

Amélioration de l'accès aux résultats biologiques : Analyse d'expression différentielle et Application Shiny

Master 1 Bioinformatique
Université de Rennes 1
2019 - 2020



David GALLIEN & Gabin COUDRAY

Résumé

L'accessibilité à l'analyse des données et à leurs résultats issus de technologies générant des données en masse est un enjeu essentiel pour la recherche de demain. En tant que futurs bioinformaticiens notre rôle est de le permettre. Il nous faut donc acquérir les compétences nécessaires à la réalisation d'outils permettant cet accès facilité. Un grand nombre de ces analyses se déroule sous R, c'est pourquoi nous avons axé notre projet sous cet environnement.

Toutefois pour avoir des résultats à analyser, il faut une analyse et des données. Nous avons donc porté notre choix sur l'analyse de données issues de séquençages de l'ARN et plus spécialement sur l'analyse d'expression différentielle. Dans un premier temps un jeu de données d'une analyse RNA-seq 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. a été utilisé afin de prendre en main et d'apprendre l'utilisation du package DESeq2 développé par Bioconductor. Ce package est un package sous R permettant de réaliser une analyse d'expression différentielle de données RNA-seq.

Une fois ce package compris et maîtrisé, nous nous sommes attaqués à l'accessibilité à ce type d'analyses avec DESeq2. Pour cela nous avons décidé de créer une application permettant d'exécuter simplement le workflow DESeq2. Afin de réaliser cette application nous avons utilisé le package Shiny qui permet de créer une application web interactive en associant des fonctionnalités de R et des fonctionnalités utilisant les langages HTML, JavaScript ou encore CSS.

mots clés : Séquençage ARN, DESeq2, NGS, Visualisation, Exploration

Abstract

Accessibility to data analysis and results from technologies generating big data is a key issue for tomorrow's research. As future bioinformaticians our role is to enable this. We therefore need to acquire the skills to create tools that allow this facilitated access. Many of these analyses take place under R, which is why we have focused our project under this environment. However, in order to have results to analyze, analysis and data are needed.

We have therefore focused on the analysis of data from RNA-seq experiments and more specifically on differential expression analysis. As a first step, a dataset from an RNA-seq analysis of the study *RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells from Himes BE, Jiang X, Wagner P, and al.* was used in order to learn how to handle and use the DESeq2 package developed by Bioconductor. This package is a package under R allowing to perform a differential expression analysis of RNA-seq data.

Once this package was understood and mastered, we tackled the accessibility of this type of analysis with DESeq2. To do this we decided to create an application to simply run the DESeq2 workflow. In order to realize this application we used the Shiny package which allows to create an interactive web application by associating R functionalities, while associating functionalities using HTML, JavaScript or CSS.

keywords : RNA sequencing, DESeq2, NGS, Graphic visualization, Exploration

ENGAGEMENT DE NON PLAGIAT

Je, soussigné (e) **Gabin COUDRAY**
Etudiant (e) en **Master 1 Bioinformatique**

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



ENGAGEMENT DE NON PLAGIAT

Je, soussigné (e) **David Gallien**
Etudiant (e) en **Master 1 Bioinformatique**

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



Table des matières

Introduction	8
Contexte	8
Objectifs	10
Matériels et méthodes	11
Analyse de l'expression différentielle de données RNA-seq	11
Jeu de données	11
DESeq2	11
Analyse de l'expression différentielle sous DESeq2	13
Création de l'application	15
Shiny	15
Shinydashboard	15
Rédaction du rapport	16
Markdown	16
Résultats	17
Analyse de l'expression différentielle de données RNA-seq	17
Application RShiny	20
Partie 1 : Informations	20
Partie 2 : Importation des données	21
Partie 3 : DESeq2	22
Partie 4 : Résultats	23
Customisation	25
Discussion	27
Conclusion et perspectives	28
Bibliographie	29
Annexe 1 : Bilan personnel	30

Annexe 2 : Script R pour l’analyse RNAseq via DESeq2 32

Annexe 3 : Scripts R pour l’application Shiny 35

3.1 Interface utilisateur 35

3.2 Serveur 43

3.3 Fonctions 51

Introduction

Contexte

De nos jours, la plupart des chercheurs n'a 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 eu besoin d'un support d'expérimentation. 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 séquençage 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 dans différentes conditions.

Le RNA-seq est un nouveau moyen permettant un séquençage de l'ARN plus rapide que les techniques qui existaient précédemment comme la méthode de Sanger. Le but principal du RNA-seq est d'étudier l'expression différentielle de gènes dans différentes conditions. Le séquençage de l'ARN a été cité pour la première fois en 2008, et depuis, le nombre de publications contenant des données de RNA-seq augmente d'années en années. Ce genre d'analyses utilise les technologies de séquençage de nouvelles générations (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 cellule. Ces reads sont stockés dans des fichiers au format fastQ et leur qualité est estimée 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, chaque ligne représente un alignement d'un read. Pour finir, un comptage des reads pour chaque position est réalisé 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 : Acquisition des données

Dans un premier temps, 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 séquençé grâce aux séquenceurs de nouvelles générations. 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 obtient des fichiers fastQ. Ce genre de fichier est composé de blocs de 4 lignes représentant 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 région du génome qui lui est la plus similaire. Cette étape permet d'obtenir des fichiers BAM dans lesquels chaque ligne correspond à un read. La moyenne du nombre de reads mappés sur une région est appelée la profondeur.

Etape 4 : Quantification

Le nombre de reads est un témoin de l'abondance de l'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 mapés 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 cartes 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 gènes 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 l'expression différentielle 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

Analyse de l'expression différentielle de données RNA-seq

Jeu de données

Nous allons tout d'abord procéder à une analyse de l'expression différentielle de gènes. 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, le but de cette étude est de comprendre le mécanisme du traitement dans les muscles lisses des voies respiratoires ainsi que son impact sur l'expression génétique dans différents types cellulaires en utilisant la technologie RNA-seq.

Cette expérience rassemble 8 échantillons : 4 traités avec du dexaméthasone (glucocorticoïde synthétique) et 4 échantillons contrôles sans traitement. 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.

Pour effectuer cette analyse nous allons travailler sous R et notamment sous l'IDE R-studio. Afin d'effectuer l'analyse différentielle nous allons utiliser le package *DESeq2* proposé et développé par Bioconductor.

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 développé par la plateforme Bioconductor qui met à disposition des utilisateurs différents packages et outils open source destinés à la bioinformatique et plus spécifiquement à l'analyse et l'étude de séquençages hauts débits comme le RNA-seq. 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. L'objectif est de connaître le nombre d'échecs ayant eu lieu avant le nombre de succès défini. Cette loi est utilisée car le comptage des reads n'est pas continu, on ne retrouve que des entiers non nuls et on ne peut donc 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.

Pour le bon fonctionnement du package DESeq2 Il est nécessaire d'avoir au préalable une table de comptage ainsi qu'une table contenant le design de l'expérience. La première étape consiste à créer un premier jeu de données DESeq contenant ces deux éléments. Une fois l'objet créé, il est possible d'accéder à la table de comptage ainsi qu'au design de l'expérience à l'aide de fonctions propres au package DESeq2 (annexe 2). La seconde étape consiste à normaliser les données 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 étudier.

```
### Create dds object ---
dds <- DESeqDataSetFromMatrix(counts_table,colData=airway_metadata,design = ~dex,tidy = TRUE)
# Set reference of experience, here "control"
colData(dds)$dex <- relevel(colData(dds)$dex , ref="control")
# Differential expression analysis and normalization ----
dds <- DESeq(dds)
# Extraction of DE results
res <- results(dds,tidy = TRUE)
```

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

Analyse de l'expression différentielle sous DESeq2

Avant d'analyser les résultats de l'analyse d'expression différentielle nous avons visualisé la distribution du comptage des reads pour chaque échantillon avant et après normalisation (figure 2) en utilisant la méthode du log-fréquence afin de faciliter la visualisation. En parallèle nous avons visualisé la profondeur de séquençage de chaque échantillon avant et après normalisation. Cela a permis de vérifier visuellement la bonne normalisation des données. Et enfin nous avons visualisé le comptage par gène pour chaque échantillon. Pour ces visualisations et les autres à venir dans l'analyse nous avons utilisé les packages *ggplot2* et *ggrepel* disponibles sur le CRAN.

```
# Extraction of count table
count_table_dds <- as.data.frame(counts(dds))
# Visualization of count distribution
ggplot(data=count_table_dds, aes(log(count_table_dds[, "SRR1039517"]+1))) +
  geom_histogram(breaks=seq(0,14,1), col="black", fill="grey") +
  theme_light()+labs(title="SRR1039517",
                     x="Count value (number of read by genes) in log(count+1)",
                     y="Count frequency") +
  theme_bw()
```

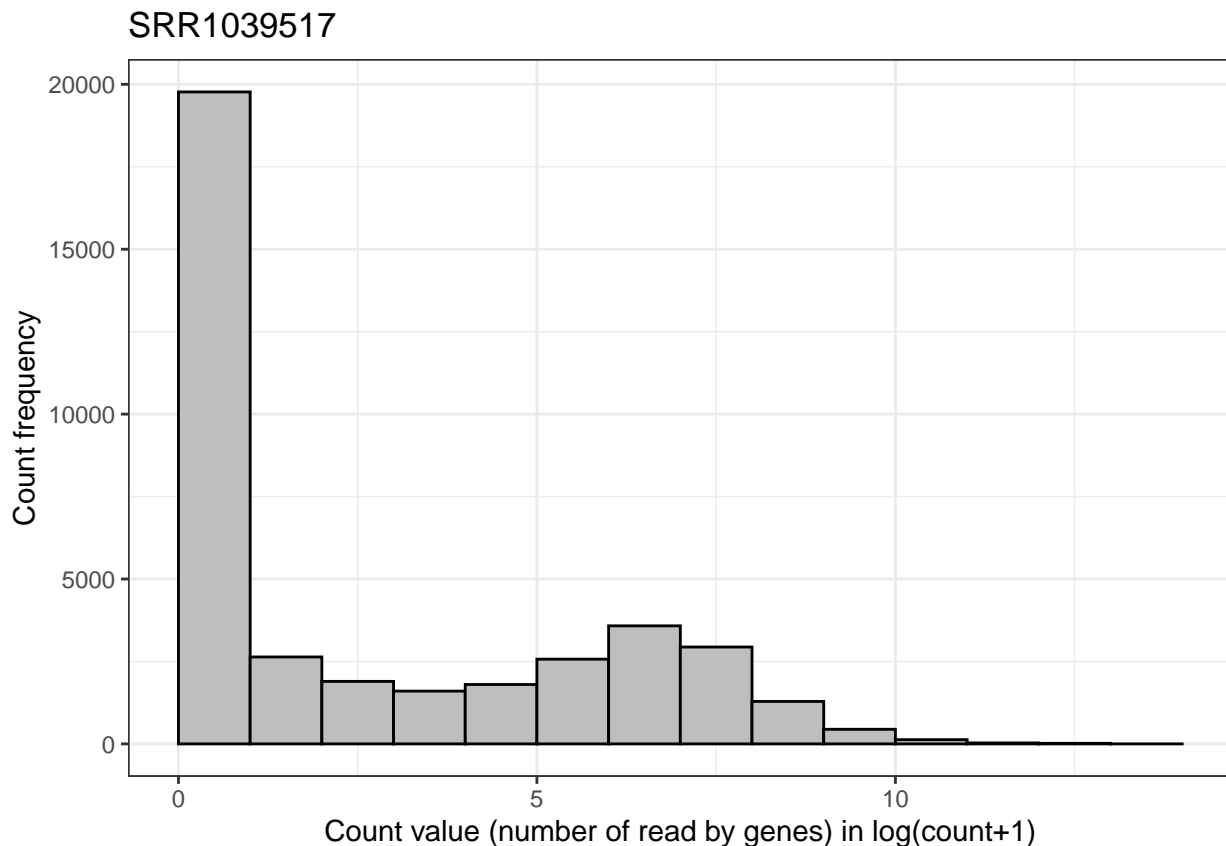


Figure 2 : Visualisation de la distribution des comptages non normalisés pour l'échantillon SRR1039517.

Suite à cette première approche nous nous sommes intéressés aux résultats propres de l'analyse de l'expression différentielle (DE). La fonction intégrée au package *DESeq2* (figure 1) nous a permis de sortir une table contenant les résultats de cette analyse. Cette table contient 7 colonnes : moyennes des échantillons, log2foldchange, erreurs standards, statistiques de tests, p-values et p-values ajustées. A partir de cette table Nous avons ressorti les gènes différentiellement exprimés au risque alpha de 0.05 à l'aide de la p.value ajusté. Le LogFoldChange nous permet de différencier les gènes sur-exprimés ou sous-exprimés par rapport à la référence ($LFC > 0$: surrégulé, $LFC < 0$: sous régulés). Afin de visualiser les résultats de l'analyse d'expression différentielle au risque alpha de 0.05 nous réalisons un MA plot ainsi qu'un volcano plot. Pour le volcano plot nous lui associerons le fichier d'annotation afin d'observer les ID des gènes les plus différentiellement exprimés directement sur le graphique ainsi que les gènes les plus sur-exprimés ou sous-exprimés.

Une fois les résultats de l'analyse de l'expression différentielle sur les gènes visualisés, nous avons analysé la différence des expressions entre les différents échantillons et non entre la condition de référence et la condition testée. Pour cela nous avons réalisé une analyse en composante principale ainsi qu'une matrice de distance représentée sous forme de heatmap. Ces 2 représentations sont faites à l'aide de la fonction PCA intégrée au package *DESeq2* et au package *gplots* couplé au package *RcolorBrewer* pour le gradient de couleur de la heatmap.

Pour ces deux visualisations nous avons utilisé la matrice de comptage transformée selon la méthode VST "Variance stabilization transformation". Enfin nous avons fini par réaliser une clustering heatmap de l'expression des 50 gènes les plus significativement différentiellement exprimés couplée au fichier d'annotation afin de visualiser les ID des gènes. Cette dernière heatmap a été réalisée à l'aide du package *NMF* pour "Non negative Matrix Factorization". La méthode utilisée pour les matrices de distances des deux heatmaps est la méthode dite de Pearson.

Création de l'application

Concernant la réalisation de l'application, nous avons préalablement créé un script contenant des fonctions générant les visualisations et résultats que nous souhaitions avoir dans l'application (annexe 3.3). Nous avons réalisé ces fonctions car pour la plus grande partie des résultats et leurs visualisations les lignes de codes correspondantes étaient imposantes ce qui aurait apporté une perte de lisibilité dans les scripts de l'application. En plus de ce script contenant les fonctions nécessaires au fonctionnement de l'application nous utilisons les packages propres à la création d'une application que sont *Shiny* et *Shinydashboard*.

Shiny

Premièrement, le package *Shiny* permet la création d'applications web interactives capables d'utiliser toutes les fonctionnalités de R. C'est un outil permettant de créer des pages web sans forcément avoir de connaissances en HTML, css ou javascript. Cependant, c'est un plus d'avoir des connaissances dans ces domaines 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'apparence de l'application (annexe 3.1).
- Le fichier sever.R contient les instruction permettant le fonctionnement de l'application (annexe 3.2).

Shinydashboard

Pour l'aspect esthétique et la facilité de navigation nous avons décidé de faire un tableau de bord à l'aide du package *Shinydashboard*. Ce package permet donc de générer un corps 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 personnalisé 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 le 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 l'expression différentielle de données RNA-seq

L'analyse de l'expression différentielle de données RNA-seq a donné lieu à plusieurs résultats que nous allons présenter dans cette partie. Premièrement, nous nous intéressons aux résultats qui concernent les gènes grâce au volcano plot et au tableau résumant l'état des gènes au sein de l'expérience. Comme on peut le voir dans le tableau de la figure 3, les résultats de l'analyse d'expression différentielle ont mis en valeur 2181 gènes différentiellement exprimés au risque alpha de 0.05 (figure 3). On remarque aussi que 1242 gènes sont sur-exprimés et 939 sont sous-exprimés par rapport à la condition de référence "control".

```
## No.DE Down.regulated Up.regulated NA.  
## 12964          939          1242 23549
```

Figure 3 : Table des résultats de l'analyse d'expression différentielle au risque alpha = 0.05.

A la vue du volcano plot ci-dessous, on remarque que les gènes les plus significativement différenciellement exprimés sont SPARCL1, PER1, ARHGEF2, et MAOA. La plupart des gènes les plus différenciellement exprimés sont sur-exprimés par rapport à la référence car ils se trouvent à droite de l'axe des ordonnées et ont donc un LFC positif. On note d'ailleurs que les deux gènes les plus sur-exprimés sont ALOX15B et AC007325.2 (figure 4)

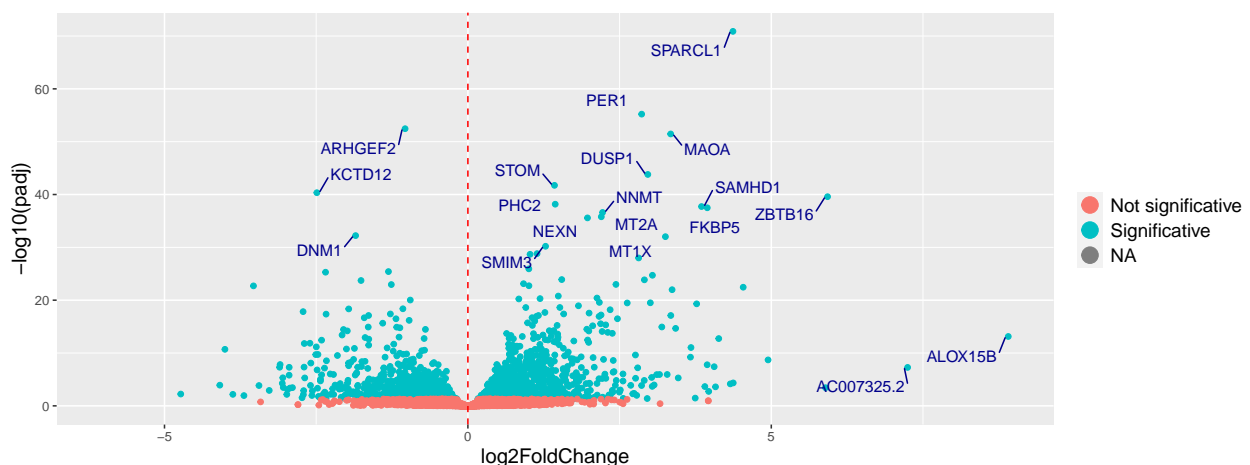


Figure 4 : Volcano plot de l'analyse d'expression différentielle. En bleu on retrouve les gènes dont l'expression diffère de la condition de référence "control" au risque alpha = 0.05 et en rouge les gènes dont l'expression ne diffère pas au risque alpha = 0.05. Les points à droite de la ligne de démarcation correspondent aux gènes sur-exprimés par rapport à la référence (LFC > 0) et les points à gauche correspondent aux gènes sous-exprimés par rapport à la référence (LFC < 0).

Entre les échantillons on remarque à l'aide de la PCA (figure 5) que deux groupes se distinguent selon la première dimension qui donne le plus gros pourcentage d'informations (32%) : un groupe contenant les échantillons traités et un groupe contenant les échantillons contrôles. Selon la seconde dimension qui a un pourcentage d'informations moins important (24%), on remarque là aussi deux groupes. L'un des groupes contient tous les échantillons sauf SRR1039517 et SRR1039516. Cela laisse penser que ces deux échantillons sont les plus éloignés des autres du fait de la différence du types cellulaires. On aurait donc 3 types cellulaires proches (N05261,N06101,N61311) et l'un plus éloignés (N08061).

On remarque d'ailleurs sur la matrice de distance obtenue avec la méthode de corrélation de pearson (figure 5) que l'échantillon SRR1039517 est le plus éloigné de ses homologues "treated" et que sa plus forte corrélation est avec son "control" associé. La matrice de distance pour les échantillons montre tout de même bien deux groupes : l'un contenant les échantillons traités et l'autre les échantillons contrôles. D'ailleurs la clustering heatmap (figure 6) sur l'expression des gènes montre cette fois bien clairement deux groupes l'un contenant les échantillons traités et l'autre les échantillons contrôles. Cette visualisation montre aussi, comme nous l'avons identifié, que les gènes les plus significativement différemment exprimés sont le plus souvent sur-exprimés dans les échantillons traités.

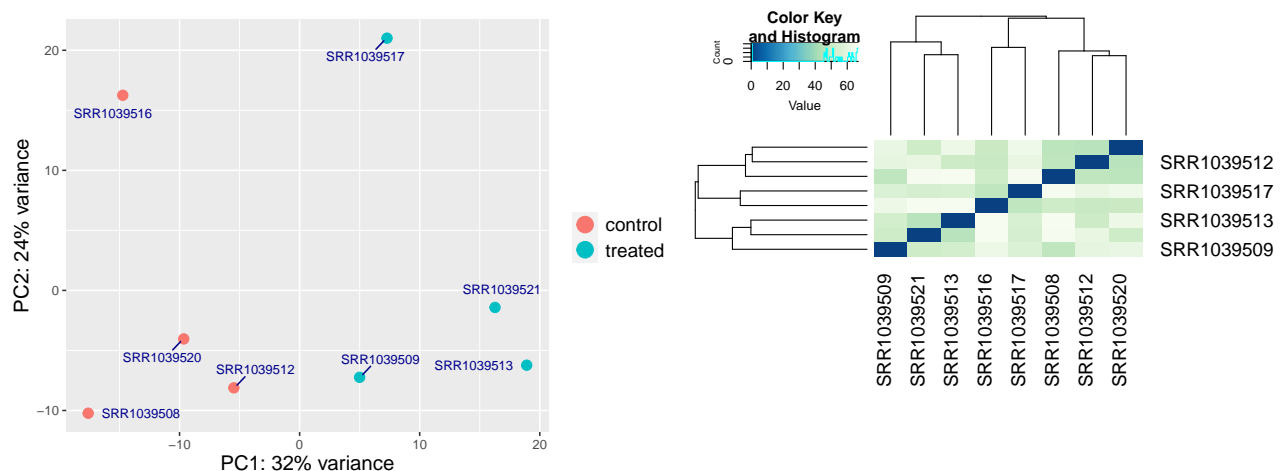


Figure 5 : Analyse en composante principale (gauche) et matrice de distance (droite) sur les 8 échantillons avec une transformation selon la méthode VST "Variance stabilization transformation". Un gradient de couleur bleu est utilisé. Plus la couleur vire au bleu plus la distance calculée est proche.

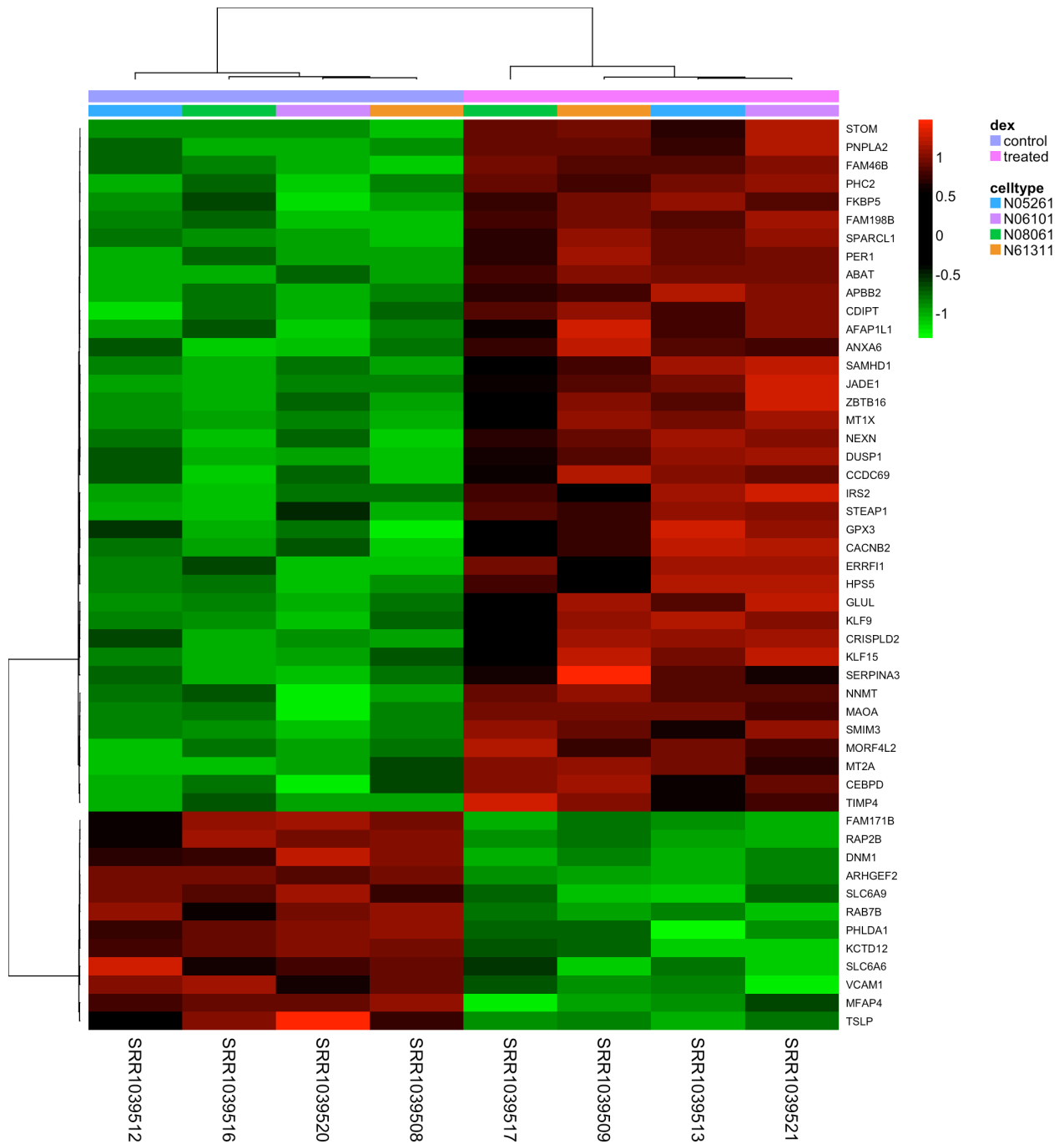


Figure 6 : Clustering Heatmap de l'expression des 50 gènes étant les plus significativement DE. Le gradient utilisé est un gradient rouge-vert. Plus l'expression est importante plus la couleur vire au rouge et inversement plus la couleur vire au vert plus l'expression est faible.

Application RShiny

Tout d’abord, l’application est disponible à l’adresse suivante :

<https://gabin-coudray.shinyapps.io/mlproject/>

Notre but était donc de mettre en place une application Shiny capable de faire l’analyse d’expression différentielle de données RNA-seq. Pour cela nous utilisons les packages *Shiny* et *Shinydashboard* afin de créer une application ergonomique et agréable à prendre en main. Pour ce faire, nous avons décidé de diviser l’application en 4 principales parties :

- Une première partie de présentation de l’application.
- Une partie pour l’import des données nécessaires.
- Une partie permettant l’utilisation de DESeq2.
- Une dernière partie avec tous les résultats.

Nous avons fait le choix de faire cette application en anglais et nous allons donc détailler chacune des parties en précisant leur mode de fonctionnement ainsi qu’en affichant certains résultats obtenus.

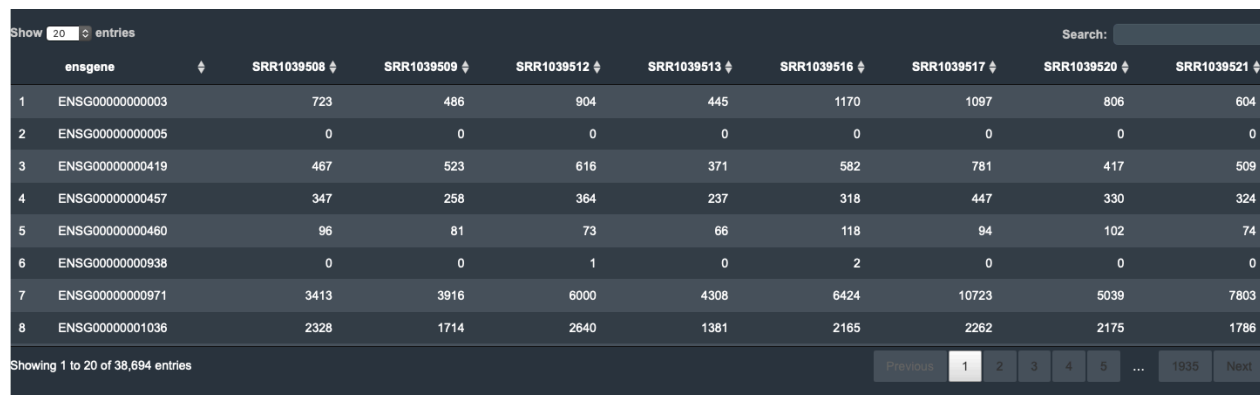
Partie 1 : Informations

La première partie sert de page d’accueil pour l’application. Elle présente rapidement le but et les fonctionnalités de cette application. Comme l’analyse de l’expression différentielle RNA-seq faite précédemment, cette application se base sur l’utilisation du package DESeq2 afin d’obtenir des résultats pouvant être interprétés à partir d’une table de comptage et d’un fichier metadata au minimum. On retrouve donc ainsi sur cette page d’accueil les types de fichiers autorisés pour permettre l’analyse ainsi que des exemples de ceux ci. Trois sortes de fichiers sont autorisés : une table de comptage de reads (count table), un fichier metadata (metadata table) et un fichier d’annotation de gènes (annotation file).

Partie 2 : Importation des données

Les 3 types de fichiers cités dans le paragraphe précédent peuvent être importés dans l'application. Cependant seulement deux sont essentiels et l'autre est optionnel. En effet, les fichiers count table et metadata table sont nécessaires au bon fonctionnement des outils du package DESeq2. Concernant le fichier d'annotation, il est facultatif donc il est demandé à l'utilisateur de préciser s'il en possède un. Ce fichier va servir à afficher le symbole des gènes si besoin pour certains résultats.

Tous les fichiers importés doivent être au format .csv, .tsv ou .txt avec pour séparation des tabulations, des virgules ou des points-virgules. Les fichiers doivent avoir des compositions spécifiques. Concernant le fichier de comptage de reads, la première colonne doit correspondre à l'identifiant du gène et toutes les autres aux comptages de reads par échantillons. Pour le fichier metadata, la première colonne doit correspondre aux échantillons de la table de comptages et une autre colonne minimum est nécessaire. Celle-ci doit être une colonne qui sépare les échantillons en 2 conditions (e.g. control et treated). Pour finir le fichier d'annotation doit répondre à une obligation : une des colonnes doit être nommée "symbol". Cette colonne correspond au symbole des gènes et est nécessaire pour pouvoir afficher ces symboles sur le volcano plot ou la clustering heatmap des gènes. Enfin, chaque fichier importé est affiché sur sa page pour permettre à l'utilisateur de parcourir les tableaux.



	ensgene	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516	SRR1039517	SRR1039520	SRR1039521
1	ENSG00000000003	723	486	904	445	1170	1097	806	604
2	ENSG00000000005	0	0	0	0	0	0	0	0
3	ENSG000000000419	467	523	616	371	582	781	417	509
4	ENSG000000000457	347	258	364	237	318	447	330	324
5	ENSG000000000460	96	81	73	66	118	94	102	74
6	ENSG000000000938	0	0	1	0	2	0	0	0
7	ENSG000000000971	3413	3916	6000	4308	6424	10723	5039	7803
8	ENSG000000001036	2328	1714	2640	1381	2165	2262	2175	1786

Figure 7 : Exemple de l'importation de la table de comptage. L'utilisateur peut parcourir les pages de 20 entités du tableaux par une recherche ou en choisissant la page.

Partie 3 : DESeq2

DESeq2 est lancé grâce à un bouton “Run DESeq2 workflow”. En appuyant sur ce bouton, un écran d’attente va s’afficher le temps que le processus se termine. Durant cette attente, la fonction `DESeqDataSetFromMatrix()` va permettre de stocker les valeurs d’entrée, les calculs intermédiaires et les résultats d’une analyse de l’expression différentielle. Afin de faire fonctionner correctement cette partie, un design est demandé lors de l’importation du fichier metadata. Ce design doit correspondre à un facteur permettant de distinguer les deux différents groupes des échantillons. Il faut donc faire le bon choix lors de l’élaboration du design et garder la bonne colonne du fichier metadata. De plus, il faut choisir la référence de l’expérience. En effet, sans possibilité de ce choix, le processus DESeq2 prend pour référence le premier facteur dans l’ordre alphabétique. Par exemple, si on a “Wild Type” et “Mutant”, c’est “Mutant” qui sera pris en référence alors que ça devrait être “Wild Type”. Si la mauvaise référence est utilisée, tous les résultats obtenus seront inversés. Avec notre jeu de données, c’est “control” qui est utilisé comme référence.

On appelle une “colonne linéaire” une colonne dont toutes les valeurs en lignes sont différentes. Une liste de sélections laisse le choix à l’utilisateur entre les colonnes du tableau correspondant aux critères nécessaires au bon déroulement de du workflow DESeq2 : les colonnes non linéaires et les colonnes dont toutes les valeurs ne sont pas égales. Avec un bon design, le DESeq2 est lancé et les résultats s’affichent après une dizaine de secondes d’attente.

Ensuite, la fonction `DESeq()` permet de faire une analyse en 3 étapes :

- Estimation des facteurs de taille
- Estimation de la dispersion
- Ajustement de la loi binomiale négative

Pour finir le processus DESeq2, les résultats sont extraits sous forme de tableau grâce à la fonction `results()`. Dans ce tableau on retrouve les moyennes des échantillons, les `log2foldchange`, les erreurs standards, les statistiques de tests, les p-values et les p-values ajustées. Ces résultats vont être importants pour permettre d’établir les graphiques et autres résultats de la section “Résultats”.

Partie 4 : Résultats

On retrouve plusieurs types de résultats sous forme de graphiques. Tous les graphiques sont réalisés grâce au package ggplot2 mais aussi grâce aux fonctions intégrées au package DESeq2 pour le graphique de dispersion et l'ACP et le package NMF pour les heatmaps. Tous ces graphiques sont directement disponibles dès la fin du processus de DESeq2 sauf 3. En effet, l'ACP et les deux heatmaps sont affichés après une seconde étape qui va permettre de lancer les fonctions capables de réaliser ces graphiques.

Le premier disponible est un graphique de la distribution du comptage de reads ("count distribution") qui montre la fréquence de reads par échantillons. Il est possible de choisir l'échantillon que nous voulons visualiser ainsi que la portée de l'axe des abscisses et la dimension des barres sur le graphique. Ensuite, on retrouve un graphique permettant de visualiser le nombre de reads par gènes et par échantillons. Il y a donc la possibilité de choisir le gène que l'on veut visualiser. Pour continuer, nous trouvons un barplot témoignant de la profondeur de séquençage pour chaque échantillon avec la possibilité de choisir la largeur des barres. Chacun de ces trois graphiques proposent une option pour visualiser les données avant et après normalisation.

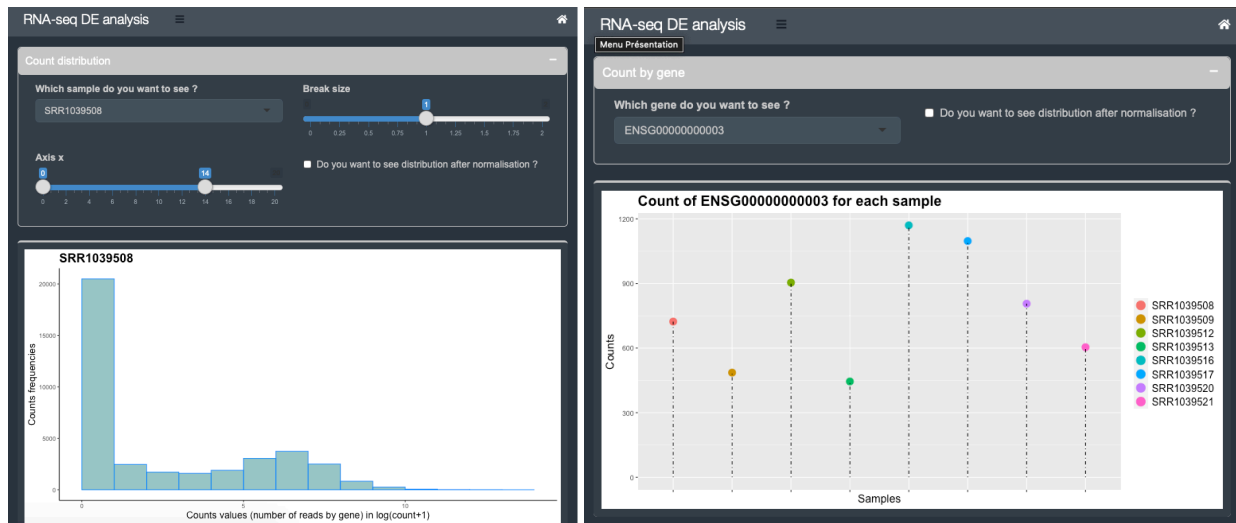


Figure 8 : Exemple de pages pour les graphiques de count distribution (gauche) et du comptage de reads par gènes (droite). On remarque les options comme la normalisation, la taille des barres ou le choix de l'échantillon ou du gène au dessus des graphiques.

La dispersion est un paramètre décrivant la déviation de la variance par rapport à la moyenne. Le graphique de dispersion permet d'avoir une idée de la dispersion des données, on peut y distinguer

une valeur intermédiaire (en bleu) entre la dispersion individuelle (en noir) et la dispersion ajustée (en rouge). Ces valeurs de dispersion sont utilisées pour le calcul des tests statistiques d'expression différentielle. Il reste alors 5 graphiques à voir dont 4 vus dans l'analyse de l'expression différentielle de données RNA-seq de la partie précédente. En effet, l'Analyse en Composante Principale et la matrice de distance sont aussi disponibles avec des options permettant de choisir le mode de transformation entre Log transformation et Variance-stabilizing transformation. Il est aussi possible de choisir le groupe pour l'ACP. De plus, la clustering heatmap d'expression de gènes est aussi visualisée avec le choix de la transformation (VTS ou LogT) ainsi que le choix de la condition pour la séparation des échantillons dans la heatmap et le nombre de gènes à afficher. Enfin, on retrouve le volcano plot. Comme pour la heatmap d'expression de gènes, le volcano plot offre la possibilité d'afficher les symboles des gènes sur le graphique pour une visualisation plus explicite (comme sur la figure 4).

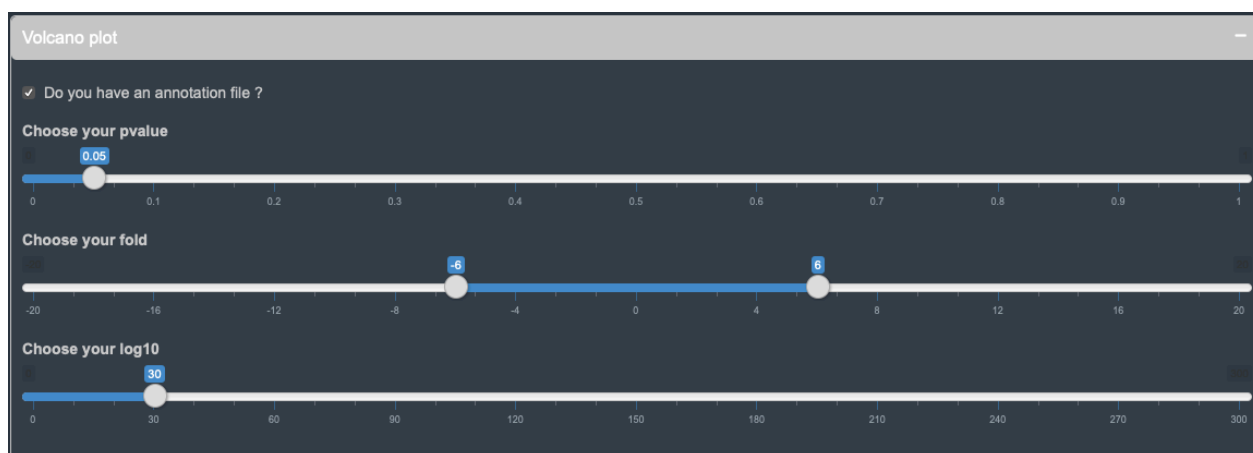


Figure 9 : Options d'affichage des symboles de gènes sur le volcano plot. "Choose your fold" permet de choisir la portée des gènes à afficher selon l'axe de abscisses afin de voir les gènes sur et sous-exprimés. "Choose your log10" permet de choisir les gènes à afficher selon l'axe des ordonnées afin de voir les gènes le plus significativement différentiellement exprimés.

Pour finir, un MA plot (figure 10) est disponible avec un tableau présentant le nombre de gènes différentiellement exprimés ainsi que les gènes sur-exprimés et sous-exprimés. Pour le MAplot et le volcano plot il est possible de choisir sa p-value même si elle est initialisée à 0.05 pour chaque. De plus, chacun des graphiques obtenus est téléchargeable au format .png en bas de chaque page.

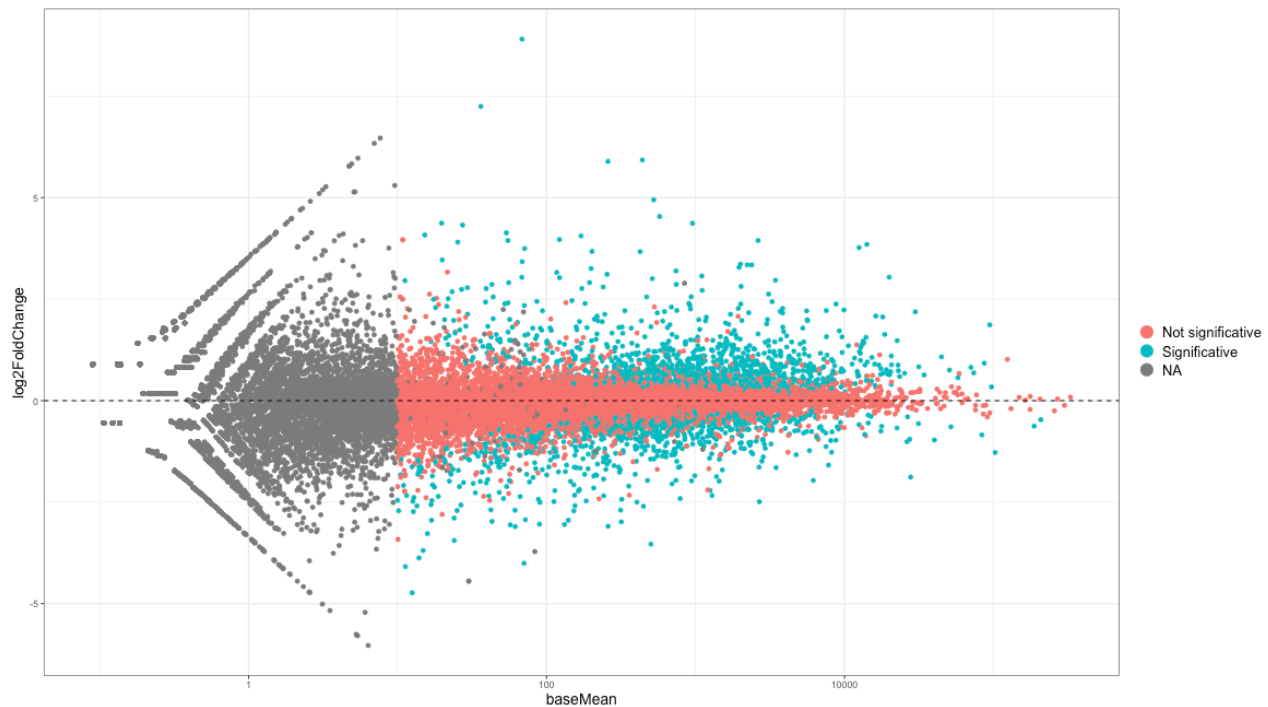


Figure 10 : MA plot obtenu sur l'application avec le jeu de données présenté dans la partie matériels et méthodes.

Customisation

Nous avons utilisé différents outils pour rendre plus agréable l'utilisation de l'application sur le plan visuel. A l'aide du langage HTML nous avons pu modifier des thèmes déjà existants. En effet, nous avons décidé de créer un bouton permettant de choisir entre 2 thèmes : un thème foncé et un thème clair selon les préférences de l'utilisateur. Pour cela nous avons utilisé le package *dashboardthemes* qui propose différents thèmes dont ceux utilisés : `grey_dark` et `grey_light`. Il a fallu modifier le thème `grey_dark` car il affichait une police noire sur un fond foncé dans les tableaux. C'était donc difficilement lisible. Pour se faire, nous avons dû récupérer le code du thème sur le [git](#) de Nik. L. présentant le package afin de pouvoir modifier ou ajouter certains codes HTML directement sur R de façon à ce que tout soit optimisé.

Nous avons aussi fait en sorte d'améliorer le visuel de nos graphiques. Pour cela nous avons donc utilisé `ggplot2`, le package permettant de faire des graphiques plus élégants. Ainsi les couleurs, les tailles des titres et des légendes et le texte sur les graphiques on pu être modifiés ou ajoutés pour compléter et améliorer au mieux chacun des graphiques.

Pour finir, nous avons mis en place un bouton accueil grâce au langage HTML. Ce bouton est placé

en haut à droite de l'écran, dans l'en-tête et permet à tout moment de retourner sur la page d'accueil qui est la page d'information sur l'application. De plus, toujours grâce au langage HTML, nous avons ajouté un pied de page dans lequel on retrouve des informations sur les développeurs de l'application ainsi que le git sur lequel on retrouve tout le travail réalisé.

Discussion

L'analyse de l'expression différentielle réalisée sur les données de l'étude RNA-Seq transcriptome profiling identifie CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells de Himes BE, Jiang X, Wagner P, et al a permis de mettre en place, tout en analysant les résultats obtenus, un workflow d'analyse d'expression différentielle utilisant le package DESeq2. Ce workflow contient notamment des étapes de visualisations des données brutes ainsi que de données normalisées de la table de comptages de reads mais aussi des étapes de visualisations des résultats de l'analyse de l'expression différentielle (MA plot, volcano plot, ACP, Heatmap). Ce workflow a été intégré dans la réalisation d'une application Shiny permettant d'entreprendre une analyse d'expression différentielle à partir d'une table de comptage de reads et son design expérimental associé auxquels l'on peut rajouter en option un fichier d'annotation. Cette application suit donc le workflow DESeq2 réalisé et génère les différents résultats et visualisations résultant de l'utilisation de ce workflow sur le jeu de données importé par l'utilisateur.

Conclusion et perspectives

Tout d'abord avant de conclure il est important de replacer le projet dans son contexte. Le but principal était d'aborder de nouveaux outils nous permettant d'évoluer et progresser sous l'environnement R ainsi que de trouver des outils permettant l'accessibilité à ces mêmes outils. Dans ce sens, ce projet nous a permis de découvrir le dépôt de packages Bioconductor qui fournit de nombreux packages destinés à la bioinformatique. Nous avons donc pu nous concentrer sur le package *DESeq2* pour une analyse d'expression différentielle. Nous avons grâce à ce package appris dans un premier temps à mieux comprendre l'analyse d'expression différentielle de données RNA-seq pour ensuite réaliser notre propre analyse d'expression différentielle à partir de données récupérées d'un article traitant la compréhension des mécanismes de traitement de l'asthme.

Et c'est à partir de cette analyse que nous avons pu répondre à notre problématique de l'accessibilité des résultats. Nous avons donc réalisé une application interactive suivant le workflow de notre analyse d'expression différentielle sous *DESeq2*. Cette application permet à tout possesseur d'une table de comptage de reads et de son design associé de réaliser un workflow de manière automatisé tout en ayant la possibilité de choisir ses propres paramètres. Toute la partie création de cette application nous a permis de découvrir le package *Shiny* et d'apprendre son utilisation et son fonctionnement. Au-delà du package en lui-même, cette démarche nous a permis l'apprentissage de bases importantes dans le langage html ainsi que quelques notions nécessaires pour la customisation dans le langage CSS. La finalité de ce projet est donc l'acquisition de nouvelles compétences dans l'analyse de données RNA-seq sous R ainsi que dans la mise en forme et l'accessibilité des résultats.

Maintenant que nous nous sommes familiarisés à ces différents outils nous pourrions aller plus loin dans notre application et dans l'analyse de données RNA-seq en étudiant d'autres outils permettant de compléter notre analyse. Notamment en apprenant à utiliser les packages permettant de réaliser les étapes précédant l'obtention de la table de comptage, ce qui permettrait d'avoir une application prête à l'utilisation dès l'obtention des fichiers fastQ.

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](https://doi.org/10.1371/journal.pone.0099625)
2. 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>
3. 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](https://doi.org/10.1165/rcmb.2017-0430TR)
4. Love, M.I., Huber, W., Anders, S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15:550. doi : [10.1186/s13059-014-0550-8](https://doi.org/10.1186/s13059-014-0550-8)
5. Nik. L. (2018, mars 4). nik01010/dashboardthemes. Consulté à l'adresse <https://github.com/nik01010/dashboardthemes>
6. Wickham, H. (2016). Mastering Shiny. Consulté à l'adresse <https://mastering-shiny.org/>
7. Bioconductor - Home. (2003). Consulté à l'adresse <https://www.bioconductor.org/>
8. Glossaire pour la statistique génomique. Montpellier GenomiX, MGX.

Annexe 1 : Bilan personnel

Tout d'abord nous avons débuté ce projet ne sachant pas encore si nous allions pouvoir réaliser notre stage ou non. Nous avons donc choisi ce sujet rapidement dans l'optique d'avoir une solution de secours si nos stages n'avaient pas lieu. Nous nous sommes donc tournés vers ce sujet car nous avions tous les deux postulé à un stage qui avait une problématique similaire (mise en place d'application shiny pour des données issues de séquençage RNA-seq et de données de puces micro-array). Dans ces conditions, nous avons donc décidé de nous tourner vers l'analyse d'expression différentielle de données RNA-seq. Pour cela il a fallu nous documenter afin de trouver des moyens de mieux comprendre cette analyse et des outils permettant de réaliser cette analyse sous R. Nous sommes alors tombés sur le package DESeq2 qui nous a semblé tout de suite idéal. Nous étions toujours dans le flou de savoir si notre stage allait avoir lieu ou non c'est pourquoi nous nous sommes dans un premier temps axé sur de la documentation brute sans encore entamer le développement de l'application. Ainsi, nous nous sommes donc documentés sur le package DESeq2 et l'analyse RNA seq pendant plusieurs jours afin d'avoir les connaissances nécessaires à la réalisation d'une analyse d'expression différentielle. Après avoir entrepris cette analyse sur un jeu de données d'une étude en libre accès et bien assimilé le package DESeq2 nous avons donc décidé de nous concentrer sur la réalisation de l'application. Avec l'annulation de nos stages respectifs nous avons pu enfin nous projeter pleinement dans notre projet. Nous avons donc commencé par nous documenter sur le package Shiny et les autres outils permettant de mettre en place une telle application. Nous avons déjà suivi un tutoriel au cours de cette année donc nous avons quelques bases mais pendant une semaine nous nous sommes mis à suivre assidument un cours disponible sur internet (n°5 dans la bibliographie). Cela nous a permis d'avoir des bases approfondies sur Shiny.

Tout s'est bien passé dans l'ensemble pendant toute la durée du projet. Cependant nous aurions pu nous documenter un peu plus ce qui nous aurait permis de peut être éviter certains blocages à certains moments. Mais notre démarche nous a permis de chercher par nous même et donc d'améliorer notre capacité d'adaptation, notre esprit de recherche et aussi notre anglais car la plupart des recherches a été faite en anglais. Ce fut aussi très gratifiant de trouver des réponses à nos problèmes seuls. Nous avons travaillé à distance en utilisant principalement Discord. Nous faisions également des réunions 1 à 2 fois par semaine afin de résumer ce que nous avons fait et ce que nous avons à faire. Nous nous partagions aussi chaque matin ce que chacun pouvait faire dans la journée. Nous avons aussi mis

en place des horaires afin d'avoir un rythme de travail sérieux et quand cela était possible nous nous mettions en vocal afin d'être disponibles pour s'aider mutuellement sur nos difficultés individuelles. Afin de partager ce que nous faisions chacun sur nos postes de travail, nous avons utilisé git. Cela nous a permis de partager directement à partir de Rstudio. Cependant nous avons rencontré quelques problèmes au début le temps d'assimiler complètement toutes les fonctionnalités.

Annexe 2 : Script R pour l'analyse RNAseq via DESeq2

```
#Library
library(DESeq2)
library(Biobase)
library(tidyverse)

### import data ----
#Firstly we import the 3 data table which we will use during this analysis.
#The "counts_table" contains the counting of read mapped on each genes by samples,
#"airway_metadata" contains the information about the samples design
#and "anno" contains the informations about the genes.
counts_table <- read.csv("Data/airway_scaledcounts.csv")
airway_metadata <- read.csv("Data/airway_metadata.csv")
anno <- read.csv("Data/annotables_grch38.csv")
anno <- anno %>% select(ensgene,symbol)

### Create the dds object ---
dds <- DESeqDataSetFromMatrix(counts_table,colData=airway_metadata,design = ~dex,tidy = TRUE)
# Set reference of experience, here "control"
colData(dds)$dex <- relevel(colData(dds)$dex , ref="control")
# To display experiment design.
colData(dds)
# To display column which biological condition is set.
design(dds)

### Explore data ----
#Transformation of dds counts table in data frame to use ggplot package
count_table_dds <- as.data.frame(counts(dds))
#vizualisation of count distribution ----
#To facilitate the vizualisation we use the log-freq of each count value "log(count+1)"
for( i in 1:8){
  p <- ggplot(data=count_table_dds, aes(log(count_dds[,i]+1))) +
    geom_histogram(breaks=seq(0,14,1),col="black",fill="grey")+
    theme_light()+
    labs(title=colnames(count_table_dds)[i],
         x="Count value (number of read by genes) in log(count+1)",
         y="Count frequency") +
    theme_bw()
  plot(p)
}
#Number for Null for each sample ----
apply(count_table_dds, 2 ,FUN = function(x) sum(x==0))
#vizualisation of depth for each sample using deph.plot function ----
depth <- colSums(count_table_dds)
depth <- as.data.frame(depth)
depth$sample <- row.names(depth)
ggplot(depth, aes( x=sample ,y=depth))+
  geom_bar(stat="identity",col="black", fill="white")+
  labs(title = "Depth of each sample", x="Sample", y="Depth")+
  theme_bw()

#Differential expression analysis and normalization ----
dds <- DESeq(dds)
#Extraction of the DE result
res <- results(dds,tidy = TRUE)
# The scale factor resulting of normalization are stock in size factor of our design.
colData(dds)
#Depth of count table after normalization ----
depth_normalize <- colSums(counts(dds, normalized= TRUE))
depth_normalize <- as.data.frame(depth_normalize)
depth_normalize$sample <- row.names(depth_normalize)
ggplot(depth_normalize, aes( x=sample ,y=depth_normalize))+
  geom_bar(stat="identity",col="black", fill="white")+
  labs(title = "Depth of each sample", x="Sample", y="Depth")+
  theme_minimal()
#vizualisation of count distribution after normalization ----
count_normalize <- counts(dds, normalized= TRUE)
count_normalize <- as.data.frame(count_normalize)
for(i in 1:8){
  p <- ggplot(data=count_normalize, aes(log(count_normalize[,i]+1))) +
    geom_histogram(breaks=seq(0,14,1),col="black",fill="grey")+
    theme_light()+
    labs(title=colnames(count_normalize)[i],
         x="Count value (number of read by genes) in log(count+1)",
         y="Count frequency")
  plot(p)
}
```



```

}

# Dispersion plot ----
# Relationship between dispersion and counts means.
# DESeq2 offer a function that can directly display a plot which describe
# the relation ship between dispersion and count mean.

DESeq2::plotDispEsts(dds, main= "Relationship between dispersion and counts means")
# We obtain a plot that show the final estimate which are obtain after
# shrink of genes estimate and we finally observe the fitted estimate. We can also observed outliers value.

### DE analysis results ----

# Show a summary of DE results at alpha = 0.05

summary(results(dds), 0.05)

# Number of genes which is differential express at 5%
up_regulated <- res %>% filter(padj <= 0.05 & log2FoldChange > 0) %>% nrow()
down_regulated <- res %>% filter(padj <= 0.05 & log2FoldChange < 0) %>% nrow()
tb <- table(res$padj <= 0.05, useNA="always")
tb.DE <- data.frame("No DE" = tb[1], "Down regulated" = down_regulated,
                    "Up regulated" = up_regulated, "NA" = tb[3] )
row.names(tb.DE) <- ""
tb.DE

# MA plot : relationship between mean count of a gene and its log2 ratio between the two conditions
res <- as.data.frame(res)
res <- res %>% mutate(sig=padj<0.05)

ggplot(res, aes(x = baseMean, y = log2FoldChange, col = sig)) +
  geom_point() +
  scale_x_log10() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "black") +
  ggtitle("MA plot") + theme_bw()

# Volcano plot at alpha 0.5
# Show ID of the most DE gene
res_v <- res %>% mutate(sig=padj<0.05) %>% arrange(padj) %>%
  inner_join(anno, by=c("row"= "ensgene"))

ggplot(res_v, aes(x=log2FoldChange, y=-log10(padj), col=sig)) +
  geom_point() +
  ggtitle("Volcano plot labelling top significant genes") +
  geom_text_repel(data = subset(res_v, (-log10(padj) > 30 |
                                     log2FoldChange > 6 |
                                     log2FoldChange < -6)),
                 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 significant", "Significant", "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))

# PCA
# Two PCA with two different transformation
vsdata <- vst(dds, blind=FALSE)
plotPCA(vsdata, intgroup="dex")

rld <- rlogTransformation(dds, blind = FALSE)
plotPCA(rld, intgroup="dex")

# Distance matrix for sample
library(RColorBrewer)
library(gplots)
dists <- dist(t(assay(vsdata)))
mat <- as.matrix(dists)
hmcol=colorRampPalette(brewer.pal(9,"GnBu"))(100)
heatmap.2(mat, trace="none", col = rev(hmcol), margin=c(13,13))

# Heatmap of gene expression for 50 better DE gene
library(NMF)

```

```

res <- tbl_df(res)
res <- res %>%
  arrange(padj) %>%
  inner_join(anno, by=c("row"="ensgene")) %>%
  filter(padj<0.05)
NMF::aheatmap(assay(vsddata)[arrange(res, padj, pvalue)$row[1:50],],
  labRow=arrange(res, padj, pvalue)$symbol[1:50],
  scale="row", distfun="pearson",
  annCol=dplyr::select(airway_metadata, dex, celltype),
  col=c("green", "black", "black", "red"))

```

Annexe 3 : Scripts R pour l'application Shiny

3.1 Interface utilisateur

```
### Library ----
### here we find all library need for proper functioning 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 composite 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 composite 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("MetadadataTable"),
          menuItemOutput("AnnotationTable")),
        menuItemOutput("menuDESeq2"),

        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(), "- Sample distance matrix",
br(), "- Gene expression 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 contain :
### - a selectInput of separator
### - a fileInput to upload count table of RNA-seq experience
### - a box which contain :
### - 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" = ",", "Tab" = "\t", "Semi-colon" = ";")),
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 menuItemOutput("")
  ### On this page we find :
  ### - a box which contain :
  ###   - a selectInput of separator
  ###   - a fileInput to upload metadata of RNA-seq experience
  ### - a box which contain :
  ###   - information about file accepted in fileInput
  ### - a box which contain :
  ###   - a select 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,
          selectInput("DesignDESeq2", "Choose your design without linear combination",
            c("")),
          dataTableOutput("MetaTable")
        )
      ),
    ### Upload annotation file ----
    ### upload page of annotation file associate with the RNA-seq experience
    ### this page is associate with menuItemOutput("")
    ### 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 contain :
    ###         - a selectInput of separator
    ###         - a fileInput to upload metadata of RNA-seq experience
    ###       - a box which contain :
    ###         - 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>
                    <br><h7>The process 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 contain :
### - 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 contain :
### - 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 contain :
### - 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 contain :
### - 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 contain :
    ### - 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 contain :
    ### - 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 contain :
    ### - 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 contain :
    ### - 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, align = "center",
                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 contain :
    ### - 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 contain :
    ### - sliderInput of P.value which set p.val of ma.plot function
    ### - a tableOutput() of number.DE.gene function
    ### - a box() which contain :
    ### - 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 contain :
### - 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 contain :
### - 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("AnnoVolcano"),
            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")))
),

### sample distance matrix heat map ----
### On this page we find distance matrix heat map and it parameters after running DESeq
### We find :
### - a box() which contain :
### - a selectInput to chose transformation for distance.matrix.heatmap
### - a actionButton to run distance.matrix.heatmap function
### - a box() which contain :
### - 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, align = "center",
        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 contain :
### - 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 contain :
### - 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))),
    uiOutput("Annoheatmap"),
    box(solidHeader = F, status = "primary",width = 12, align = "center",
      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>
      <p align="center" width="4"> Git : <a href="https://github.com/Gabin-c/RNA-seq-DE-analysis"
      target="_blank">https://github.com/Gabin-c/RNA-seq-DE-analysis</a> </p>'
    ),
    style =
      "
      position:relative;
      width:100%;
      background-color: rgb(70,80,90);
      color: rgb(205,205,205);"
  ))
)
)

```

3.2 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

  ### Verify for each column of metadata() if the values of each row are different or not
  allValuesDifferent <- reactive({
    apply(metadata(), 2, function(a) length(unique(a))==nrow(metadata()))
  })

  ### Select the FALSE of "allValuesDifferent" so the columns with equal values
  notAllValuesDifferent <- reactive({
    metadata()[allValuesDifferent() == FALSE]
  })

  ### Verify for each column notAllValuesDifferent if the values of each row are equals
  uniqueValue <- reactive({
    apply(notAllValuesDifferent(), 2, function(a) length(unique(a))==1)
  })

  ### Select the false of "uniqueValue" so the columns without unique values
  notUniqueValue <- reactive({
    notAllValuesDifferent()[uniqueValue() == FALSE]
  })

  ### Update selectinput with columns of notUniqueValue() for the DESeq2 design
```

```

observeEvent(input$MetadataFile,{
  updateSelectInput(session,"DesignDESeq2",
    choices = paste("~ ",paste(colnames(notUniqueValue()))))
})

### 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'))

### Display DESeq2 page after count table uploaded and a message to upload metadata file and choose design
observeEvent(input$CountDataTable,{
  output$menuDESeq2 <- renderMenu({
    if(input$DesignDESeq2 == ""){
      showNotification("Upload a metadata file and choose a design.")
    }
    menuItem(text = "2 Run DESeq2", tabName = "deseq2", icon = icon("play-circle"))
  })
})

### Running DESeq2 clicking on the button ----
observeEvent(input$RunDESeq2,{
  if(input$DesignDESeq2 == ""){
    showNotification("Upload a metadata file and choose a design.")
  }
  else{
    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$countss_dds <-as.data.frame(counts(dds$DESeq2))
    dds$countss_dds_n <-as.data.frame(counts(dds$DESeq2,normalized=TRUE))
    ### Transposed counts data frame normalized or not
    dds$countss_turnup <- as.data.frame(t(dds$countss_dds))

```

```

dds$counts_turnup_n <- as.data.frame(t(dds$counts_dds_n))

### 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
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()

})

### 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$annotationVolcano,{
  if(input$CheckAnnotation== TRUE){
    if(input$annotationVolcano==TRUE){
      output$SliderFoldVolcano <- renderUI({
        sliderInput("sliderfold", "Choose your fold", min=-20, max=20, value=c(-6,6))
      })
      output$SliderLogVolcano <- renderUI({
        sliderInput("sliderlog", "Choose your log10", min=0, max=300, value=30)
      })
    }
  }
  if(input$CheckAnnotation== FALSE){
    if(input$annotationVolcano==TRUE){
      output$SliderFoldVolcano <- renderUI({})
      output$SliderLogVolcano <- renderUI({})
      output$AnnoVolcano <- renderUI({
        HTML("<center><h3> You don't have annotation file. </h3></center>")
      })
    }
  }
  if(input$CheckAnnotation== FALSE){
    if(input$annotationVolcano==FALSE){
      output$SliderFoldVolcano <- renderUI({})
      output$SliderLogVolcano <- renderUI({})
      output$AnnoVolcano <- 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")
    }
  )

  ### Sample 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(){
    sample.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)
    }
  })

  observeEvent(input$annotationHeatmap,{
    if(input$annotationHeatmap==TRUE){
      if(input$CheckAnnotation==FALSE){
        output$Annoheatmap <- renderUI(
          box(width = 12, solidHeader = F,
            HTML("<center><h3> You don't have annotation file. </h3></center>"))
        )
      }
    }else{
      output$Annoheatmap <- renderUI({})
    }
  })

  ### 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$themes <- renderUI({
      theme_grey_dark2
    })
  }else{
    output$themes <-renderUI({
      theme_grey_light2
    })
  }
})

### "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()
})

### 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("Sample distance matrix",tabName = "DistanceMatrix",
                    icon = icon("far fa-check-square"))
    }else{
        menuSubItem("Sample 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("Gene expression Heatmap",tabName = "Heatmap",
                    icon = icon("far fa-check-square"))
    }else{
        menuSubItem("Genne expression Heatmap",tabName = "Heatmap")
    }
})
}

```

3.3 Fonctions

```
library(DESeq2)
library(Biobase)
library(gplots)
library(RColorBrewer)
library(shiny)
library(tidyverse)
library(NMF)
library(ggrepel)
library(factoextra)
### 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 your DESeq2 dataset
### - depth()
### - count_distribution()
### - plotcount()
### - DESeq2 : results object after run DESeq() on your DESeq2 dataset and results() or DESeq() object
### - maplot()
### - volcanoplot()
### - number_of_DE
### - DESeq2 : after run DESeq() on your DESeq2 dataset 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 visualization) 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")
}

### number of differential 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){
  up_regulated <- dds.result %>% filter(padj <= p.val & log2FoldChange > 0) %>% nrow()
  down_regulated <- dds.result %>% filter(padj <= p.val & log2FoldChange < 0) %>% nrow()
  tb <-table(dds.result$padj <= p.val ,useNA="always")
  tb.DE <- data.frame("No DE" = tb[1],
                     "Down regulated" = down_regulated,
                     "Up regulated" = up_regulated,
                     "NA" = tb[3] )
  row.names(tb.DE) <- ""
  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="dotted")+
    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 significant", "Significant", "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)))
}

### VolcanoPlot ----
### 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(padj), col=sig)) +
      geom_point() +
      ggtitle("Volcano plot labelling top significant genes") +
      geom_text_repel(data = subset(dds.res, (-log10(padj) > 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 significant", "Significant", "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)))
}
}

### 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 <- get_dist(t(assay(dds.resTransf)),method = "pearson")
  mat <- as.matrix(dists)
  hmccl= colorRampPalette(brewer.pal(9,"GnBu"))(100)
  return(heatmap.2(mat,trace="none",col = rev(hmccl),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"),treeheight = c(500,100))
    }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"),treeheight = c(500,100))
    }
  }
}

## Theme ----
shinyDashboardThemeDIY <- function(
  appFontFamily, appFontColor, logoBackColor, bodyBackColor, headerButtonBackColor, headerButtonIconColor,
  headerButtonBackColorHover, headerButtonIconColorHover, headerBackColor, headerBoxShadowColor,
  headerBoxShadowSize, sidebarBackColor, sidebarPadding, sidebarShadowRadius, sidebarShadowColor,
  sidebarMenuBackColor, sidebarMenuPadding, sidebarMenuBorderRadius, sidebarUserTextColor,
  sidebarSearchBackColor, sidebarSearchIconColor, sidebarSearchBorderColor, sidebarTabTextColor,
  sidebarTabTextSize, sidebarTabBorderStyle, sidebarTabBorderColor, sidebarTabBorderWidth,
  sidebarTabBackColorSelected, sidebarTabTextColorSelected, sidebarTabRadiusSelected,
  sidebarTabTextColorHover, sidebarTabBackColorHover, sidebarTabBorderStyleHover, sidebarTabBorderRadiusHover,
  sidebarTabBorderWidthHover, sidebarTabRadiusHover, boxBackColor, boxBorderRadius,
  boxShadowSize, boxShadowColor, boxTitleSize, boxDefaultColor, boxPrimaryColor, boxSuccessColor,
  boxWarningColor, boxDangerColor, tabBoxTabColor, tabBoxTabTextSize, tabBoxTabTextColor,
  tabBoxTabTextColorSelected, tabBoxBackColor, tabBoxHighlightColor, tabBoxBorderRadius, buttonBackColor,
  buttonTextColor, buttonBorderColor, buttonBorderRadius, buttonBackColorHover, buttonTextColorHover,
  buttonBorderRadiusHover, buttonHeight = 34, buttonPadding = "6px 12px", textboxBackColor,
  textboxBorderColor, textboxBorderRadius, textboxBackColorSelect, textboxBorderRadiusSelect,
  textboxHeight = 34, textboxPadding = "6px 12px", tableBackColor, tableBorderColor,
  tableBorderTopSize, tableBorderRowSize, primaryFontColor = "auto", successFontColor = "auto",
  warningFontColor = "auto", dangerFontColor = "auto", infoFontColor = "auto", boxInfoColor = "auto",
  dataFontColor
) {

  htmltools::tags$head(

    htmltools::tags$style(

      htmltools::HTML(

        paste0(

          '
          /* font: google import [OPTIONAL] */
          /* @import url("//fonts.googleapis.com/css?family=Roboto"); */
          /* font */
          body, label, input, button, select, box,
          .h1, .h2, .h3, .h4, .h5, h1, h2, h3, h4, h5 {
            font-family: ',appFontFamily,';
            color: ', appFontColor, ';
          }
          /* font: fix for h6 */
          /* messes up sidebar user section if included above */
          .h6, h6 {
            font-family: ',appFontFamily,';
          }
          /* sidebar: logo */
          .skin-blue .main-header .logo {
            background: ', logoBackColor, ';
          }
          /* sidebar: logo hover */
          .skin-blue .main-header .logo:hover {
            background: ', logoBackColor, ';
          }
          /* sidebar: collapse button */
          .skin-blue .main-header .navbar .sidebar-toggle {
            background: ', headerButtonBackColor, ';
            color: ', headerButtonIconColor, ';
          }
          /* sidebar: collapse button hover */
          .skin-blue .main-header .navbar .sidebar-toggle:hover {
            background: ', headerButtonBackColorHover, ';
            color: ', headerButtonIconColorHover, ';
          }
        '
      )
    )
  )
}

```

```

/* header */
.skin-blue .main-header .navbar {
background: ', headerBackColor, ';
box-shadow: ', headerBoxShadowSize, ' ', headerBoxShadowColor, ';
}
/* sidebar*/
.skin-blue .main-sidebar {
background: ', sidebarBackColor, ';
box-shadow: ', sidebarShadowRadius, " ", sidebarShadowColor, ';
padding-left: ', sidebarPadding, 'px;
padding-right: ', sidebarPadding, 'px;
/* padding-top: 60px; */
}
/* sidebar menu */
.main-sidebar .user-panel, .sidebar-menu, .sidebar-menu>li.header {
white-space: nowrap;
background: ', sidebarMenuBackColor, ';
padding: ', sidebarMenuPadding, 'px;
border-radius: ', sidebarMenuBorderRadius, 'px;
}
/* fix for user panel */
.user-panel>.info>p, .skin-blue .user-panel>.info {
color: ', sidebarUserTextColor, ';
font-size: 12px;
font-weight: normal;
}
section.sidebar .user-panel {
padding: 10px;
}
/* sidebar: tabs */
.skin-blue .main-sidebar .sidebar .sidebar-menu a {
color: ', sidebarTabTextColor, ';
font-size: ', sidebarTabTextSize, 'px;
border-style: ', sidebarTabBorderStyle, ';
border-color: ', sidebarTabBorderColor, ';
border-width: ', sidebarTabBorderWidth, 'px;
}
/* sidebar: tab selected */
.skin-blue .main-sidebar .sidebar .sidebar-menu .active > a {
color: ', sidebarTabTextColorSelected, ';
font-size: ', sidebarTabTextSize, 'px;
border-radius: ', sidebarTabRadiusSelected, ';
border-style: ', sidebarTabBorderStyleHover, ';
border-color: ', sidebarTabBorderColorHover, ';
border-width: ', sidebarTabBorderWidthHover, 'px;
}
.skin-blue .sidebar-menu > li:hover > a,
.skin-blue .sidebar-menu > li.active > a {
color: ', sidebarTabTextColorSelected, ';
background: ', sidebarTabBackColorSelected, ';
border-radius: ', sidebarTabRadiusHover, ';
}
/* sidebar: tab hovered */
.skin-blue .main-sidebar .sidebar .sidebar-menu a:hover {
background: '
,sidebarTabBackColorHover, ' ';
,'color: ', sidebarTabTextColorHover, ' ';
font-size: ', sidebarTabTextSize, 'px;
border-style: ', sidebarTabBorderStyleHover, ' ';
border-color: ', sidebarTabBorderColorHover, ' ';
border-width: ', sidebarTabBorderWidthHover, 'px;
border-radius: ', sidebarTabRadiusHover, ' ';
}
/* sidebar: subtab */
.skin-blue .sidebar-menu > li > .treeview-menu {
margin: 0px;
background: ', sidebarMenuBackColor, ' ';
}
.skin-blue .treeview-menu > li > a {
background: ', sidebarMenuBackColor, ' ';
}
/* sidebar: subtab selected */
.skin-blue .treeview-menu > li.active > a,
.skin-blue .treeview-menu > li > a:hover {
background: ', sidebarTabBackColorSelected, ' ';
}
/* sidebar: search text area */
.skin-blue .sidebar-form input[type=text] {
background: ', sidebarSearchBackColor, ' ';
}

```

```

color: ', appFontColor, ';
border-radius: ', textboxBorderRadius, 'px 0px 0px ', textboxBorderRadius, 'px;
border-color: ', sidebarSearchBorderColor, ';
border-style: solid none solid solid;
}
/* sidebar: search button */
.skin-blue .input-group-btn > .btn {
background: ', sidebarSearchBackColor, ';
color: ', sidebarSearchIconColor, ';
border-radius: 0px ', textboxBorderRadius, 'px ', textboxBorderRadius, 'px 0px;
border-style: solid solid solid none;
border-color: ', sidebarSearchBorderColor, ';
}
/* sidebar form */
.skin-blue .sidebar-form {
border-radius: 0px;
border: 0px none rgb(255,255,255);
margin: 10px;
}
/* body */
.content-wrapper, .right-side {
background: ', bodyBackColor, ';
}
/* box */
.box {
background: ', boxBackColor, ';
border-radius: ', boxBorderRadius, 'px;
box-shadow: ', boxShadowSize, ' ', boxShadowColor, ';
}
/* box: title */
.box-header .box-title {
font-size: ', boxTitleSize, 'px;
}
/* tabbox: title */
.nav-tabs-custom>.nav-tabs>li.header {
color: ', appFontColor, ';
font-size: ', boxTitleSize, 'px;
}
/* tabbox: tab color */
.nav-tabs-custom, .nav-tabs-custom .nav-tabs li.active:hover a,
.nav-tabs-custom .nav-tabs li.active a {
background: ', tabBoxTabColor, ';
color: ', appFontColor, ';
border-radius: ', boxBorderRadius, 'px;
}
.nav-tabs-custom {
box-shadow: ', boxShadowSize, ' ', boxShadowColor, ';
}
/* tabbox: active tab bg */
.nav-tabs-custom>.nav-tabs>li.active {
border-radius: ', tabBoxBorderRadius, 'px;
border-top-color: ', tabBoxHighlightColor, ';
# box-shadow: ', boxShadowSize, ' ', boxShadowColor, ';
}
/* tabbox: font color */
.nav-tabs-custom>.nav-tabs>li.active:hover>a, .nav-tabs-custom>.nav-tabs>li.active>a {
border-bottom-color: ', tabBoxTabColor, ';
border-top-color: ', tabBoxHighlightColor, ';
border-right-color: ', tabBoxHighlightColor, ';
border-left-color: ', tabBoxHighlightColor, ';
color: ', tabBoxTabTextColorSelected, ';
font-size: ', tabBoxTabTextSize, 'px;
border-radius: ', tabBoxBorderRadius, 'px;
}
/* tabbox: inactive tabs background */
.nav-tabs-custom>.nav-tabs>li>a {
color: ', tabBoxTabTextColor, ';
font-size: ', tabBoxTabTextSize, 'px;
}
/* tabbox: top area back color */
.nav-tabs-custom, .nav-tabs-custom>.tab-content, .nav-tabs-custom>.nav-tabs {
border-bottom-color: ', tabBoxHighlightColor, ';
background: ', tabBoxBackColor, ';
}
/* tabbox: top area rounded corners */
.nav-tabs-custom>.nav-tabs {
margin: 0;
border-radius: ', tabBoxBorderRadius, 'px;
}

```



```

/* infobox */
.info-box {
background: ', boxBackColor, ';
border-radius: ', boxBorderRadius, 'px;
box-shadow: ', boxShadowSize, ' ', boxShadowColor, ';
}
/* valuebox */
.small-box {
border-radius: ', boxBorderRadius, 'px;
box-shadow: ', boxShadowSize, ' ', boxShadowColor, ';
}
/* valuebox: main font color */
.small-box h3, .small-box p {
color: rgb(255,255,255)
}
/* box: default color */
.box.box-solid>.box-header, .box>.box-header {
color: ', appFontColor, ';
}
.box.box-solid>.box-header {
border-radius: ', boxBorderRadius, 'px;
}
.box.box-solid, .box {
border-radius: ', boxBorderRadius, 'px;
border-top-color: ', boxDefaultColor, ';
}
/* box: info color */
.box.box-solid.box-info>.box-header h3, .box.box-info>.box-header h3 {
color: ', infoFontColor, ';
}
.box.box-solid.box-info>.box-header {
background: ', boxInfoColor, ';
border-radius: ', boxBorderRadius, 'px;
}
.box.box-solid.box-info, .box.box-info {
border-color: ', boxInfoColor, ';
border-left-color: ', boxInfoColor, ';
border-right-color: ', boxInfoColor, ';
border-top-color: ', boxInfoColor, ';
border-radius: ', boxBorderRadius, 'px;
}
/* box: primary color */
.box.box-solid.box-primary>.box-header h3, .box.box-primary>.box-header h3 {
color: ', primaryFontColor, ';
}
.box.box-solid.box-primary>.box-header {
background: ', boxPrimaryColor, ';
border-radius: ', boxBorderRadius, 'px;
}
.box.box-solid.box-primary, .box.box-primary {
border-color: ', boxPrimaryColor, ';
border-left-color: ', boxPrimaryColor, ';
border-right-color: ', boxPrimaryColor, ';
border-top-color: ', boxPrimaryColor, ';
border-radius: ', boxBorderRadius, 'px;
}
/* box: success color */
.box.box-solid.box-success>.box-header h3, .box.box-success>.box-header h3 {
color: ', successFontColor, ';
}
.box.box-solid.box-success>.box-header {
background: ', boxSuccessColor, ';
border-radius: ', boxBorderRadius, 'px;
}
.box.box-solid.box-success, .box.box-success {
border-color: ', boxSuccessColor, ';
border-left-color: ', boxSuccessColor, ';
border-right-color: ', boxSuccessColor, ';
border-top-color: ', boxSuccessColor, ';
border-radius: ', boxBorderRadius, 'px;
}
/* box: warning color */
.box.box-solid.box-warning>.box-header h3, .box.box-warning>.box-header h3 {
color: ', warningFontColor, ';
}
.box.box-solid.box-warning>.box-header {
background: ', boxWarningColor, ';
border-radius: ', boxBorderRadius, 'px;
}

```

```

.box.box-solid.box-warning, .box.box-warning {
border-color: ', boxWarningColor, ';
border-left-color: ', boxWarningColor, ';
border-right-color: ', boxWarningColor, ';
border-top-color: ', boxWarningColor, ';
border-radius: ', boxBorderRadius, 'px;
}
/* box: danger color */
.box.box-solid.box-danger>.box-header h3, .box.box-danger>.box-header h3 {
color: ', dangerFontColor, ';
}
.box.box-solid.box-danger>.box-header {
background: ', boxDangerColor, ';
border-radius: ', boxBorderRadius, 'px;
}
.box.box-solid.box-danger, .box.box-danger {
border-color: ', boxDangerColor, ';
border-left-color: ', boxDangerColor, ';
border-right-color: ', boxDangerColor, ';
border-top-color: ', boxDangerColor, ';
border-radius: ', boxBorderRadius, 'px;
}
/* button */
.btn-default {
background: ', buttonBackColor, ';
color: ', buttonTextColor, ';
border-color: ', buttonBorderColor, ';
border-radius: ', buttonBorderRadius, 'px;
height: ', buttonHeight, 'px;
padding: ', buttonPadding, ';
}
/* button: hover */
.btn-default:hover {
background: ', buttonBackColorHover, ';
color: ', buttonTextColorHover, ';
border-color: ', buttonBorderColorHover, ';
}
/* button: focus */
.btn-default:focus, .action-button:focus {
background: ', buttonBackColor, ';
color: ', buttonTextColor, ';
border-color: ', buttonBorderColor, ';
}
/* button: active */
.btn-default:active, .action-button:active {
background: ', buttonBackColor, ';
color: ', buttonTextColor, ';
border-color: ', buttonBorderColor, ';
}
/* button: visited */
.btn-default:visited {
background: ', buttonBackColor, ';
color: ', buttonTextColor, ';
border-color: ', buttonBorderColor, ';
}
/* textbox */
.form-control, .selectize-input, .selectize-control.single .selectize-input {
background: ', textboxBackColor, ';
color: ', appFontColor, ';
border-color: ', textboxBorderColor, ';
border-radius: ', textboxBorderRadius, 'px;
height: ', textboxHeight, 'px;
min-height: ', textboxHeight, 'px;
padding: ', textboxPadding, ';
}
/* textbox: selected */
.form-control:focus, .selectize-input:focus {
color: ', appFontColor, ';
background: ', textboxBackColorSelect, ';
border-color: ', textboxBorderColorSelect, ';
-webkit-box-shadow: inset 0px 0px 0px, 0px 0px 0px;
box-shadow: inset 0px 0px 0px, 0px 0px 0px;
}
/* multi-row selectize input */
.selectize-control.multi .selectize-input.has-items {
height: auto;
}
/* verbatim text output */
.qt pre, .qt code {

```

```

font-family: ', appFontFamily, ' !important;
}
pre {
color: ', appFontColor, ' ;
background-color: ', textboxBackColor, ' ;
border: 1px solid ', textboxBorderColor, ' ;
border-radius: ', textboxBorderRadius, 'px;
}
/* drop-down menu */
.selectize-dropdown, .selectize-dropdown.form-control {
background: ', textboxBackColor, ' ;
border-radius: 4px;
}
/* table */
.table {
background: ', tableBackColor, ' ;
border-radius: ', textboxBorderRadius, 'px;
}
/* table: row border color*/
.table>tbody>tr>td, .table>tbody>tr>th, .table>tfoot>tr>td,
.table>tfoot>tr>th, .table>thead>tr>td, .table>thead>tr>th {
border-top: ', tableBorderRowSize, 'px solid ', tableBorderColor, ' ;
}
/* table: top border color*/
.table>thead>tr>th {
border-bottom: ', tableBorderTopSize, 'px solid ', tableBorderColor, ' ;
}
/* table: hover row */
.table-hover>tbody>tr:hover {
background-color: ', tableBorderColor, ' ;
}
/* table: stripe row */
.table-striped>tbody>tr:nth-of-type(odd) {
background-color: ', tableBorderColor, ' ;
}
/* table: body colour */
table.dataTable tbody tr {
background-color: ', tableBackColor, ' !important;
}
/* table: footer border colour */
table.dataTable {
border: 0px !important;
}
/* datatable: selected row */
table.dataTable tr.selected td, table.dataTable td.selected {
background-color: ', boxSuccessColor, ' !important;
color: rgb(0,0,0) !important;
}
/* datatable: hover row */
table.dataTable tr.hover td, table.dataTable td.hover {
background-color: ', tableBorderColor, ' !important;
}
table.dataTable.hover tbody tr:hover, table.dataTable.display tbody tr:hover {
background-color: ', tableBorderColor, ' !important;
}
table.dataTable.row-border tbody th, table.dataTable.row-border tbody td,
table.dataTable.display tbody th, table.dataTable.display tbody td {
border-top: 1px solid ', tableBorderColor, ' !important;
}
/* datatable: stripe row */
table.dataTable.stripe tbody tr.odd, table.dataTable.display tbody tr.odd {
background-color: ', tableBorderColor, ' !important;
}
/* datatable: page control */
.dataTables_wrapper .dataTables_paginate .paginate_button {
background-color: ', tableBorderColor, ' !important;
color: ', dataFontColor, ' !important;
}
/* datatable: table info */
.dataTables_wrapper .dataTables_paginate .paginate_button.disabled,
.dataTables_wrapper .dataTables_paginate .paginate_button.disabled:hover,
.dataTables_wrapper .dataTables_paginate .paginate_button.disabled:active {
color: ', dataFontColor, ' !important;
}
.dataTables_wrapper .dataTables_length, .dataTables_wrapper .dataTables_filter,
.dataTables_wrapper .dataTables_info, .dataTables_wrapper .dataTables_processing,
.dataTables_wrapper .dataTables_paginate {
color: ', dataFontColor, ' !important;
}

```

```

        .dataTables_wrapper .dataTables_length, .dataTables_wrapper .dataTables_filter,
        .dataTables_wrapper .dataTables_info, .dataTables_wrapper .dataTables_processing,
        .dataTables_wrapper .dataTables_paginate {
          color: ', dataFontColor, ' !important;
        }
        thead {color: ', dataFontColor, ' ; } tbody {color: ', dataFontColor, ' ; }

        /* datatable search box */
        .dataTables_wrapper .dataTables_filter input {
          background-color: ', textboxBackColor, ' ;
          border: 1px solid ', textboxBorderColor, ' ;
          border-radius: ', textboxBorderRadius, 'px;
        }
        thead {color: ', dataFontColor, ' ; } tbody {color: ', dataFontColor, ' ; }
        /* notification and progress bar */
        .progress-bar {
          background-color: ', boxSuccessColor, ' ;
        }
        .shiny-notification {
          height: 80px;
          font-family: ', appFontFamily, ' ;
          font-size: 15px;
          color: rgb(0,0,0);
          background-color: rgb(225,225,225);
          border-color: rgb(205,205,205);
          border-radius: 10px;
          margin-left: -450px !important;
        }
        /* horizontal divider line */
        hr {
          border-top: 1px solid rgb(215,215,215);
        }
        /* modal */
        .modal-body {
          background-color: ', boxBackColor, ' ;
        }
        .modal-footer {
          background-color: ', boxBackColor, ' ;
        }
        .modal-header {
          background-color: ', boxBackColor, ' ;
        }
      '
    )
  )
}

```

```

theme_grey_dark2 <- shinyDashboardThemeDIY(
  ### general
  appFontFamily = "Arial"
  ,appFontColor = "rgb(205,205,205)"
  ,primaryFontColor = "rgb(255,255,255)"
  ,infoFontColor = "rgb(255,255,255)"
  ,successFontColor = "rgb(255,255,255)"
  ,warningFontColor = "rgb(255,255,255)"
  ,dangerFontColor = "rgb(255,255,255)"
  ,bodyBackColor = "rgb(45,55,65)"

  ### header
  ,logoBackColor = "rgb(70,80,90)"

  ,headerButtonBackColor = "rgb(70,80,90)"
  ,headerButtonIconColor = "rgb(25,35,45)"
  ,headerButtonBackColorHover = "rgb(40,50,60)"
  ,headerButtonIconColorHover = "rgb(0,0,0)"

  ,headerBackColor = "rgb(70,80,90)"
  ,headerBoxShadowColor = ""
  ,headerBoxShadowSize = "0px 0px 0px"

  ### sidebar
  ,sidebarBackColor = "rgb(52,62,72)"
  ,sidebarPadding = 0

  ,sidebarMenuBackColor = "transparent"
  ,sidebarMenuPadding = 0
  ,sidebarMenuBorderRadius = 0

```

```

,sidebarShadowRadius = ""
,sidebarShadowColor = "0px 0px 0px"

,sidebarUserTextColor = "rgb(205,205,205)"

,sidebarSearchBackColor = "rgb(45,55,65)"
,sidebarSearchIconColor = "rgb(153,153,153)"
,sidebarSearchBorderColor = "rgb(45,55,65)"

,sidebarTabTextColor = "rgb(205,205,205)"
,sidebarTabTextSize = 14
,sidebarTabBorderStyle = "none"
,sidebarTabBorderColor = "none"
,sidebarTabBorderWidth = 0

,sidebarTabBackColorSelected = "rgb(70,80,90)"
,sidebarTabTextColorSelected = "rgb(255,255,255)"
,sidebarTabRadiusSelected = "5px"

,sidebarTabBackColorHover = "rgb(55,65,75)"
,sidebarTabTextColorHover = "rgb(255,255,255)"
,sidebarTabBorderStyleHover = "none"
,sidebarTabBorderColorHover = "none"
,sidebarTabBorderWidthHover = 0
,sidebarTabRadiusHover = "5px"

### boxes
,boxBackColor = "rgb(52,62,72)"
,boxBorderRadius = 5
,boxShadowSize = "0px 0px 0px"
,boxShadowColor = ""
,boxTitleSize = 16
,boxDefaultColor = "rgb(52,62,72)"
,boxPrimaryColor = "rgb(200,200,200)"
,boxInfoColor = "rgb(80,95,105)"
,boxSuccessColor = "rgb(155,240,80)"
,boxWarningColor = "rgb(240,80,210)"
,boxDangerColor = "rgb(240,80,80)"

,tabBoxTabColor = "rgb(52,62,72)"
,tabBoxTabTextSize = 14
,tabBoxTabTextColor = "rgb(205,205,205)"
,tabBoxTabTextColorSelected = "rgb(205,205,205)"
,tabBoxBackColor = "rgb(52,62,72)"
,tabBoxHighlightColor = "rgb(70,80,90)"
,tabBoxBorderRadius = 5

### inputs
,buttonBackColor = "rgb(230,230,230)"
,buttonTextColor = "rgb(0,0,0)"
,buttonBorderColor = "rgb(50,50,50)"
,buttonBorderRadius = 5

,buttonBackColorHover = "rgb(180,180,180)"
,buttonTextColorHover = "rgb(50,50,50)"
,buttonBorderColorHover = "rgb(50,50,50)"

,textboxBackColor = "rgb(68,80,90)"
,textboxBorderColor = "rgb(76,90,103)"
,textboxBorderRadius = 5
,textboxBackColorSelect = "rgb(80,90,100)"
,textboxBorderColorSelect = "rgb(255,255,255)"

### tables
,tableBackColor = "rgb(52,62,72)"
,tableBorderColor = "rgb(70,80,90)"
,tableBorderTopSize = 1
,tableBorderRowSize = 1

### datatables
,dataFontColor = "rgb(255, 255, 255)"
)

theme_grey_light2 <- shinyDashboardThemeDIY(

### general
appFontFamily = "Arial"
appFontColor = "rgb(45,45,45)"
,primaryFontColor = "rgb(15,15,15)"

```

```

,infoFontColor = "rgb(15,15,15)"
,successFontColor = "rgb(15,15,15)"
,warningFontColor = "rgb(15,15,15)"
,dangerFontColor = "rgb(15,15,15)"
,bodyBackColor = "rgb(240,240,240)"

### header
,logoBackColor = "rgb(120,120,120)"

,headerButtonBackColor = "rgb(120,120,120)"
,headerButtonIconColor = "rgb(220,220,220)"
,headerButtonBackColorHover = "rgb(100,100,100)"
,headerButtonIconColorHover = "rgb(60,60,60)"

,headerBackColor = "rgb(120,120,120)"
,headerBoxShadowColor = "#dfdfdf"
,headerBoxShadowSize = "3px 5px 5px"

### sidebar
,sidebarBackColor = "rgb(255,255,255)"
,sidebarPadding = 0

,sidebarMenuBackColor = "transparent"
,sidebarMenuPadding = 0
,sidebarMenuBorderRadius = 0

,sidebarShadowRadius = "3px 5px 5px"
,sidebarShadowColor = "#dfdfdf"

,sidebarUserTextColor = "rgb(115,115,115)"

,sidebarSearchBackColor = "rgb(240,240,240)"
,sidebarSearchIconColor = "rgb(100,100,100)"
,sidebarSearchBorderColor = "rgb(220,220,220)"

,sidebarTabTextColor = "rgb(100,100,100)"
,sidebarTabTextSize = 14
,sidebarTabBorderStyle = "none"
,sidebarTabBorderColor = "none"
,sidebarTabBorderWidth = 0

,sidebarTabBackColorSelected = "rgb(230,230,230)"
,sidebarTabTextColorSelected = "rgb(0,0,0)"
,sidebarTabRadiusSelected = "0px"

,sidebarTabBackColorHover = "rgb(245,245,245)"
,sidebarTabTextColorHover = "rgb(0,0,0)"
,sidebarTabBorderStyleHover = "none solid none none"
,sidebarTabBorderColorHover = "rgb(200,200,200)"
,sidebarTabBorderWidthHover = 4
,sidebarTabRadiusHover = "0px"

,boxBackColor = "rgb(248,248,248)"
,boxBorderRadius = 5
,boxShadowSize = "none"
,boxShadowColor = ""
,boxTitleSize = 18
,boxDefaultColor = "rgb(225,225,225)"
,boxPrimaryColor = "rgb(95,155,213)"
,boxInfoColor = "rgb(180,180,180)"
,boxSuccessColor = "rgb(112,173,71)"
,boxWarningColor = "rgb(237,125,49)"
,boxDangerColor = "rgb(232,76,34)"

,tabBoxTabColor = "rgb(248,248,248)"
,tabBoxTabTextSize = 14
,tabBoxTabTextColor = "rgb(100,100,100)"
,tabBoxTabTextColorSelected = "rgb(45,45,45)"
,tabBoxBackColor = "rgb(248,248,248)"
,tabBoxHighlightColor = "rgb(200,200,200)"
,tabBoxBorderRadius = 5

### inputs
,buttonBackColor = "rgb(215,215,215)"
,buttonTextColor = "rgb(45,45,45)"
,buttonBorderColor = "rgb(150,150,150)"
,buttonBorderRadius = 5

,buttonBackColorHover = "rgb(190,190,190)"

```

```
,buttonTextColorHover = "rgb(0,0,0)"
,buttonBorderColorHover = "rgb(150,150,150)"

,textboxBackColor = "rgb(255,255,255)"
,textboxBorderColor = "rgb(118,118,118)"
,textboxBorderRadius = 5
,textboxBackColorSelect = "rgb(245,245,245)"
,textboxBorderColorSelect = "rgb(108,108,108)"

### tables
,tableBackColor = "rgb(248,248,248)"
,tableBorderColor = "rgb(238,238,238)"
,tableBorderTopSize = 1
,tableBorderRowSize = 1

### datatables
,dataFontColor = "rgb(108,108,108)"
)
```