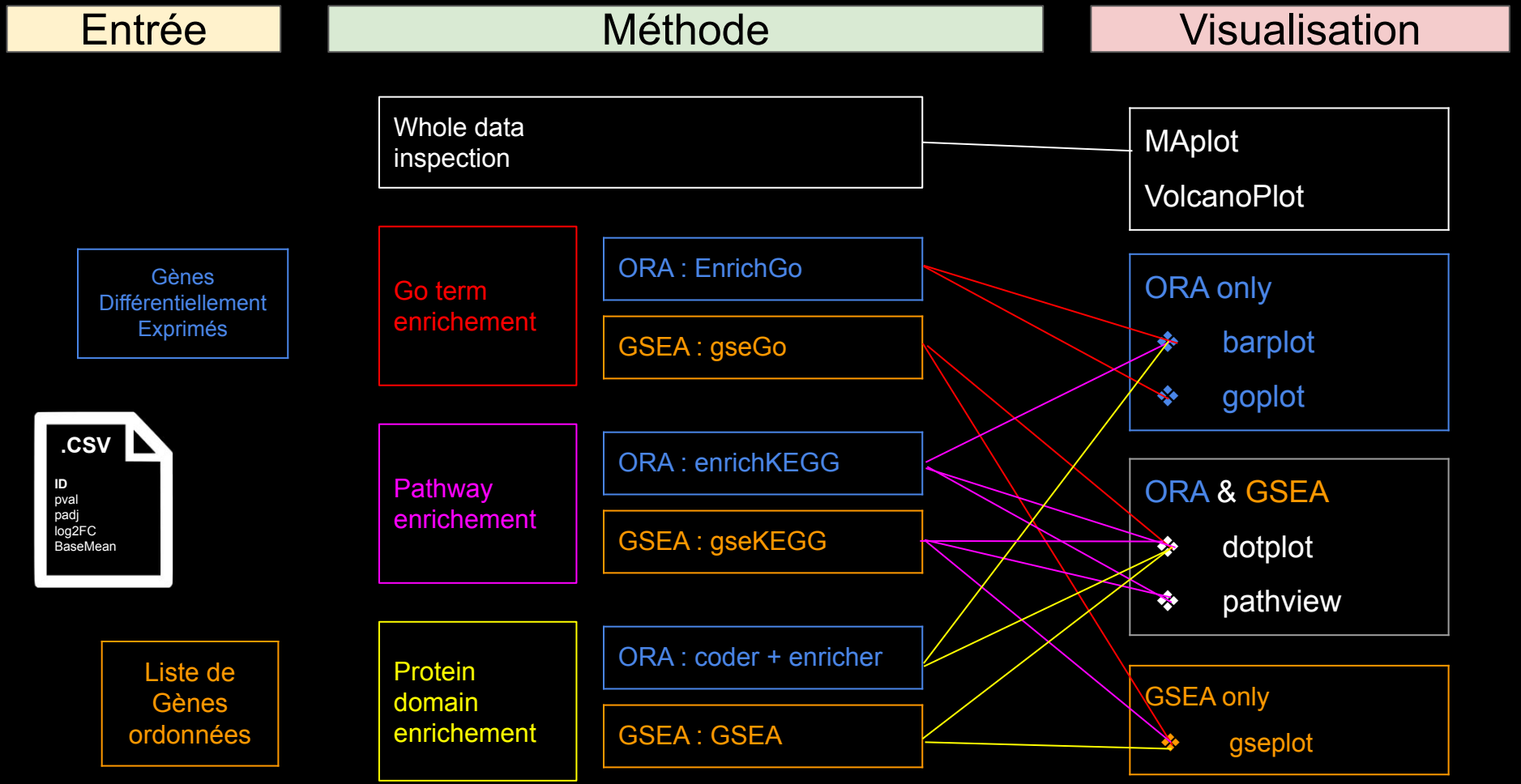
Master 2 BioInformatique Modélisation et Statistique (BIMS), Université de Rouen Normandie

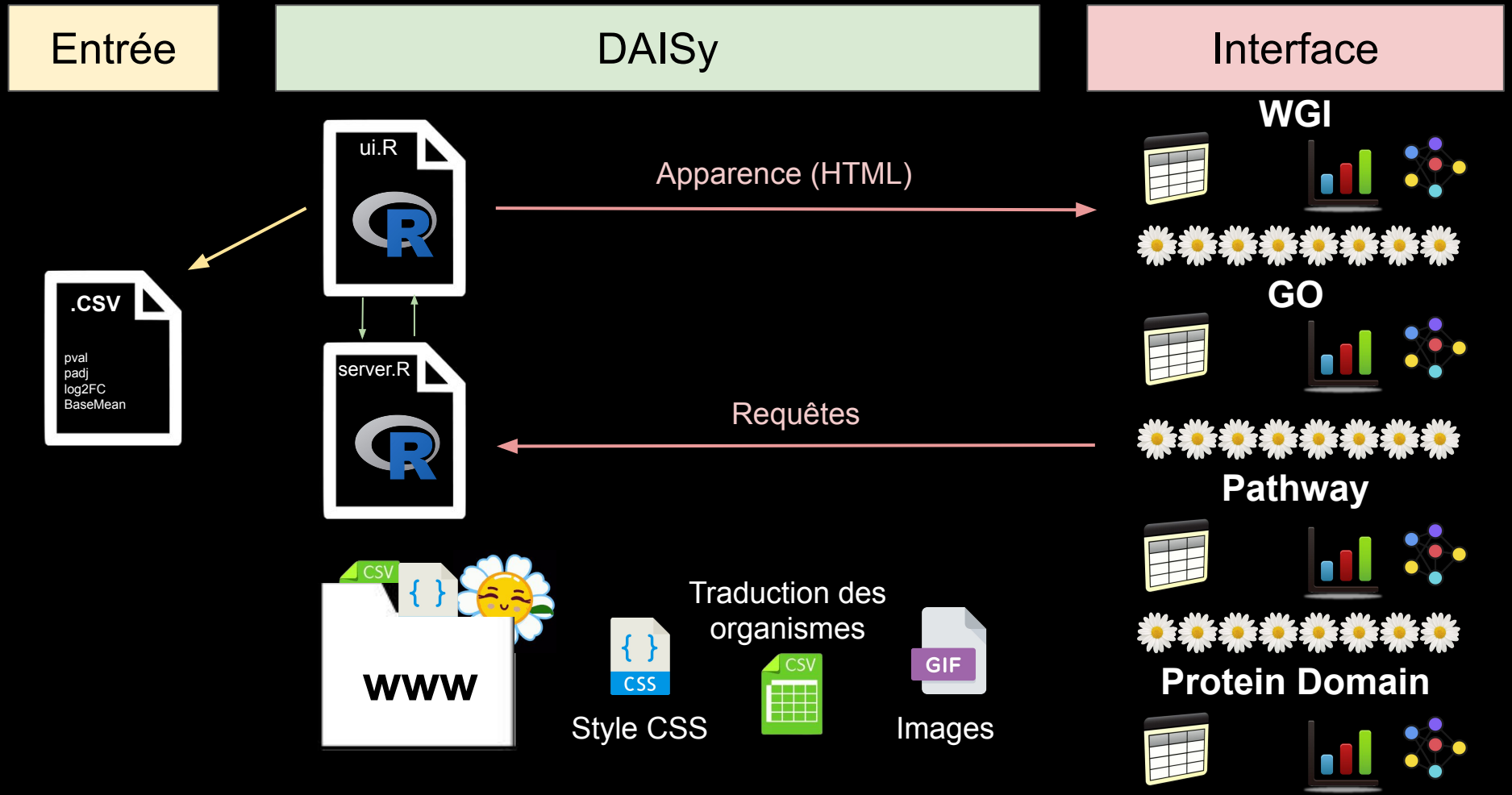
Functional Enrichment Analysis - Development of a Shiny application

RNA DAiSY seq

Differential expression Analysis In ShinY for RNAseq

Alizée Bardon, Fiona Bottin et Sara Bencheikh


Entrée
Méthode
Visualisation
Whole data inspection
MAplot
VolcanoPlot
Gènes Différentiellement Exprimés
Go term enrichement
ORA : EnrichGo
GSEA : gseGo
ORA only
barplot
goplot
.csv
ID
pval
padj
log2FC
BaseMean
Pathway enrichement
ORA : enrichKEGG
GSEA : gseKEGG
ORA & GSEA
dotplot
pathview
Liste de Gènes ordonnées
Protein domain enrichement
ORA : coder + enricher
GSEA : GSEA
GSEA only
gseplot

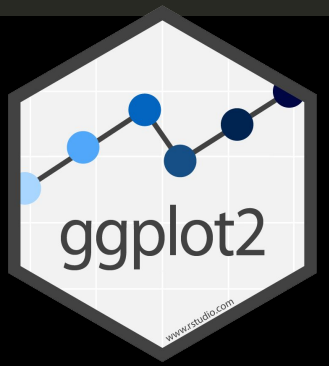

Entrée
DAISy
Interface
ui.R
server.R
.csv
pval
padj
log2FC
BaseMean
Apparence (HTML)
Requêtes
WGI
GO
Pathway
Protein Domain
CSV
{ }
www
CSS
Style CSS
Traduction des organismes
CSV
GIF
Images

Lancement de la démonstration

Whole genome inspection

```
re <- reactive({
        file <- input$file1
        ext <- tools::file_ext(file$datapath)
        req(file)
        validate(need(ext == "csv", "Invalid file. Please upload a .csv file"))
        data <- read.csv(file$datapath, header = TRUE, sep = ";")
        data <- na.omit(data)
        required_columns <- c("ID", "baseMean", "log2FC", "pval", "padj")
        column_names <- colnames(data)
        shiny::validate(need(all(required_columns %in% column_names), "Missing required columns"))
        data
})
```

```
p <- ggplot(
      data = data_plot_plotly,
      mapping = aes(x = log2FC, y=-log10(padj), col=diffexpressed, key = key)) +
      geom_point(size = 0.45) +
      theme_bw()  +
      scale_color_manual(values=c("red", "black", "green")) +
      xlab("log2 fold change") +
      ylab("-log10(p-value)")
```



Gene Ontology enrichement

```
goGse_annot <-  eventReactive(input$Run_Annotation_go, if(input$method_go == 2){
      espece_id <- espece_id()
      orga_translate_table <- orga_translate_table()
      go <- re()
      original_gene_list <- go$log2FC
      names(original_gene_list) <- go$ID
      gene_list<-na.omit(original_gene_list)
      gene_list <- sort(gene_list, decreasing=TRUE)

      gse <- gseGO(geneList=gene_list,
                   ont = input$Ontology,
                   keyType = 'ENSEMBL',
                   pvalueCutoff = input$pvalue_go,
                   minGSSize = 3,
                   maxGSSize = 800,
                   verbose = TRUE,
                   OrgDb = orga_translate_table[espece_id,2],
                   pAdjustMethod = input$Ajustement_go)

})
```

GSEA

```
goGse_enrich <-  eventReactive(input$Run_Annotation_go, if(input$method_go == 1){
      espece_id <- espece_id()
      orga_translate_table <- orga_translate_table()
      go <- re()
      original_gene_list <- go$log2FC
      names(original_gene_list) <- go$ID
      gene_list<-na.omit(original_gene_list)
      gene_list = sort(gene_list, decreasing = TRUE)
      sig_genes_df = subset(go, padj < input$pvalue_go)
      genes <- sig_genes_df$log2FC
      names(genes) <- sig_genes_df$ID
      genes <- na.omit(genes)
      tresholdLog2FoldChange <- tresholdLog2FoldChange()

      if (input$type_go == "over"){
        genes <- names(genes)[genes > tresholdLog2FoldChange]
      }
      else if(input$type_go == "under") {
        genes <- names(genes)[genes < -tresholdLog2FoldChange]
      }
      else if(input$type_go == "both") {
        genes <- names(genes)[abs(genes) > tresholdLog2FoldChange]
      }
      go_enrich <- enrichGO(gene = genes,
                            universe = names(gene_list),
                            OrgDb = orga_translate_table[espece_id,2],
                            keyType = 'ENSEMBL',
                            readable = T,
                            ont = input$Ontology,
                            pvalueCutoff = input$pvalue_go,
                            pAdjustMethod = input$Ajustement_go
                            )
    })
```

ORA

KEGG, Pathway enrichement

```
kegg_data <- eventReactive(input$Run_Pathway, {
      data <- re()
      espece_id <- espece_id()
      orga_translate_table <- orga_translate_table()
      adj_method <- input$kegg_adj_method

      ids = bitr(data$ID, fromType = "ENSEMBL", toType = "ENTREZID",
                 OrgDb=orga_translate_table[espece_id,2])

      dedup_ids = ids[!duplicated(ids[c("ENSEMBL")]),]
      df2 = data[data$ID %in% dedup_ids$ENSEMBL,]

      df2$Y = dedup_ids$ENTREZID

      kegg_gene_list <- df2$log2FC
      names(kegg_gene_list) <- df2$Y

      kegg_gene_list<-na.omit(kegg_gene_list)

      kegg_gene_list = sort(kegg_gene_list, decreasing = TRUE)

GSEA  if(input$method == 2){
        res <- gse_kegg(kegg_gene_list, adj_method, orga_translate_table[espece_id,])
      }
ORA   if(input$method == 1){
        res <- ora_kegg(df2, kegg_gene_list, adj_method, orga_translate_table[espece_id,])
      }
      list(res =res, kegg_gene_list=kegg_gene_list)
})
```

```
gse_kegg <- function(kegg_gene_list, adj_method, orga_translate_table) {
    if (input$db == 1){
    gse_result <- gseKEGG(geneList     = kegg_gene_list,
                          organism     = orga_translate_table[1,4],
                          nPerm        = 10000,
                          minGSSize    = 3,
                          maxGSSize    = 800,
                          pvalueCutoff = input$pvalue_gsea,
                          pAdjustMethod = adj_method,
                          keyType      = "ncbi-geneid")
    }
    else {
    gse_result <- gsePathway(geneList     = kegg_gene_list,
                             organism     = orga_translate_table[1,5],
                             nPerm        = 10000,
                             minGSSize    = 3,
                             maxGSSize    = 800,
                             pvalueCutoff = input$pvalue_gsea,
                             pAdjustMethod = adj_method)
    }
    return(gse_result)
}
```

KEGG

Reactome

GSEA

clusterProfiler

KEGG, Pathway enrichement

```r
kegg_data <- eventReactive(input$Run_Pathway, {
    data <- re()
    espece_id <- espece_id()
    orga_translate_table <- orga_translate_table()
    adj_method <- input$kegg_adj_method

    ids = bitr(data$ID, fromType = "ENSEMBL", toType = "ENTREZID",
               OrgDb=orga_translate_table[espece_id,2])

    dedup_ids = ids[!duplicated(ids[c("ENSEMBL")]),]
    df2 = data[data$ID %in% dedup_ids$ENSEMBL,]

    df2$Y = dedup_ids$ENTREZID

    kegg_gene_list <- df2$log2FC
    names(kegg_gene_list) <- df2$Y

    kegg_gene_list<-na.omit(kegg_gene_list)

    kegg_gene_list = sort(kegg_gene_list, decreasing = TRUE)

    if(input$method == 2){
      res <- gse_kegg(kegg_gene_list, adj_method, orga_translate_table[espece_id,])
    }
    if(input$method == 1){
      res <- ora_kegg(df2, kegg_gene_list, adj_method, orga_translate_table[espece_id,])
    }
    list(res =res, kegg_gene_list=kegg_gene_list)
})
```

GSEA

ORA

```r
ora_kegg <- function(df2, kegg_gene_list, adj_method, orga_translate_table) {
        # Exctract significant results from previous pvalue adjusted threshold
        kegg_sig_genes_df = subset(df2, padj < input$pvalue)
        # Vector of log2foldChange with ENTREZ IDs as names
        kegg_genes <- kegg_sig_genes_df$log2FC
        names(kegg_genes) <- kegg_sig_genes_df$Y
        kegg_genes <- na.omit(kegg_genes)
        # filter on log2fold change (under or over expressed DEG)
        if (input$type == 1){
          kegg_genes <- names(kegg_genes)[kegg_genes > input$tresholdLog2FoldChange]
        }
        else if(input$type == 2) {
          kegg_genes <- names(kegg_genes)[kegg_genes < - input$tresholdLog2FoldChange]
        }
        else {
          kegg_genes <- names(kegg_genes)[kegg_genes < - input$tresholdLog2FoldChange
                                          || kegg_genes > input$tresholdLog2FoldChange]
        }
        #Database to use for annotation
        if (input$db == 1){
          ora_result <- enrichKEGG(gene=kegg_genes, universe=names(kegg_gene_list),
                                   organism=orga_translate_table[1,4],
                                   pvalueCutoff = input$pvalue_gsea,
                                   keyType = "ncbi-geneid",
                                   pAdjustMethod = adj_method)
        }
        else {
          ora_result <- enrichPathway(gene=kegg_genes, universe=names(kegg_gene_list),
                                      organism=orga_translate_table[1,5],
                                      pvalueCutoff = input$pvalue_gsea,
                                      pAdjustMethod = adj_method)
        }
        return(ora_result)
}
```

KEGG

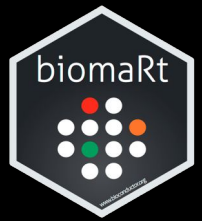
Reactome

ORA

clusterProfiler

Protein domain enrichement

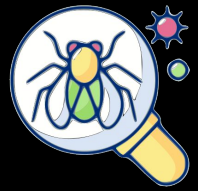
```
interpro_id_raw <- getBM(
  attributes=c('interpro', 'interpro_description', 'ensembl_transcript_id', 'ensembl_gene_id'),
  filters = 'ensembl_gene_id',
  values = resOrdered$ID,
  uniqueRows = FALSE,
  mart = ensembl)

create_table_enrichment = function(GeneList, GeneRef){

  Gene.Bg.ratio = get_Gene_and_Bg_ratio(GeneList = GeneList, GeneRef = GeneRef)
  Bg.ratio = signif(100 * Gene.Bg.ratio$m/(Gene.Bg.ratio$m + Gene.Bg.ratio$n), 3)
  Gene.ratio = signif(100 * Gene.Bg.ratio$x / Gene.Bg.ratio$k, 3)

  test = hypergeom_test(x = Gene.Bg.ratio$x,k = Gene.Bg.ratio$k,
                        m = Gene.Bg.ratio$m, n = Gene.Bg.ratio$n)

  table.enrich = data.frame(interpro_ID = Gene.Bg.ratio$Term,
                            pvalue = signif(test$pvalue_fisher.test, 3),
                            padj = signif(test$padj, 3),
                            BgRatio = Bg.ratio,
                            GeneRatio = Gene.ratio,
                            count = Gene.Bg.ratio$x,
                            )

return (table.enrich[order(table.enrich$pval), ])
```

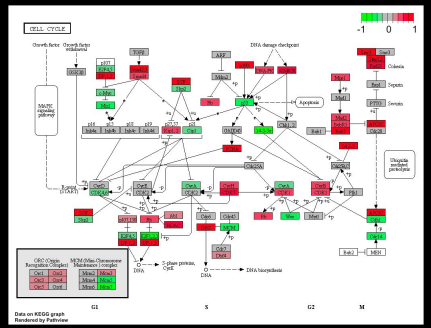


What next ?

Fichier d'entré :
trop spécifique (ENSEMBL)

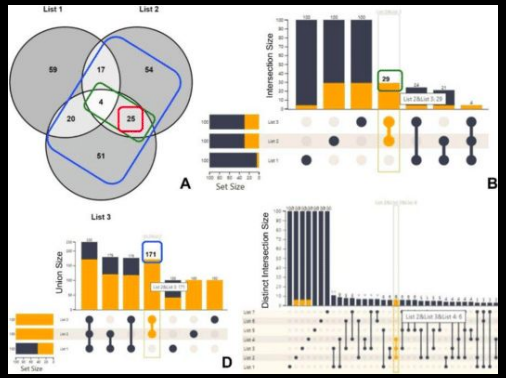
Nombre d'organismes modèles restreint …
et les autres ?



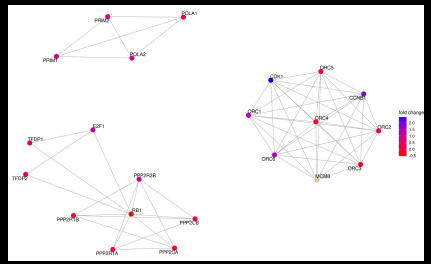


Pathview :
optimiser l'affichage
des réseaux

Plus de deux conditions
à comparer ?



https://www.researchgate.net/publication/352068166/figure/fig1/AS:1030430237151232@1622685121380/UpSet-plot-vs-Venn-diagrams-A-Intersection-of-the-three-gene-lists-100-genes-each.png



Reactome :
Obtenir un
réseau lisible
(viewPathway())

Merci de votre attention

Annotation fonctionnelle pour :

- GO
- KEGG
- Domaines Protéiques

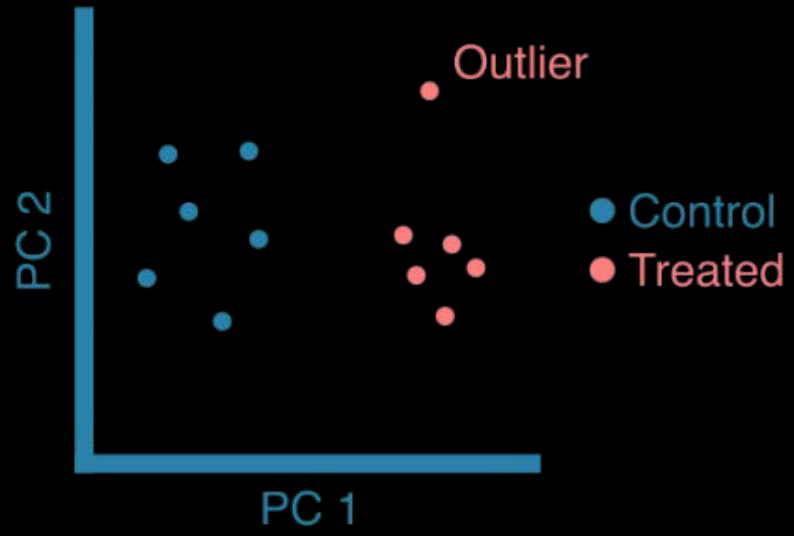
- Fichier d'entrée : trop spécifique (ENSEMBL)
- KEGG
  - Pathview : affichage des réseaux
  - Reactom : fonction viewPathway()

Organismes modèles… ok, et les autres ?

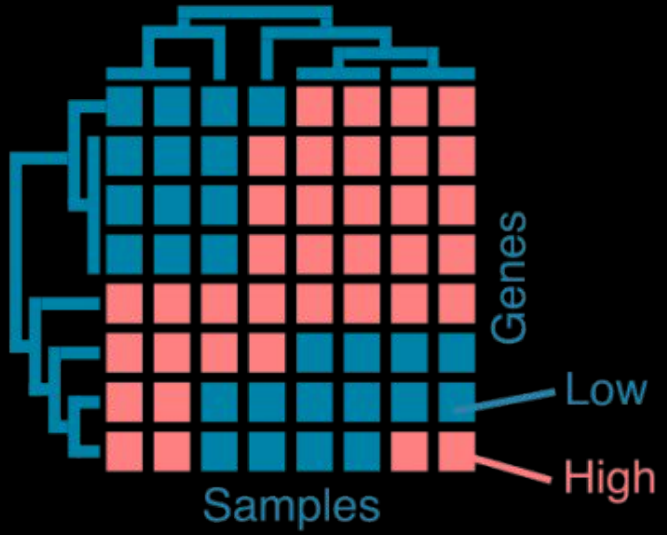
Restreint à un petit nombre d'organismes modèles

Enrichissement fonctionnel : souvent plus de deux conditions à comparer

Principal component analysis



Expression heatmap



GO Term Enrichement : Plots

server.R

```r
output$dotplot_gsea_go <-renderPlot({
      gse<-goGse_annot()
      require(DOSE)
      dotplot(gse, showCategory = input$showCategory_dotplot, title = "gsea dotplot" , split=".sign") + facet_grid(.~.sign)
    })

    output$ridgeplot_go<-renderPlot({
      gse<-goGse_annot()
      require(DOSE)
      ridgeplot(gse, showCategory =input$showCategory_ridgeplot)
    })

    output$gsea_plot_go <-renderPlot({
      gse<-goGse_annot()
      require(DOSE)
      gseaplot2(gse, geneSetID = input$showCategory_gseaplot)
})
```

uii.R

```r
conditionalPanel(

  condition = "input.method_go == 2",

  box(title = "Dot Plot GSEA", solidHeader = T, status = "success", width = 12, collapsible = T,id = "dotplot",
      fluidRow(
          column(3,
              wellPanel(
                  numericInput("showCategory_dotplot", "number of categories to show", value = 5))),
          column(9,
              wellPanel(
                  shinycustomloader::withLoader(plotlyOutput("dotplot_gsea_go", height = "450px"), type = "image", loader = "wait.gif")
```

GO Term Enrichement : Table

server.R

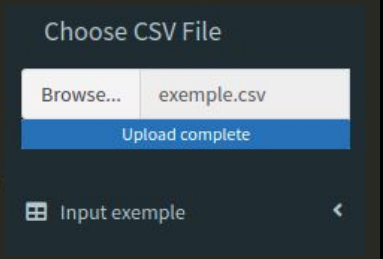
```
output$go_enrich_table_test <- DT::renderDataTable(DT::datatable({
      if (input$method_go == 1){
        data <- as.data.frame(goGse_enrich() )%>%
        mutate(interpro_link = paste0("<a href='https://amigo.geneontology.org/amigo/term/", Description,"' target='_blank'>", Description,"</a>"))
        col_a_afficher = c("URL",
                           "Description",
                           "GeneRatio",
                           "BgRatio",
                           "pvalue",
                           "p.adjust",
                           "qvalue")
        data[col_a_afficher]
      }
      else if (input$method_go == 2){
        data <- as.data.frame(goGse_annot() )%>%
        mutate(interpro_link = paste0("<a href='https://amigo.geneontology.org/amigo/term/", Description,"' target='_blank'>", Description,"</a>"))
        col_a_afficher = c("URL",
                           "Description",
                           "enrichmentScore",
                           "pvalue",
                           "p.adjust",
                           "rank")
        data[col_a_afficher]
      }
      updateSelectInput(session,"paths")
      DT::datatable(data)
      },
escape = FALSE))

```

uii.R

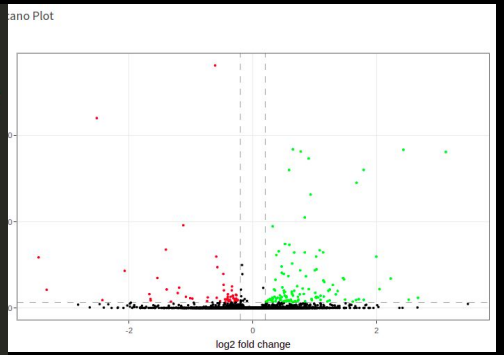
```
box (
  shinycustomloader::withLoader(dataTableOutput("go_enrich_table"), type = "image", loader = "wait.gif"),
width = 12) ,
```

Annexes

```
re <- reactive({
        file <- input$file1
        ext <- tools::file_ext(file$datapath)
        req(file)
        validate(need(ext == "csv", "Invalid file. Please upload a .csv file"))
        data <- read.csv(file$datapath, header = TRUE, sep = ";")
        data <- na.omit(data)
        required_columns <- c("ID", "baseMean", "log2FC", "pval", "padj")
        column_names <- colnames(data)
        shiny::validate(need(all(required_columns %in% column_names), "Missing required columns"))
        data
})
```

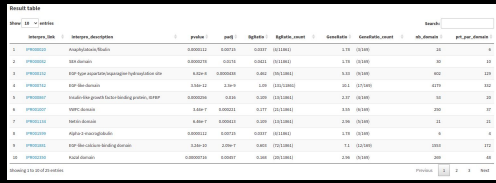


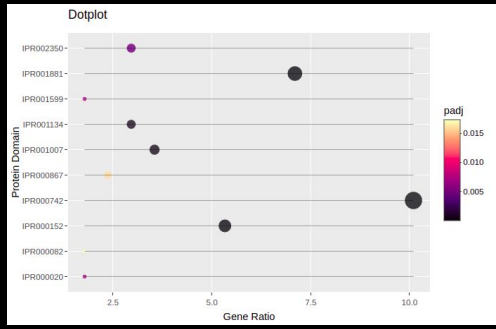

```
p <- ggplot(
      data = data_plot_plotly,
      mapping = aes(x = log2FC, y=-log10(padj), col=diffexpressed, key = key)) +
      geom_point(size = 0.45) +
      theme_bw()  +
      scale_color_manual(values=c("red", "black", "green")) +
      xlab("log2 fold change") +
      ylab("-log10(p-value)")
```

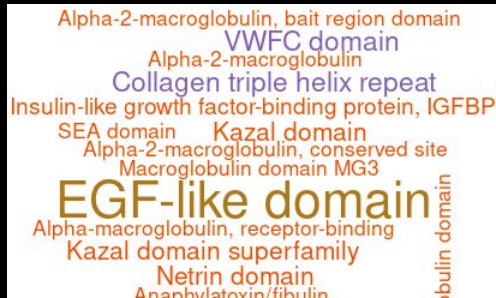


Enrichissement en domaines protéiques

```
interpro_id_raw <- getBM(
  attributes=c('interpro', 'interpro_description', 'ensembl_transcript_id', 'ensembl_gene_id'),
  filters = 'ensembl_gene_id',
  values = resOrdered$ID,
  uniqueRows = FALSE,
  mart = ensembl)

create_table_enrichment = function(GeneList, GeneRef){

  Gene.Bg.ratio = get_Gene_and_Bg_ratio(GeneList = GeneList, GeneRef = GeneRef)
  Bg.ratio = signif(100 * Gene.Bg.ratio$m/(Gene.Bg.ratio$m + Gene.Bg.ratio$n), 3)
  Gene.ratio = signif(100 * Gene.Bg.ratio$x / Gene.Bg.ratio$k, 3)

  test = hypergeom_test(x = Gene.Bg.ratio$x,k = Gene.Bg.ratio$k,
                        m = Gene.Bg.ratio$m, n = Gene.Bg.ratio$n)

  table.enrich = data.frame(interpro_ID = Gene.Bg.ratio$Term,
                            pvalue = signif(test$pvalue_fisher.test, 3),
                            padj = signif(test$padj, 3),
                            BgRatio = Bg.ratio,
                            GeneRatio = Gene.ratio,
                            count = Gene.Bg.ratio$x,
                            )

return (table.enrich[order(table.enrich$pval), ])
```







Procédure [Irizarry *et al.*, 2009, Mootha *et al.*, 2003, Subramanian *et al.*, 2005]

- Soit un ensemble de gènes S pré-défini (ex : Terme GO)

1. Calcul d'un score d'enrichissement (Y et N sont de même signe)
   - Gènes sont ordonnés (par p-value, par LFC ou autre)
   - Score calculé :

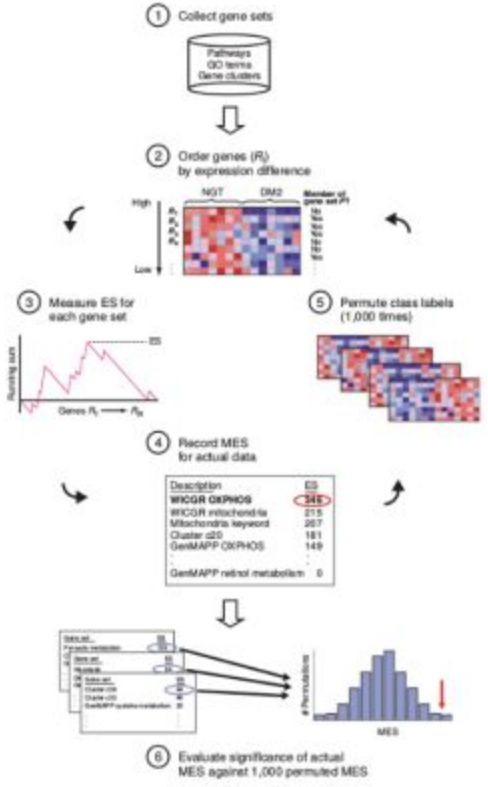
$$S_{c_i} = \begin{cases} +Y & \text{si le gène} i \in S \\ -N & \text{sinon.} \end{cases}$$

$$ES = \max_{1 \le i \le n} \sum_{i=1}^{j} S_{c_i} \text{ si } Y > 0 \text{ ou } ES = \min_{1 \le i \le n} \sum_{i=1}^{j} S_{c_i} \text{ si } Y < 0$$

2. Estimation du niveau de significativité du score ES
   - Permutation de l'ensemble S (1 000)
   - Calcul du score ES pour les ensembles permutés qu'on stocke
3. Comparaison du score ES calculé à celui des ensembles permutés
   - P-valeur calculée :

$$p = \frac{1}{1000} \sum_{j=1}^{1000} \mathbf{1}_{ES \ge ES_j} \text{ si } Y > 0 \text{ ou } p = \frac{1}{1000} \sum_{j=1}^{1000} \mathbf{1}_{ES \le ES_j} \text{ si } Y < 0$$

Étapes



Mootha *et al.*, 2003

Résumé

- Test de randomisation/Kolmogorov-Smirnov
  - Détermine si un échantillon suit bien une loi donnée
  - Hypothèses sont :

$$H_0 : D_1 = D_2 \quad \text{vs} \quad H_1 : D_1 \neq D_2$$

  - Hypothèse nulle : si l'enrichissement est présent, cela est dû au hasard

3 App 1: Connecting `ui` and `server`