# Note

Ce rapport en pdf intéractif a été créé en R Markdown, il permet d’utiliser à la fois le langage LaTeX et R.

Le code ci-dessous a dû être retravaillé pour être correctement afficher.

La dernière version du code est disponible à l’adresse suivante : <https://github.com/EcoNum/coral_growth001>

# Annexe

## ui.R

library(shiny)  
library(shinyWidgets)  
library(DT)  
library(plotly)  
library(shinythemes)  
library(shinyWidgets)  
  
  
shinyUI(  
 navbarPage(  
 #theme = shinytheme("slate"),  
 title = "Coral growth", # Titre onglet 1  
 #### Onglet principal : Graphique  
 tabPanel(title = "Plot",  
 ## Sidebar : volet de gauche - Input  
 sidebarPanel(  
 uiOutput(outputId = "u\_choice\_project"),  
 uiOutput(outputId = "u\_choice\_condition"),  
 uiOutput(outputId = "u\_choice\_status"),  
 uiOutput(outputId = "u\_choice\_id"), # Sélection des ID  
 uiOutput(outputId = "u\_choice\_plot")#Sélection du graphique  
 uiOutput(outputId = "u\_choice\_nbr\_day"),# Sélection de Xvar  
 uiOutput(outputId = "u\_choice\_date") #Sélection date  
 ),  
 ## MainPanel : Volet de droite - Output  
 mainPanel(  
 tabsetPanel(  
 # Sous-onglet  
 tabPanel(title = "Main plot",  
 plotlyOutput(outputId = "u\_plot",   
 height = "600px" ),  
 #sortie console  
 verbatimTextOutput(outputId = "u\_info"))  
 #tabPanel(title = "Test plot")  
 )  
 )  
 ),  
 ### Onglet principal : Tableau de donnée  
 tabPanel("Data table",  
 # Sidebar : Volet de gauche - Input  
 # sidebarPanel(  
 # ),  
 # MainPanel : Volet de droite - Output  
 mainPanel(  
 tabsetPanel(  
 tabPanel(title = "Table", DTOutput(outputId = "u\_table")  
 )  
 )  
 )  
 ),  
 ### Onglet principal : Aide  
 tabPanel(title = "Help",  
 fluidRow(  
 column(12, includeMarkdown(  
 "../../analysis/Notebook/Notebook-Manuel.Rmd"))))  
 )  
)

## server.R

library(shiny)  
library(ggplot2)  
library(lubridate)  
library(tidyverse)  
library(dplyr)  
library(plotly)  
library(shinyWidgets)  
SciViews::R  
  
  
  
### ------------------\_\_Partie logique du serveur\_\_-----------------------  
shinyServer(function(input, output, session) {  
  
 # Tableur de Madeleine :  
 # coral\_url <- "https://docs.google.com/spreadsheets/d/e/2PACX-1vTJLtfjj  
 #UM4VK6aM177ly9GCKyMHFrFqQdsqjhJCtpe4DUGuZWOe2fZWB5xTZEx3WAcW08BVEBFfn2C  
 #/pub?gid=0&single=true&output=csv"  
  
 # Tableur de Jordan :  
 coral\_url <- "https://docs.google.com/spreadsheets/d/e/2PACX-1vSoBfvhztF  
 gALk1fcljBbYP03D-fRIEy7mu1DrHKZ--BXYZWHFxUujac\_-gFSteM99p7CFQILT\_eXcC/pu  
 b?gid=0&single=true&output=csv"  
  
 #Importation et format des colonnes  
 read\_csv(coral\_url,  
 col\_types = cols( .default = col\_character(),  
 date = col\_datetime(),  
 weight = col\_double(),  
 temperature = col\_double(),  
 salinity = col\_double() )) %>.%  
 mutate(.,  
 project = factor(project), author = factor(author),  
 aqua = factor(aqua),  
 condition = factor(condition),  
 species = factor(species),  
 id = factor(id, levels = 1:length(unique(id))),  
 status = factor(status)  
 ) -> df  
  
 ### Calcul du poids squelettique :  
 #a corriger : rho\_aragonite  
 #P = Pression hydrostatique, elle vaut 0 a la surface  
 skeleton\_weight <- function(S, T, P = 0,  
 buoyant\_weight,  
 rho\_aragonite = 2930){  
 rho\_water <- seacarb::rho(S = S, T = T , P = P)  
 skl\_wgt <- buoyant\_weight / (1 - (rho\_water / rho\_aragonite))  
 skl\_wgt <- round(skl\_wgt, digits = 3)  
 return(skl\_wgt)  
 }  
  
 # Ajout de la colonne du poids squelettique  
 df <- mutate(df,  
 skw = skeleton\_weight(S = salinity,  
 T = temperature,  
 buoyant\_weight = weight))  
  
 # Nombre de ID different  
 nbr\_id <- unique(df$id)  
  
 # Conditions  
 nbr\_condition <- unique(df$condition)  
  
 # Projet  
 nbr\_projet <- unique(df$project)  
  
 # Statut  
 nbr\_status <- unique(df$status)  
  
 # Taux de croissance  
 df %>.%  
 group\_by(., id) %>.%  
 arrange(., date) %>.%  
 mutate(.,  
 delta\_date = (as.numeric(difftime(date,   
 date[1], units = "days"))),  
 ratio = round(((skw-skw[1])/skw[1]/delta\_date)\*100,digits = 3),  
 delta\_date = round(delta\_date, digits = 0)) %>.%  
 ungroup(.) -> df  
  
  
 ### ----------\_\_Fin traitement du tableau de données\_\_ ------------- ###  
  
 #======================================================================#  
  
 # ----------------------- Selection des dates --------------------------  
 output$u\_choice\_date <- renderUI({  
  
 dateRangeInput(inputId = "s\_choice\_date",  
 label = 'Date range input: ',  
 start = min(df$date), end = max(df$date),  
 min = min(df$date), max = Sys.Date()  
 )  
 })  
 # ------------------------- Selection Xvar -----------------------------  
 output$u\_choice\_nbr\_day <- renderUI({  
  
 radioButtons(inputId = "s\_choice\_nbr\_day",  
 label = 'Xvar : ',  
 choices = c("Date", "Number of days"),  
 selected = "Number of days"  
 )  
 })  
  
 #--------------------------Selection id---------------------------------  
 output$u\_choice\_id <- renderUI({  
 pickerInput(inputId = "s\_choice\_id",  
 label = "Choice ID :",  
 choices = nbr\_id,  
 options = list(`actions-box` = TRUE),  
 multiple = T,  
 selected = c(8, 9, 55, 9))  
  
 })  
  
 # ------------------------- Choix des ID -------------------------------  
 observe({  
 print(input$s\_choice\_id)  
 })  
  
 #----------------------Choix graphique (variable y)---------------------  
 output$u\_choice\_plot <- renderUI({  
  
 radioButtons(inputId = "s\_choice\_plot", label = "Yvar :",  
 choices = c("Buoyant mass", "Skeleton mass",  
 "Growth rate"),  
 selected = "Buoyant mass")  
 })  
  
 #--------------------------Choix projet---------------------------------  
 output$u\_choice\_project <- renderUI({  
  
 selectInput(inputId = "s\_choice\_project",  
 label = "Project :",  
 choices = nbr\_projet,  
 multiple = TRUE,  
 selected = nbr\_projet)  
 })  
  
 #-------------------------Choix condition-------------------------------  
 output$u\_choice\_condition <- renderUI({  
  
 selectInput(inputId = "s\_choice\_condition",  
 label = "Condition :",  
 choices = nbr\_condition,  
 multiple = TRUE,  
 selected = nbr\_condition)  
 })  
  
 #--------------------------Choix statut---------------------------------  
 output$u\_choice\_status <- renderUI({  
  
 selectInput(inputId = "s\_choice\_status",  
 label = "Status :",  
 choices = nbr\_status,  
 multiple = TRUE,  
 selected = nbr\_status)  
 })  
  
  
  
 ###---------------------Output de mon graphique--------------------###  
 output$u\_plot <- renderPlotly({  
  
# Filtre en fonction des choix  
 df %>.%  
 filter(.,  
 project %in% input$s\_choice\_project,  
 condition %in% input$s\_choice\_condition,  
 status %in% input$s\_choice\_status,  
 date >= input$s\_choice\_date[1]&date<=input$s\_choice\_date[2],  
 id %in% input$s\_choice\_id  
 ) -> df  
  
 # Choix de la masse squelettique  
 if ("Skeleton mass" %in% input$s\_choice\_plot) {  
 yvar = df$skw  
 y\_axis\_name <- "Skeleton mass (g)"  
 }  
  
 # Choix de la masse immergée  
 if ("Buoyant mass" %in% input$s\_choice\_plot) {  
 yvar = df$weight  
 y\_axis\_name <- "Buoyant mass (g)"  
 }  
  
 # Choix du taux de croissance  
 if ("Growth rate" %in% input$s\_choice\_plot) {  
 yvar = df$ratio  
 y\_axis\_name <- "Growth rate"  
 }  
  
 # Choix par nombre de jour  
 if ("Number of days" %in% input$s\_choice\_nbr\_day) {  
 xvar = df$delta\_date  
 xlabel = "Day"  
 }  
  
 # Choix par date du jour  
 if ("Date" %in% input$s\_choice\_nbr\_day) {  
 xvar = df$date  
 xlabel = "Date"  
 }  
  
 ggplot(df, aes(x = xvar, y = yvar, colour = id)) +  
 geom\_point(size = 2, show.legend = FALSE, na.rm = TRUE) +  
 geom\_line(show.legend = FALSE, na.rm = TRUE) +  
 xlab(xlabel) + ylab(y\_axis\_name) -> p  
  
 p <- ggplotly(p, show.legend = FALSE)  
 })  
  
  
 ###-------------------------Sortie console---------------------------###  
 output$u\_info <- renderPrint({  
  
 #Affichage de la formule utilisé  
 formule <- ""  
  
 if ("Buoyant mass" %in% input$s\_choice\_plot) {  
 formule <- "Buoyant mass (g)"  
 }  
 if ("Skeleton mass" %in% input$s\_choice\_plot) {  
 formule <- "Skeleton mass (g)"  
 }  
 if ("Growth rate" %in% input$s\_choice\_plot) {  
 formule <- "Growth rate = ( (skeleton\_mass\_n - skeleton\_mass\_n-1) /  
 skeleton\_mass\_n-1 ) / (time\_n - time\_n-1) \* 100"  
 }  
  
 # Calculs boutures mortes  
 nbr\_dead <- as.numeric(count(unique(subset(df, status == "dead",id))))  
 death\_rate <- as.numeric(round  
 ((nbr\_dead / length(levels(nbr\_id))) \* 100,  
 digits = 2))  
 id\_dead <- unique(subset(df, status == "dead",id))  
 id\_dead <- id\_dead$id  
  
 cat("Yvar : ", formule, "\n", "\n",  
 "Species :", as.character(unique(df$species)), "\n", "\n",  
 "Number of deads cuttings :", nbr\_dead, "\n",  
 "ID dead cuttings : ", paste(id\_dead, collapse = ", "), "\n",  
 "Death rate :", death\_rate, "%")  
 })  
  
  
 # -------------------------Onglet tableau------------------------------#  
 output$u\_table <- renderDT({  
 datatable(df, filter = "top")  
 })  
  
 # Recuperation de l'ID du fichier ui.R  
 output$u\_choice\_table <- renderUI({  
  
 radioButtons(inputId = "s\_choice\_table", label = "Filtrer",  
 choices = c("Yes", "No"),  
 selected = "No")  
 })  
  
 output$u\_subchoice\_table <- renderUI({  
  
 dropdown(  
 radioButtons(inputId = "s\_subchoice\_table",  
 label = "by",  
 choices = c("skeleton weight", "growth rates"),  
 selected = c("skeleton weight")),  
 width = "200px",  
 size = "default",  
 label = "Variable type",  
 tooltip = tooltipOptions(placement = "right",  
 title = "Choice variable type")  
 )  
 })  
  
 output$u\_choice\_var <- renderUI({  
  
 numericInput(inputId = "s\_choice\_var",  
 label = if (input$s\_subchoice\_table == "growth rates")  
 {"Growth rates higher than :"}  
 else {"Skeleton weight higher than :"},  
 value = 1)  
 })  
})