

---

output: pdf\_document

```
library(shiny)
library(shinyWidgets)
library(DT)
library(plotly)
library(shinythemes)

shinyUI(
  navbarPage(
    # theme = shinytheme("slate"),
    title = "Evolution de la croissance des coraux", # Titre onglet 1
    tabPanel("Graphique", # Onglet principal 1
      # Sidebar : volet de gauche - Input
      sidebarPanel(
        uiOutput(outputId = "ID"), # Selection des ID a afficher
        uiOutput(outputId = "Ratio")
      ),

      # mainPanel : Volet de droite - Output
      mainPanel(
        tabsetPanel( # Sous-onglet
          tabPanel("Sous-onglet 1 : Graphique",
            plotlyOutput(outputId = "monplot"),
            #sortie console
            verbatimTextOutput(outputId = "boutures_mortes")),
          tabPanel("Sous-onglet 2 : ")
        )
      )
    ),
    tabPanel("Tableau de donnee", # Onglet principal 2
      # sidebar : volet de gauche - Input
      sidebarPanel(
      ),
      # mainPanel : Volet de droite - Output
      mainPanel(
        tabsetPanel(
          tabPanel("Beau tablo", DT::dataTableOutput("tableau"))
        )
      )
    )
  )
)
```

```
#           /                               #
#           /                               #
#         ,/.                               #
#       ,\|/.                               #
#     ,'.V.'\                               #
#   /.'      \. \                           #
# /.'      \.' \                           #
#
```

```

#           ,'.:      ;,`.
#           |@|/  . . |(@|
#           ,-. `.-';      :`-,',-.-
#           '---`-\ /,-----.\ /-' ---`
#           (---- _/ ||___|| _/ ----)
#           `-,',-'\ `.-.-' / `.-.,'
#           `-.____,-'
#
#####__INFO__#####
# Application Shiny
# creer un graphique et un tableau a partir d'un fichier .csv
# les valeurs manquantes "NA" sont detecter comme etant des boutures mortes.
#
# Pour utiliser correctement l'application,
# il est important de respecter la syntaxe des noms des colonnes qui sont :
# |      ID      |      weight      |      temp      |      salinity      |      date      |
#
# Le format de la date doit etre de type :
# dd/mm/yyyy
#
# Il est egalement necessaire de commenter les lignes :
#cp_tablo[81:84, 2] <- "oublie"
#cp_tablo[16, 2] <- "a rejeter"
#botablo[81:84, 2] <- "oublie"
#botablo[16, 2] <- "a rejeter"
#
# Ces lignes sont specifiques a mon jeux de donnees
#####
#
# Titre : Croissance des coraux
# Auteur : Jordan Benrezkallah
# Date debut : 04/03/2019
# Date fin : 06/05/2019
#
#####

# Importation des librairies :

library(shiny)
library(ggplot2)
library(lubridate)
library(tidyverse)
library(dplyr)
library(plotly)
library(google sheets)
SciViews::R
# library(scales)

# #Fonction de Raphael :
# source(file = "../R/fonctions.R")
# #Mes fonctions
# source(file = "../R/fonction.R")

```

```

# Importation de mes donnees (format csv)
#correction a faire : chemin relatif
#tablo <- gdata::read.xls("~/shared/Github/coral_growth001/data/raw/monBordel/tablo.xls")

tablo <- read.table("~/shared/Github/coral_growth001/data/my_data/tablogs.csv", header = TRUE, sep = ";")

# GOOGLE SHEETS#
# tablo <- gs_title("tablo")
# tablo <- gs_read(tablo)
# tablo

# Determination du nombre de ligne de tableau a utiliser
# /!\ Baser sur la premiere valeur NA rencontre dans la colonne "temp" /!\
# Fonction a ameliorer de facon a ne garder seulement les lignes completes (ID, weight, temp, salinity,

ma_derniere_ligne <- function(){
  a <- 0
  for (i in tablo$temp) {
    if (!is.na(i)) {
      a <- a + 1
    }
  }
  return(a)
}

# Extraction des 5 colonnes (id, weight,temp,salinity et date) jusqu'a la derniere
# ligne de la colonne "temp" du fichier .csv
tablo <- tablo[1:ma_derniere_ligne(), 1:5]

### Calcul du poids squelettique :
#a corriger : rho_aragonite
#P = Pression hydrostatique, elle vaut 0 a la surface
skeleton_weight <- function(S = tablo$salinity, T = tablo$temp, P = 0,
                             buoyant_weight = tablo$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
tablo <- mutate(tablo, skw = skeleton_weight())

# changer le type de l'ID de "int" a "factor"
tablo$ID <- factor(tablo$ID)

#changer le type (mode) de la date
tablo$date <- dmy_hm(tablo$date)

#parse_date_time(tablo$date, locale = locale("fr"), orders = "dmy HMS")
tablo$date <- as_datetime(tablo$date)

```

```

# arrondir la datetime a l'heure pres
# tablo$date <- round_date(tablo$date, "hour")

# Nombre de ID different
nbr_ID <- unique(tablo$ID)

#Je fais une copie pour pouvoir travailler dessus sans creer de probleme d'affichage
cp_tablo <- tablo
botablo <- tablo

# affiche dans le tablo a presenter
botablo[81:84, 2] <- "oublie"

#la valeur de la bouture 16 est a rejeter
botablo[16, 2] <- "a rejeter"

#Remplace les valeurs manquantes par "Bouture morte"
botablo[is.na(botablo)] <- "Bouture morte"

#Tableau a afficher sur l'app Shiny :
botablo <- transmute(botablo,
                      ID = botablo$ID,
                      "Masse immerge (g)" = botablo$weight,
                      "Masse squelettique (g)" = skeleton_weight(),
                      "Temperature (c)" = botablo$temp,
                      "Salinite (g/L)" = botablo$salinity,
                      Date = botablo$date)

# Taux de croissance
tablo %>%
  group_by(., ID) %>%
  arrange(., date) %>%
  mutate(., delta_date = difftime(date, date[1], units = "days" ),
          ratio = (skw - skw[1]) / skw[1] / as.double(delta_date)) -> tablo1
# a cause du group_by je ne peux pas modifier directement "tablo"
tablo <- mutate(tablo, ratio = tablo1$ratio)

#tablo$ratio[is.nan(tablo$ratio)] <- "HOHOH"

###-----###
### ----- Partie logique du serveur ----- ###
shinyServer(function(input, output, session) {

# -----Selection des ID-----

# Recuperation de l'ID du fichier ui.R
output$ID <- renderUI({

#Menu deroulant
  dropdown(
    checkboxGroupInput(inputId = "choix_id", label = NULL,
                      choices = c("All", "None", nbr_ID), selected = c("All")),
    width = "200px", size = "default", label = "ID",

```

```

    tooltip = tooltipOptions(placement = "right", title = "Choix des ID")
  )
})

#-----Choix taux de croissance-----
output$Ratio <- renderUI({
  radioButtons(inputId = "choix_ratio", label = NULL,
    choices = c("Masse squelettique", "Taux de croissance"),
    selected = "Taux de croissance")
})

# -----Output de mon graphique-----
output$monplot <- renderPlotly({

  #Filtrer les lignes par rapport a ce qui a ete selectionne
  if ("All" %in% input$choix_id) {
    updateCheckboxGroupInput(session, inputId = "choix_id", label = "select All",
      choices = c("All", "None", nbr_ID), selected = c("All", nbr_ID)
    )
  }

  if ("None" %in% input$choix_id) {
    updateCheckboxGroupInput(session, inputId = "choix_id", label = "select All",
      choices = c("All", nbr_ID), selected = NULL
    )
  }

  else {
    tablo <- filter(tablo, tablo$ID %in% input$choix_id)
    yvar = tablo$skw
    y_nom_axe <- "Masse squelettique (g)"
  }

  # Choix du taux de croissance
  if ("Taux de croissance" %in% input$choix_ratio) {
    #tutu <- filter(tutu, tutu$ID %in% input$choix_id)
    yvar = tablo$ratio
    y_nom_axe <- "Taux de croissance"
  }

  # Tableau
  p <- ggplot(tablo, aes(x = tablo$date, y = yvar, colour = tablo$ID)) +
    geom_point(size = 2, show.legend = FALSE) + geom_line(show.legend = F) +
    xlab("Date") + ylab(y_nom_axe)
  #+ theme(axis.line = element_line(color = "darkgray", size = 2, linetype = "solid"))

  #p + scale_x_date(labels = date_format("%d-%m-%y"))

  #Pour remettre plotly, il faut changer : renderPlotly (server.R), plotlyOutput (ui.R) et dec commenter
  p <- ggplotly(p)

  ### Legende qui ne fonctionne pas, probleme d'attribution...

```

```

# Legende lorsque l'on passe son curseur :
# ma_legende <- paste("ID :", factor_ID, "\n", "Poids :", tablo$weight, "\n", "Date :", madate)
# pp <- ggplotly(p)
# pp <- style(pp, text = ma_legende, hoverinfo = "text")
})
#-----Sortie console-----#
output$boutures_mortes <- renderPrint({
  ### Cette partie sert a compter les boutures mortes

  #remplacer les weight de valeur NA des id 81 a 84 par "oublie"
  #cela va servir pour ne pas les compter dans les boutures mortes
  cp_tablo[81:84, 2] <- "oublie"

  #la valeur de la bouture 16 est a rejeter
  cp_tablo[16, 2] <- "a rejeter"

  #les 2 lignes ci-dessous empeche la visualisation du graphique si je ne met pas cp_tablo
  ID_NA <- subset(cp_tablo, is.na(weight) == TRUE, ID)
  ID_NA <- unique(ID_NA)
  ID_NA <- ID_NA$ID

  #nombre de boutures mortes :
  nbr_bouture_morte <- length(ID_NA)

  #Taux de mortalite :
  Taux_mort <- round((nbr_bouture_morte / length(as.numeric(unique(cp_tablo$ID)))) * 100, digits = 1)

  cat("Nombre de bouture morte : ", nbr_bouture_morte, "\nma_derniere_ligne() :",
      ma_derniere_ligne(), "\n", "\n", "\nTaux de mortalite : ",
      Taux_mort, "%", "\nID bouture morte : ", paste(ID_NA, collapse = ", "))
})

# -----Tableau-----
output$tableau <- DT::renderDataTable({DT::datatable(tablo)
})
})

```