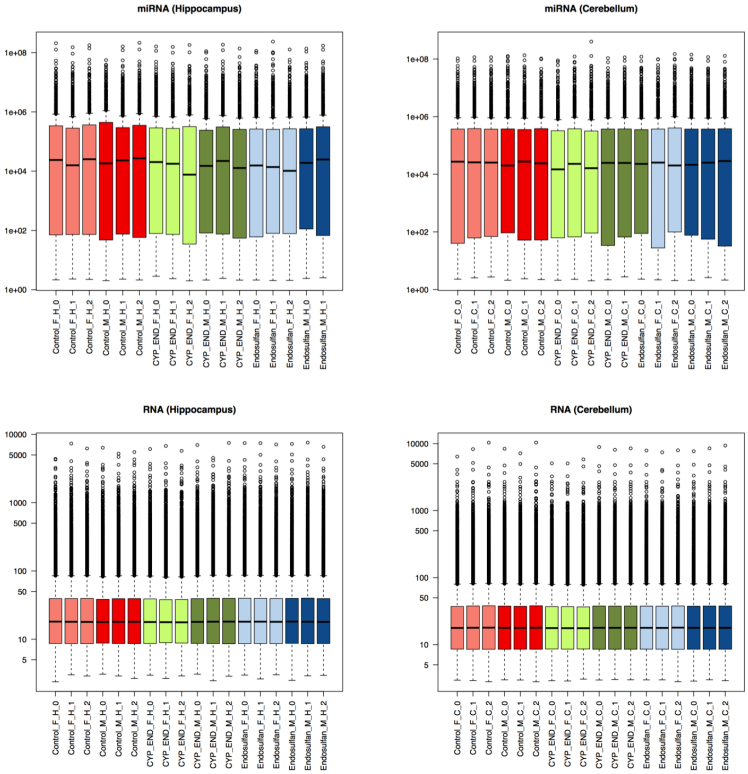
After filter:



After normalization:



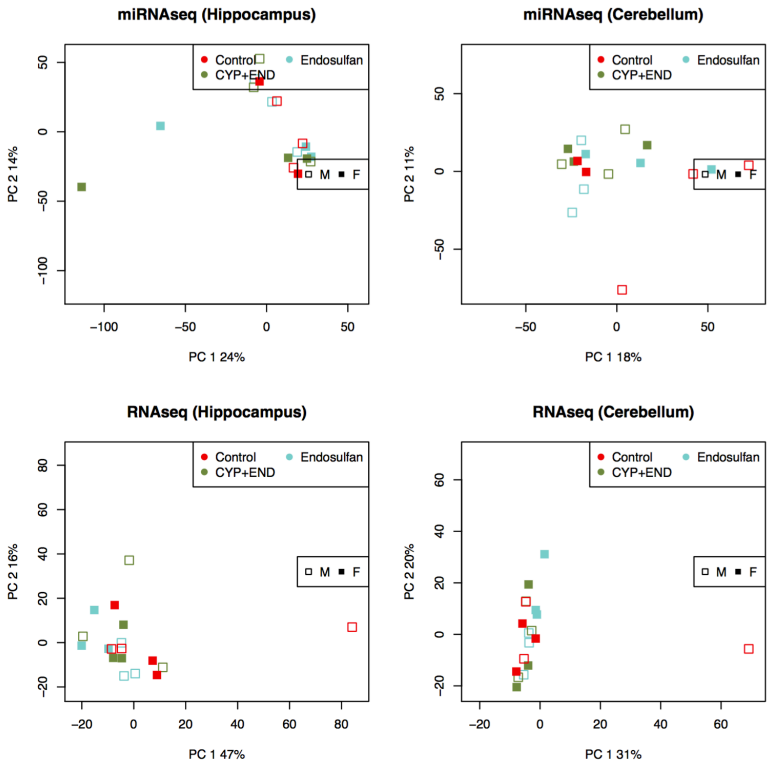
After arsynseq and normalization:



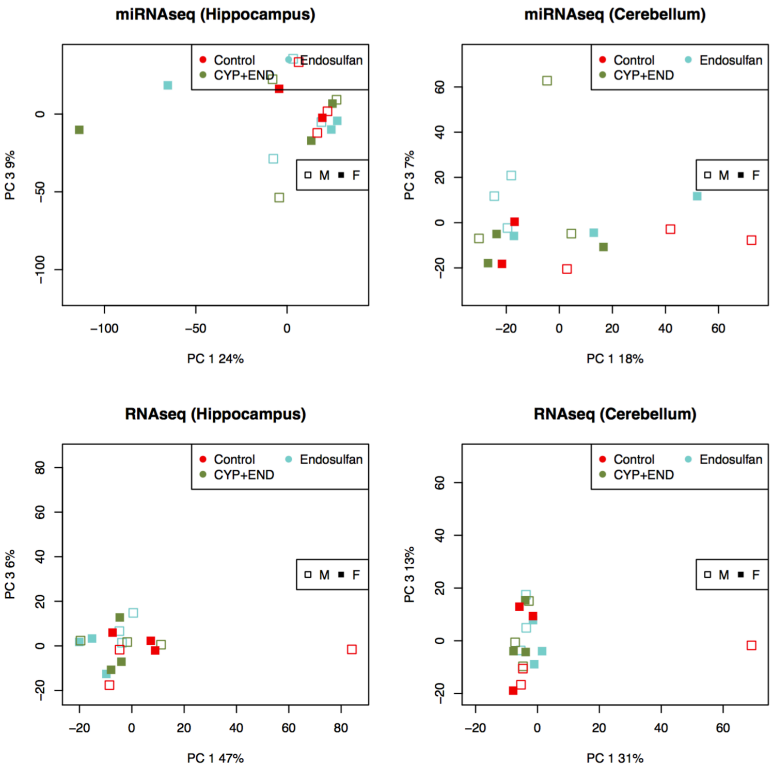
7.2.6. Attachment XVII: PCA transcriptomics

Before pre-processing:

PCA 1 & 2

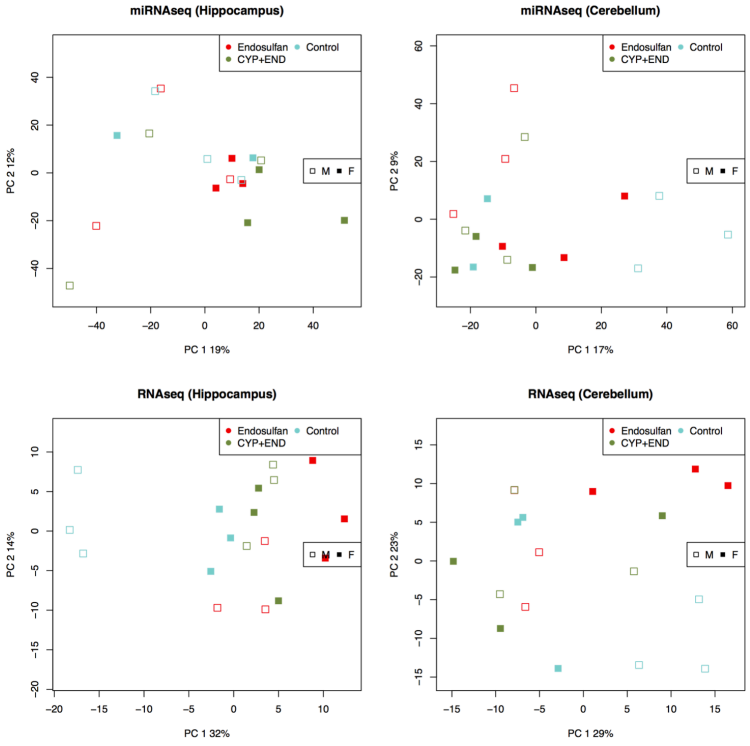


PCA 1 & 3

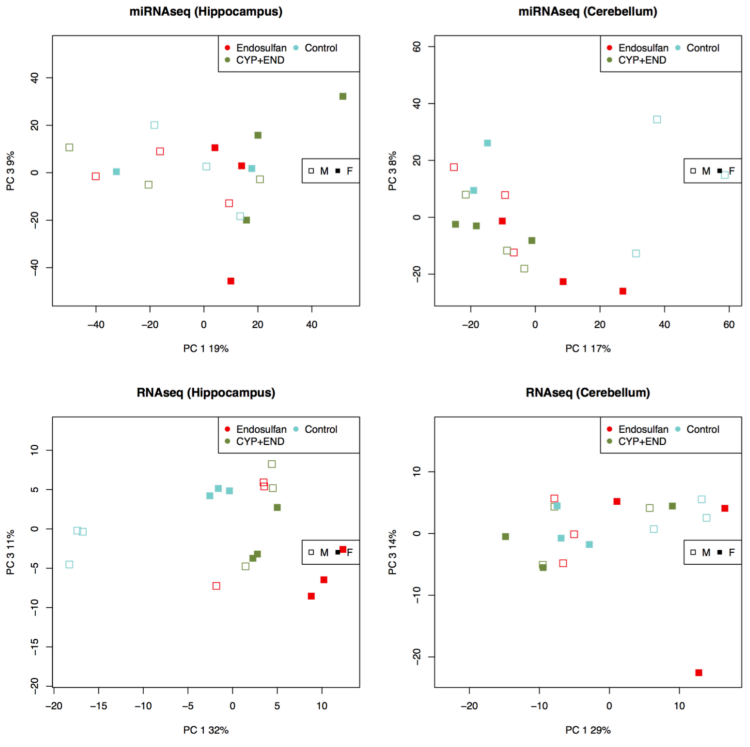


After pre-processing:

PCA 1 & 2

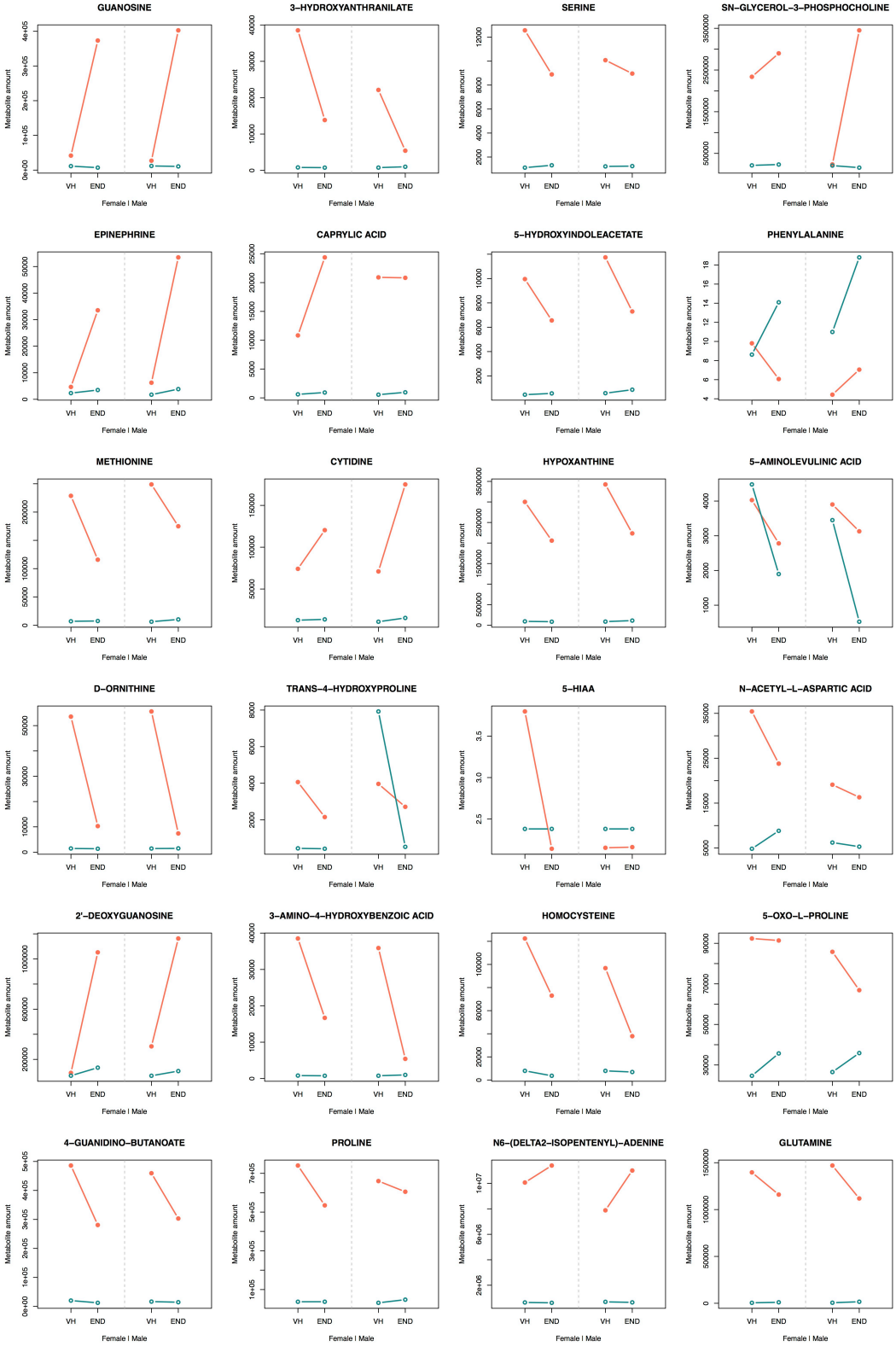


PCA 1 & 3

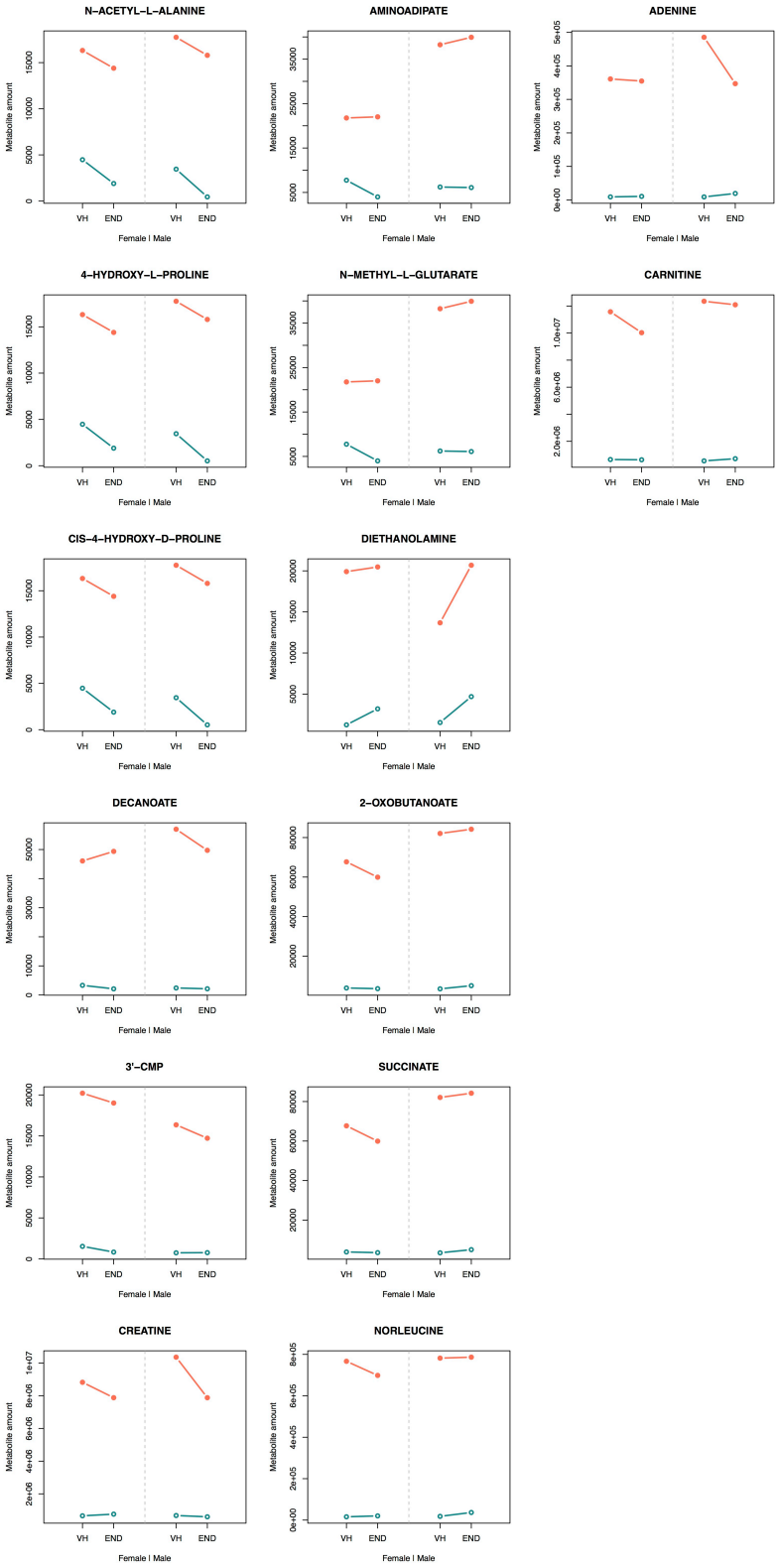


7.2.7. Attachment XVIII: Expression profiles

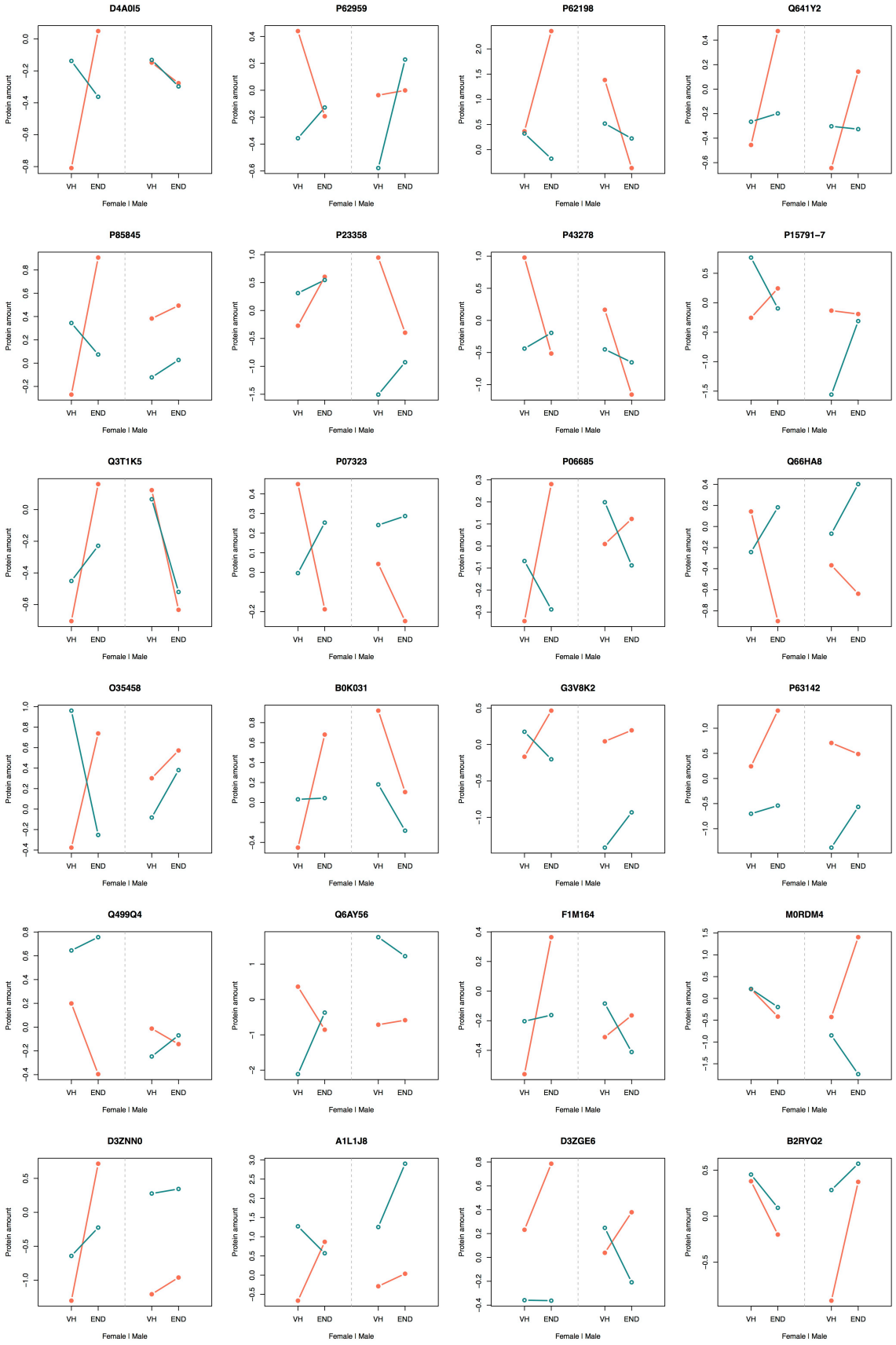
Metabolomics



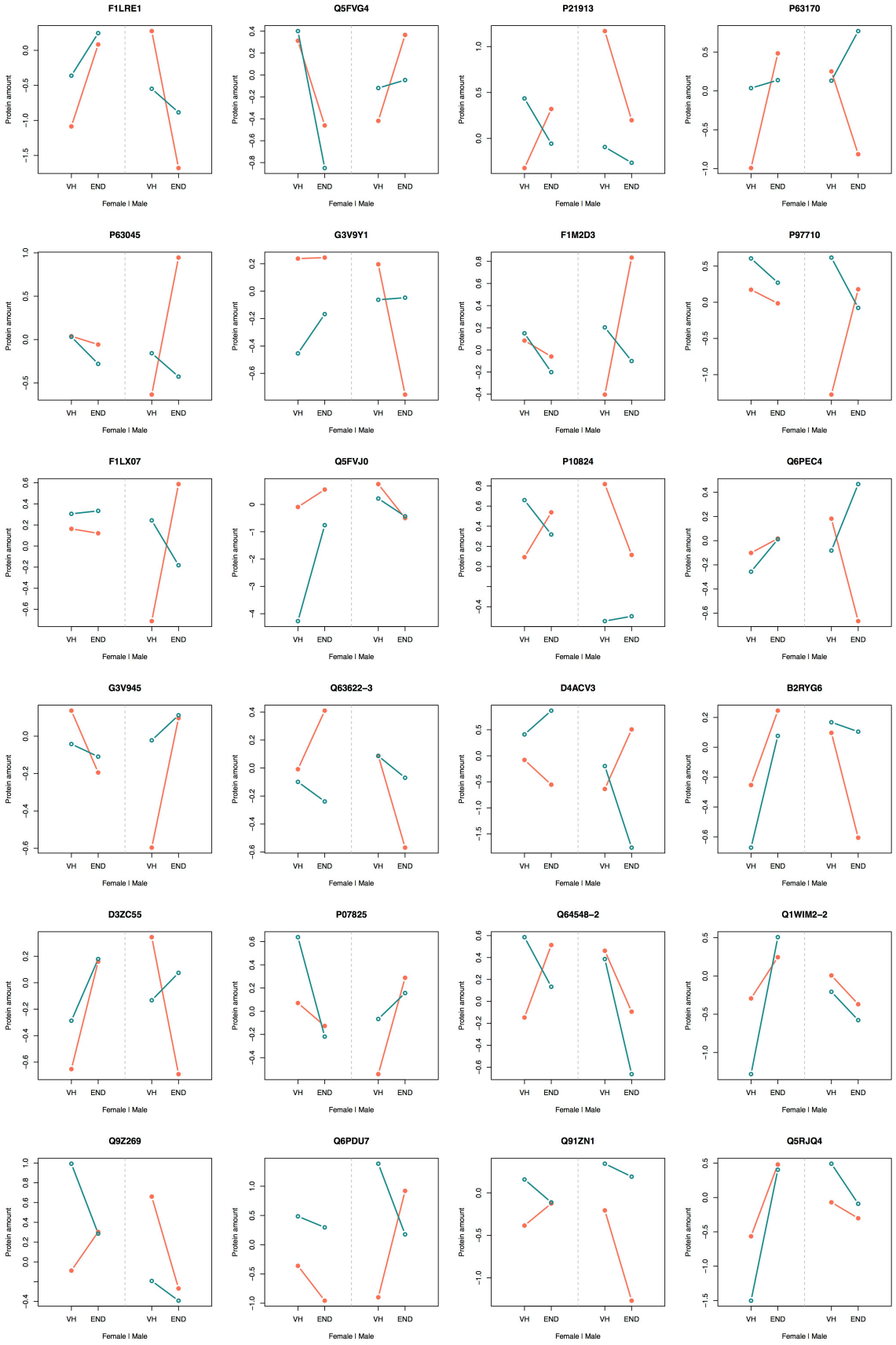
Metabolomics



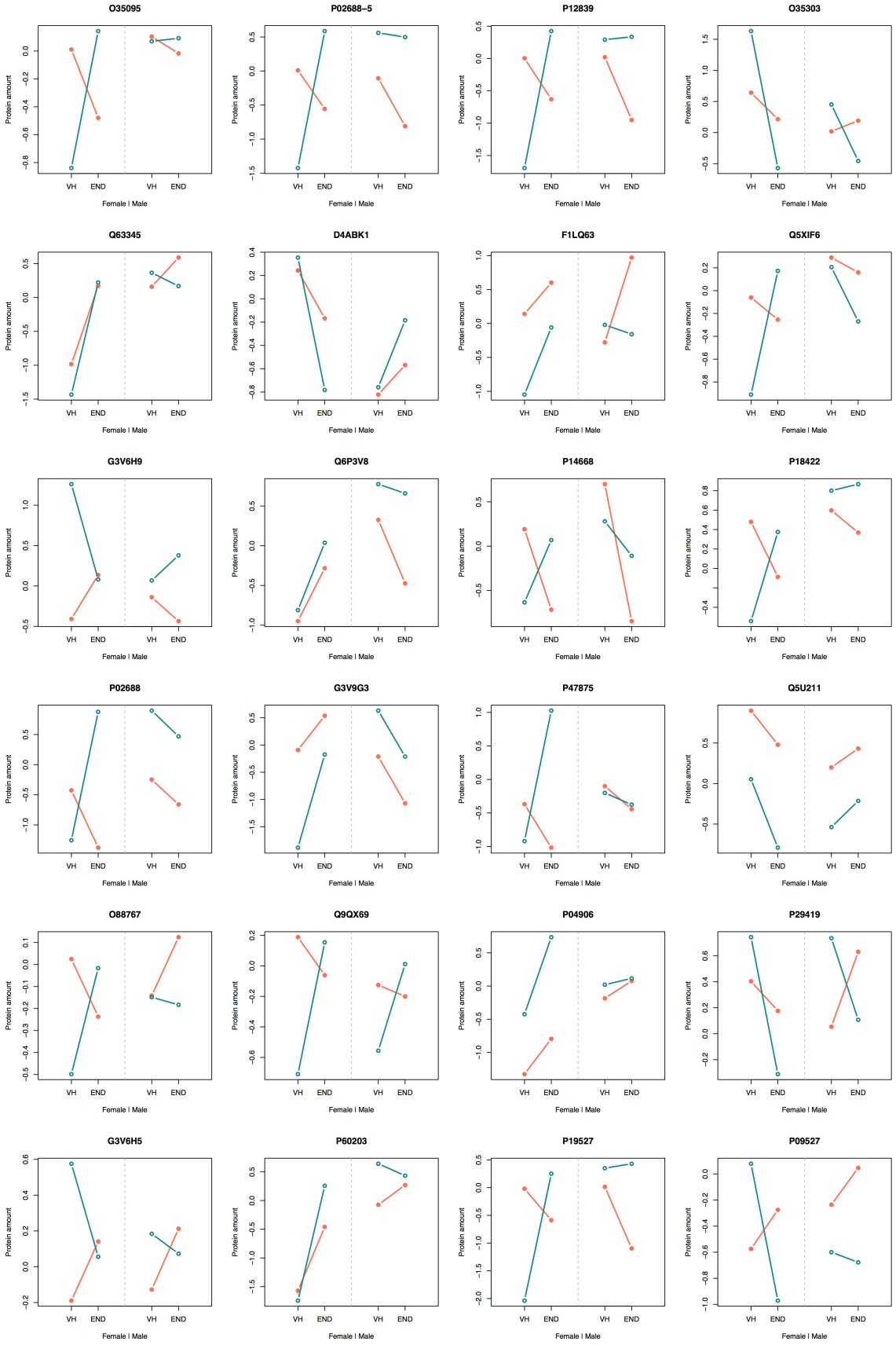
Proteomics


D4A0I5
P62959
P62198
Q641Y2
P85845
P23358
P43278
P15791−7
Q3T1K5
P07323
P06685
Q66HA8
O35458
B0K031
G3V8K2
P63142
Q499Q4
Q6AY56
F1M164
M0RDM4
D3ZNN0
A1L1J8
D3ZGE6
B2RYQ2
Protein amount
VH
END
Female | Male

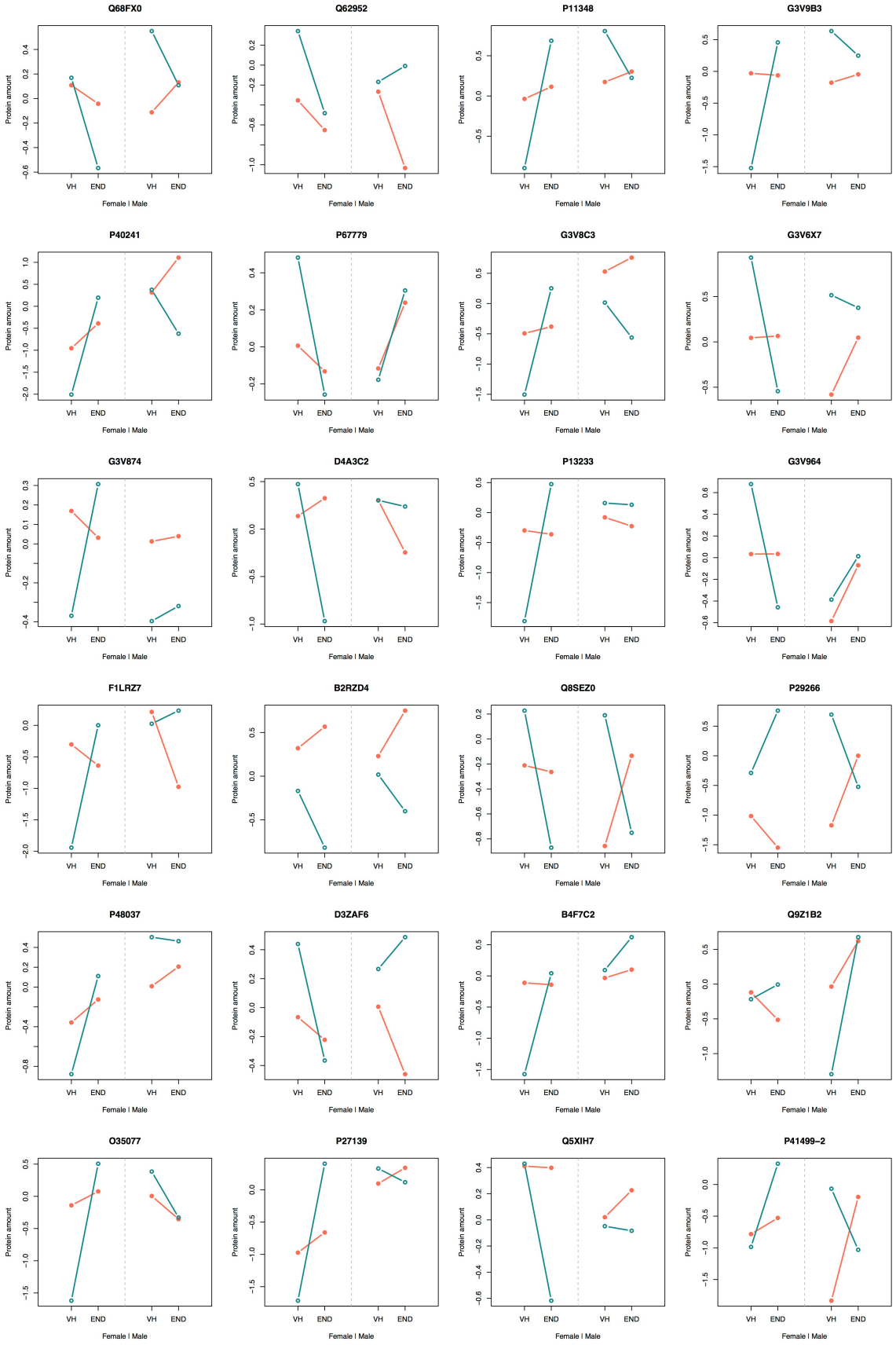
Proteomics


F1LRE1
Q5FVG4
P21913
P63170
P63045
G3V9Y1
F1M2D3
P97710
F1LX07
Q5FVJ0
P10824
Q6PEC4
G3V945
Q63622–3
D4ACV3
B2RYG6
D3ZC55
P07825
Q64548–2
Q1WIM2–2
Q9Z269
Q6PDU7
Q91ZN1
Q5RJQ4
Protein amount
VH
END
Female | Male

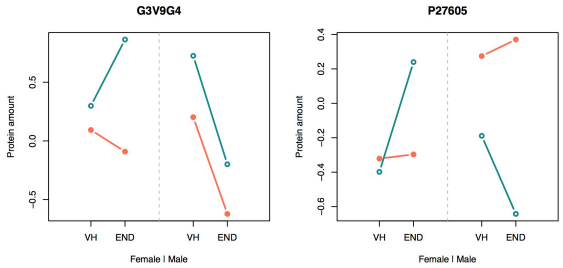
Proteomics


O35095
P02688−5
P12839
O35303
Q63345
D4ABK1
F1LQ63
Q5XIF6
G3V6H9
Q6P3V8
P14668
P18422
P02688
G3V9G3
P47875
Q5U211
O88767
Q9QX69
P04906
P29419
G3V6H5
P60203
P19527
P09527
Protein amount
VH
END
Female | Male

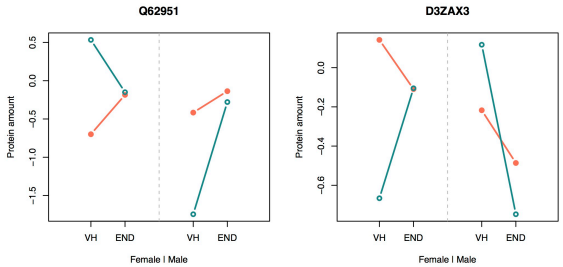
Proteomics


Q68FX0
Q62952
P11348
G3V9B3
P40241
P67779
G3V8C3
G3V6X7
G3V874
D4A3C2
P13233
G3V964
F1LRZ7
B2RZD4
Q8SEZ0
P29266
P48037
D3ZAF6
B4F7C2
Q9Z1B2
O35077
P27139
Q5XIH7
P41499-2
Protein amount
VH
END
Female | Male

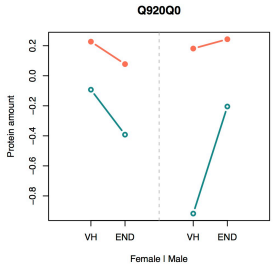
Proteomics

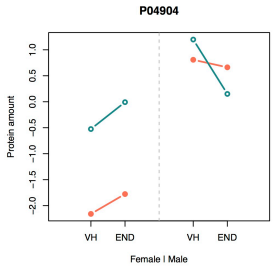

G3V9G4
Protein amount
0.5
0.0
−0.5
VH
END
VH
END
Female | Male

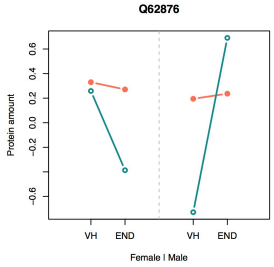
P27605
Protein amount
0.4
0.2
0.0
−0.2
−0.4
−0.6
VH
END
VH
END
Female | Male

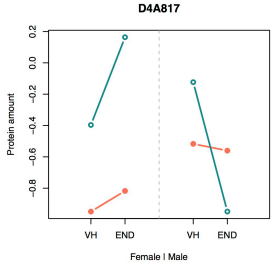

Q62951
Protein amount
0.5
0.0
−0.5
−1.0
−1.5
VH
END
VH
END
Female | Male

D3ZAX3
Protein amount
0.0
−0.2
−0.4
−0.6
VH
END
VH
END
Female | Male

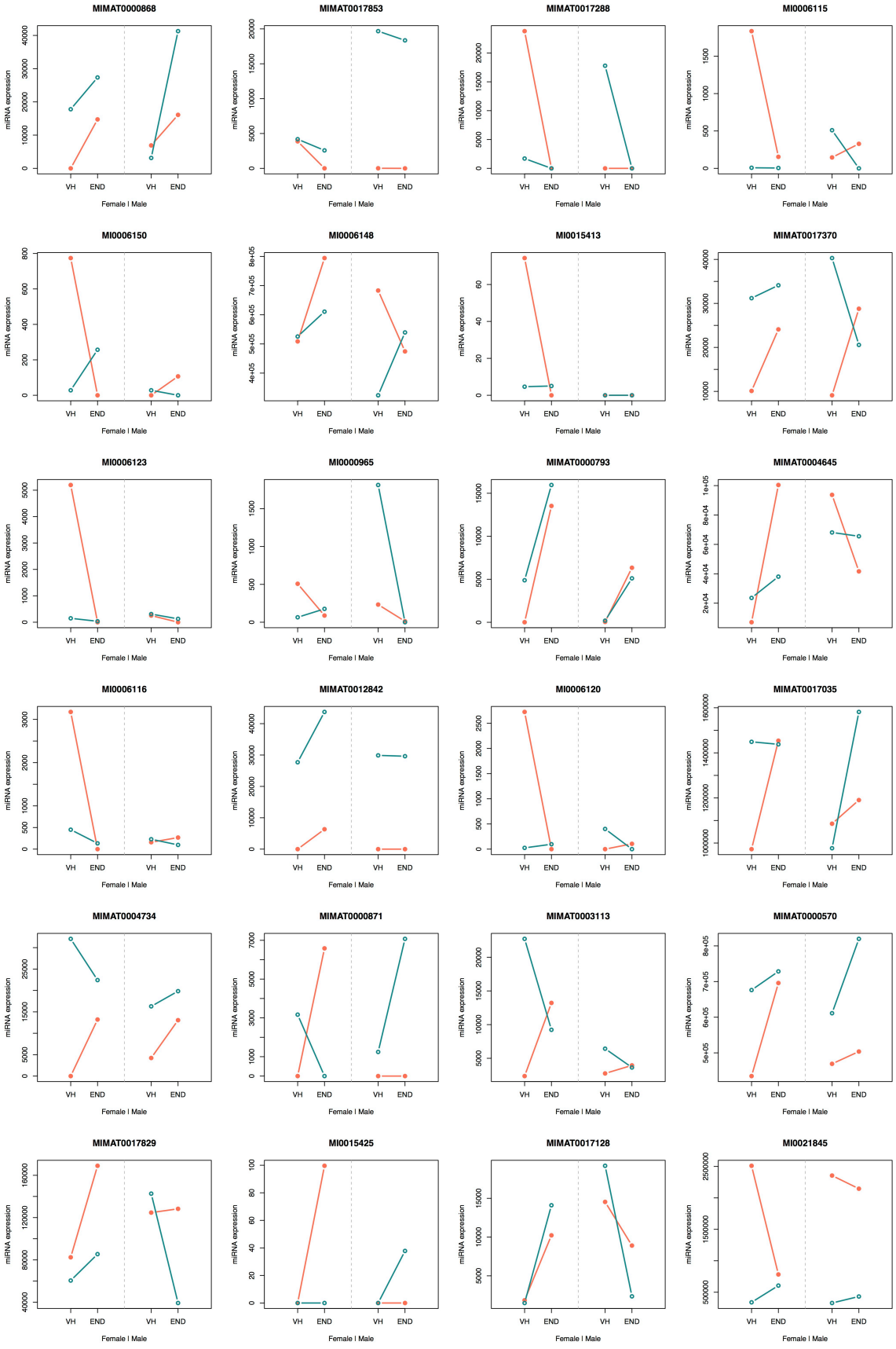

Q920Q0
Protein amount
0.2
0.0
−0.2
−0.4
−0.6
−0.8
VH
END
VH
END
Female | Male


P04904
Protein amount
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
VH
END
VH
END
Female | Male

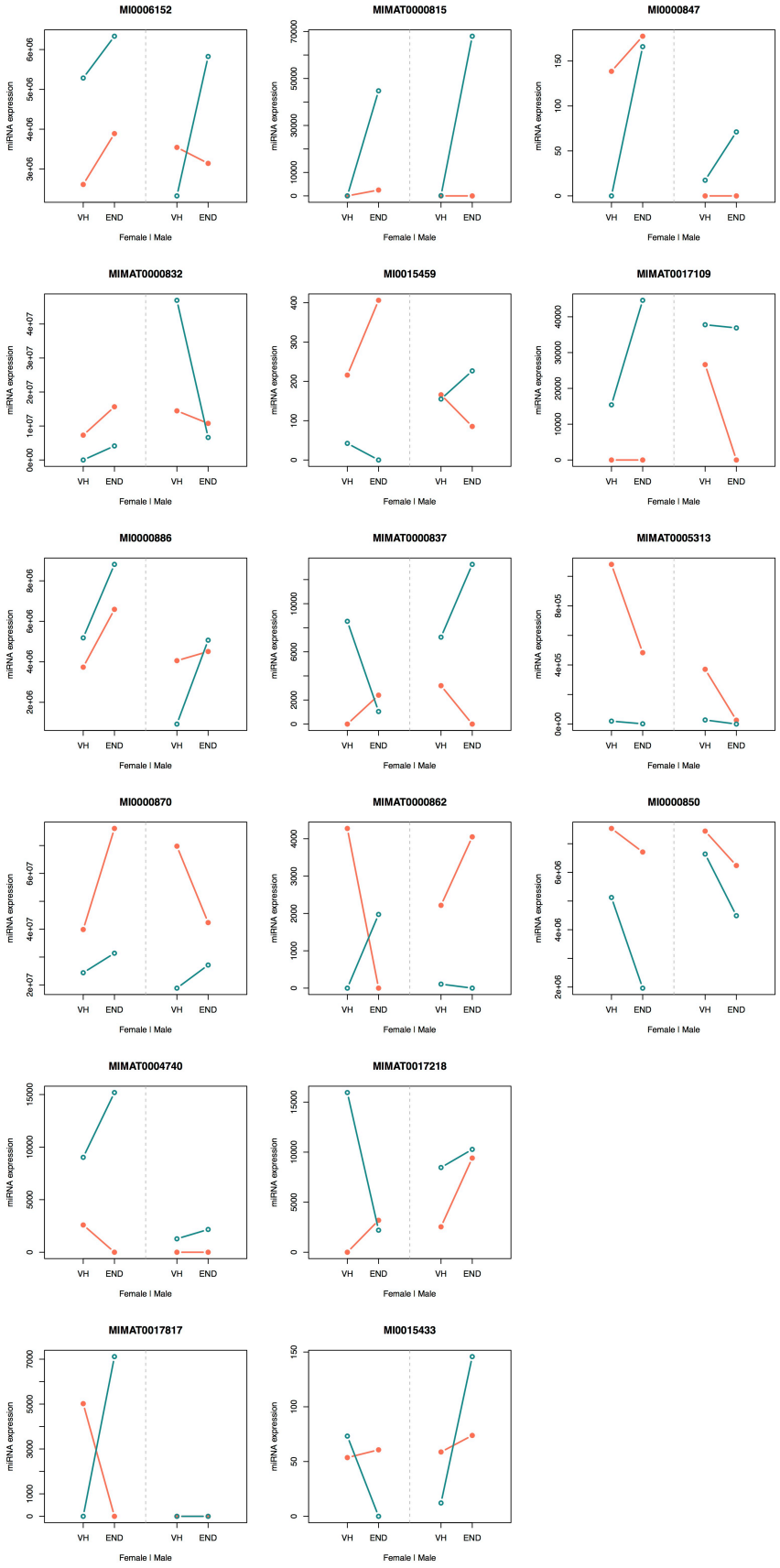

Q62876
Protein amount
0.6
0.4
0.2
0.0
−0.2
−0.6
VH
END
VH
END
Female | Male


D4A817
Protein amount
0.2
0.0
−0.2
−0.4
−0.6
−0.8
VH
END
VH
END
Female | Male

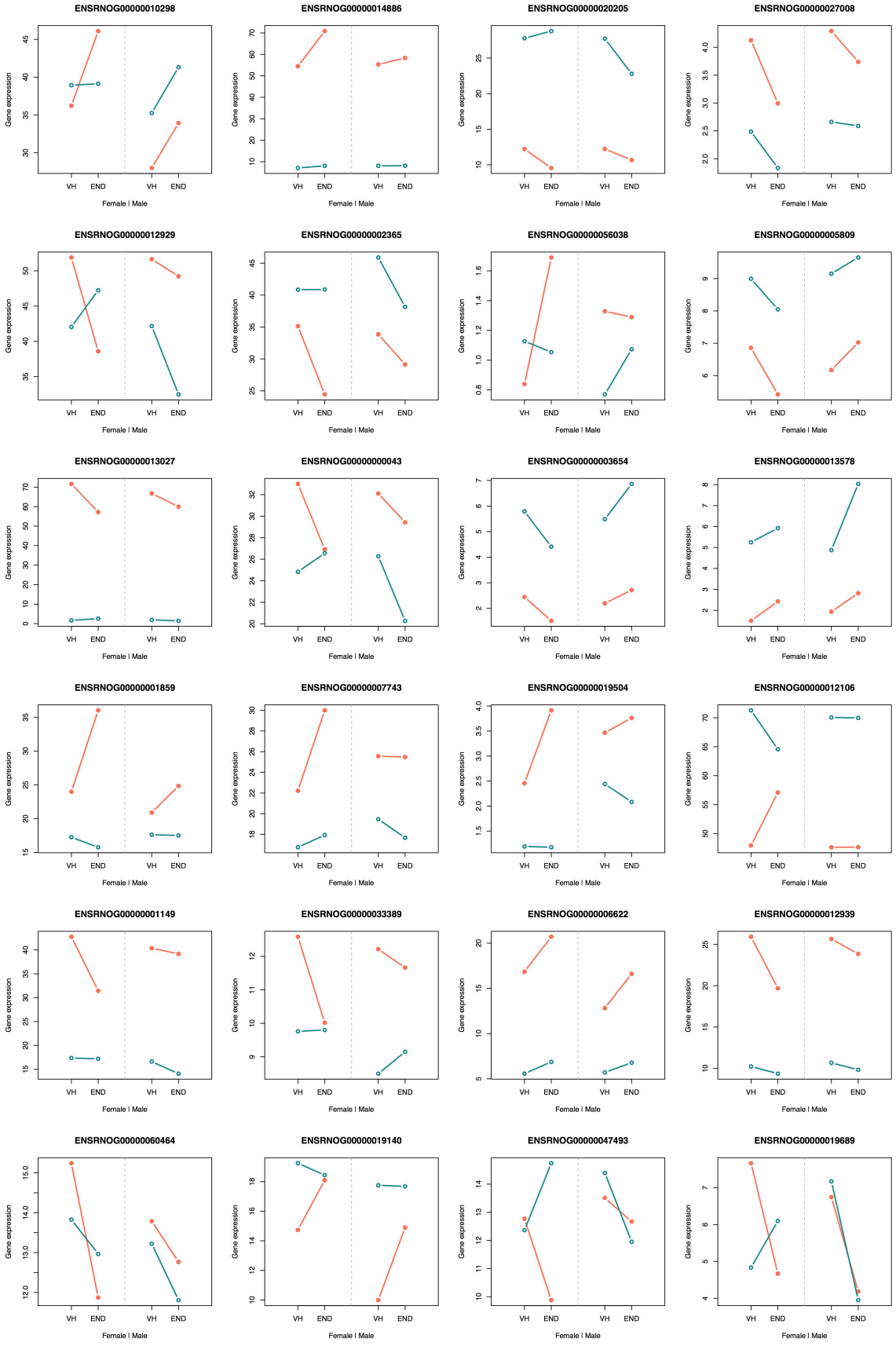
miRNA-seq


MIMAT0000868
MIMAT0017853
MIMAT0017288
MI0006115
MI0006150
MI0006148
MI0015413
MIMAT0017370
MI0006123
MI0000965
MIMAT0000793
MIMAT0004645
MI0006116
MIMAT0012842
MI0006120
MIMAT0017035
MIMAT0004734
MIMAT0000871
MIMAT0003113
MIMAT0000570
MIMAT0017829
MI0015425
MIMAT0017128
MI0021845
miRNA expression
VH END VH END
Female | Male

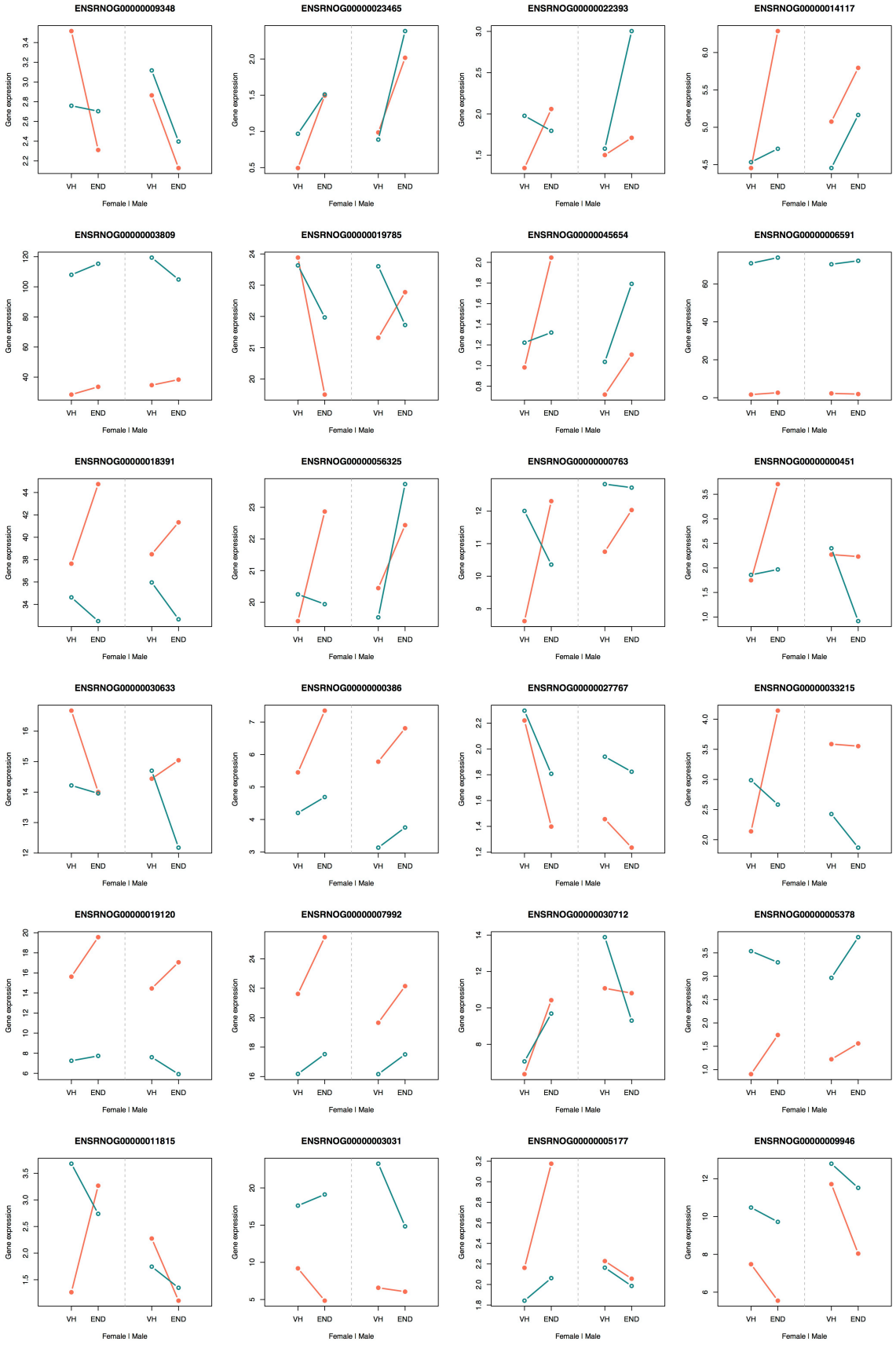
miRNA-seq


MI0006152
MIMAT0000815
MI0000847
MIMAT0000832
MI0015459
MIMAT0017109
MI0000886
MIMAT0000837
MIMAT0005313
MI0000870
MIMAT0000862
MI0000850
MIMAT0004740
MIMAT0017218
MIMAT0017817
MI0015433
miRNA expression
VH
END
Female | Male

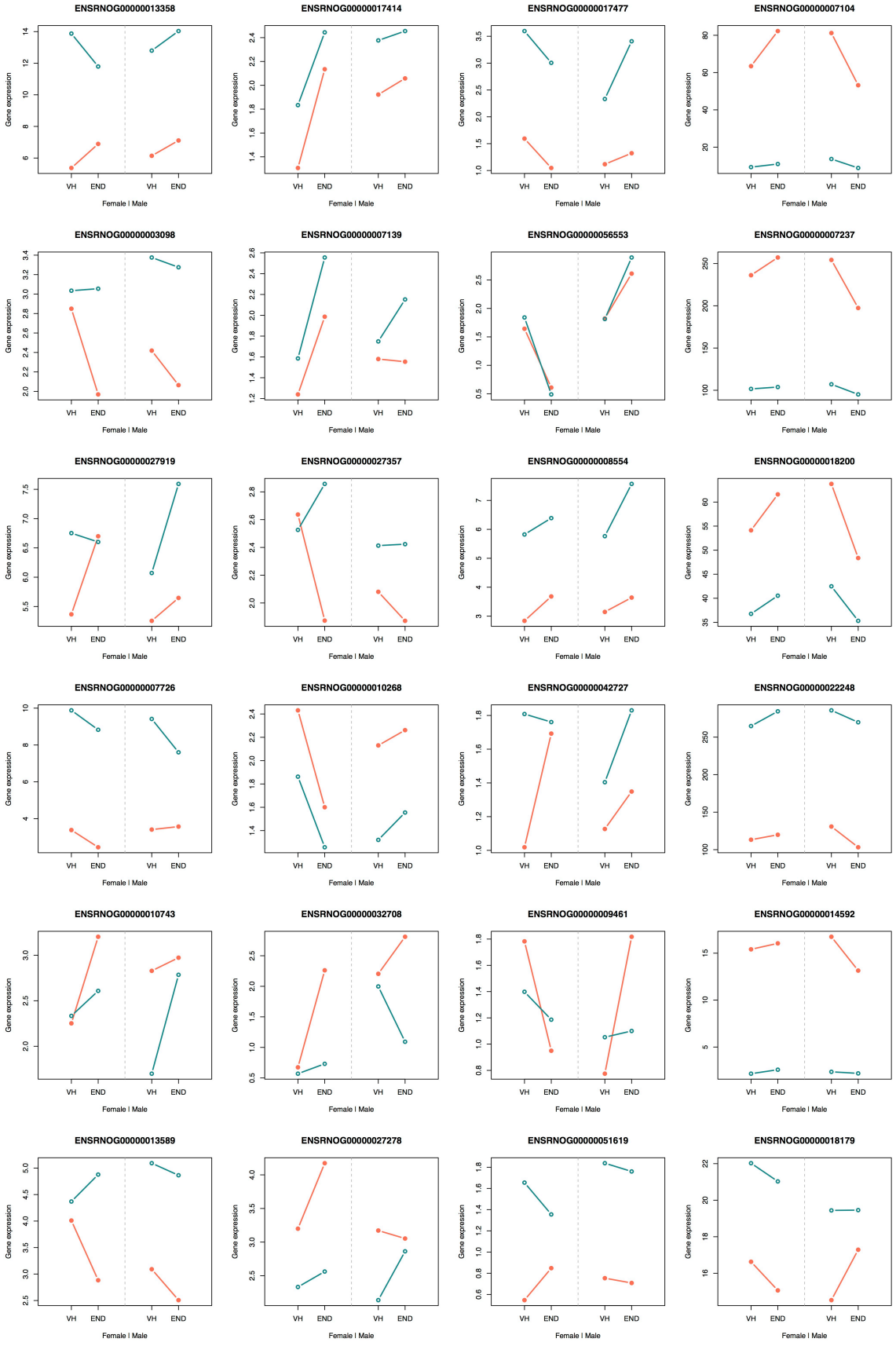
RNA-seq


ENSRNOG00000010298
ENSRNOG00000014886
ENSRNOG00000020205
ENSRNOG00000027008
ENSRNOG00000012929
ENSRNOG00000002365
ENSRNOG00000056038
ENSRNOG00000005809
ENSRNOG00000013027
ENSRNOG00000000043
ENSRNOG00000003654
ENSRNOG00000013578
ENSRNOG00000001859
ENSRNOG00000007743
ENSRNOG00000019504
ENSRNOG00000012106
ENSRNOG00000001149
ENSRNOG00000033389
ENSRNOG00000006622
ENSRNOG00000012939
ENSRNOG00000060464
ENSRNOG00000019140
ENSRNOG00000047493
ENSRNOG00000019689
Gene expression
VH
END
Female | Male

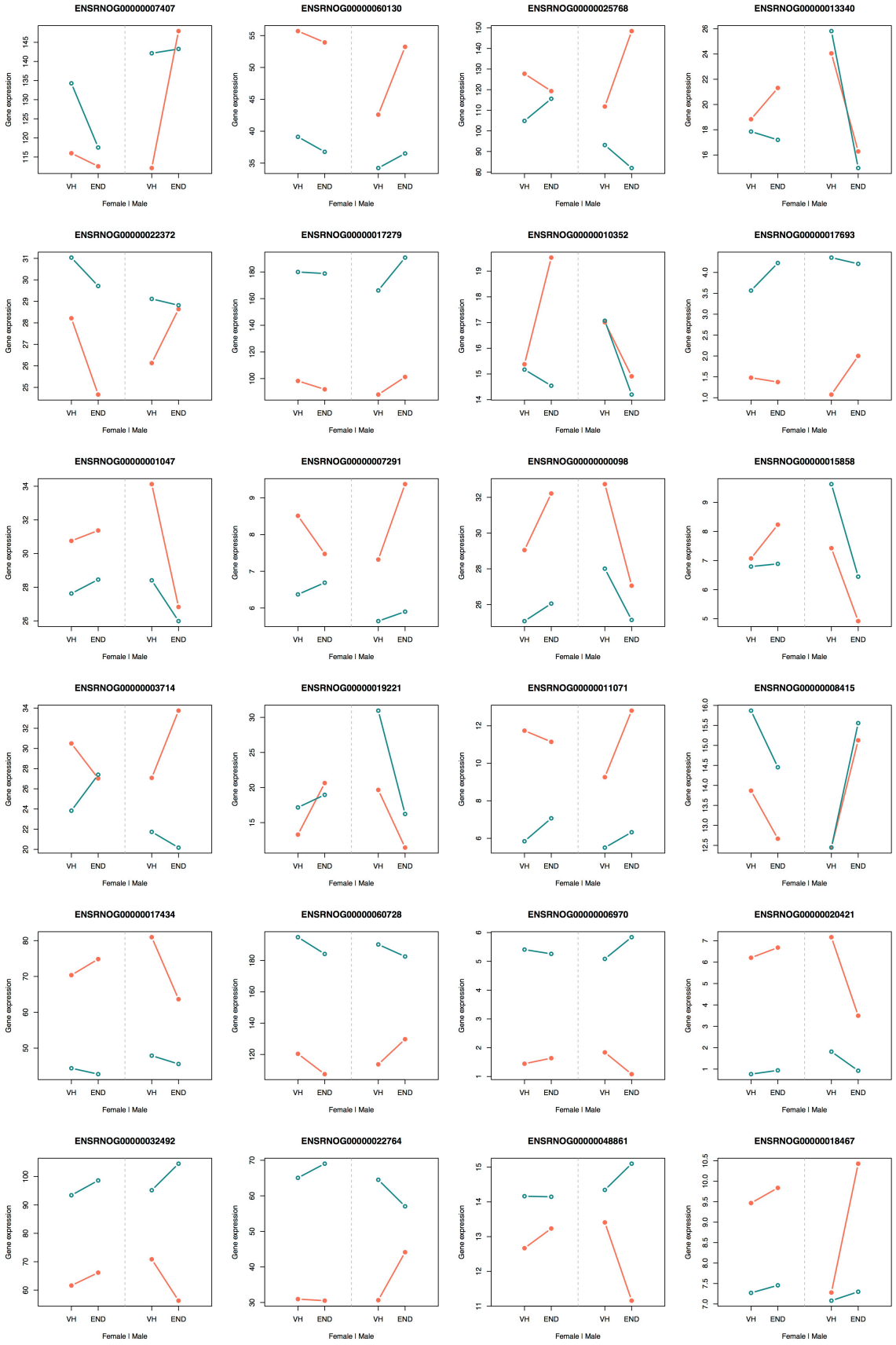
RNA-seq


ENSRNOG00000009348
ENSRNOG00000023465
ENSRNOG00000022393
ENSRNOG00000014117
ENSRNOG00000003809
ENSRNOG00000019785
ENSRNOG00000045654
ENSRNOG00000006591
ENSRNOG00000018391
ENSRNOG00000056325
ENSRNOG00000000763
ENSRNOG00000000451
ENSRNOG00000030633
ENSRNOG00000000386
ENSRNOG00000027767
ENSRNOG00000033215
ENSRNOG00000019120
ENSRNOG00000007992
ENSRNOG00000030712
ENSRNOG00000005378
ENSRNOG00000011815
ENSRNOG00000003031
ENSRNOG00000005177
ENSRNOG00000009946
Gene expression
VH
END
Female | Male

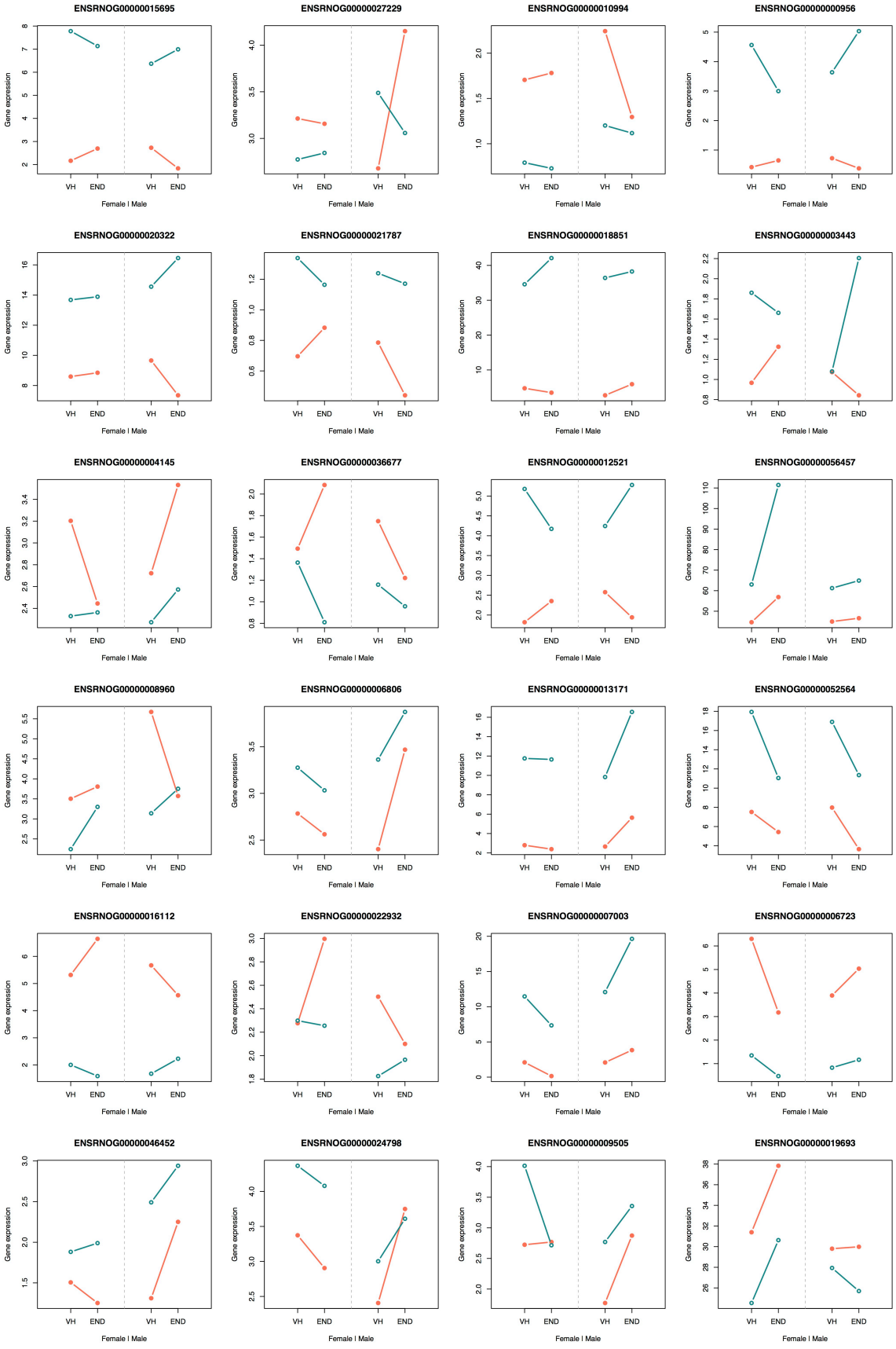
RNA-seq


ENSRNOG00000013358
ENSRNOG00000017414
ENSRNOG00000017477
ENSRNOG00000007104
ENSRNOG00000003098
ENSRNOG00000007139
ENSRNOG00000056553
ENSRNOG00000007237
ENSRNOG00000027919
ENSRNOG00000027357
ENSRNOG00000008554
ENSRNOG00000018200
ENSRNOG00000007726
ENSRNOG00000010268
ENSRNOG00000042727
ENSRNOG00000022248
ENSRNOG00000010743
ENSRNOG00000032708
ENSRNOG00000009461
ENSRNOG00000014592
ENSRNOG00000013589
ENSRNOG00000027278
ENSRNOG00000051619
ENSRNOG00000018179
Gene expression
VH
END
Female | Male

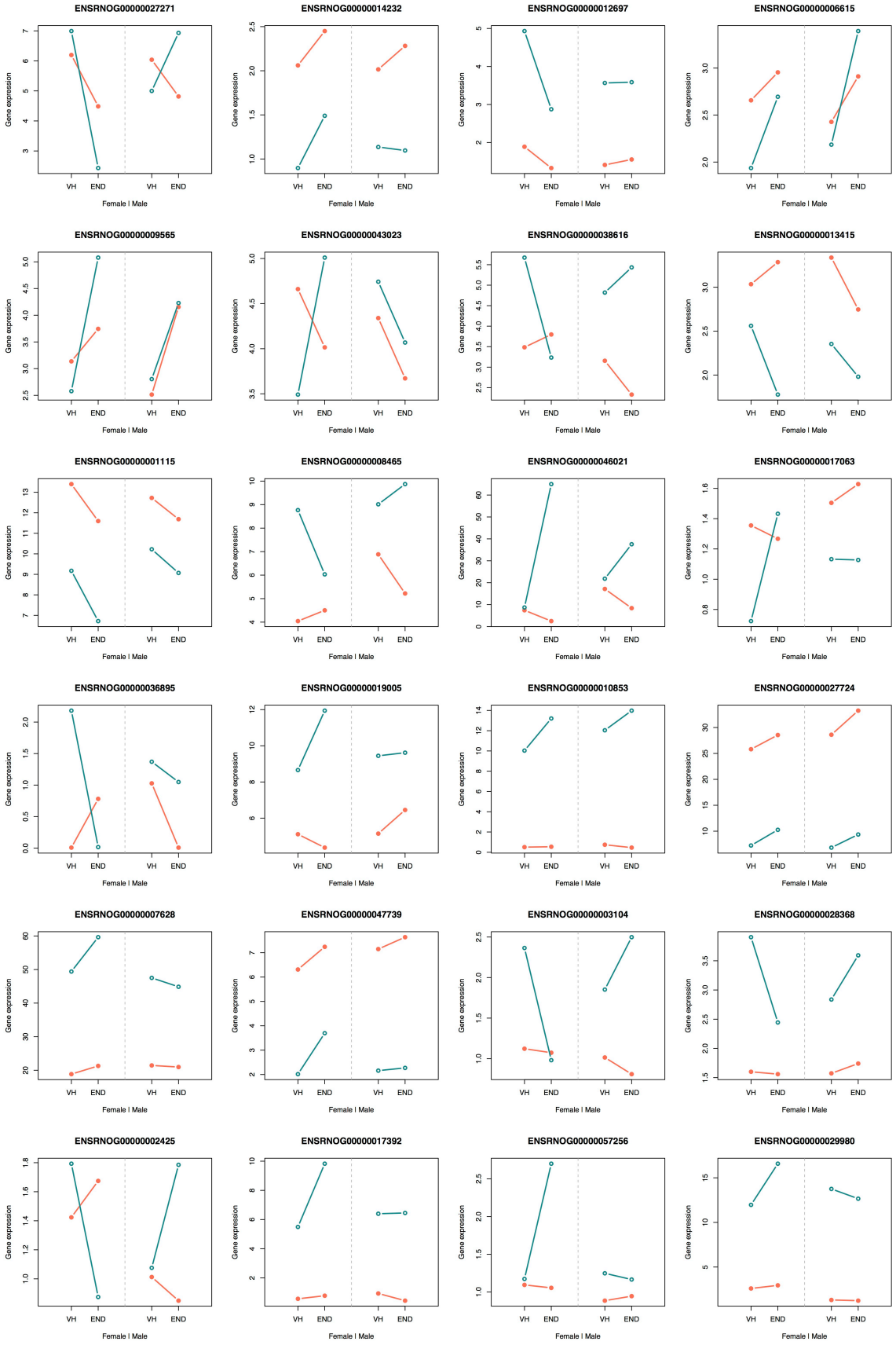
RNA-seq


ENSRNOG00000007407
ENSRNOG00000060130
ENSRNOG00000025768
ENSRNOG00000013340
ENSRNOG00000022372
ENSRNOG00000017279
ENSRNOG00000010352
ENSRNOG00000017693
ENSRNOG00000001047
ENSRNOG00000007291
ENSRNOG00000000098
ENSRNOG00000015858
ENSRNOG00000003714
ENSRNOG00000019221
ENSRNOG00000011071
ENSRNOG00000008415
ENSRNOG00000017434
ENSRNOG00000060728
ENSRNOG00000006970
ENSRNOG00000020421
ENSRNOG00000032492
ENSRNOG00000022764
ENSRNOG00000048861
ENSRNOG00000018467
Gene expression
VH
END
Female | Male

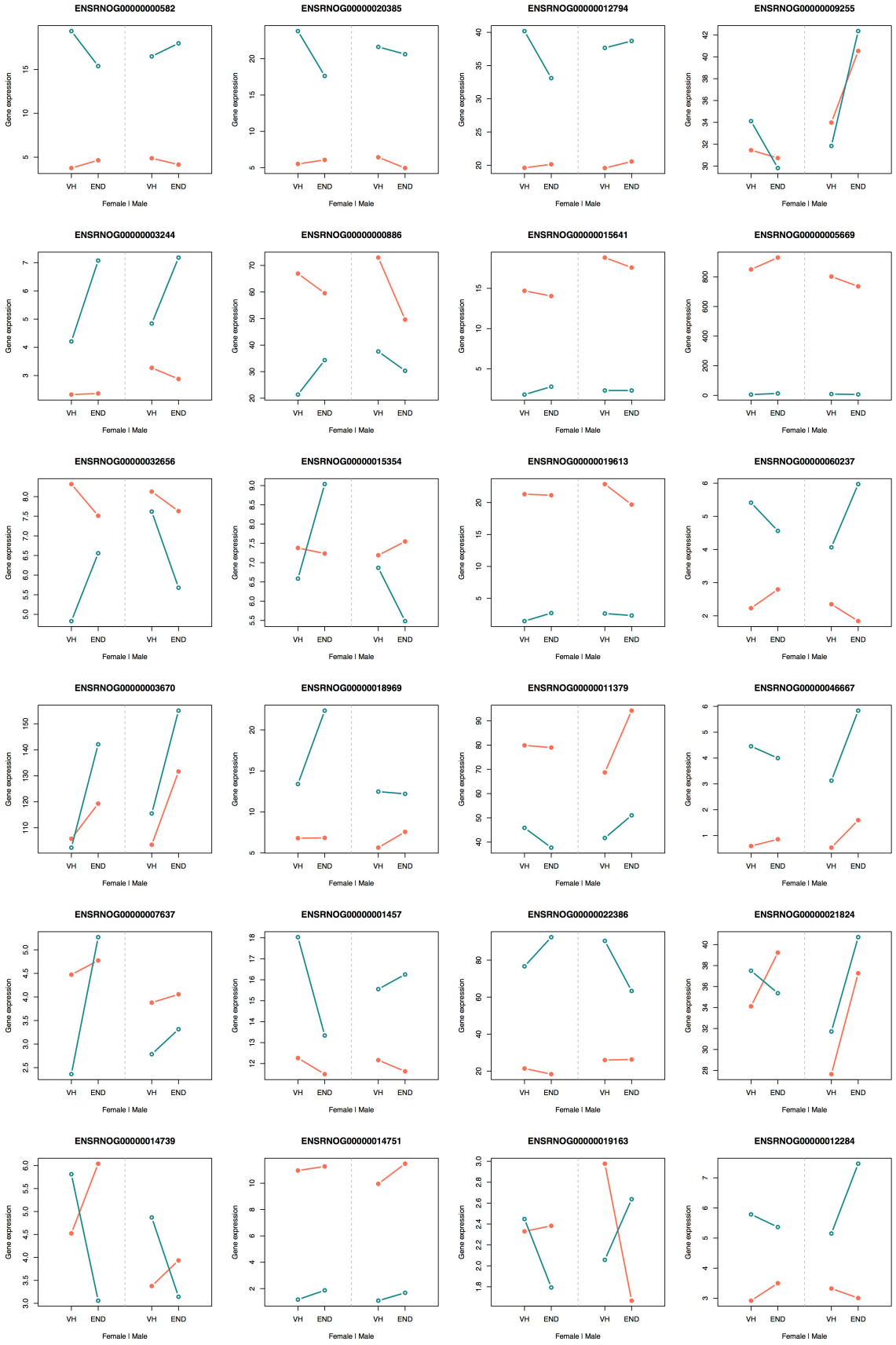
RNA-seq


ENSRNOG00000015695
ENSRNOG00000027229
ENSRNOG00000010994
ENSRNOG00000000956
ENSRNOG00000020322
ENSRNOG00000021787
ENSRNOG00000018851
ENSRNOG00000003443
ENSRNOG00000004145
ENSRNOG00000036677
ENSRNOG00000012521
ENSRNOG00000056457
ENSRNOG00000008960
ENSRNOG00000006806
ENSRNOG00000013171
ENSRNOG00000052564
ENSRNOG00000016112
ENSRNOG00000022932
ENSRNOG00000007003
ENSRNOG00000006723
ENSRNOG00000046452
ENSRNOG00000024798
ENSRNOG00000009505
ENSRNOG00000019693
Gene expression
VH END VH END
Female | Male

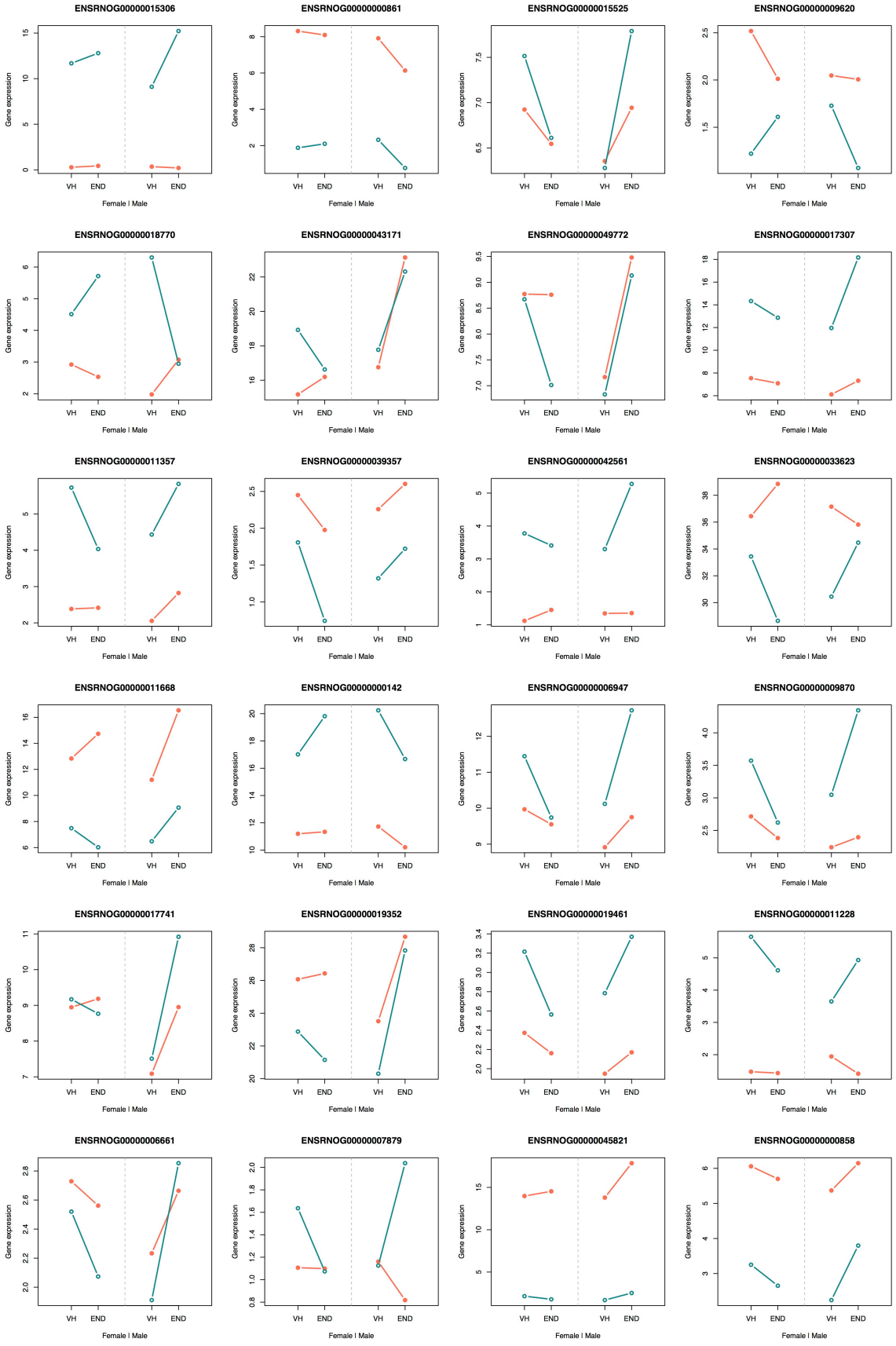
RNA-seq


ENSRNOG00000027271
ENSRNOG00000014232
ENSRNOG00000012697
ENSRNOG00000006615
ENSRNOG00000009565
ENSRNOG00000043023
ENSRNOG00000038616
ENSRNOG00000013415
ENSRNOG00000001115
ENSRNOG00000008465
ENSRNOG00000046021
ENSRNOG00000017063
ENSRNOG00000036895
ENSRNOG00000019005
ENSRNOG00000010853
ENSRNOG00000027724
ENSRNOG00000007628
ENSRNOG00000047739
ENSRNOG00000003104
ENSRNOG00000028368
ENSRNOG00000002425
ENSRNOG00000017392
ENSRNOG00000057256
ENSRNOG00000029980
Gene expression
VH
END
Female | Male

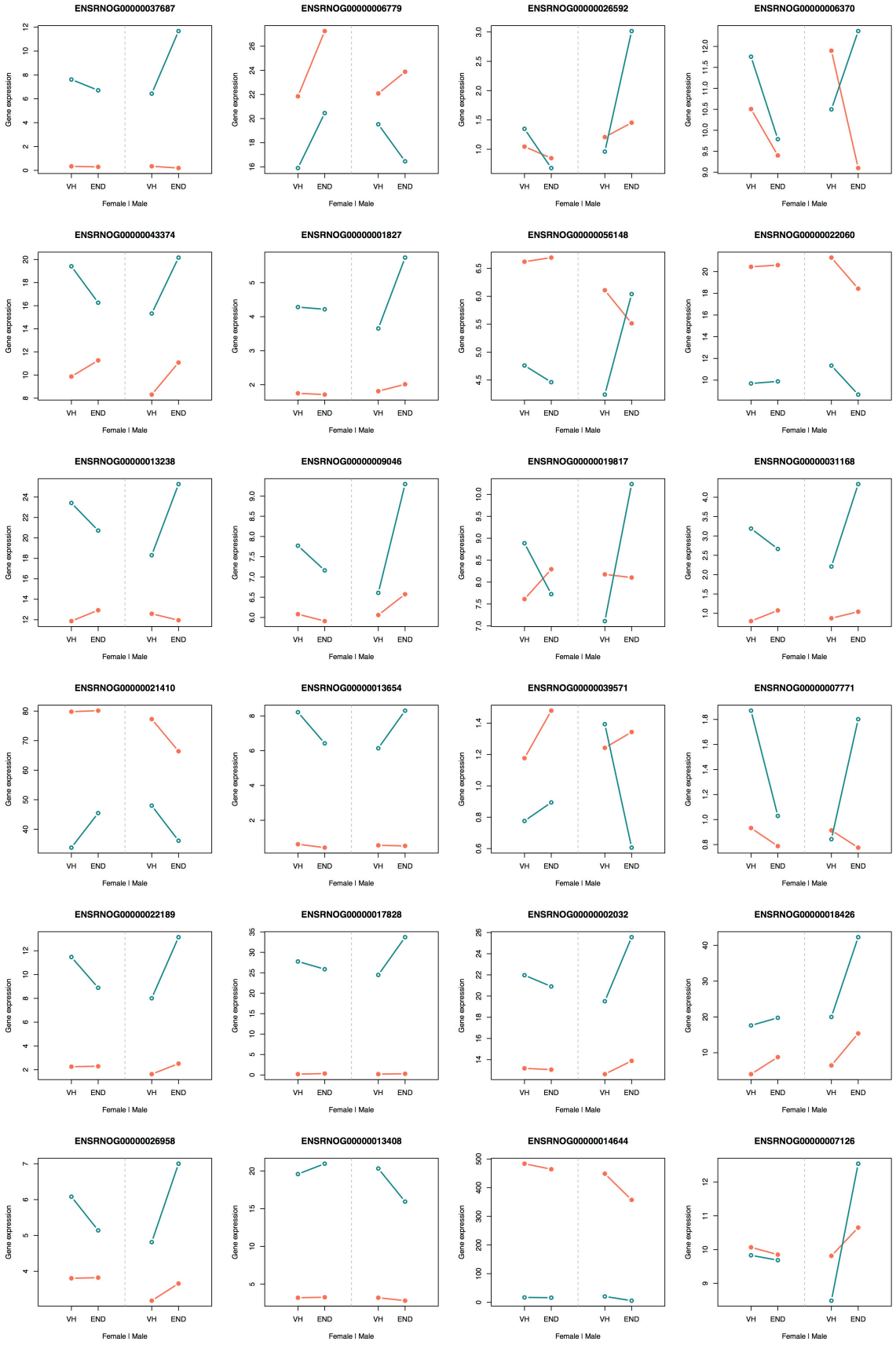
RNA-seq


ENSRNOG00000000582
ENSRNOG00000020385
ENSRNOG00000012794
ENSRNOG00000009255
ENSRNOG00000003244
ENSRNOG00000000886
ENSRNOG00000015641
ENSRNOG00000005669
ENSRNOG00000032656
ENSRNOG00000015354
ENSRNOG00000019613
ENSRNOG00000060237
ENSRNOG00000003670
ENSRNOG00000018969
ENSRNOG00000011379
ENSRNOG00000046667
ENSRNOG00000007637
ENSRNOG00000001457
ENSRNOG00000022386
ENSRNOG00000021824
ENSRNOG00000014739
ENSRNOG00000014751
ENSRNOG00000019163
ENSRNOG00000012284
Gene expression
VH
END
Female | Male

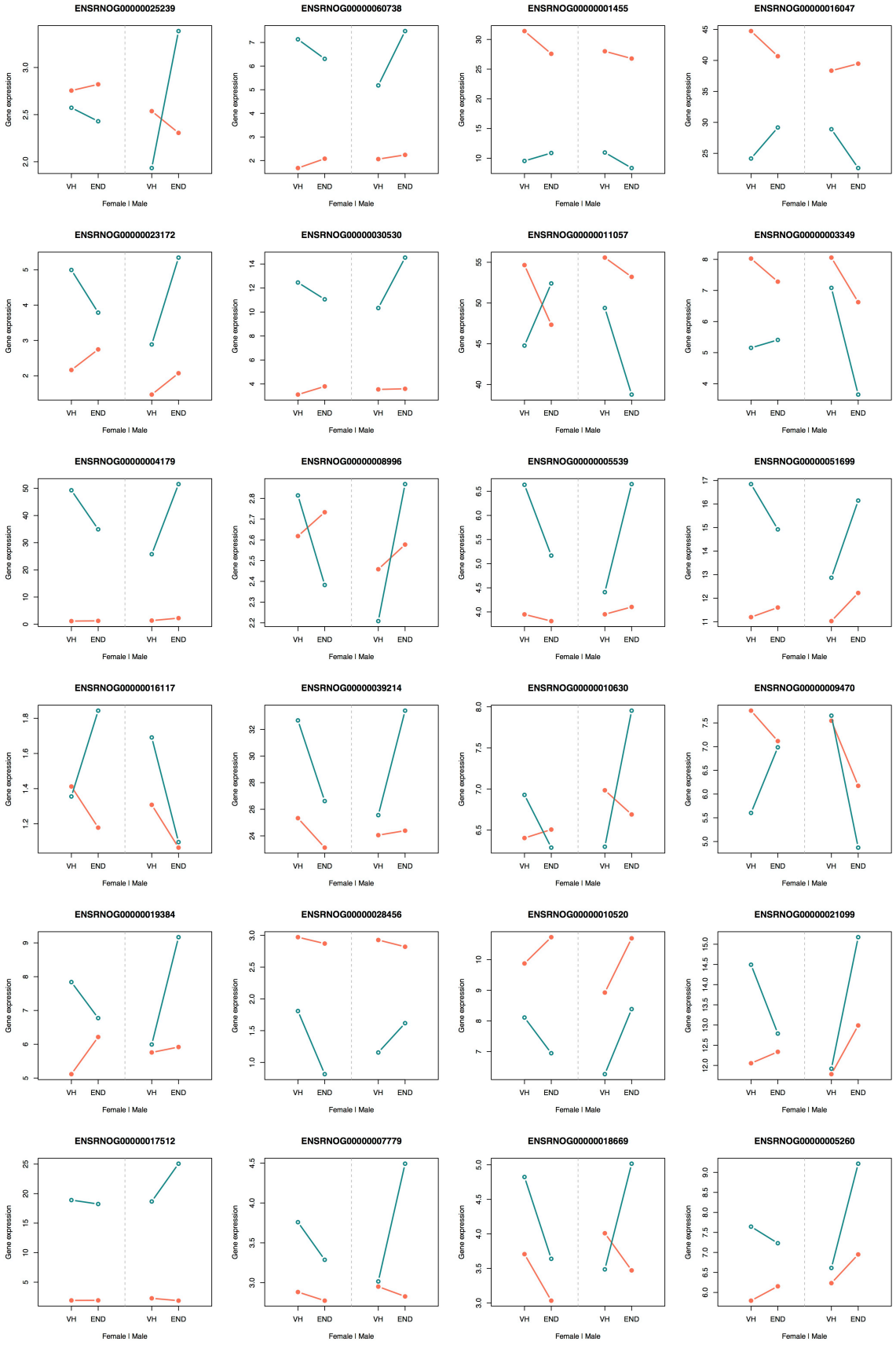
RNA-seq


ENSRNOG00000015306
ENSRNOG00000000861
ENSRNOG00000015525
ENSRNOG00000009620
ENSRNOG00000018770
ENSRNOG00000043171
ENSRNOG00000049772
ENSRNOG00000017307
ENSRNOG00000011357
ENSRNOG00000039357
ENSRNOG00000042561
ENSRNOG00000033623
ENSRNOG00000011668
ENSRNOG00000000142
ENSRNOG00000006947
ENSRNOG00000009870
ENSRNOG00000017741
ENSRNOG00000019352
ENSRNOG00000019461
ENSRNOG00000011228
ENSRNOG00000006661
ENSRNOG00000007879
ENSRNOG00000045821
ENSRNOG00000000858
Gene expression
VH
END
Female | Male

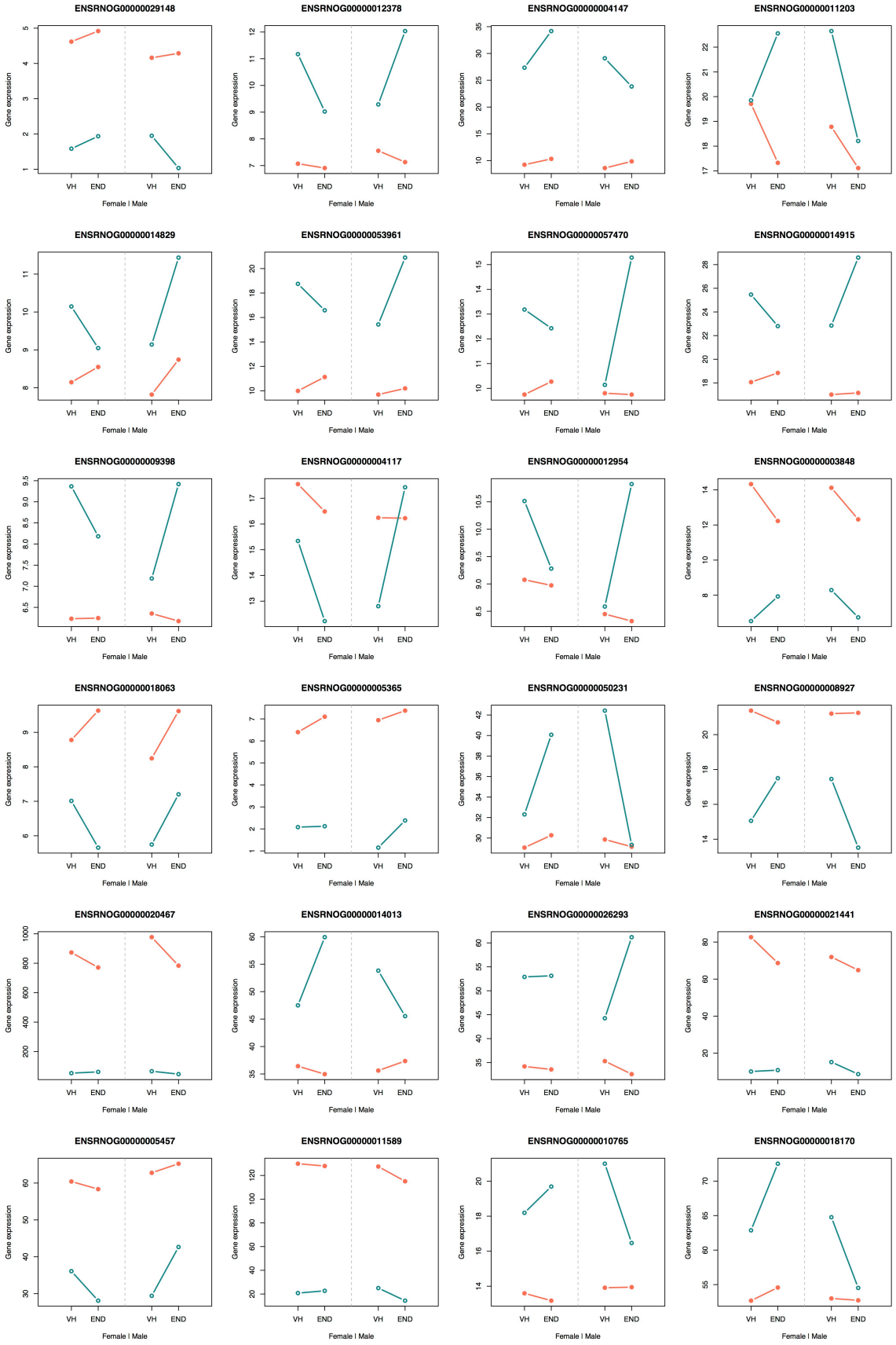
RNA-seq


ENSRNOG00000037687
ENSRNOG00000006779
ENSRNOG00000026592
ENSRNOG00000006370
ENSRNOG00000043374
ENSRNOG00000001827
ENSRNOG00000056148
ENSRNOG00000022060
ENSRNOG00000013238
ENSRNOG00000009046
ENSRNOG00000019817
ENSRNOG00000031168
ENSRNOG00000021410
ENSRNOG00000013654
ENSRNOG00000039571
ENSRNOG00000007771
ENSRNOG00000022189
ENSRNOG00000017828
ENSRNOG00000002032
ENSRNOG00000018426
ENSRNOG00000026958
ENSRNOG00000013408
ENSRNOG00000014644
ENSRNOG00000007126
Gene expression
VH
END
Female | Male

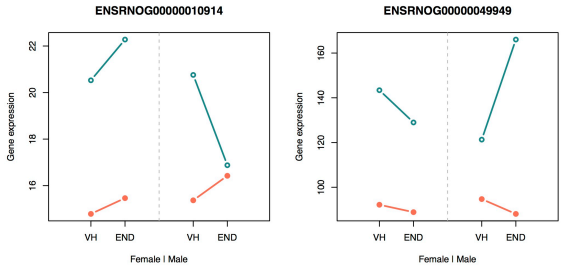
RNA-seq


ENSRNOG00000025239
ENSRNOG00000060738
ENSRNOG00000001455
ENSRNOG00000016047
ENSRNOG00000023172
ENSRNOG00000030530
ENSRNOG00000011057
ENSRNOG00000003349
ENSRNOG00000004179
ENSRNOG00000008996
ENSRNOG00000005539
ENSRNOG00000051699
ENSRNOG00000016117
ENSRNOG00000039214
ENSRNOG00000010630
ENSRNOG00000009470
ENSRNOG00000019384
ENSRNOG00000028456
ENSRNOG00000010520
ENSRNOG00000021099
ENSRNOG00000017512
ENSRNOG00000007779
ENSRNOG00000018669
ENSRNOG00000005260
Gene expression
VH
END
Female | Male

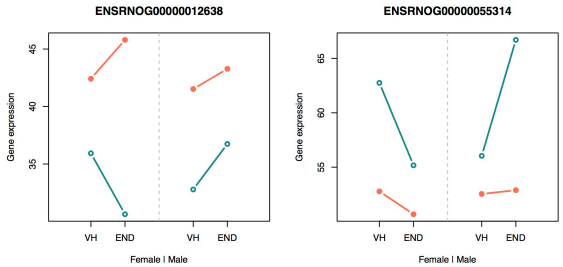
RNA-seq


ENSRNOG00000029148
ENSRNOG00000012378
ENSRNOG00000004147
ENSRNOG00000011203
ENSRNOG00000014829
ENSRNOG00000053961
ENSRNOG00000057470
ENSRNOG00000014915
ENSRNOG00000009398
ENSRNOG00000004117
ENSRNOG00000012954
ENSRNOG00000003848
ENSRNOG00000018063
ENSRNOG00000005365
ENSRNOG00000050231
ENSRNOG00000008927
ENSRNOG00000020467
ENSRNOG00000014013
ENSRNOG00000026293
ENSRNOG00000021441
ENSRNOG00000005457
ENSRNOG00000011589
ENSRNOG00000010765
ENSRNOG00000018170
Gene expression
VH
END
Female | Male

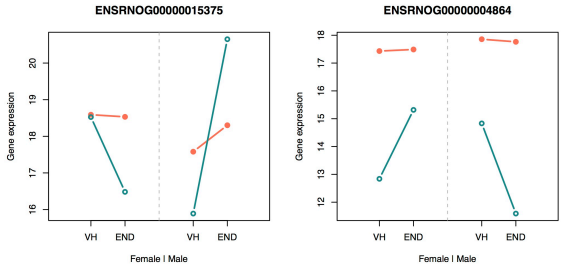
RNA-seq


ENSRNOG00000010914
Gene expression
VH
END
VH
END
Female | Male

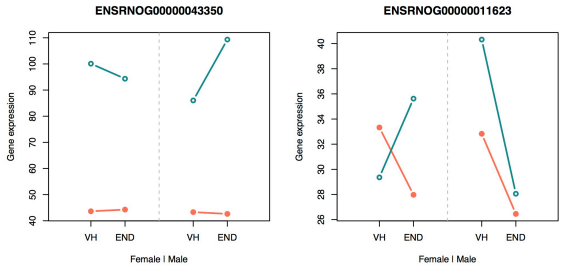
ENSRNOG00000049949
Gene expression
VH
END
VH
END
Female | Male


ENSRNOG00000012638
Gene expression
VH
END
VH
END
Female | Male

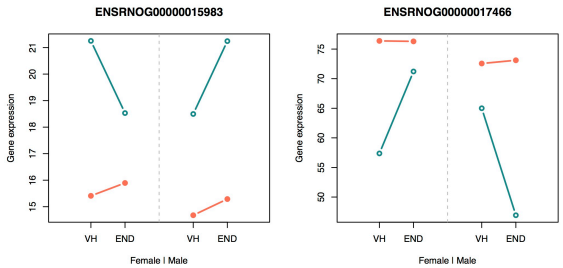
ENSRNOG00000055314
Gene expression
VH
END
VH
END
Female | Male


ENSRNOG00000015375
Gene expression
VH
END
VH
END
Female | Male

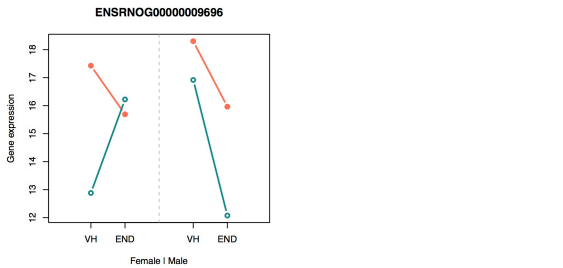
ENSRNOG00000004864
Gene expression
VH
END
VH
END
Female | Male


ENSRNOG00000043350
Gene expression
VH
END
VH
END
Female | Male

ENSRNOG00000011623
Gene expression
VH
END
VH
END
Female | Male


ENSRNOG00000015983
Gene expression
VH
END
VH
END
Female | Male

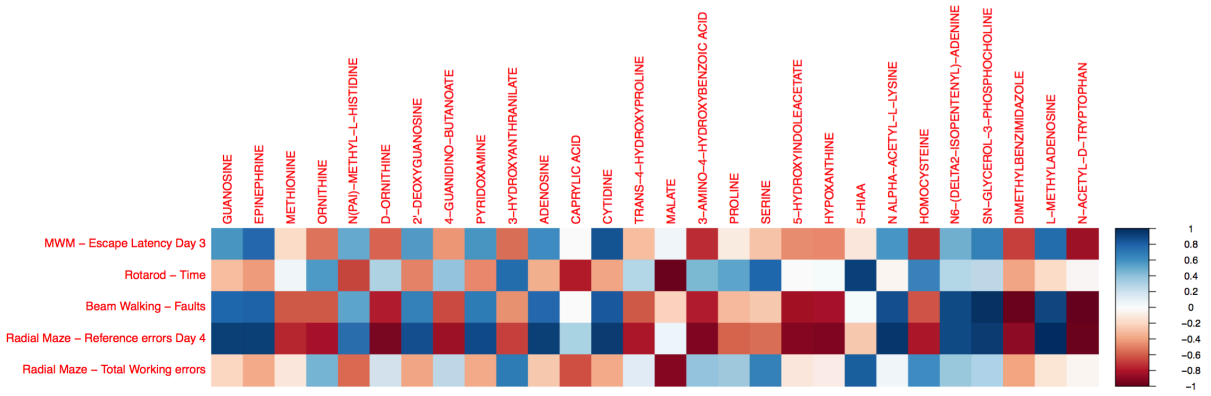
ENSRNOG00000017466
Gene expression
VH
END
VH
END
Female | Male


ENSRNOG00000009696
Gene expression
VH
END
VH
END
Female | Male

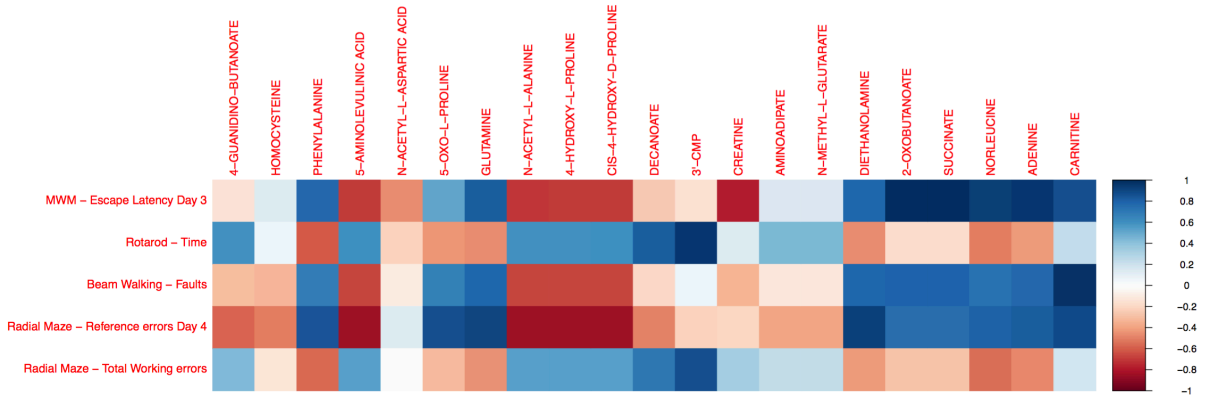
7.2.8. Attachment XIX: Correlation plots

Metabolomics

Cerebellum

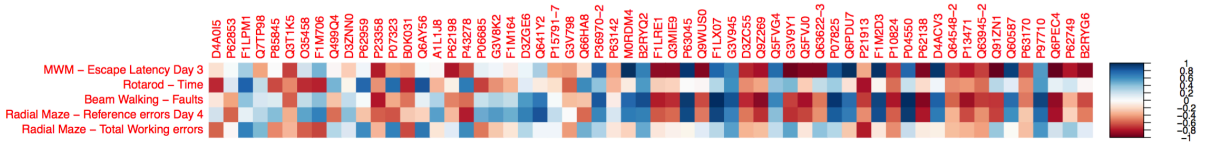


Hippocampus

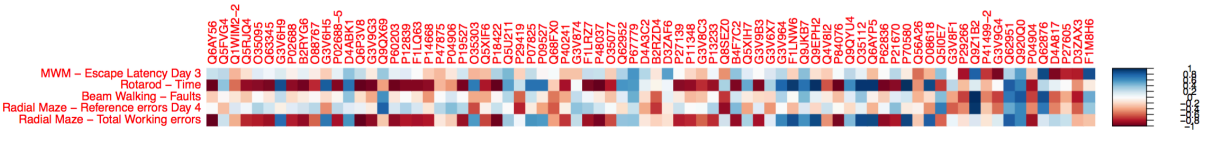


Proteomics

Cerebellum

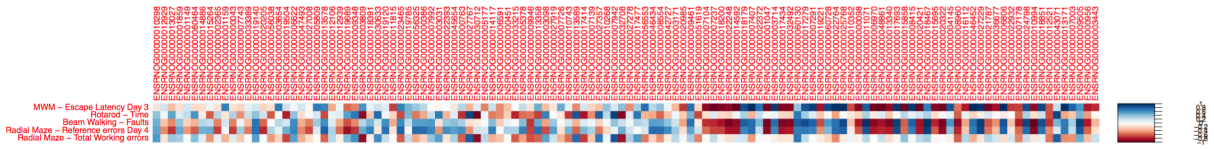


Hippocampus

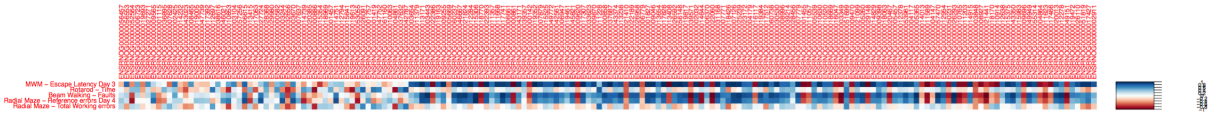


RNA-seq

Cerebellum

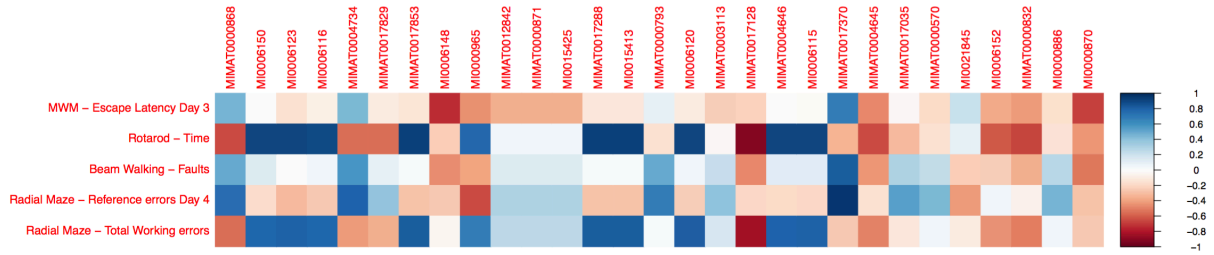


Hippocampus

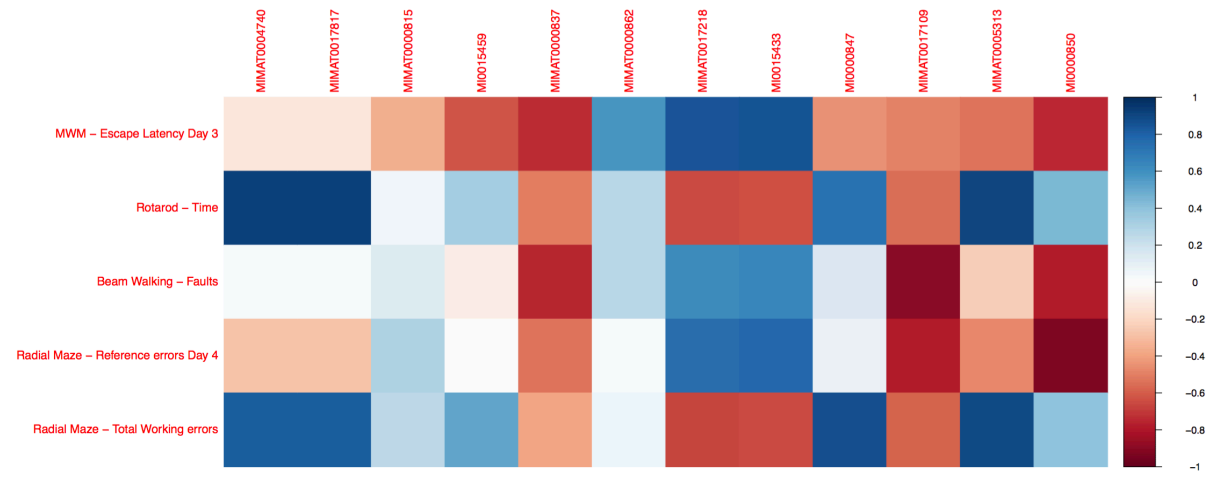


miRNA-seq

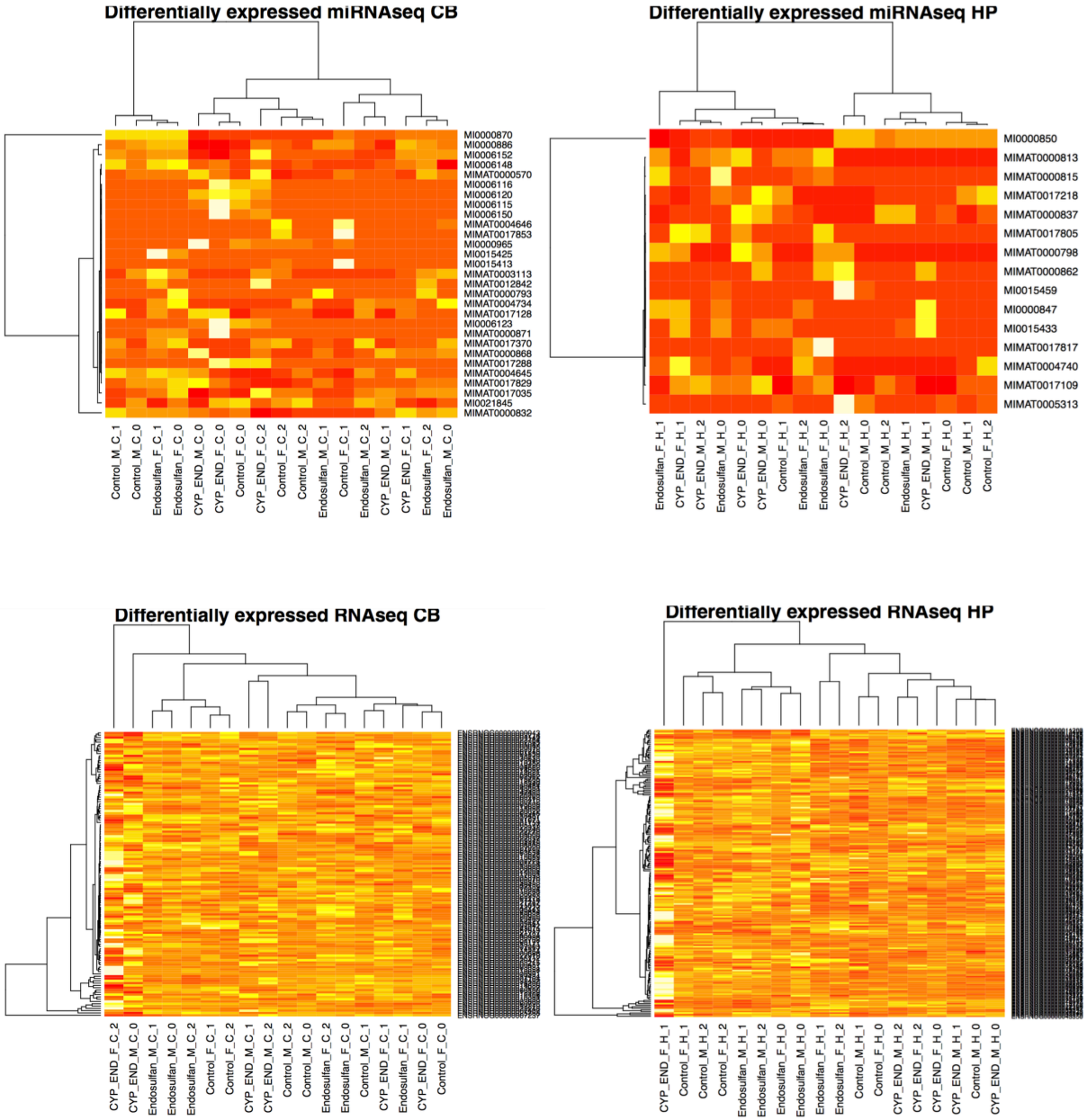
Cerebellum



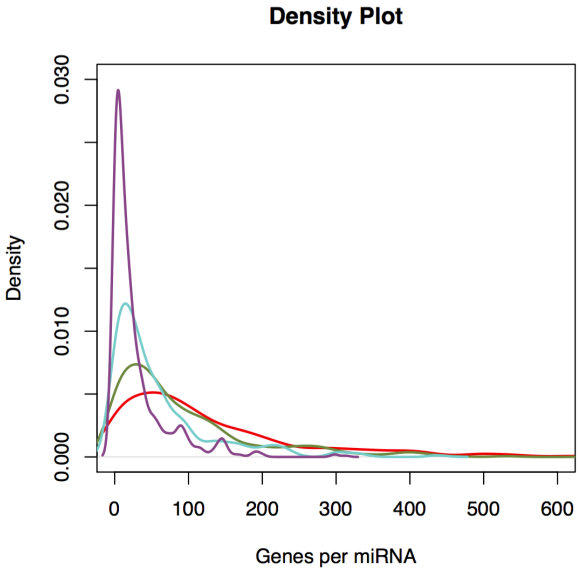
Hippocampus

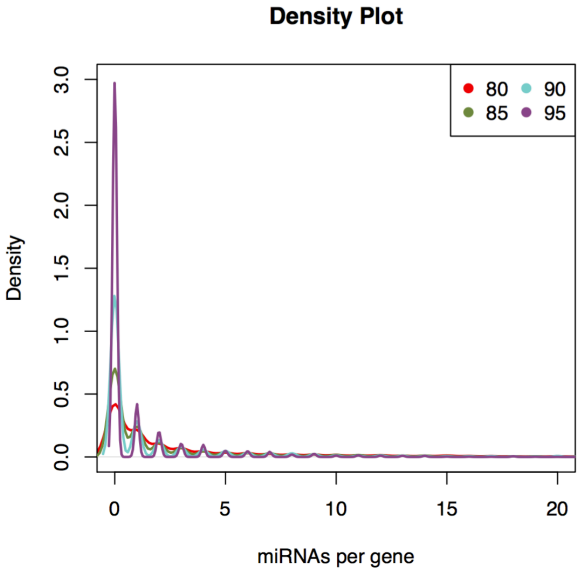


7.2.9. Attachment XX: Heatmaps DE transcriptomics

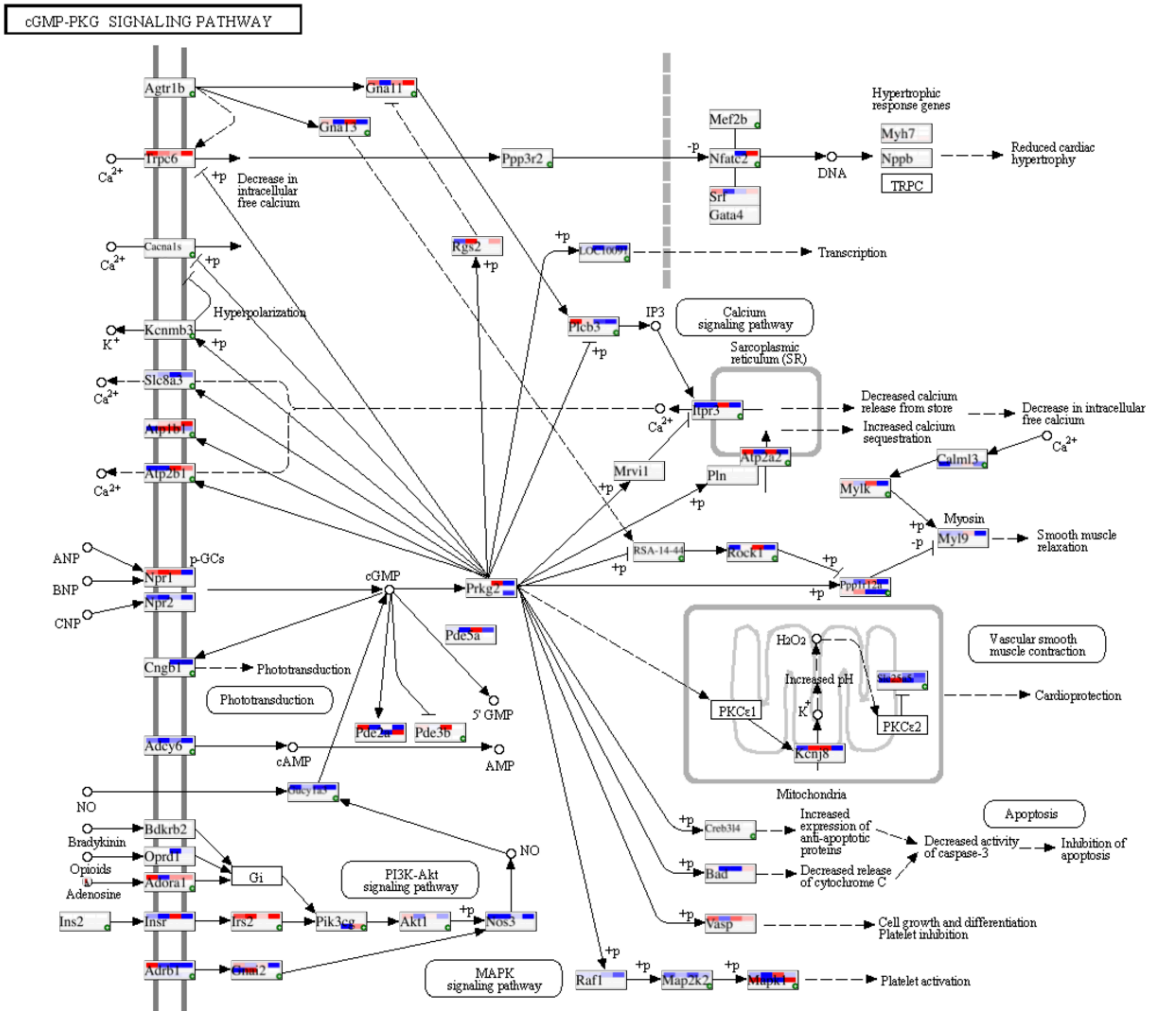


7.2.10. Attachment XXI: miRDB density plots





7.2.11. Attachment XXII: cGMP-PKC signaling pathway



7.2.12. Attachment XXIII: Parkinson's disease pathway