---

output: pdf_document

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"
```

```r
#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)
  })
})
```