Amélioration de l'accès aux résultats biologiques :

Séquençage ARN et Application Shiny

Master 1 Bioinformatique

Université de Rennes 1

2019 - 2020

David GALLIEN & Gabin COUDRAY

UMR CNRS 6290
Institut de Génétique et Développement de Rennes (IGDR)

Annabelle MONNIER
Equipe Oncogénomique translationnelle

UNIVERSITÉ DE RENNES 1

ENGAGEMENT DE NON PLAGIAT

Je, soussigné (e) ……
Etudiant (e) en ……

Déclare être pleinement informé (e) que le plagiat de documents ou d'une partie de documents publiés sous toute forme de support (y compris l'internet), constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour la rédaction de ce document.

Signature

Laboratoire de Génomique Médicale
BMT-HC - CHU Pontchaillou

2 rue Henri le Guilloux
35033 Rennes Cedex
FRANCE

Annabelle MONNIER
annabelle.monnier@univ-rennes1.fr
TÉL. 33 (0)2 99 28 92 54

UMR CNRS 6290
Institut de Génétique et Développement de Rennes (IGDR)

Annabelle MONNIER
Equipe Oncogénomique translationnelle

UNIVERSITÉ DE RENNES 1

ENGAGEMENT DE NON PLAGIAT

Je, soussigné (e) ……
Etudiant (e) en ……

Déclare être pleinement informé (e) que le plagiat de documents ou d'une partie de documents publiés sous toute forme de support (y compris l'internet), constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour la rédaction de ce document.

Signature

Laboratoire de Génomique Médicale
BMT-HC - CHU Pontchaillou

2 rue Henri le Guilloux
35033 Rennes Cedex
FRANCE

Annabelle MONNIER
annabelle.monnier@univ-rennes1.fr
TÉL. 33 (0)2 99 28 92 54

Table des matières

Introduction

Contexte

Aujourd'hui, la plupart des chercheurs n'ont en général pas le temps d'utiliser les outils permettant une analyse des données générées par les nouvelles technologies. C'est pour cela qu'il est important de leur offrir la possibilité d'avoir accès à des outils facilitant l'analyse et leur permettant d'être plus efficaces. Le but principal de notre projet est de mettre en place une application qui pourra aider ces scientifiques à explorer leurs résultats sous l'environnement R. Pour cela le package Shiny nous permet de créer une application web interactive.

Néanmoins, afin de créer une application interactive, nous avons besoin de quelques choses à montrer. Dans le domaine de la recherche médicale qui est le principal axe de recherche, les techniques de séquençage de seconde génération sont souvent utilisées. Ces techniques génèrent de nombreuses données qui ont besoin d'être explorées et analysées. Nous avons donc décidé de concentrer notre travail sur le séquençange de l'ARN (RNA-seq). Cette technique de séquençage de seconde génération a pour principal but de détecter des expressions différentielles entre des types cellulaires de différentes conditions.

Le RNA-seq est un nouveau moyen permettant un séquençage de l'ARN plus rapide que les techniques qui existaienet précédemment comme la méthode de Sanger. Le but principal du RNA-seq est d'étudier l'expression différentielle de gènes entre différentes conditions. Le séquançage de l'ARN a été cité pour la première en 2008, et depuis, le nombre de publication contenant des données de RNA-seq augmentent d'années en années. Ce genre d'analyses utilises les technologies de séquançage de nouvelles génération (NGS) comme Illumina, Roche 454 ou encore Ion Torrent.

Une analyse RNA-seq présente 3 grandes étapes :

- Fragmentation aléatoire des ARN matures
- Amplification de ces fragments par PCR
- Séquençage de ces fragments donnant des millions de reads

Le nombre de reads obtenu est proportionnel à l'abondance des ARN dans la cellules. Ces reads sont stockés dans des fichiers au format fastQ et leur qualité est estimé grâce à des outils spécifiques. Ensuite, chaque read est mappé sur le génome de référence de l'organisme étudié. Après ce mapping, des fichiers BAM sont obtenus. Dans ces fichiers, chaques ligne représante un alignement d'un read. Pour finir, un comptage des reads pour chaque position est realisé afin de remplir une table de comptage permettant l'analyse des données RNA-seq.

Analyse de données RNA-seq

Les étapes de l'analyse RNA-seq présentées ci-dessous sont inspirées du mode d'emploi d'analyse de données RNA-seq sur le site bioinfo-fr.net.

Etape 1 : Les données

Premièrement, pour obtenir le jeu de données, l'ARN est extrait des cellules et l'ARNm est isolé grâce à sa queue poly-adénylée. Une fois extrait, l'ARNm est fragmenté et subit une reverse transcription en ADNc. Ensuite, l'ADNc est séquencé grâce aux NGS. Aujourd'hui, le plus utilisé est la technologie Illumina qui utilise une amplification clonale et un séquençage par synthèse. Le séquençage peut être "single end" (chaque read est indépendant) ou "paired-end" (les reads sont pairés). Après le séquençage, des millions de reads sont obtenus.

Etape 2 : Contrôle qualité

A la sortie du séquenceur on retrouve des fichiers fastQ. Ce genre de fichier est composé de blocs de 4 lignes représentants un read. Grâce aux fichiers fastQ, la qualité du séquençage peut être estimée à l'aide de programmes comme FastQC.

Etape 3 : Mapping

Cette partie de l'analyse consiste à aligner tous les reads sur le génome de l'organisme étudié. Un read est mappé sur la régions du génome qui lui est la plus similaire. Cette étape permet d'obtenir des fichiers BAM dans lesquels chaque ligne correspond à un read. De plus, la moyenne du nombre de reads mappés sur une régions est appelée la profondeur.

Etape 4 : Quantification

Le nombre de reads est un témoin de l'abondance d'ARN dans la cellule. Ainsi, il est possible d'estimer le niveau d'expression d'un gène. C'est pourquoi il est important de compter les reads mappés pour chaque gène. Le but de cette étape est de remplir une table de comptage afin de pouvoir la manipuler facilement avec R par exemple.

Etape 5 : Statistiques

Différents résultats statistiques peuvent être obtenus ainsi que des graphiques ou encore des carte de densité pour les exons. Il est important de normaliser les données et de comparer les p-value ajustées obtenues après différents tests. Tout ceci permet d'établir une liste de gène différentiellement exprimés.

Objectifs

Aujourd'hui, de plus en plus d'études utilisent le séquençage de l'ARN, il en résulte de plus en plus de données à analyser. Pour essayer de répondre à cette problématique nous avons décidé d'élaborer une application interactive grâce au package RShiny. Cette application a pour but d'aider à l'analyse de données RNA-seq le plus profondément possible en répondant aux plus de questions possibles et permettre une visualisation intuitive des résultats. Le but de ce projet est de nous permettre d'en apprendre plus sur cette nouvelle technique de séquençage de l'ARN et son analyse. Cela nous permettra aussi d'apprendre à utiliser différents packages disponibles sous R comme Shiny pour la conception de l'application ou Markdown pour la rédaction du rapport.

Matériels et méthodes

Jeu de données

Nous allons tout d'abord procéder à une analyse de l'expression différentielle de gènes en utilisant le package DESeq2. Pour cela, nous avons récupéré un jeu de données de l'étude RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells de Himes BE, Jiang X, Wagner P, et al. Les glucocorticoïdes sont utilisés pour traiter l'asthme et le but de cette étude est de comprendre le mécanisme dans les muscles lisses des voies respiratoires en utilisant la technologie RNA-seq.

Cette expérience rassemble 8 échantillons : 4 traités avec du dexamethasone (glucocorticoïde synthétique) et 4 échantillons contrôles sans traitements. Nous avons donc une table de comptage de reads dans laquelle on trouve le nombre de reads mappés sur chaque gène pour chaque échantillon. De plus, nous avons un fichier d'annotation des gènes contenant des informations sur tous les gènes.

Packages

Durant ce projet nous avons utilisé Rstudio afin de réaliser une analyse de données RNA-seq et de mettre en place une application interactive permettant de le faire. Pour cela nous avons donc du utiliser différents packages que nous présentons dans cette partie.

Analyse RNAseq

Pour procéder à l'analyse de données RNA-seq, nous avons besoin de packages DESeq2, gplot, ggrepel, Tidyverse et NMF. Ces différents packages vont permettre de réaliser une étude d'expression différentielle mais aussi d'afficher des graphiques importants pour avoir des résultats et de manipuler les données.

DESeq2

DESeq2 est un package sous R capable de procéder à une analyse de l'expression différentielle de gènes basée sur la loi binomiale négative. Ce package est disponible chez Bioconductor, un site internet qui met à disposition des utilisateurs différents logiciels bioinformatiques open source et plus spécifiquement pour l'analyse et l'étude de séquençages hauts débits comme le RNA-seq ici. Avec les différents outils de ce package, il est possible d'estimer des relations moyenne-variance dans le jeu de données et de tester l'expression différentielle basée sur la loi binomiale négative.

Cette loi binomiale négative est une alternative à la loi de Poisson. En effet, il s'agit d'une loi de probabilité discrete. Pour résumer, on considère une expérience qui peut aboutir à un succès de probabilité p ou un échec. Cette expérience se termine après un nombre choisi de succès. Le but est de savoir le nombre d'échecs ayant eu lieu avant le nombre de succès défini. C'est cette loi qui est utilisée car le comptage des reads n'est pas continu, on ne retrouve que des entier non nuls donc on ne peut pas utiliser une distribution normale. Dans la loi binomiale négative, la variance est toujours supérieure ou égale à la moyenne. Pour finir, dans une analyse RNA-seq, les gènes sous-exprimés ont une variance plus élevée que les gènes sur-exprimés.

Il est donc nécessaire d'avoir au préalable une table de comptage ainsi qu'une table contenant le design de l'expérience car la première étape consiste a crée un premier jeu de données DESeq contenant ces deux éléments. Une fois l'objet créé, il est possible d'acceder à la table de comptage ainsi qu'au design de l'expérience à l'aide de fonctions propres au jeu de données DESeq2 (annexe 1). La seconde étape consiste à normaliser les doonées de la table de comptage et à réaliser l'analyse de l'expression différentielle. Le package DESeq2 permet de réaliser ces deux étapes d'une traite. Une fois cette seconde étape réalisée on peut extraire les résultats de l'analyse d'expression différentielle, et commencer à les analyser.

```
dds <- DESeqDataSetFromMatrix(countData = counts_table,colData=airway_metadata,design = ~dex,tidy = TRUE)
dds <- DESeq(dds)
res <- results(dds)
```

Figure 1 : code des trois étapes principales de l'analyse d'expression différentielle avec le package DESeq2

ggplot2 et ggrepel

Disponibles sur le CRAN, ces deux packages permettent la visualisations de résultats sous la formes de graphiques élégants et optimisés. En effet, ggplot2 permet de réaliser des graphiques plus complexes qu'avec les fonctions de base sous R. Quant à ggrepel, il permet de mettre du texte sur ces graphiques.

Tidyverse

Tidyverse est un package également disponible sur le CRAN et regroupant plusieurs packages comme ggplot2, dplyr, tidyr, readr, purrr, tibble, stringr, ou forcats. Tous ces packages vont permettre de modéliser, transformer et visualiser les données.

NMF

Le package NMF pour "Non negative Matrix Factorization" est aussi disponible sur le CRAN et va nous être utile pour réaliser des heatmaps.

Création de l'application

Concernant l'application, nous utilisons toujours les packages présentés précédemment pour l'analyse de données RNA-seq. Mais nous utilisons également des packages propres à la création d'une application comme Shiny ou Shinydashboard.

Shiny

Premièrement, le package Shiny permet la création d'application web interactive capables d'utiliser toutes les fonctionnalités de R. C'est un outil permettant de créér des pages web sans forcément avoir de connaissance en HTML, css ou javascript. Cependant, c'est un plus d'avoir des connaissances dans ce domaine si on veut customiser ces pages web afin d'avoir une application plus attirante. Une application shiny est composée de deux fichiers :

- Le fichier ui.R (user interface) permet de contrôler l'apparance de l'application (annexe 2.1).
- Le fichier sever.R contient les instruction permettant le fonctionnement de l'application (annexe 2.2).

Shinydashboard

Shinydashboard permet d'avoir un modèle d'application sous forme de tableau de bord. On y retrouve un en-tête avec le titre de l'application, une barre latérale servant de menu et le corps principal avec lequel on va pouvoir interagir. Le menu latéral va permettre de naviguer entre les différentes pages de l'application. Le tableau de bord peut être personalisé comme on le souhaite à l'aide de css ou javascript afin d'avoir un environnement agréable à prendre en main.

Rédaction du rapport

Markdown

Pour la rédaction du rapport, nous utilisons RMarkdown qui nous permet de faire un rapport automatisé de notre travail. Les packages markdown et knitr permettent d'assembler rapport sous forme de texte et de code R. Ainsi nous pouvons intégrer notre code et les résultats obtenus lors de l'analyse. Nous avons choisi un format de sortie sous forme de PDF, ce qui nous autorise à utiliser LaTeX afin d'avoir un document final avec une mise en page optimale que nous auront définie.

Résultats

Analyse de données RNA-seq

Application RShiny

Conclusion et perspectives

Bibliographie

1. Himes BE, Jiang X, Wagner P, et al. RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells. PLoS One. 2014;9(6):e99625. Published 2014 Jun 13. doi: 10.1371/journal.pone.0099625

2. Koch CM, Chiu SF, Akbarpour M, et al. A Beginner's Guide to Analysis of RNA Sequencing Data. Am J Respir Cell Mol Biol. 2018;59(2):145-157. doi: 10.1165/rcmb.2017-0430TR

3. Julien Delafontaine. (2013, septembre 11). Analyse de données RNA-seq : mode d'emploi. Consulté à l'adresse https://bioinfo-fr.net/lanalyse-de-donnees-rna-seq-mode-demploi

4. Bioconductor - Home. (2003). Consulté à l'adresse https://www.bioconductor.org/

5. Wickham, H. (2016). Mastering Shiny. Consulté à l'adresse https://mastering-shiny.org/

Résumé

Abstract

Annexe 1 : Script R pour l’analyse RNAseq via DESeq2

Annexe 2 : Scripts R pour l'application Shiny

2.1 Interface utilisateur

```
### Library ----
### here we find all library need for proper functionning of app
library(shiny)
library(shinydashboard)
library(shinyjs)
library(dplyr)
library(tidyverse)
library(vroom)
library(DT)
library(DESeq2)
library(shinyWidgets)
library(shinythemes)
library(waiter)
library(dashboardthemes)
library(shinycssloaders)
library(shinydashboardPlus)
### Files with all the function needed to make plots ----
source("function_dds.R")

### Annotation pannel ----
parameters_Annotation <- tagList(
  tags$style("#paramsAnno { display:none; }"),
### parameter_tabs is a tabsetPanel for input annotation page
### it allow to switch between panel when we use updateTabsetPanel()
### in our case it is when we check right annotation boxes of input annotation page
  tabsetPanel(
    id="paramsAnno",
    tabPanel("nothing"),
    tabPanel("annotation",
             fluidRow(
               column(width= 6,
                      box(title="Upload annotation file",width = 12, solidHeader = TRUE,collapsible = TRUE,
                          column(width=5,selectInput("sep_Anno", "Separator:", c("Comma" = ",", "Tab" = "\t",
                     "Semi-colon" = ";"))),
                          fileInput("AnnotationFile", "Upload annotation file",
                                    accept = c(".csv",".txt",".tsv")))),
               column(width= 6,
                      box(
                        title = "Accepted files :", width = 12,
                        HTML(
                          "<li> .csv / .tsv / .txt files </li>
                          <li> Separated by tabulation, comma or semi-colon </li>
                          <li> One column with genes symbols named 'symbol'</li>
                          "),
                        height = 160
                        )
                      )
                    )
                )
            )
          )

### User Interface  ----
### UI is a shinydashboard, we use library shinydashboard to set up a dashboard UI
### a dashboard is compound of :
###   - a Header
###   - a sidebar
###   - a body
ui <- tagList(
  ### Parameters of the dashboard ----
  div(
    id = "app",
    dashboardPage(
      ### Customize the header ----
      ### header is composate of :
      ###   - a title
      ###   - a home bouton which on click return user on introduction page
      dashboardHeader(title = "RNA-seq DE analysis",
                      uiOutput("themes"),
```

```
                ### Home button ----
                tags$li(a(onclick = "openTab('Intro')",
                          href = NULL,
                          icon("home"),
                          title = "Homepage",
                          style = "cursor: pointer;"),
                        class = "dropdown",
                        tags$script(HTML("
                                         var openTab = function(tabName){
                                         $('a', $('.sidebar')).each(function() {
                                         if(this.getAttribute('data-value') == tabName) {
                                         this.click()
                                         };
                                         });
                                         }")))
),

### Sidebar ----
### Sider bar  is composate of a sidebar menu
### in this sidebar menu we have menu item which is associate with a tab of body dashboard
dashboardSidebar(
  sidebarMenu(id="mysidebar",
      menuItem(text = "Informations", tabName = "Intro", icon = icon("info-circle")),
      menuItem(text = "1 Upload data", tabName = "upload", icon = icon("arrow-circle-up"),
               startExpanded = TRUE,
                  # menuItem which are set up in the server function
                menuItemOutput("CountTable"),
                menuItemOutput("MetadataTable"),
                menuItemOutput("AnnotationTable")),
      menuItem(text = "2 Run DESeq2", tabName = "deseq2", icon = icon("play-circle")),
      menuItemOutput("menuResults"),
      tags$hr(),# a simple line
      # this menu generate a switch button to set theme of dashboard
      # two options are available a light or dark mode
      menuItem(icon = NULL,
      materialSwitch(inputId = "theme", label = "Theme", status = "default", value= TRUE)
      ),tags$hr()
  )
),
### Dashboard body ----
### Organization of the differents pages associate with their menuItem
dashboardBody(
  useShinyjs(),
  fluidRow(
    tabItems(
      ### Introduction ----
      ### introduction page associate with menuItem "Informations"
      ### In this page we find all information about application
      ### in particular how it works and different tool used to generate DE analysis
      ### to generate layout html tag provide by shiny library are used
      ### withSpinner() of shinycssloaders library is use to generate waiting screen during load of img
      tabItem(tabName = "Intro",
              fluidPage(
                h2("Introduction"),
                p("This is an R Shiny web interactive application developed as part of a ",
                  strong("course project."), "The purpose of this application is to perform a ",
                  strong("differential expression analysis from a counts table"),
                  "in order to help
                  researchers getting interpretable results.",
                  align = "justify"),
                p("This application uses the package ",
                  a("DESeq2", href="https://bioconductor.org/packages/release/bioc/html/DESeq2.html"),
                  "from Bioconductor. It is a package to study differential gene expression analysis
                  based on the negative binomial distribution. It allows a quick visualization of results
                  in order to analyze the counts table data. The results will be given in different forms
                  like graphics, heatmaps, MAplot or even Volcano plot.",
                  align = "justify"),
                tags$hr(),
                h3("1. Upload data", style="padding-left: 1em"),
                p("The input data files accepted for this App are 3 files in '.txt', '.csv' or '.tsv'
                format separated by comma, tabulation or semi-colon.
                  This App necessarily requires a 'Count Data Table' and a 'Metadata Table'. An optional
                'Annotation File' can be added", style="padding-left: 2em", align = "justify"),
                h4("1.1 Count Data Table", style="padding-left: 3em"),
                p("The Count Data Table must contain the count for each sample of the experiment for each
                  gene and the first column must be gene ID or gene name as below :",
                  style="padding-left: 5em", align = "justify"),
                column( 12, style="padding-left: 5em" ,withSpinner(tableOutput("countExample"))),
                br(),
```

```
              h4("1.2 Metadata Table", style="padding-left: 3em"),
              p("The Metadata table must contain the information of the experiment with at least
              2 columns. The first one corresponds to the samples in the same order as the columns of
              the Count Table.
                The second one is a condition column. You can add as many columns as you have factors
              in your experiment.",style="padding-left: 5em", align = "justify"),
              column( 12, style="padding-left: 5em" ,withSpinner(tableOutput("metadataExample"))),
              h4("1.2  Annotation File", style="padding-left: 3em"),
              p("The Annotation File contains informations about the genes. If you have one, it must
                contain a column named 'symbol' in which we can find the symbol of each gene.",
                style="padding-left: 5em", align = "justify"),
              column( 12, style="padding-left: 5em" ,withSpinner(tableOutput("annoExample"))),
              h3("2. Results", style="padding-left: 1em"),
              p("The results will be display after running DESeq2. You will obtain 9 differents
                results :", style="padding-left: 2em", align = "justify"),
              p("- Count distribution",
                br(), "- Count by gene",
                br(), "- Depth of sample",
                br(), "- Dispersion",
                br(), "- PCA",
                br(), "- MA plot",
                br(), "- Volcano plot",
                br(), "- Distance matrix",
                br(), "- Heatmap",style="padding-left: 5em", align = "justify"),
              p("You can download all the results plots at the bottom of all these pages.",
                style="padding-left: 2em", align = "justify")
              )
            ),
      ### Upload count table ----
      ### upload page of count table of RNA-seq experience
      ### this page is associate with menuItemOuput("")
      ### On this page we find :
      ###   - a box which countain :
      ###       - a selectInput of separator
      ###       - a fileInput to upload count table of RNA-seq experience
      ###   - a box which countain :
      ###       - information about file accepted in fileInput
      ###   - a dataTableOutput of count table input in fileInput
      tabItem(tabName = "CountData",
              column(width = 6,
                     box(title="Upload count table",width = 12, solidHeader = TRUE,collapsible = TRUE,
                         column(width=5,
                                selectInput("separator_Count", "Separator:", c("Comma" = ",",
                         fileInput("CountDataTable", "Upload count table",
                                   accept = c(".csv",".txt",".tsv"))
                     )),
              column(width = 6,
                     box(
                       title = "Accepted files :", width = 12,
                       HTML(
                         "<li> .csv / .tsv / .txt files </li>
                         <li> Separated by tabulation, comma or semi-colon </li>
                         <li> First column has to be gene ID or gene name</li>
                         <li> All others columns are count for each sample</li>"),
                       height = 160
                       )),
              dataTableOutput("CountReadTable")
              ),
      ### Upload metadata table ----
      ### upload page of metadata file of RNA-seq experience
      ### this page is associate with menuItemOuput("")
      ### On this page we find :
      ###   - a box which countain :
      ###       - a selectInput of separator
      ###       - a fileInput to upload metadata of RNA-seq experience
      ###   - a box which countain :
      ###       - information about file accepted in fileInput
      ###   - a box which countain :
      ###       - a text input to chose design formula to set for DESeq2 dataset object
      ###   - a dataTableOutput of metadata input in fileInput
            tabItem(tabName = "Metadata",
              column(width = 6,
                     box(title="Upload metadata table",width = 12, solidHeader = TRUE,collapsible = TRUE,
                         column(width=5,
                                selectInput("separator_Metadata", "Separator:",
                                            c("Comma" = ",", "Tab" = "\t", "Semi-colon" = ";"))),
                         fileInput("MetadataFile", "Upload metadata table",
                                   accept = c(".csv",".txt",".tsv"))
                     )),
```

```
                column(width = 6,
                       box(
                         title = "Accepted files :", width = 12,
                         HTML(
                           "<li> .csv / .tsv / .txt files </li>
                           <li> Separated by tabulation, comma or semi-colon </li>
                           <li> At least metadata table contains two columns</li>
                           <li> At least one column has to be factor</li>"),
                         height = 160
                         )),
                column(width = 12,
                       box(width = 12,
                           textInput("DesignDESeq2","Choose your design without linear combination",
                                     placeholder = "Conditions"))),
                dataTableOutput("MetaTable")
                ),
        ### Upload annotation file ----
        ### upload page of annotation file associate with the RNA-seq experience
        ### this page is associate with menuItemOuput("")
        ### on this page we find :
        ###   - a box with :
        ###       - a checkboxInput :
        ###            - if box is check on : parameter_tabs is set on annotation
        ###               - On this page we find :
        ###                   - a box which countain :
        ###                       - a selectInput of separator
        ###                       - a fileInput to upload metadata of RNA-seq experience
        ###                   - a box which countain :
        ###                       - information about file accepted in fileInput
        ###            - if box is check off : parameter_tabs is set on nothing
        ###               - On this page we find : nothing
          tabItem(tabName = "Annotation",
                fluidPage(
                  box(width = 12,
                      checkboxInput("CheckAnnotation","Do you have an annotation file ?",value=FALSE)),
                  fluidRow(
                    parameters_Annotation),
                  dataTableOutput("AnnoTable")
                )
        ),
        ### Run DESeq2 ----
        ### On this page we find little information about how run DESeq2 workflow
        ### a actionButton when press it run DESeq2 workflow
        ### a uiOutput("") set on server function
        tabItem(tabName = "deseq2",
                waiter::use_waiter(),
                fluidPage(
                  box(width = 12, solidHeader = F,
                      HTML(" <center><h3>Here you gonna run DESeq2 workflow.</h3> </pre>
                           <br><p> Check if your design chosen previously is correct.
                           <br>If it is not, the application will crash.</p>
                           <br><h7>This will take a few seconds.</h7></center>")),
                  box(width = 12,

                      actionButton("RunDESeq2","Run DESeq2 Workflow ",icon = icon("fas fa-user-astronaut"),
                                   class="btn btn-danger btn-lg btn-block ")),
                  uiOutput("SuccessMessage")
                      )
                ),
        ### Count distribution page ----
        ### On this page we find count distribution plot and it parameters after running DESeq
        ### On this page we find :
        ###     - a box() which countain :
        ###         - selectInput of sample which set sample on
        ###           count.distribution.plot function
        ###         - sliderInput of break width which set break.width on count.distribution.plot function
        ###         - sliderInput range of x.axis which set x.min and x.max on count.distribution.plot
        ###           function
        ###     - a box() which countain :
        ###         - return of count.distribution.plot
        tabItem(tabName = "Count_Distribution",
                box(title="Count distribution",solidHeader = T, status = "primary",width=12,
                    collapsible = TRUE,
                    column(width = 6,
                           selectInput("sample","Which sample do you want to see ?", choices = c())
                    ),
                    column(width = 6,
                           sliderInput("breaksDistribution","Break size",min=0,max=2,value=1.0,step = 0.25)
                    ),
```

```
                column(width = 6,
                       sliderInput("axis","Axis x",min=0,max=20,value=c(0,14))
                ),
                column(width = 6, checkboxInput("normalizeDistribution","Do you want to see distribution
                                                after normalisation ?",value=FALSE)
                )
            ),
            box(width=12,status = "primary",withSpinner(plotOutput("CountDistributionPlot"))),
            column(width= 4,
                   downloadButton("downloadDistribution",'Download plot',class = "btn-warning")
            )
    ),
    ### Count by gene page ----
    ### On this page we find count by gene plot and it parameters after running DESeq
    ### On this page we find :
    ###     - a box() which countain :
    ###         - selectizeInput of gene which set sample on count.gene.plot function
    ###         - a checkBox normalization if checkbox is check ON dds.count use is normalize, else
    ###           dds.count use is not normalize
    ###     - a box() which countain :
    ###         - return of count.gene.plot
    ###     - a downloadButton to download  the generate plot
    tabItem(tabName = "Count_Gene",
            box(title="Count by gene",solidHeader = T, status = "primary",width=12,collapsible = TRUE,
                column(width = 6,
                       selectizeInput("gene","Which gene do you want to see ?", choices = NULL)
                ),
                column(width = 6, checkboxInput("normalizeCountGene","Do you want to see distribution
                                                after normalisation ?",value=FALSE)
                )
            ),
            box(width=12,status = "primary",withSpinner(plotOutput("CountGenePlot"))),
            column(width= 4,
                   downloadButton("downloadCountgene",'Download plot',class = "btn-warning")
            )
    ),
    ### Depth plot ----
    ### On this page we find depth sample plot and it parameters after running DESeq
    ### On this page we find :
    ###     - a box() which countain :
    ###         - sliderInput of break width which set break.width on depth.plot function
    ###         - a checkBox normalization if checkbox is check ON dds.count use is normalize, else
    ###           dds.count use is not normalize
    ###     - a box() which countain :
    ###         - return of depth.plot
    ###     - a downloadButton to download  the generate plot
    tabItem(tabName = "Depth",
            box(title="Depth of Sample",width = 12,solidHeader = T, status = "primary",
                collapsible = TRUE,
                sliderInput("breaksDepth","Bar size",min=0,max=4,value=0.75,step = 0.25),
                checkboxInput("normalizeDepth","Do you want to see depth after normalisation ?",
                              value=FALSE)
            ),
            box(width=12,status = "primary",withSpinner(plotOutput("depth",height = 500))),
            column(width= 4,
                   downloadButton("downloadDepth",'Download plot',class = "btn-warning")
            )
    ),
    ### PCA plot ----
    ### On this page we find PCA plot and it parameters after running DESeq
    ### We find :
    ###     - a box() which countain :
    ###         - a selectInput of intgroup for pca.plot function
    ###         - a selectInput to chose transformation for pca.plot
    ###         - a actionButton to run pca?plot function
    ###     - a box() which countain :
    ###         - return of pca.plot
    ###     - a downloadButton to download  the generate plot
    tabItem(tabName = "pca",
            box(width = 12,
                title = "PCA", solidHeader = T, status = "primary",collapsible = TRUE,
                selectInput("TransformationPCA",label= "Choose your transformation",
                            choices = c("Variance-stabilizing transformation"="vst",
                                        "Log transformation"="rld")),
                selectInput("conditionpca","Choose your intgroup for PCA ?", choices = c()),
                actionButton("runPCA","Run PCA")
            ),
            box(solidHeader = F, status = "primary",width = 12,
                withSpinner(plotOutput("PCAplot",height = 650))),
```

```
        column(width= 4,
               downloadButton("downloadPCA",'Download plot',class = "btn-warning")
        )
),
### Dispersion plot ----
### On this page we find dispersion plot after running DESeq
### We find :
###     - a box() which countain :
###         - return of dispersion.plot
###     - a downloadButton to download  the generate plot
tabItem(tabName = "Dispersion",
        box(width = 12,
            title = "Dispersion", solidHeader = T, status = "primary",collapsible = TRUE,
            withSpinner(plotOutput("dispersionPlot",height = 650))),
        column(width= 4,
               downloadButton("downloadDispersion",'Download plot',class = "btn-warning")
        )
),
### MA plot ----
### On this page we find MA plot and it parameters after running DESeq
### We find :
###     - a box() which countain :
###         - sliderInput of P.value which set p.val of ma.plot function
###         - a tableOutput() of number.DE.gene function
###     - a box() which countain :
###         - return of ma.plot
###     - a downloadButton to download  the generate plot
tabItem(tabName = "MAplot",
        box(width = 12,
            title = "MA plot", solidHeader = T, status = "primary",collapsible = TRUE,
            sliderInput("pvalueMAplot", "Choose your pvalue", min=0, max=1, value=0.05),
            tableOutput("numberDEgenes")
        ),
        box(solidHeader = F, status = "primary",width = 12,
            withSpinner(plotOutput("MAplot",height = 650))),
        column(width= 4,
               downloadButton("downloadMaplot",'Download plot',class = "btn-warning"))
),
### Volcano plot ----
### On this page we find MA plot and it parameters after running DESeq
### We find :
###     - a box() which countain :
###         - sliderInput of P.value which set p.val of volcano.plot function
###         - a checkbox Yes/No if there an annotation
###           - if check Yes uiOutput("sliderFoldVolcano") and uiOutput("SliderLogVolcanon") appear
###     - a box() which countain :
###         - return of volcano.plot
###     - a downloadButton to download  the generate plot
tabItem(tabName = "Volcanoplot",
        fluidPage(
          box(width = 12,
              title = "Volcano plot", solidHeader = T, status = "primary",collapsible = TRUE,
              checkboxInput("annotationVolcano","Do you have an annotation file ?",value=FALSE),
              sliderInput("pvalueVolcano", "Choose your pvalue", min=0, max=1, value=0.05),
              uiOutput("SliderFoldVolcano"),
              uiOutput("SliderLogVolcano")
          ),
          box( solidHeader = F, status = "primary",width = 12,
               withSpinner(plotOutput("volcanoPlot",height = 650))),
          column(width= 4,
                 downloadButton("downloadVolcano",'Download plot',class = "btn-warning")))
),
### distance matrix heat map ----
### On this page we find distance matrix heat map and it parameters after running DESeq
### We find :
###     - a box() which countain :
###         - a selectInput to chose transformation for distance.matrix.heatmap
###         - a actionButton to run distance.matrix.heatmap function
###     - a box() which countain :
###         - return of distance.matrix.heatmap
###     - a downloadButton to download  the generate plot
tabItem(tabName = "DistanceMatrix",
        box(width = 12,
            title = "Heat map", solidHeader = T, status = "primary",collapsible = TRUE,
            selectInput("TransformationMatrix",label= "Choose your transformation",
                        choices = c("Variance-stabilizing transformation"="vst",
                                    "Log transformation"="rld")),
            actionButton("RunMatrix","Run Heat map")),
        box(solidHeader = F, status = "primary",width = 12,
```

```
                        withSpinner(plotOutput("DistanceMatrixMap",height = 650))
                    ),
                    column(width= 4,
                           downloadButton("downloadDistanceMatrix",'Download plot',class = "btn-warning")
                    )
            ),
            ### Gene expression heatmap ----
            ### On this page we find gene expression heatmap and it parameters after running DESeq
            ### We find :
            ###     - a box() which countain :
            ###         - a selectInput to chose transformation for gene.expression.heatmap
            ###         - a selectInput to chose condition for gene.expression.heatmap
            ###         - a annotation checkboxInput() for is.Anno of gene.expressions.heatmap function
            ###         - a actionButton to run gene.expression.heatmap function
            ###         - a sliderInput to set the number of genes to display in gene.expression.heatmap
            ###     - a box() which countain :
            ###         - return of distance.matrix.heatmap
            ###     - a downloadButton to download  the generate plot
            tabItem(tabName = "Heatmap",
                    waiter::use_waiter(),
                    box(width = 12,
                        title = "Heat map", solidHeader = T, status = "primary",collapsible = TRUE,
                        column(width=6, selectInput("TransformationHeatmap",label= "Choose your transformation",
                                                    choices = c("Variance-stabilizing transformation"="vst",
                                                                "Log transformation"="rld")),
                               checkboxInput("annotationHeatmap","Do you have a Annotation file ?",value=FALSE)
                        ),
                        column(width=6,
                               selectInput("conditionHeatmap","Choose your condition for Heat map ?",
                                           choices = c()),
                               actionButton("RunHeatmap","Run Heat map")),
                        column(width=12,
                               sliderInput("nbGenes",label="Choose the number of genes you want to display",
                                           min = 0,
                                           max = 200, value = c(0, 60)))),
                    box(solidHeader = F, status = "primary",width = 12,
                        withSpinner(plotOutput("Heatmap", height = 1000, width = 1000))
                    ),
                    column(width= 4,
                           downloadButton("downloadHeatmap",'Download plot',class = "btn-warning")
                    ))
          )
        )
      )
    )
  ),
  ### footer settings ----
  tags$footer(
    wellPanel(
      HTML('
           <p align="center" width="4">Developed by
<a href="https://www.linkedin.com/in/david-gallien-2096b9193/" target="_blank">David Gallien</a> and
<a href="https://www.linkedin.com/in/gabin-coudray-a1941913b/" target="_blank">Gabin Coudray</a>. </p>
           <p align="center" width="4">First year of
<a href="http://bioinfo-rennes.fr/" target="_blank">Bioinformatics Master<span>&#39;</span>s degree</a>
in Rennes. </p>
           <p align="center" width="4">
<a href="https://www.univ-rennes1.fr/" target="_blank">University of Rennes 1.</a> </p>'
      ),
      style = "position:relative;
               width:100%;
               background-color: #2d3741;"))
)
```

2.2 Serveur

```
### Server  ----
server <- function(input, output,session) {
  ### Create dds object with no values
  dds <- reactiveValues()
  ### Increase the authorized size for upload ----
  options(shiny.maxRequestSize=30*1024^2)

  ### Display a count table example in the introduction
  output$countExample <- renderTable({
    countex <- read.csv("countexample.csv",sep=",")
    countex
  })

  ### Display a metadata table example in the introduction
  output$metadataExample <- renderTable({
    metaex <- read.csv("metadataexample.csv",sep=",")
    metaex
  }, na = "")

  ### Display an annotation table example in the introduction
  output$annoExample <- renderTable({
    annoex <- read.csv("annoexample.csv",sep=",")
    annoex
  })

  ### Import the count file ----
  ### csv, tsv or txt files separated by comma, tab or semi-colon
  count_table <- reactive({
    req(input$CountDataTable)
    countTable <- read.csv(input$CountDataTable$datapath, sep = input$separator_Count)
  })

  ### Display the count file ----
  output$CountReadTable <- DT::renderDataTable(count_table(), options = list(pageLength = 20,
                           autoWidth = FALSE,scrollX = TRUE, scrollY = '300px'))

  ### Import the metadata file ----
  ### csv, tsv or txt files separated by comma, tab or semi-colon
  metadata <- reactive({
    req(input$MetadataFile)
    meta_table <- read.csv(input$MetadataFile$datapath, sep = input$separator_Metadata, row.names=NULL)
  })
  ### Display the metadata file ----
  output$MetaTable <- DT::renderDataTable(metadata(),options = list(pageLength = 20, autoWidth = FALSE,
                                                                    scrollX = TRUE, scrollY = '300px'))

  ### Design condition for DESeq2 ----
  ### Corresponds to the columns of the metadata table
  observeEvent(input$MetadataFile,{
    updateTextInput(session,"DesignDESeq2", value = paste("~ ",paste(colnames(metadata()), collapse=" + ")))
  })

  ### Check if the user has annotation file to upload
  observeEvent(input$CheckAnnotation, {
    if(input$CheckAnnotation == TRUE){
      updateTabsetPanel(session, "paramsAnno", selected = "annotation")
    }else{
      updateTabsetPanel(session, "paramsAnno", selected = "nothing")
    }
  })

  ### Import annotation file ----
  ### csv, tsv or txt files separated by comma, tab or semi-colon
  anno <- reactive({
    req(input$AnnotationFile)
    read.csv(input$AnnotationFile$datapath, sep = input$sep_Anno)
  })
  output$AnnoTable <- DT::renderDataTable(anno(),options = list(pageLength = 20, autoWidth = FALSE,
                                                                scrollX = TRUE, scrollY = '300px'))

  ### Running DESeq2 clicking on the button  ----
  observeEvent(input$RunDESeq2,{
    req(input$RunDESeq2)
    ### Waiting screen
    waiter <- waiter::Waiter$new(html = spin_ball())
    waiter$show()
```

```
  ### DESeq2 process
  dds$dds <- DESeqDataSetFromMatrix(count_table(),colData=metadata(),design=as.formula(input$DesignDESeq2),
                                    tidy=TRUE)
  dds$DESeq2 <- DESeq(dds$dds)
  dds$results <- results(dds$DESeq2,tidy=TRUE)

  ### Display success message after running DESeq2
  output$SuccessMessage <- renderUI({
    box(width = 12, solidHeader = F,
        HTML("<center><h3>DESeq2 workflow successfully completed</h3></center>"))
  })

  ### Samples choices for count distribution
  updateSelectInput(session,"sample",choices = metadata()[,1])

  ### Genes choices for count by gene
  updateSelectizeInput(session,"gene",choices = count_table()[,1], server = TRUE)

  ### Factor choices for PCA
  updateSelectInput(session,"conditionpca",choices = colnames(metadata()))

  ### Factor choices heatmap
  updateSelectInput(session,"conditionHeatmap",choices = colnames(metadata()))

  ### Counts data frame normalized or not
  dds$counts_dds <-as.data.frame(counts(dds$DESeq2))
  dds$counts_dds_n <-as.data.frame(counts(dds$DESeq2,normalized=TRUE))
  ### Transposed counts data frame normalized or not
  dds$counts_turnup <- as.data.frame(t(dds$counts_dds))
  dds$counts_turnup_n <- as.data.frame(t(dds$counts_dds_n))

  ### End of the waiting screen
  on.exit(waiter$hide())
})

### Count distribution ----
### Normalize or not the data
normcount <- reactive({
  if(input$normalizeDistribution==TRUE){
    dds$counts_dds <-as.data.frame(counts(dds$DESeq2,normalized=TRUE))
  }
  else if(input$normalizeDistribution==FALSE){
    dds$counts_dds <-as.data.frame(counts(dds$DESeq2))
  }
})

### Display count distribution plot using count.distribution.plot() function (see function_dds.R)
distribution <- function(){count.distribution.plot(normcount(), sample=input$sample,x.min=input$axis[1],
                                                   x.max=input$axis[2],
                                                   break.width = input$breaksDistribution)}

output$CountDistributionPlot <- renderPlot({
  validate(
    need(dds$DESeq2, "Please run DESeq2")
  )
  distribution()})
### Download distribution plot in .png format
output$downloadDistribution <- downloadHandler(
  filename = function(){
    paste(input$sample,'.png',sep = '')
  },
  content = function(file){
    ggsave(file, plot = distribution(), device = "png")
  }
)

### Count by gene ----
### Normalize or not the data
normCountGene <- eventReactive(input$normalizeCountGene,{
  if(input$normalizeCountGene==TRUE){
    dds$counts_turnup_n
  }
  else if(input$normalizeCountGene==FALSE){
    dds$counts_turnup
  }
})
### Display Count by gene plot using gene.count.plot() function (see function_dds.R)
countg <- function() {
  gene.count.plot(normCountGene(), input$gene)}
```

```
output$CountGenePlot <- renderPlot({
  validate(
    need(dds$DESeq2, "Please run DESeq2")
  )
  countg()})
### Download Counts by gene plot in .png format
output$downloadCountgene <- downloadHandler(
  filename = "CountByGene.png",
  content = function(file){
    ggsave(file, plot = countg(), device = "png")
  }
)

### Depth ----
### Normalize or not the data
normdepth <- eventReactive(input$normalizeDepth,{
  if(input$normalizeDepth==TRUE){
    dds$counts_dds_n <-as.data.frame(counts(dds$DESeq2,normalized=TRUE))
  }
  else if(input$normalizeDepth==FALSE){
    dds$counts_dds <-as.data.frame(counts(dds$DESeq2))
  }
})
### Display depth plot using depth.plot() function from function_dds.R
depthFunction <- function(){
  depth.plot(normdepth(),break.width= input$breaksDepth)
}
output$depth <- renderPlot({
  validate(
    need(dds$counts_dds, "Please run DESeq2")
  )
  depthFunction()
})
### Download depth file
output$downloadDepth <- downloadHandler(
  filename = "Depth.png",
  content = function(file){
    ggsave(file, plot = depthFunction(), device = "png")
  }
)

### Dispersion ----
### Display dispersion plot using dispersion() function from function_dds.R
dispersionFunction <- function(){
  dispersion(dds$DESeq2)
}
output$dispersionPlot <- renderPlot({
  validate(
    need(dds$DESeq2, "Please run DESeq2")
  )
  dispersionFunction()
})

### Download dispersion plot
output$downloadDispersion <- downloadHandler(
  filename = "Dispersion.png",
  content = function(file){
    png(file)
    dispersionFunction()
    dev.off()
  }
)

### PCA ----
### Needs to run PCA
### 2 possibles transformations : vst or rlog
observeEvent(input$runPCA,{
  if(input$TransformationPCA=="vst"){
    dds$TransformationPCA <- vst(dds$DESeq2, blind=FALSE)
  }else{
    dds$TransformationPCA <- rlogTransformation(dds$DESeq2,blind=FALSE)
  }
})

### Display PCA plot usin pca.plot() function from function_dds.R
PCAfunction <- function(){
  pca.plot(dds$TransformationPCA,input$conditionpca)

}
```

```
output$PCAplot <- renderPlot({
  withProgress(message = "Running PCA , please wait",{

    validate(
      need(dds$TransformationPCA, "Please run DESeq2 and PCA")
    )
    PCAfunction()
  })})
### Possibility to download the PCA plot
output$downloadPCA <- downloadHandler(
  filename = "PCA.png",
  content = function(file){
    ggsave(file, plot = PCAfunction(), device = "png")
  }
)

### MA plot ----
### Display MA plot using ma.plot() function from function_dds.R
MAplotFunction <- function(){
  ma.plot(dds$results,p.val=input$pvalueMAplot)
}
output$MAplot <- renderPlot({
  validate(
    need(dds$results, "Please run DESeq2")
  )
  MAplotFunction()
})
### Display a table with the number of differential expressed genes
output$numberDEgenes <- renderTable({
  number.DE.gene(dds$results,input$pvalueMAplot)
})
### Download MAplot in .png format
output$downloadMaplot <- downloadHandler(
  filename = "Maplot.png",
  content = function(file){
    ggsave(file, plot = MAplotFunction(), device = "png")
  }
)

### Volcano plot ----
### Display parameters for volcano
### If there is an annotation file, display sliders to choose parameters
observeEvent(input$annotationVolcano,{
  if(input$annotationVolcano== TRUE){
    output$SliderFoldVolcano <- renderUI({
      sliderInput("sliderfold", "Choose your fold", min=-20, max=20, value=c(-6,6))
    })
    output$SliderLogVolcano <- renderUI({
      sliderInput("sliderlog", "Choose your log10", min=0, max=300, value=30)
    })
  }else{
    output$SliderFoldVolcano <- renderUI({})
    output$SliderLogVolcano <- renderUI({})
  }
})

### Display volcano plot using volcano.plot() function from function_dds.R
VolcanoplotFunction <- function(){
  volcano.plot(dds$results,is.anno = input$annotationVolcano, anno = anno() ,p.val=input$pvalueVolcano,
               minlogF=input$sliderfold[1], maxlogF=input$sliderfold[2], minlogP=input$sliderlog,
               count.tb=colnames(count_table()))
}
output$volcanoPlot <- renderPlot({
  validate(
    need(dds$results, "Please run DESeq2")
  )
  VolcanoplotFunction()
})
### Download Volcano plot in .png format
output$downloadVolcano <- downloadHandler(
  filename = "Volcanoplot.png",
  content = function(file){
    ggsave(file, plot = VolcanoplotFunction(), device = "png")
  }
)
```

```
### Distance matrix heatmap ----
### Chose vst or rlog transformation
observeEvent(input$RunMatrix,{
  if(input$TransformationMatrix=="vst"){
    dds$TransformationMatrix <- vst(dds$DESeq2, blind=FALSE)
  }else{
    dds$TransformationMatrix <- rlogTransformation(dds$DESeq2,blind=FALSE)
  }
})
### Display distance matrix using the fonction distance.matrix.heatmap() from function_dds.R
distanceCluster <- function(){
  distance.matrix.heatmap(dds$TransformationMatrix)
}
output$DistanceMatrixMap <- renderPlot({
  withProgress(message = "Running heatmap , please wait",{
    validate(
      need(dds$TransformationMatrix, "Please run DESeq2 and Heat map")
    )
    distanceCluster()
  })})
### Download distance matrix in .png format
output$downloadDistanceMatrix <- downloadHandler(
  filename = "DistanceMatrix.png",
  content = function(file){
    png(file)
    distanceCluster()
    dev.off()
  }
)

### Gene expression heatmap ----
### Chose vst or rlog transformation
observeEvent(input$RunHeatmap,{
  if(input$TransformationHeatmap=="vst"){
    dds$TransformationHeatmap <- vst(dds$DESeq2, blind=FALSE)
  }else{
    dds$TransformationHeatmap <- rlogTransformation(dds$DESeq2,blind=FALSE)
  }
})

### Display heatmap using gene.expression.heatmap() function from function_dds.R
heatmapCluster <- function() {
  input$RunHeatmap
  gene.expression.heatmap(dds$results,dds$TransformationHeatmap,is.anno = input$annotationHeatmap,
                          metadata=metadata(),condition = input$conditionHeatmap,
                          count.tb=colnames(count_table()),min=input$nbGenes[1],max=input$nbGenes[2],
                          anno=anno())
}
output$Heatmap <- renderPlot({
  validate(
    need(dds$TransformationHeatmap, "Please run DESeq2 and Heat map")
  )
  heatmapCluster()
})
### Download heatmap in .png format
output$downloadHeatmap <- downloadHandler(
  filename = "Heatmap.png",
  content = function(file){
    png(file)
    heatmapCluster()
    dev.off()
  }
)

### Theme ----
### Choice between dark or light theme with a switcher button
observeEvent(input$theme,{
  if(input$theme==TRUE){
    output$themes <- renderUI({
      shinyDashboardThemes("grey_dark")
    })
  }else{
    output$themes <-renderUI({
      shinyDashboardThemes("grey_light")
    })
  }
})
```

```
### "Input count table" menu in the sidebar
### Change the icon with a check icon when the counts file is imported
menuCount <- reactive({
  if(is.null(input$CountDataTable)==TRUE){
    menuSubItem(text = "1.1 Input count table", tabName = "CountData")

  }else{
    menuSubItem(text = "1.1 Input count table", tabName = "CountData", icon = icon("far fa-check-square"))
  }
})
output$CountTable <- renderMenu({
  menuCount()
})

### "Input metadata table" menu in the sidebar
### Change the icon with a check icon when the metadata file is imported
menuMetadata <- reactive({
  if(is.null(input$MetadataFile)==TRUE){
    menuSubItem(text = "1.2 Input metadata table", tabName = "Metadata")

  }else{
    menuSubItem(text = "1.2 Input metadata table", tabName = "Metadata", icon = icon("far fa-check-square"))
  }
})
output$MetadataTable <- renderMenu({
  menuMetadata()
})

### "Input annotation file" menu in the sidebar
### Change the icon with a check icon when the annotation file is imported
menuAnnotation <- reactive({
  if(is.null(input$AnnotationFile)==TRUE){
    menuSubItem(text = "1.3 Input annotation file", tabName = "Annotation")

  }else{
    menuSubItem(text = "1.3 Input annotation file", tabName = "Annotation",
                icon = icon("far fa-check-square"))
  }
})
output$AnnotationTable <- renderMenu({
  menuAnnotation()
})

### Display "Results" menu when DESeq2 is successfully run
### Put a check icon for the menu where the plots are already display
### The others (PCA and both heatmap) needs to be run
observeEvent(input$RunDESeq2,{
  output$menuResults <- renderMenu({  menuItem(text = "3 Results", tabName = "deseq2", icon = icon("poll"),
                                               startExpanded = TRUE,
                                      menuSubItem("Count distribution",tabName = "Count_Distribution",
                                                  icon = icon("far fa-check-square")),
                                      menuSubItem("Count by gene", tabName = "Count_Gene",
                                                  icon = icon("far fa-check-square")),
                                      menuSubItem("Depth of sample",tabName = "Depth",
                                                  icon = icon("far fa-check-square")),
                                      menuSubItem("Dispersion",tabName = "Dispersion",
                                                  icon = icon("far fa-check-square")),
                                      menuPCA(),
                                      menuSubItem("MA Plot",tabName = "MAplot",
                                                  icon = icon("far fa-check-square")),
                                      menuSubItem("Volcano Plot",tabName = "Volcanoplot",
                                                  icon = icon("far fa-check-square")),
                                      menuDistanceMatrix(),
                                      menuHeatmap()
  )
  })
})

### Menu for PCA menu in sidebar
### Change the icon with check icon when pca is run successfully
menuPCA <- reactive({
  if(input$runPCA){
    menuSubItem("PCA",tabName = "pca",icon = icon("far fa-check-square"))
  }else{
    menuSubItem("PCA",tabName = "pca")
  }
})
```

```
  ### Menu for Distance matrix in the sidebar
  ### Change the icon with check icon when heatmap is run successfully
  menuDistanceMatrix <- reactive({
    if(input$RunMatrix){
      menuSubItem("Distance matrix",tabName = "DistanceMatrix",icon = icon("far fa-check-square"))
    }else{
      menuSubItem("Distance matrix",tabName = "DistanceMatrix")
    }
  })

  ### Menu for heatmap in the sidebar
  ### Change the icon with check icon when heatmap is run successfully
  menuHeatmap <- reactive({
    if(input$RunHeatmap){
      menuSubItem("Heatmap",tabName = "Heatmap", icon = icon("far fa-check-square"))
    }else{
      menuSubItem("Heatmap",tabName = "Heatmap")
    }
  })
}
```

2.3 Fonctions

```
library(DESeq2)
library(Biobase)
library(gplots)
library(RColorBrewer)
library(shiny)
library(tidyverse)
library(NMF)
library(ggrepel)


### Preamble ----
### All of this function can be use after running a DESeq2 workflow.
### You need a DESeq2 dataset to be able to run it.
###   - DESeq2 : count table after run counts() on yout DESeq2 dataset
###      - depth()
###      - count_distribution()
###      - plotcount()
###   - DESeq2 : results object after run DESeq() on your DESeq2 dataset and results() on
###              DESeq() object
###      - maplot()
###      - volcanoplot()
###      - number_of_DE
###   - DESeq2 : after run DESeq() on your DESeq2 datasetn and do vst() or rlogtransformation() on
###              DESeq() object
###      - pca()
###      - heatmap()
###      - clustering_heatmap()
###      - dispersion()

### Depth plot ----
### depth return a barplot of depth sample with ggplot library
### depth need two argument
###   - dds which is a count table of a RNAseq experience which have in column : sample, and in row :  gene
###   - breaksize which is width of the bar
depth.plot <- function(dds.count,break.width=1){
  depth <- as.data.frame(colSums(dds.count))
  depth$Sample <- row.names(depth)

  return(ggplot(depth, aes( x=Sample ,y=depth[,1]))+
           geom_bar(stat="identity",fill=brewer.pal(n=length(depth$Sample),name="YlGn"),width = break.width)+
           labs(title = "Depth of each sample", x="Samples", y="Depth")+theme_bw()+
           theme(plot.title = element_text(face = "bold", size= 18)) +
           theme(axis.title.x = element_text(size=14)) +
           theme(axis.title.y = element_text(size=14)))
}


### Count distribution plot ----
### count_distribution return an histogram of count values distribution in the log(count+1)
### (to facilitate vizualisation) format for one sample
### count_distribution need five arguments
###   - dds which is a count table of an RNAseq experience which have in column : sample, and in row :  gene
###   - breaksize which is width of histogram bar
###   - sample for which we display the count distribution
###   - min which is the min of x axis
###   - max which is the max of x axis

count.distribution.plot <- function(dds.count, sample,x.min=0,x.max=14,break.width=1){

  return(ggplot(data=dds.count, aes(log(dds.count[,sample]+1))) +
           geom_histogram(breaks=seq(x.min,x.max,break.width),position="identity",alpha=0.5,fill="darkcyan",
                          color="dodgerblue1")+
           theme_classic() +
           labs(title=sample,
                x="Counts values (number of reads by gene) in log(count+1)",
                y="Counts frequencies") +
           theme(plot.title = element_text(face = "bold", size= 18)) +
           theme(axis.title.x = element_text(size=14)) +
           theme(axis.title.y = element_text(size=14))
  )
}
```

```
### Dispersion plot ----
### dispersion return plot obtained by DESeq2::plotDispEsts() function of DESeq2 package
### dispersion just need one argument : a DESeq() object
dispersion <- function(dds){
  DESeq2::plotDispEsts(dds, main= "Relationship between dispersion and counts means")
}


### nummber of differemtial express gene ----
### number.DE.gene return a table of DE results with number of TRUE,FALSE and NA in function of pvalue
### number.DE.gene need two arguments
###   - dds.result which is a results table obtain by function results() on a DESeq() object
###   - p.val which is pvalue accept un DE analysis, padje is set at 0.05

number.DE.gene <- function(dds.result,p.val = 0.05){
  tb.DE <- as.data.frame(table(dds.result$padj <= p.val ,useNA="always"))
  colnames(tb.DE) = c("DE","Genes")
  return(tb.DE)
}


### Plotcount ----
### plotcount return a point plot of read count for one gene in each sample
### plotcount need two argument
###   - dds.count which is a count table of a RNAseq experience which have in column : sample, and in row : gene
###   - gene which is gene which we want to display read count for each sample

gene.count.plot <- function(dds.count,gene){
  dds1 <- dds.count
  dds1[,"name"] <- row.names(dds1)
  return(
    ggplot(dds1, aes(x=dds1[,"name"], y=dds1[,gene])) +
      geom_point(size=4,aes(colour=factor(name))) +
      geom_segment(aes(x=dds1[,"name"], xend=dds1[,"name"], y=0, yend=dds1[,gene]),linetype="dotdash")+
      theme(axis.text.x = element_blank() )+
      labs(title=paste("Count of",gene,  "for each sample"),x="Samples",y="Counts")+
      guides(color= guide_legend(title = "Sample", override.aes = list(size=5))) +
      theme(plot.title = element_text(face = "bold", size= 18)) +
      theme(axis.title.x = element_text(size=14)) +
      theme(axis.title.y = element_text(size=14)) +
      theme(legend.text=element_text(size=13)) +
      theme(legend.title=element_blank())
  )
}


### Maplot ----
### maplot return a MA plot of DE
### maplot need two arguments
###   - dds.results which is a results table obtain by function results() on a DESeq() object
###   - p.val which is pvalue accept un DE analysis, padje is set at 0.05
ma.plot <- function(dds.results,p.val=0.05){
  dds.res <- dds.results %>% mutate(sig=padj<p.val)
  return(ggplot(dds.res, aes(x = baseMean, y = log2FoldChange, col = sig)) +
           geom_point() +
           scale_x_log10() +
           geom_hline(yintercept = 0, linetype = "dashed",color = "black") +
           theme_bw() +
           scale_colour_discrete(name="",labels=c("Not significative", "Significative", "NA"))+
           guides(color = guide_legend(override.aes = list(size=5))) +
           theme(legend.text=element_text(size=13))+
           theme(axis.title.x = element_text(size=14)) +
           theme(axis.title.y = element_text(size=14)))
}
```

```
### VolcanonPlot ----
### volcano.plot generate a volcano plot of DE
### volcano.plot need 8 arguments
###   - dds.results which is a results table obtain by function results() on a DESeq() object
###   - is.anno if there is an annotation file
###   - anno an annotation
###   - p.val which is pvalue accept un DE analysis, padje is set at 0.05
###   - count.tb which is a count table of a RNAseq experience which have in column : sample, and in row : gene
###   - maxlogF : max Fold change in x axis which we set gene ID on geom point
###   - minlogF : max Fold change in x axis which we set gene ID on geom point
###   - minlogP : min Log10 in y which we set gene ID on geom point
volcano.plot <-function(dds.results, is.anno=FALSE,anno,p.val=0.05,maxlogF=6,minlogF=0,minlogP=30,count.tb){
  if(is.anno == TRUE){
    dds.res <- dds.results %>% mutate(sig=padj<p.val) %>%  arrange(padj) %>%
      inner_join(anno,by=c("row"=count.tb[1]))
    return(ggplot(dds.res, aes(x=log2FoldChange, y=-log10(pvalue), col=sig)) +
             geom_point() +
             ggtitle("Volcano plot labelling top significant genes") +
             geom_text_repel(data = subset(dds.res,
                                           (-log10(pvalue) > minlogP |
                                              log2FoldChange > maxlogF |
                                              log2FoldChange < minlogF)),
                             aes(label = symbol),
                             size = 4,
                             box.padding = unit(0.35, "lines"),
                             point.padding = unit(0.3, "lines"), color = "darkblue") +
             scale_colour_discrete(name="",
                                   labels=c("Not significative", "Significative", "NA")) +
             guides(color = guide_legend(override.aes = list(size=5))) +
             geom_vline(xintercept=0,linetype="dashed", color = "red")+
             theme(legend.text=element_text(size=13)) +
             theme(axis.title.x = element_text(size=14)) +
             theme(axis.title.y = element_text(size=14)))
  }else{
    dds.res <- dds.results %>% mutate(sig=padj<p.val) %>%  arrange(padj)
    return(ggplot(dds.res, aes(x=log2FoldChange, y=-log10(pvalue), col=sig)) +
             geom_point()+
             scale_colour_discrete(name="",
                                   labels=c("Not significative", "Significative", "NA")) +
             geom_vline(xintercept=0,linetype="dashed", color = "red") +
             guides(colour = guide_legend(override.aes = list(size = 5))) +
             theme(legend.text=element_text(size=13))+
             theme(axis.title.x = element_text(size=14)) +
             theme(axis.title.y = element_text(size=14)))
  }
}

### PCA ----
### pca.plot generate a pca plot
### pca.plot need 2 arguments
###   - dds.resTransf which is an DESeq2 transformate object with vst() or rLogtransformation()
pca.plot <- function(dds.resTransf,intgroup){
  return(
    plotPCA(dds.resTransf, intgroup=intgroup) +
      theme(axis.title.x = element_text(size=14)) +
      theme(axis.title.y = element_text(size=14)) +
      scale_colour_discrete(name="")+
      guides(colour = guide_legend(override.aes = list(size = 5))) +
      theme(legend.text=element_text(size=13))+
      geom_text_repel(
        aes(label = colnames(dds.resTransf)),
        size = 3,
        box.padding = unit(0.35, "lines"),
        point.padding = unit(0.3, "lines"), color = "darkblue")
  )
}

### Distance matrix ----
### distance.matrix.heatmap generate a heatmap of dist between different sample
### distance.matrix.heatmap need just one argument
###   - dds.resTransf which is an DESeq2 transformate object with vst() or rLogtransformation()
distance.matrix.heatmap <- function(dds.resTransf){
  dists <- dist(t(assay(dds.resTransf)))
  mat <- as.matrix(dists)
  hmcol=colorRampPalette(brewer.pal(9,"GnBu"))(100)
  return(heatmap.2(mat,trace="none",col = rev(hmcol),margin=c(13,13)))
}
```

```
### Heatmap ----
### gene.expression.heatmap generate a gene expression distance heatmap
### gene.expression.heatmap need 10 argument
###   - dds.results which is a results table obtain by function results() on a DESeq() object
###   - is.anno if there is an annotation file
###   - anno an annotation
###   - p.val which is pvalue accept un DE analysis, padje is set at 0.05
###   - count.tb which is a count table of a RNAseq experience which have in column : sample, and in row : gene
###   - metadata which a design table of an RNA-seq experience
###   - dds.resTransf which is an DESeq2 transformate object with vst() or rLogtransformation()
###   - condition : design formula of RNA-seq experience
###   - min : num of row in results table starting from the top
###   - max : num of row in results table starting from the top
gene.expression.heatmap <- function(dds.results, dds.resTransf,is.anno=FALSE,anno,p.val=0.05,metadata,
                                    condition,count.tb,min,max){
  res <- tbl_df(dds.results)
  if(is.anno==TRUE){
    res <- res %>%
      arrange(padj) %>%
      inner_join(anno,by=c("row"=count.tb[1])) %>%
      filter(padj<p.val)

    NMF::aheatmap(assay(dds.resTransf)[arrange(res, padj, pvalue)$row[min:max],],
                  labRow=arrange(res, padj, pvalue)$symbol[min:max],
                  scale="row", distfun="pearson",
                  annCol=dplyr::select(metadata, condition),
                  col=c("green","black","black","red"))

  }else{
    res <- res %>%
      arrange(padj) %>% filter(padj<p.val)

    NMF::aheatmap(assay(dds.resTransf)[arrange(res, padj, pvalue)$row[min:max],],
                  labRow=arrange(res, padj, pvalue)$symbol[min:max],
                  scale="row", distfun="pearson",
                  annCol=dplyr::select(metadata, condition),
                  col=c("green","black","black","red"))
  }
}
```