Retour d’expérience sur le développement d’une application Shiny

Exemple de Easy16S

Cédric Midoux

22 mars 2019

Cédric Midoux – Présentation

- Ingénieur d'études à Irstea – 60%
  - Unité PROSE (ex-HBAN)
  - Études des bioprocédés environnementaux
  - Bioinformaticien en charge des méta-omics

- Intégré au sein de la plateforme Migale – 40%
  - Unité MaIAGE
  - Service d'analyses de données

_01

Introduction à Shiny

Shiny, pourquoi faire ?

- Vous avez écrit un script R que vous voulez partager avec des non-initiés ?
- Vos collaborateurs souhaitent pouvoir changer les paramètres sans être confronté au code ?
- Vous voulez rentre interactif le code d'un projet ?
- Vous souhaitez déployer un outils pour explorer des données ?

➔ **Le package Shiny de R peut répondre à vos besoins !**

Shiny

- Package développé par l'équipe RStudio
- Création d'applications web interactive avec R
- On peut refaire tout ce que l'on fait habituellement avec R
- L'utilisateur n'est pas confronté au code
- Votre code R devient votre site web
  - Exemple iris
- Personnalisation de l'app shiny avec html, CSS, JavaScript, API, shinydashboard
  - Exemple bus

runExample("01_hello")

ui.R
user interface

Apparence et mise en page

```
1  library(shiny)
2
3  # Define UI for app that draws a histogram ----
4  ui <- fluidPage(
5
6    # App title ----
7    titlePanel("Hello Shiny!"),
8
9    # Sidebar layout with input and output definitions ----
10   sidebarLayout(
11
12     # Sidebar panel for inputs ----
13     sidebarPanel(
14
15       # Input: Slider for the number of bins ----
16       sliderInput(inputId = "bins",
17                   label = "Number of bins:",
18                   min = 1,
19                   max = 50,
20                   value = 30)
21
22     ),
23
24     # Main panel for displaying outputs ----
25     mainPanel(
26
27       # Output: Histogram ----
28       plotOutput(outputId = "distPlot")
29
30     )
31   )
32 )
```

```
34 # Define server logic required to draw a histogram ----
35 server <- function(input, output) {
36
37   # Histogram of the Old Faithful Geyser Data ----
38   # with requested number of bins
39   # This expression that generates a histogram is wrapped in a call
40   # to renderPlot to indicate that:
41   #
42   # 1. It is "reactive" and therefore should be automatically
43   #    re-executed when inputs (input$bins) change
44   # 2. Its output type is a plot
45   output$distPlot <- renderPlot({
46
47     x    <- faithful$waiting
48     bins <- seq(min(x), max(x), length.out = input$bins + 1)
49
50     hist(x, breaks = bins, col = "#75AADB", border = "white",
51          xlab = "Waiting time to next eruption (in mins)",
52          main = "Histogram of waiting times")
53
54     })
55
56 }
57
58 # Create Shiny app ----
59 shinyApp(ui = ui, server = server)
60
```

server.R

Calculs et constructions des outputs

Exécution de l'app

_02

Easy16S

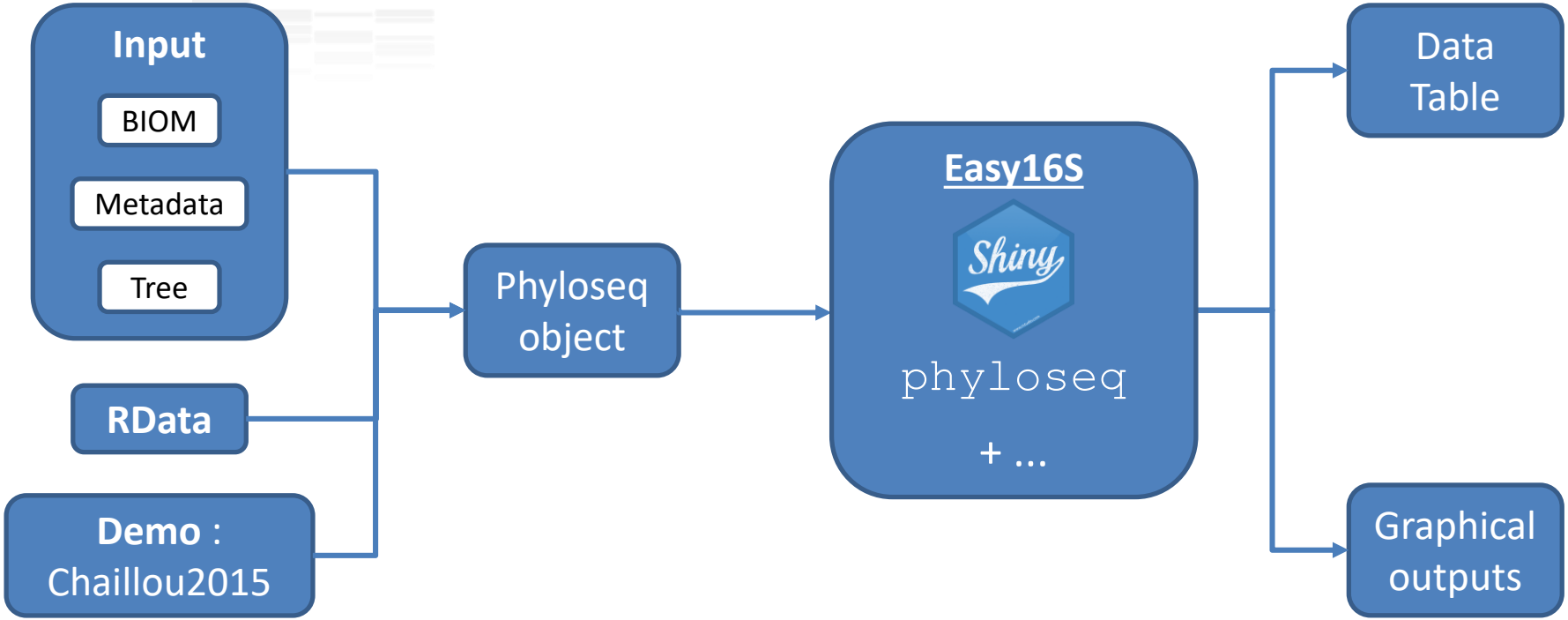
Contexte et Contraintes

- Demandes régulières de quelques analyses / figures par mon entourage scientifique

  pour analyses metabarcoding (étape post-FROGS)

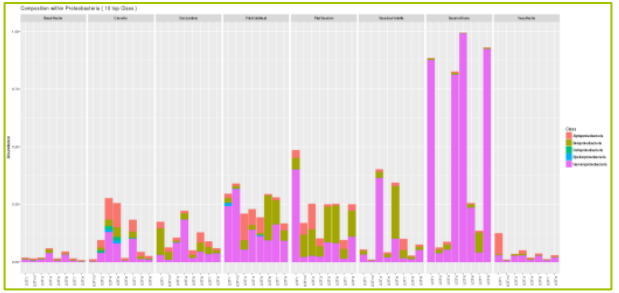
- Recherche d'une solution clé en main (coté utilisateur) et avec un développement facile :
  - Contraintes :
  - Interface user-friendly
  - Interactif
  - Utilisateur ne doit pas s'occuper des installations / mises à jour
  - Utilisation de R et phyloseq pour les figures
  - Développement continu de nouvelles *features* pour répondre aux retours

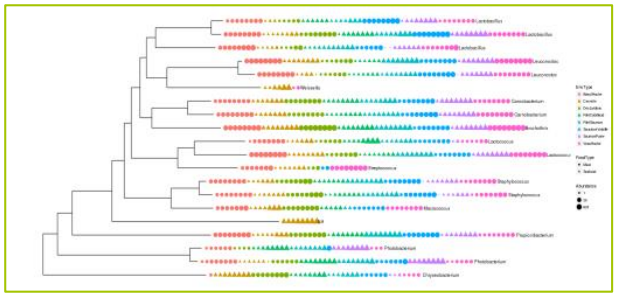
Inputs & Outputs

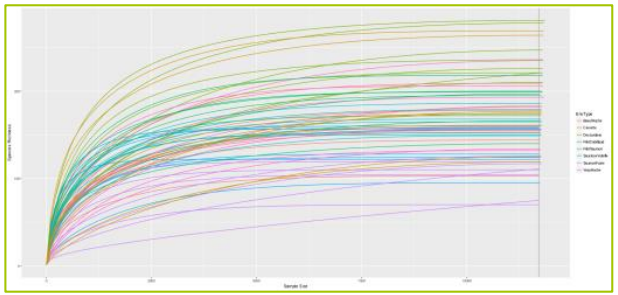


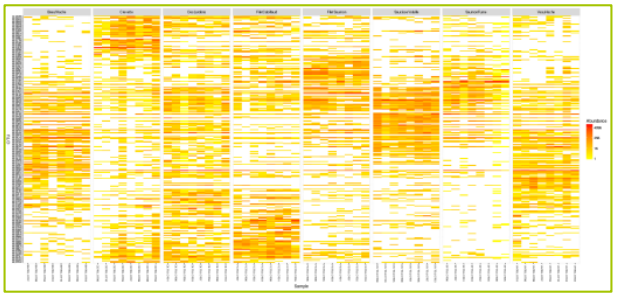
Graphical outputs

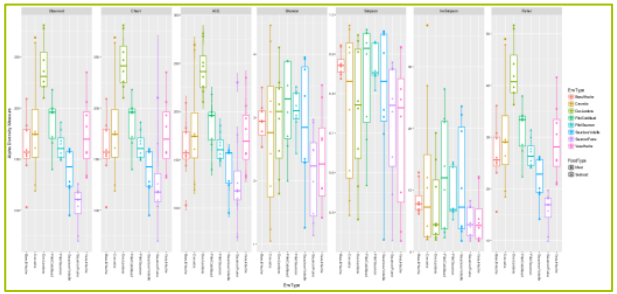
Gestion des couleurs, formes, ordre, sous-plot et autres paramètres en fonction des métadata dynamiquement par l'utilisateur

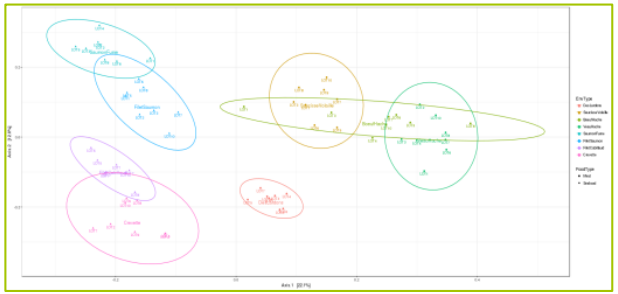


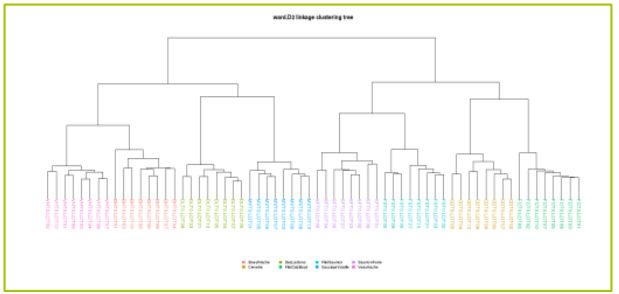












Démonstration

- http://genome.jouy.inra.fr/shiny/easy16S/
- https://gitlab.irstea.fr/cedric.midoux/easy16S

Perspectives et ToDo list

- Meilleure gestion des logs de connections
- Montée en charge du serveur shiny sur le cluster migale
- Conteneurisation de l'app avec docker
- Déploiement des graphiques dynamiques (plotly)
- Optimisation du téléchargement des figures
- Rédaction d'un manuel d'utilisation
- Intégration de nouvelles fonctionnalités remontées par les utilisateurs …

_03

Solutions de déploiement

Solutions de déploiement

- **Mise à disposition du code** (via `git` par exemple)
  - Exécution en local, dépendance à la charge de l'utilisation
- **Shinyapp.io** ($0 ~ $3 300 / an)
  - Exécution sur les serveurs Rstudio
  - Déploiement « Easy to Use »
  - Formule gratuite 25h/mois & plusieurs formules payantes
- **Shiny server open-source** (gratuit)
  - Déploiement d'une application web sur un serveur mais sans multiprocesses
- **Shiny server Pro** ($10 000 / an) (20 concurrent users max)
  - Déploiement d'une application web sur un serveur
  - Multiple R processes
  - Authentification (SSL and LDAP, Active Directory, Google OAuth, PAM, proxied authentication, or passwords)
  - Metrics & Session Management
- **Shiny-proxy**
  - Déploiement de shiny server open-source dans un docker « a la volée »

Take Home Message

☑ **Applications interactives en R facile à créer**

☑ Documentation et galerie fournie

☑ Solution shinyapp.io facile (mais contrainte)

☒ Déploiement et maintenance du server difficile

☒ Contraintes inhérentes à R
(temps de calculs, utilisation d'autres langages, …)

☒ Forte dépendance à RStudio