Retour d'expériences sur le développement d'une application Shiny – *Exemple de Easy16S*

Cédric Midoux

27 septembre 2018
Journée Intégration & Visualisation

Shiny

Shiny, pourquoi faire ? Et comment ?

- Création d'applications dynamiques pour le web avec R
- On peut refaire tout ce que l'on fait habituellement avec R
- L'utilisateur n'est pas confronté au code

- Exemple Iris
- Exemple bus

- Le code est séparé en deux :
  - `ui.R` : user interface, apparence et mise en page
  - `server.R` : calculs et constructions des outputs

runExample("01_hello")

```r
library(shiny)

# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # App title ----
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(

      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),

    # Main panel for displaying outputs ----
    mainPanel(

      # Output: Histogram ----
      plotOutput(outputId = "distPlot")

    )
  )
)
```

```r
# Define server logic required to draw a histogram ----
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

    })

}

# Create Shiny app ----
shinyApp(ui = ui, server = server)
```

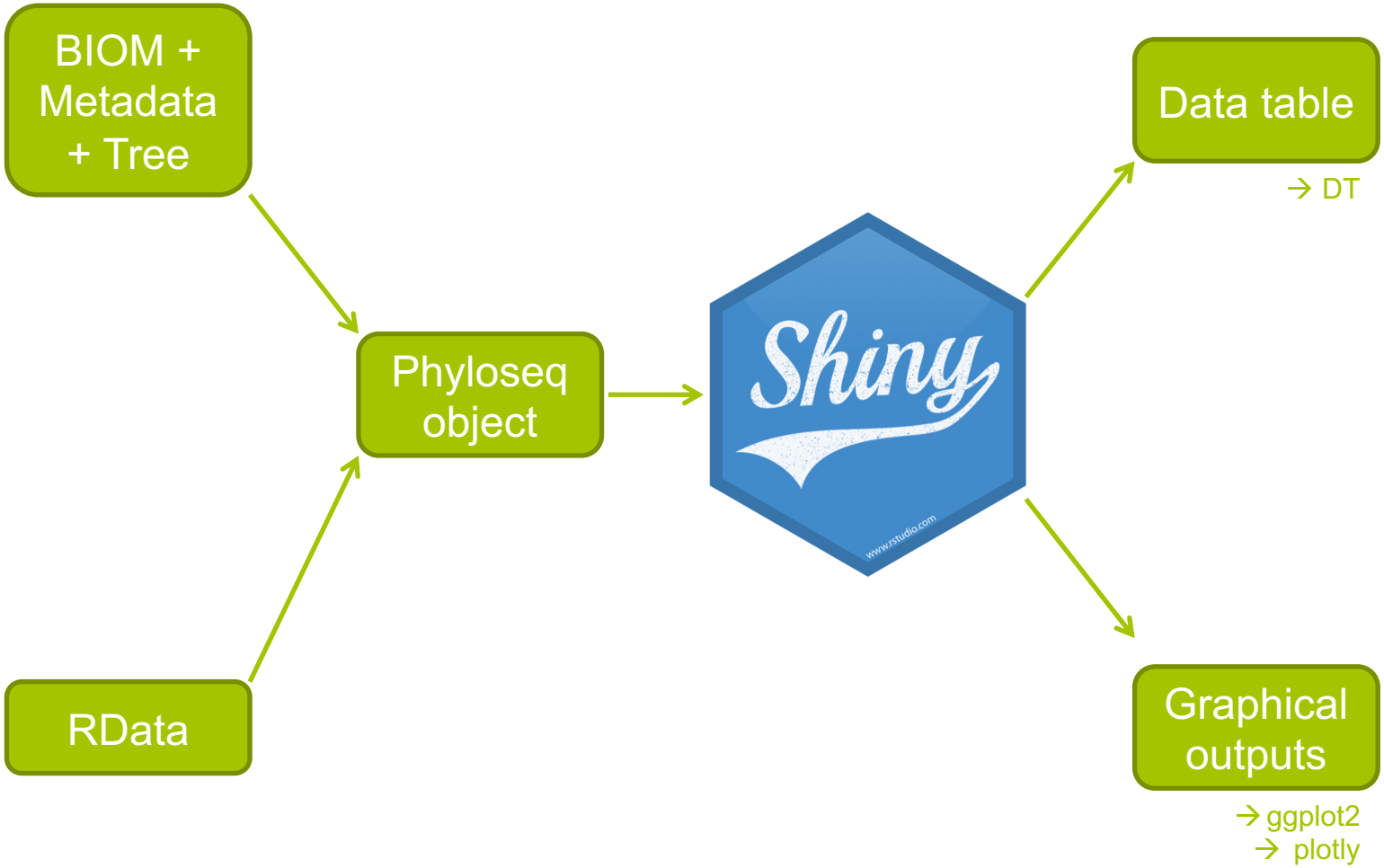
Exemple de Easy16S

Contexte et contraintes

- Demande régulière de quelques analyses / figures par mon entourage scientifique pour analyses métabarcoding (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 pour les figures
  - Développement continu de nouvelles features pour répondre aux retours
  - Développement en interaction avec l'équipe FROGS

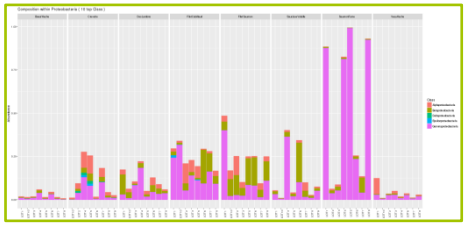
Inputs & Outputs

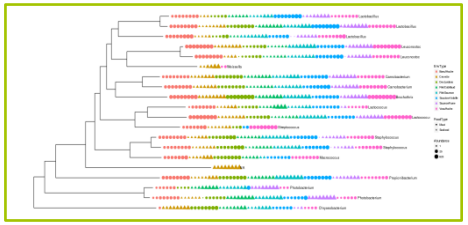


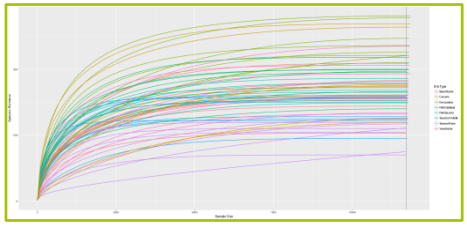
irstea

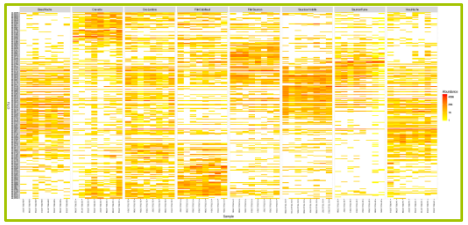
Graphical outputs

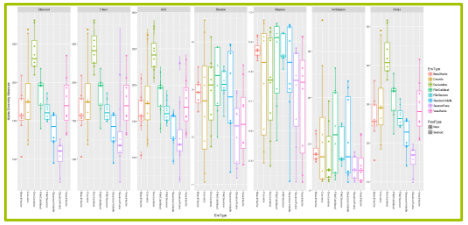
Gestion des couleurs, formes, ordre, sous-plot et autres paramètres en fonction des métadata dynamiquement par l'utilisateur

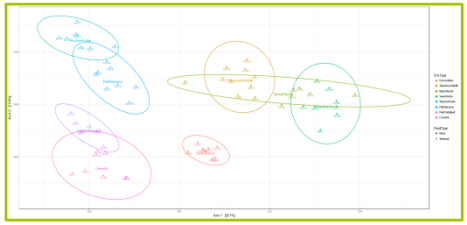


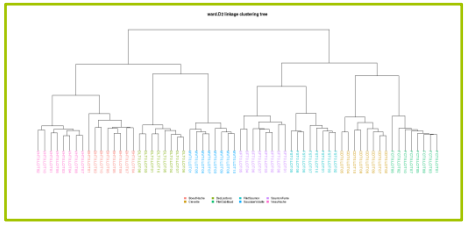












Besoins que vous pourriez rencontrer lors du développement ?

- Besoin de widgets d'inputs spécifiques ?

  → Visitez la galerie très variée
- Besoin d'une mise en page plus aboutie ?

  → Utilisation de `shinydashbord`
- Besoin d'ajuster la réactivité ?

  → `reactive()` permet de calculer des objets dynamiquement
- Besoin de gérer les erreurs ?

  → `validate(need())` permet de spécifier les messages d'erreurs
- Besoin de construire dynamiquement l'user interface ?

  → `renderUI()` et `uiOutput()` permettent de construire les widgets en fonction des data / inputs (exemple)

Solutions d’hébergement

$ 0

Hébergement

- Mise à disposition du code (*via* `git` *par exemple*)
  - Les calculs sont exécutés en local
  - L'utilisateur est responsable de l'installation des packages
  - Pas d'administration serveur à assurer

irstea

$ 0 – $ 3300 / yr

Hébergement

- Shinyapps.io
  - Déploiement d'une application web
  - Les calculs sont exécutés sur un serveur Rstudio
  - « Easy to Use »
  - Interface d'administration, logs et metrics d'utilisations
  - Possibilité d'authentification

irstea

$ 0 – $ 3300 / yr

Hébergement – shinyapps.io

	FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
Applications	5	25	Unlimited		
Active Hours	25	100	500	2000	10000
Authentification	no			Authentification	
Performance	Classic		Performance Boost		
Support	Community	Premium			
Others	RStudio Branding				Account Sharing Custom URL
$ / month	**0**	**9**	**39**	**99**	**299**

- Active hours : Active Hours are hours your applications are not idle. They are capped per account per month to protect the service from the exceptional resource consumer. If you exceed them on a paid account, an overage charge will be assessed. If you exceed them on a Free account your applications will not be available to your users again until the following month cycle.
- Performance boost : Multiple worker processes per application, up to 8GB of RAM and the ability to add additional instances to keep your applications responsive as more people use them.
- Account sharing : Share one account with many shiny developers.

$ 0

Hébergement

- Shiny server open-source
  - Déploiement d'une application web
  - Les calculs sont exécutés sur le serveur

irstea

Hébergement

$ 9995 / yr

→ 20 users simultanés
(pack de +20 = $4995
et +150 = $14995)

- Shiny server Pro
  - Déploiement d'une application web
  - Les calculs sont exécutés sur le serveur
  - Multiple R processes
  - Authentification (SSL and LDAP, Active Directory, Google OAuth, PAM, proxied authentication, or passwords)
  - Metrics & Session Management

irstea

Compare RStudio Connect, Shiny Server Pro, Shiny Server and Shinyapps.io

Category	Description	RStudio Connect	Shiny Server Pro	Shiny Server Open Source	Shinyapps.io
Overview	Commercial License (not AGPL)	●	●		●
	RStudio Support	●	●		●
	Deploy Shiny applications to the Web	●	●	●	●
	Push-button publishing from RStudio IDE	●			●
	Deploy and access shiny apps, dashboards, R Markdown reports, static plots, and APIs in one place	●			
	Scheduled updates and distribution of reports	●			
	Self-managed content – see and manage what you've published or can access from others	●			Publishers Only
	Professional Drivers – connect to some of the most popular databases	●	●		
Security & Authentication	Password protect applications	●	●		●*
	Deploy Shiny applications behind firewalls	●	●	●	
	Controlled access via SSL and LDAP, Active Directory, Google OAuth, PAM, proxied authentication, or passwords	●	●		
Tuning & Scaling	Scale applications across multiple R processes	●	●		●
	Persistent R processes for faster load times	●			
Metrics & Management	Performance and resource metrics	●	●		●
	Health check endpoint	●	●		

* For shinyapps.io plans that include authentication, your application users must have a Google, Github or a shinyapps.io account