Nouvelles technologies sous R - avec shiny

Laurent DELSOL

Chapitre 2: Documents interactifs avec Shiny

Table of contents

- Generalites
- Interface utilisateur - Mise en page
- Interface utilisateur - Inputs and Widgets
- Interface utilisateur - Output
- Serveur
- Gestion plus fine de la réactivité - Mise a jour plus adaptee des observations
- Widget dépendant d'un autre…
- Enregistrer les observations et Charger des observations
- Mettre a jour des widgets
- Mise en ligne de l'application et du code
- Pour aller un peu plus loin…
- Exercices

Generalites

Motivations

Shiny permet de creer des documents

- reproductibles, automatises, … et interactifs!
- utilisables sans installation de R, sans aucune ligne de code a taper
- au format HTML ce qui facilite leur lecture et leur partage sur internet (via un serveur shiny)
- sans trop de complexite au niveau de leur preparation

Application shiny et reactivite

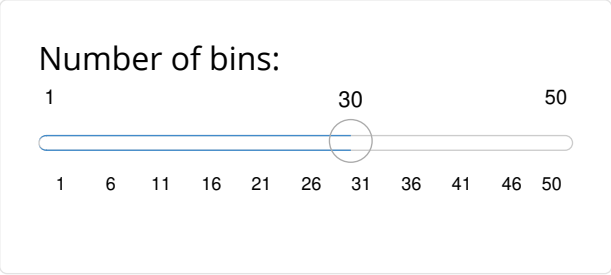
Les resultats presentes dans une application shiny sont reactifs. C'est a dire que si une nouvelle valeur d'entree (**input**) est donnee par l'utilisateur via un **widget** (window gadget) alors les resultats (**outputs**) qui en dependent seront mis a jour.

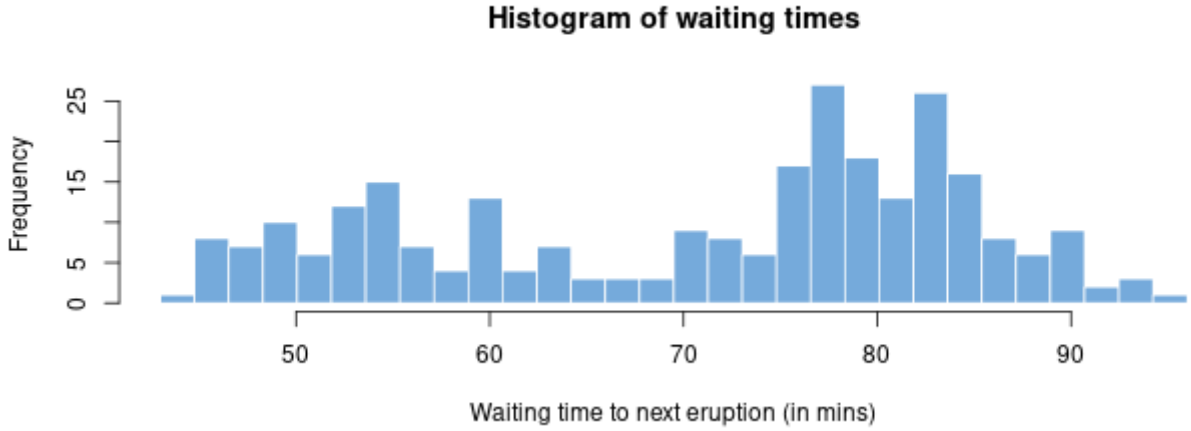
Un premier exemple

Pour comprendre l'interet de ce genre de documents, commencons par un premier exemple inclus dans le package **shiny**.

```
library(shiny)
runExample("01_hello")
```

Hello Shiny!





Un premier exemple (suite)

On cree ainsi une interface **interactive** pouvant permettre:

- de modifier des parametres
- de charger de nouvelles donnees
- de faire l'essai d'une methode sans l'avoir implementee ni avoir installe R
- de reproduire et d'automatiser le traitement des donnees
- d'aider l'utilisateur a prenre en main l'outil propose

De nombreux exemples sont disponibles a l'adresse **https://shiny.rstudio.com/gallery/ (https://shiny.rstudio.com/gallery/)**

Remarques generales

- Une application **shiny** etant au format HTML, elle ne peut etre incorporee dans des documents non interactifs (formats word, powerpoint, pdf, …)
- Il est possible de l'inclure dans une presentation au format HTML pour obtenir des pages interactives (voir plus loin)
- On peut aussi inclure des applications ou des syntaxes de type shiny dans des documents RMarkdown
- La redaction de telles applications:
  - demande de respecter une syntaxe bien particuliere
  - combine des outils classiques de R avec des commandes specifiques pour gerer les interactions

Interface utilisateur et serveur

Une application **shiny** se structure en deux parties ayant des roles specifiques:

- Un cote **UI** qui regroupe tous les elements de mise en forme et d'affichage del'interface utilisateur (notamment l'affichage des **inputs** et des **outputs**)

- Un cote **Serveur** qui regroupe les codes R permettant de construire les **outputs** (graphiques, etudes statistiques, tables, …) et de les mettre a jour si la valeur d'un des **inputs** dont ils dependent est modifiee.

Une grande panoplie de **widgets** permettra a l'utilisateur de prendre en main et adapter a sa convenance les **inputs** utilises.

Interface utilisateur et serveur

Pour plus de clarte, il est conseille de separer ces deux parties de l'application en deux fichiers bien distincts: **ui.R** et **server.R**

On les enregistre dans un meme dossier portant le nom de l'application que l'on souhaite creer.

On pourra egalement ajouter dans le meme dossier:

- un fichier **global.R** dans lequel on mettra les commandes a executer une seule fois au lancement de l'application
- un sous-dossier **www** contenant les documents utiles au navigater qui ne sont pas issus de scripts R.
- des sous-dossiers contenant les donnees, les resultats enregistres par l'application, ...

Interface utilisateur - Mise en page

Contenu

Cette partie de l'application est tres specifique a Shiny.

Elle traite de :

- la mise en forme de l'interface utilisateur
- la creation des inputs (et widgets associes)
- la creation des outputs (dont la generation/mise a jour sera faite par la partie serveur)

Interface utilisateur

Un premier exemple:

```
shinyUI(fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  sidebarLayout(
      sidebarPanel(
         sliderInput("bins","Number of bins:", min = 1,
                     max = 50,value = 30)
                  ),
      mainPanel(plotOutput("distPlot"))
                )
))
```

Syntaxe et elements de base

- Shiny utilise la fonction **fluidPage** pour creer un affichage qui s'ajuste automatiquement aux dimensions de la fenetre du navigateur de l'utilsateur.
- L'interface utilisateur est ensuite construite en placant les elements souhaites a l'interieur de la fonction **fluidPage**.
- Differentes possibilites de mise en forme s'offrent a vous:
  - titlePanel (panneau titre)
  - sidebarLayout (barre laterale, panneau principal)
  - fluidRow (decoupage personnalise de la fenetre en colonnes)
  - tabsetPanel (creation d'onglets)
  - navlistPanel (creation d'un menu de navigation)

Documentation[1] sur le site de shiny.

Interface utilisateur - Mise en page

sidebarLayout 1/3

Parametres de la fonction sidebarLayout

- **sidebarPanel**: barre laterale
- **mainPanel**: panneau principal
- **position**: ‘left’ ou ‘right’ - emplacement de la barre laterale
- **fluid**: TRUE ou FALSE - ajustement autommatique de l’affichage

Exemple:

```
sidebarLayout
    (
      sidebarPanel("Barre laterale",width=5),
      mainPanel("Panneau principal",width=7),
      position='right'
    )
```

sidebarLayout 3/3

Nouvelle version de **ui.R**

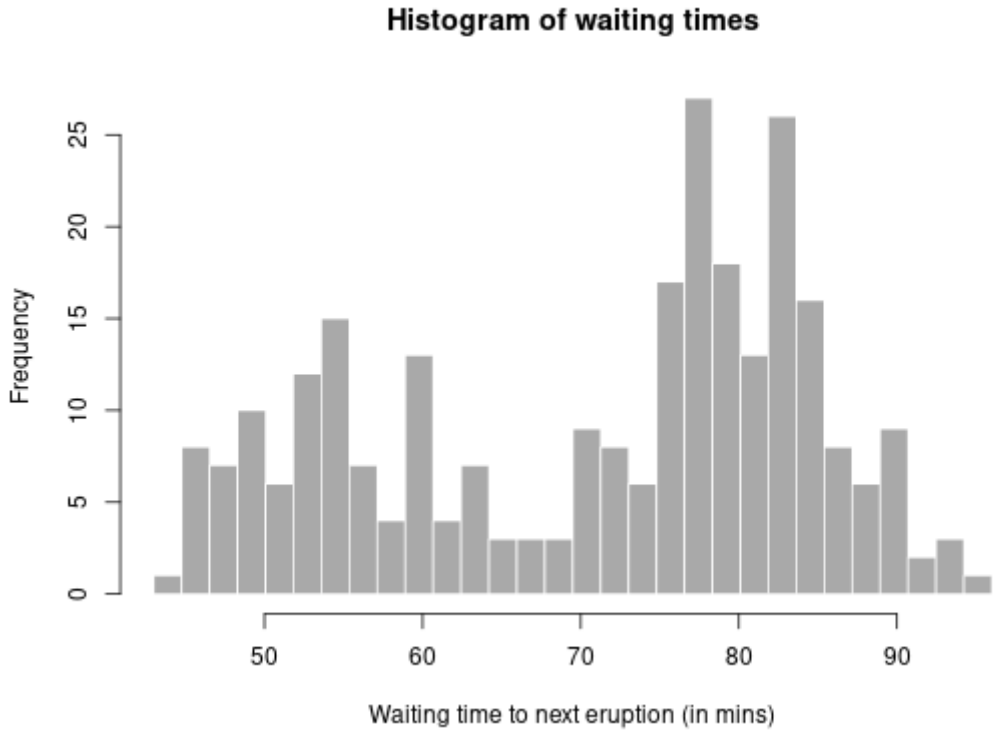
```
library(shiny)
fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  sidebarLayout
    (
      sidebarPanel("Barre laterale",
                   sliderInput("bins","Number of bins:", min = 1,max = 50,value = 30),
                   width=5),
      mainPanel("Panneau principal",
                plotOutput("distPlot"),
                width=7),
      position='right'
    )
)
```

sidebarLayout 3/3

Résultat:

Old Faithful Geyser Data

Panneau principal



Barre laterale

Number of bins:

1 30 50

1 6 11 16 21 26 31 36 41 46 50

fluidRow (decoupage personnalise) 1/3

On utilsera les fonctions

- **column (width,...,offset=0)**: definition d'une colonne
  - **width**: largeur de la colonne entre 1 et 12
  - **...**: contenu de celle-ci
  - **offset**: decalage par rapport a la colonne precedente
- **wellPanel**: cree un panneau interne

Exemple:

```
fluidRow
    (
      column(1,"colonne 1"),
      column(4,wellPanel("colonne 2"),2),
      column("colonne 3",width=5)
    )
```

fluidRow 2/3

Nouvelle version de **ui.R**

```
library(shiny)
fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  fluidRow
    (
      column(2,wellPanel("colonne 1",
            sliderInput("bins","Number of bins:", min = 1,max = 50,value = 30))
            ),
      column(2,wellPanel("colonne 2"),offset=1),
      column(wellPanel("colonne 3",
             plotOutput("distPlot")),
             width=7)
    )
)
```

fluidRow 3/3

Resultat:

Old Faithful Geyser Data

colonne 1

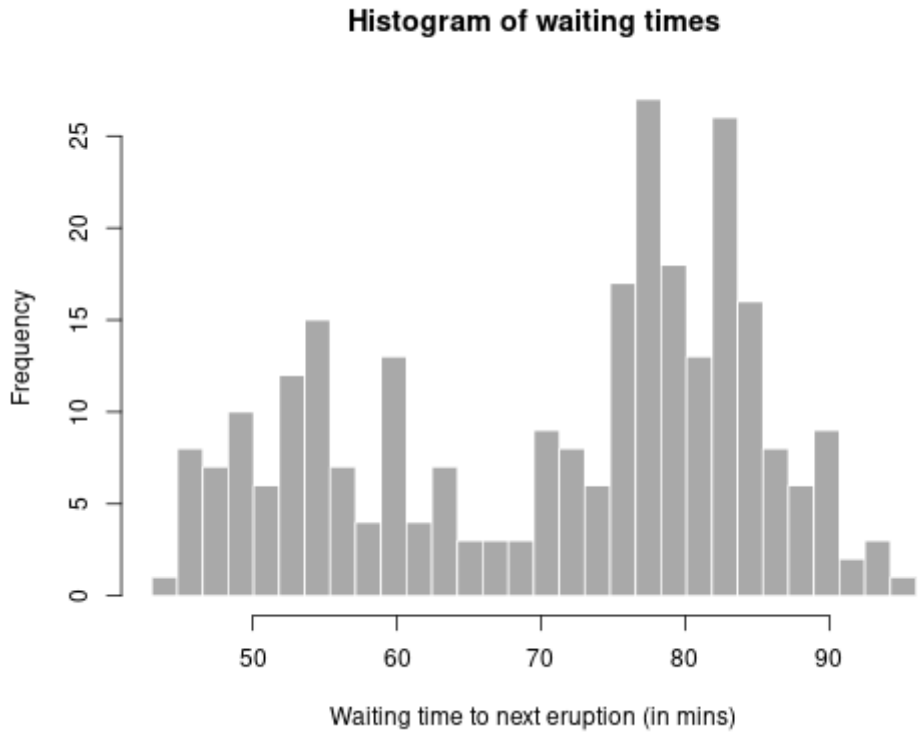
Number of bins:

1 30 50

1 11 21 31 41 50

colonne 2

colonne 3



tabsetPanel (creation d'onglets) 1/3

La fonction **tabsetPanel** prend les arguments suivants

- **...**: elements a inclure dans l'onglet
- **id**: si renseignee, ***input$id*** renverra la valeur de l'onglet actif.
- **selected**: onglet selectionne par defaut (si NULL, le premier).
- **type**: "tabs" ou "pills" apparence des onglets.

Exemple:

```
tabsetPanel
     (
        tabPanel("Plot", plotOutput("plot")),
        tabPanel("Summary", verbatimTextOutput("summary")),
        tabPanel("Table", tableOutput("table")),
        selected="Summary",id='Choix',type="pills"
      )
```

tabsetPanel 2/3

Nouvelle version de **ui.R**

```
library(shiny)
fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  tabsetPanel
     (
        tabPanel("Plot", plotOutput("distPlot")),
        tabPanel("Bins number",
                 sliderInput("bins","Number of bins:", min = 1,max = 50,value = 30)),
       tabPanel("Paper",'
                 Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Fa
                ),
        selected="Paper",id='Choix',type="pills"
      )
)
```

tabsetPanel 3/3

Resultat:

Old Faithful Geyser Data

Plot Bins number Paper

Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. Applied Statistics, 39, 357–365.

navlistPanel (cree un menu de navigation) 1/4

La fonction **navlistPanel** prend les arguments suivants

- **...** contenu des pages internes
- **id**: si renseignee, ***input$id*** renverra la valeur de l'onglet actif.
- **selected**: onglet selectionne par defaut (si NULL, le premier).
- **well**: TRUE ou FALSE indique si l'on doit separer le panneau contenant le menu.
- **fluid**: TRUE ou FALSE - ajustement autommatique de l'affichage
- **widths**: largeurs respectives du menu et des pages internes

navlistPanel 2/4

Exemple

```
navlistPanel
  (
      "Entete",
      tabPanel("Premier", h1("C'est le premier panneau")),
      tabPanel("Second",h2("C'est le second panneau")),
      tabPanel("Troisieme",h3("C'est encore un panneau")),
      well=TRUE,widths=c(2,10),selected="Troisieme"
  )
```

navlistPanel 3/4

Nouvelle version de **ui.R**

```
library(shiny)
fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  navlistPanel
    (
      "Menu",
       tabPanel("Plot", plotOutput("distPlot")),
       tabPanel("Bins number",
                 sliderInput("bins","Number of bins:", min = 1,max = 50,value = 30)),
      tabPanel("Paper",'
                 Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Fa
               ),
      well=TRUE,widths=c(2,10),selected="Paper"
      )
)
```

navlistPanel 4/4

Resultat:

Old Faithful Geyser Data

Menu

Plot

Bins number

Paper

Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. Applied Statistics, 39, 357–365.

navbarPage (comportant differentes pages) 1/4

- **navbarPage** permet de creer differentes pages avec des organisations pouvant etre differentes au sein d'une meme application shiny.

De nombreuses options de personnalisation sont disponibles (id = NULL, selected = NULL, position = c("static-top", "fixed-top", "fixed-bottom"), header = NULL, footer = NULL, inverse = FALSE, collapsible = FALSE, collapsable, fluid = TRUE, responsive = NULL, theme = NULL, windowTitle = title)

- On pourra aussi creer des sous-rubriques avec la fonction **navbarMenu**.

navbarPage (Exemple) 2/4

```
navbarPage
      (
          "Mon Application",
           tabPanel("Composant 1"),
           tabPanel("Composant 2"),
           navbarMenu
                (
                "Plus",
                tabPanel("Sous-Composant A"),
                tabPanel("Sous-Composant B")
                )
       )
```

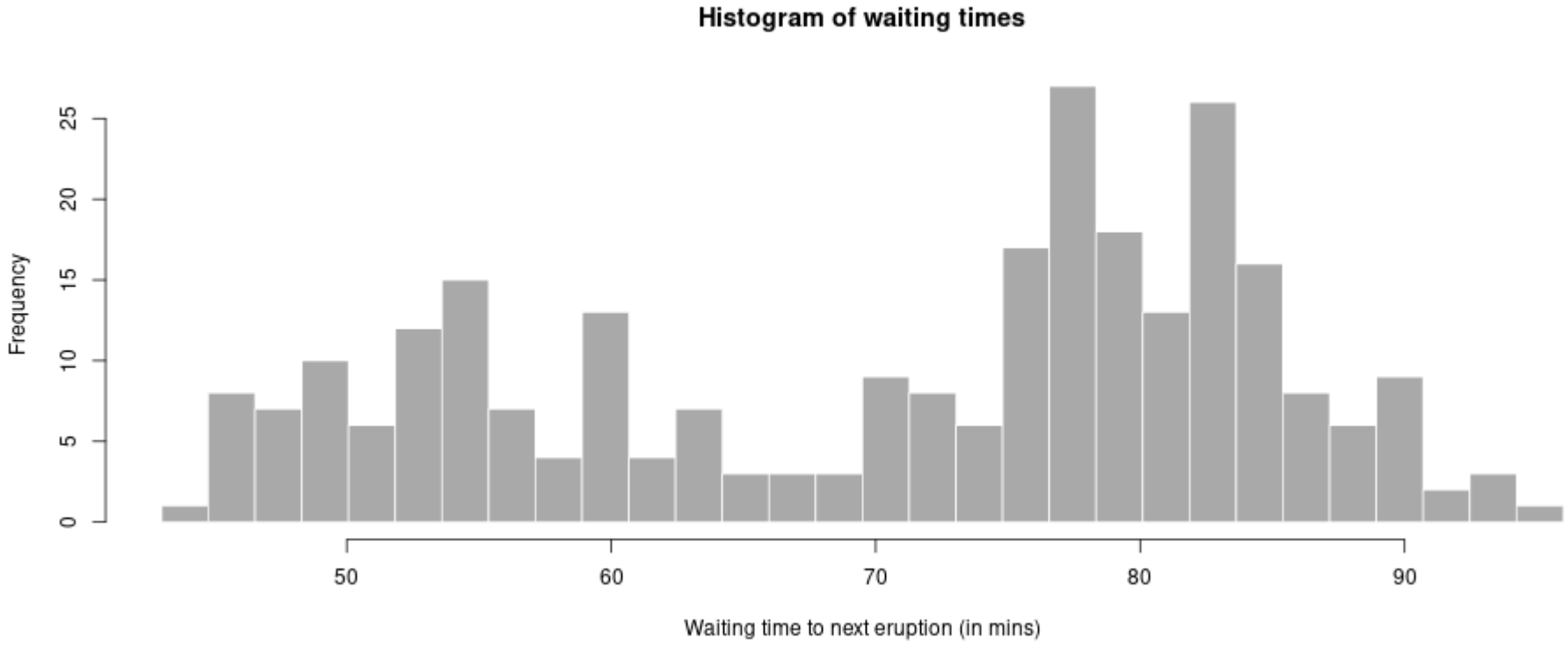
navbarPage 3/4

Nouvelle version de **ui.R**

```
library(shiny)
fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  navbarPage
     (
         "Old Faithful Geyser Data",
          tabPanel("Plot", plotOutput("distPlot")),
          tabPanel("Bins number",
                 sliderInput("bins","Number of bins:", min = 1,max = 50,value = 30)),
          navbarMenu
               (
               "References",
               tabPanel("Härdle",'Härdle, W. (1991). Smoothing Techniques with Implementa
               tabPanel("Azzalini and Bowman",'Azzalini, A. and Bowman, A. W. (1990). A l
               )
      ))
```

navbarPage 4/4

Resultat:



Incorporation de code HTML 1/2

Fonction	Effet
p	insertion d'un paragraphe
hi	titre de niveau i, $i = 1, \ldots, 6$
a	insertion d'un lien HTML
br	insertion d'une ligne vide
div	changement de style d'un paragraphe de texte
span	changement de style de texte dans un paragraphe
pre	texte tel qu'il est

Incorporation de code HTML 2/2

Fonction	Effet
code	insertion de code sans modifier son format
img	insertion d'image
strong	texte en gras
em	texte en italique

Documentation detaillee[2] et exemples sur le site de shiny.

Interface utilisateur - Inputs and Widgets

Inputs and Widgets (win. gadgets) 1/2

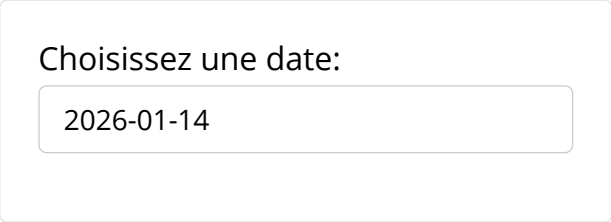
Fonction	Effet
actionButton	bouton d'action
checkboxGroupInput	ensemble de cases a cocher
checkboxInput	case a cocher
dateInput	choix d'une date
dateRangeInput	choix d'une plage de dates
fileInput	outil permettant de charger un fichier
helpText	message d'aide associe a un formulaire

Inputs and Widgets (win. gadgets) 2/2

Fonction	Effet
numericInput	champ de saisie une valeur numerique
radioButtons	choix d'une options parmi plusieurs
selectInput	menu deroulant
sliderInput	barre de defilement
submitButton	bouton de soumission
textInput	champ de saisie d'un texte

Plus de details ici (http://rmarkdown.rstudio.com/authoring_shiny.html).

Selection de dates

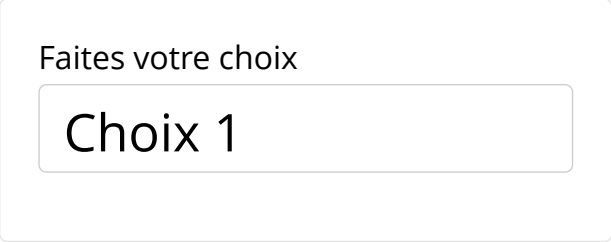


dateInput(inputId = 'Date', label = 'Choisissez une date:')



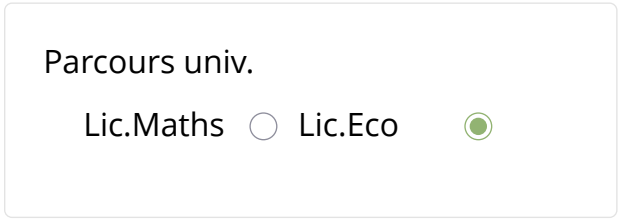
dateRangeInput(inputId = 'Date', label = 'Choisissez une date:')

Selection par menu deroulant



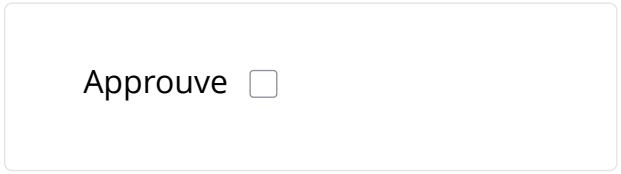
selectInput(inputId='Menu', label='Faites votre choix', choices=c('Choix 1','Choix 2','Choix 3') ,multiple = TRUE, selected='Choix 1')

Boutons radio

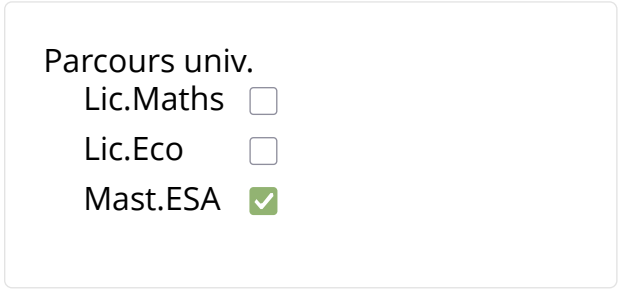



```
radioButtons(inputId = 'Cases', label = 'Parcours
univ.',choices=c('Lic.Maths', 'Lic.Eco')
,selected='Lic.Eco',inline=TRUE)
```

Cases a cocher

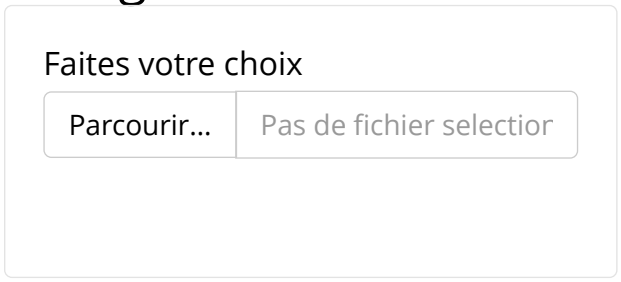



```
checkboxInput(inputId = 'Case', label =
'Approuve',value=FALSE)
```




```
checkboxGroupInput(inputId = 'Cases', label =
'Parcours univ.',c('Lic.Maths', 'Lic.Eco', 'Mast.ESA'
),selected='Mast.ESA')
```

Chargement de fichiers



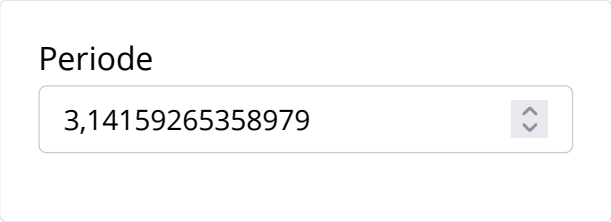

```
fileInput(inputId='File_s', label='Faites votre choix',
multiple = TRUE, buttonLabel = 'Parcourir...',
placeholder = 'Pas de fichier selectionne')
```

Saisie d'un texte



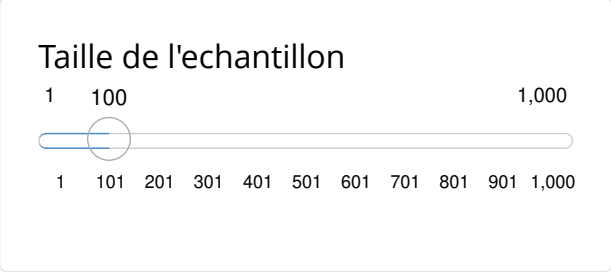

```
textInput(inputId='Texte', label='Nom et
```

Saisie d’une valeur numerique



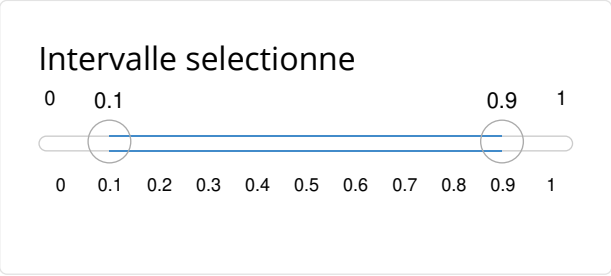

```
numericInput(inputId='Per', label='Periode',value=pi)
```

Selection d’une valeur numerique




```
sliderInput(inputId='n', label='Taille de l'echantillon',
min=1, max=1000, value=100, step=1)
```

Selection d’un intervalle




```
sliderInput(inputId='Int', label='Intervalle
selectionne', min=0, max=1, value=c(0.1,0.9),
step=.01)
```

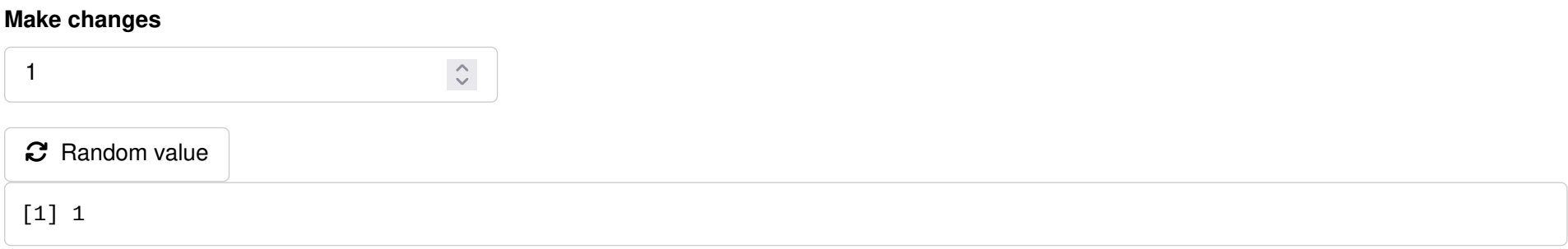
Bouton de soumission

```
shinyApp(
  ui =  basicPage(
            sliderInput(inputId='n', label='Taille de l\'echantillon', min=1, max=1000, v
            submitButton(text='Mettre a jour', icon=icon('arrow-alt-circle-right'), width
            verbatimTextOutput("value")
            ),
server = function(input, output) {
  output$value <-renderPrint({input$n})
                                  }
)
```



Bouton d’action

```
shinyApp(
  ui = basicPage(
    numericInput("num", label = "Make changes", value = 1),
    actionButton("update" ,"Random value", icon("refresh")),
    verbatimTextOutput("value")
  ),
  server = function(input, output) {
     output$value <- renderPrint({input$update;rbinom(1,max(1,isolate(input$num)),.5)})
  }
)
```



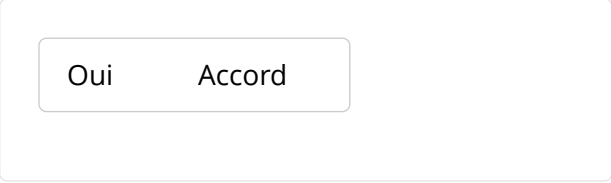
Quelques autres widgets (package shinyWidgets)

Boutons radio comme onglets



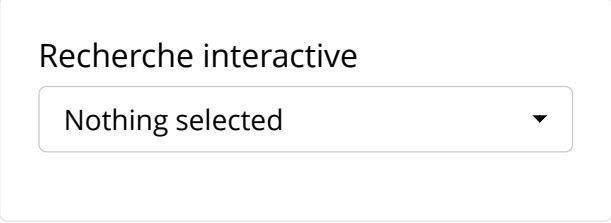

```
radioGroupButtons( inputId = 'Avis', label =
'Opinion:', choices = c('Oppose', 'Indifferent',
'Favorable'),checkIcon = list( yes = icon('ok', lib
= 'glyphicon')) )
```

Choix oui/non colore



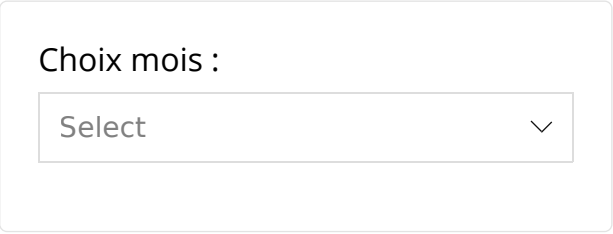

```
switchInput( inputId = 'Int', onLabel = 'Oui', offLabel =
'Non', label = 'Accord', onStatus = 'success', offStatus
= 'danger', value = TRUE, labelWidth = '80px' )
```

Menu deroulant avec recherche et selection/deselection groupée



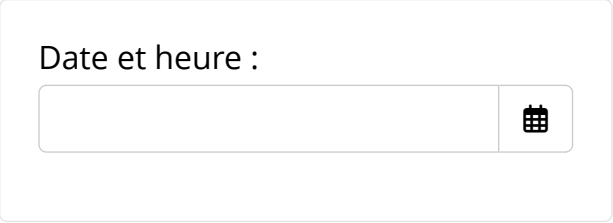

```
pickerInput( inputId = 'Menu', label = 'Recherche
interactive', choices = (data()[[3]])[,3], multiple=TRUE,
options = pickerOptions(container = 'body',
liveSearch = TRUE,actionsBox=TRUE,size = 4) )
```

Menu deroulant avec selection/deselection par groupes



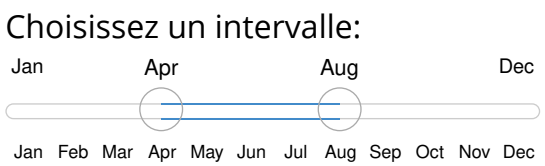

```
virtualSelectInput( inputId = 'Menu', label = 'Choix mois :', choices = list( Winter = month.name[1:3], Spring = month.name[4:6], Summer = month.name[7:9], Fall = month.name[10:12]), multiple = TRUE, allowNewOption = TRUE, dropboxWrapper = 'body')
```

Choix date et heure




```
airDatepickerInput( inputId = 'Date_&_Time', label = 'Date et heure :', dateFormat = 'dd/MM/yyyy', language = 'fr', timepicker = TRUE,minDate =Sys.Date(), disabledDates = Sys.Date(), disabledDaysOfWeek =c(0,6), timepickerOpts = timepickerOptions(hoursStep = 1,minutesStep = 10)
```

Barre de défilement pour variable qualitative




```
sliderTextInput( inputId = 'Mois', label = 'Choisissez un intervalle:', choices = month.abb, selected = month.abb[c(4, 8)], grid = TRUE, force_edges = TRUE )
```

Boutons d'action

Cliquez ici!

```
actionBttn( inputId = 'Bouton1', label = 'Cliquez ici!', style = 'fill', color = 'primary', )
```

Cliquez ici!

```
actionBttn( inputId = 'Bouton2', label = 'Cliquez ici!', style = 'jelly', color = 'success', icon = icon('sliders') )
```

Cliquez ici!

```
actionBttn( inputId = 'Bouton3', label = 'Cliquez ici!', style = 'stretch', color = 'warning', icon = icon('bars') )
```

```
actionBttn( inputId = 'Bouton4', style = 'material-circle', color = 'danger', icon = icon('gear') )
```

Bouton menu



Code

```
dropdownButton(
    tags$h3('Liste des parametres'),
    selectInput(inputId = 'xcol',label = ' Variable X',choices = names(iris)),
    selectInput(inputId = 'ycol',label = ' Variable Y',choices = names(iris),
                selected = names(iris)[[2]]),
    sliderInput(inputId = 'clusters',label = 'Nombre de clusters',value = 3,
                min = 1,max = 9),
    circle = TRUE, status = 'primary',icon = icon('gear'), width = '300px',
    tooltip = tooltipOptions(title = 'Cliquez pour voir les parametres !')
  )
```

Bouton menu avec animation



Code

```
dropdown(
    tags$h3('Liste des parametres'),
    selectInput(inputId = 'xcol',label = ' Variable X',choices = names(iris)),
    selectInput(inputId = 'ycol',label = ' Variable Y',choices = names(iris),
                selected = names(iris)[[2]]),
    sliderInput(inputId = 'clusters',label = 'Nombre de clusters',value = 3,
                min = 1,max = 9),
    style = "unite", icon = icon("gear"), status = "warning", width = "400px",
    animate = animateOptions
      (
        enter = animations$fading_entrances$fadeInLeftBig,
        exit = animations$fading_exits$fadeOutRightBig
      ),
    tooltip = tooltipOptions(title = 'Cliquez pour voir les parametres !')
  )
```

Plus d'exemples: `shinyWidgetsGallery()`

Interface utilisateur - Output

Outputs

fonction	output cree
dataTableOutput	Table de donnees
htmlOutput	HTML brut
imageOutput	image
plotOutput	graphique
tableOutput	table
textOutput	texte
uiOutput	HTML brut
verbatimTextOutput	texte “verbatim”

Serveur

Serveur

Un premier exemple:

```
library(shiny)
function(input, output, session) {
    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- faithful[, 2]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)
        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = 'darkgray', border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
             main = 'Histogram of waiting times')

    })

}
```

Contenu

Cette partie de l'application est celle qui ressemble plus a du R standard.

La difference majeure reside dans:

- la generation et la mise a jour des outputs.
- la gestion de la reactivite via la prise en compte des inputs

Cela se fait via la commande **output$Nom_output=render...({Code_R })** ou **...** est remplacé par le type d'output souhaité (voir plus loin).

La reactivite s'obtient en inserant, dans les codes R generant les outputs, les valeurs des inputs mises a jour via les widgets. Cela se fait via la commande **input$inputID**.

Generation des outputs

fonction	output genere
renderDataTable	Table de donnees
renderImage	images
renderPlot	graphiques
renderPrint	tout resultat “affiche”
renderTable	data frame, matrice, …
renderText	chaine de caracteres
renderUI	objet de type shiny

Gestion de la reactivite

La reactivite s'obtient en inserant, dans les codes R generant les outputs, les valeurs des inputs mises a jour via les widgets.

Cela se fait via la commande **input$inputID**.

On pourra utiliser la valeur d'un input sans mettre a jour les output a chaque changement de celui-ci en utilisant la fonction **isolate**.

La fonction **req** peut etre utile pour ne pas evaluer les lignes qui suivent lorsqu'une valeur n'est pas encore attribuee à une variable.

On pourra enfin utiliser des fonctions telles que **reactive**, **reactiveValues**, **eventReactive**, **observe** et **observeEvent**.

Exemple avec l'application Kmeans

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

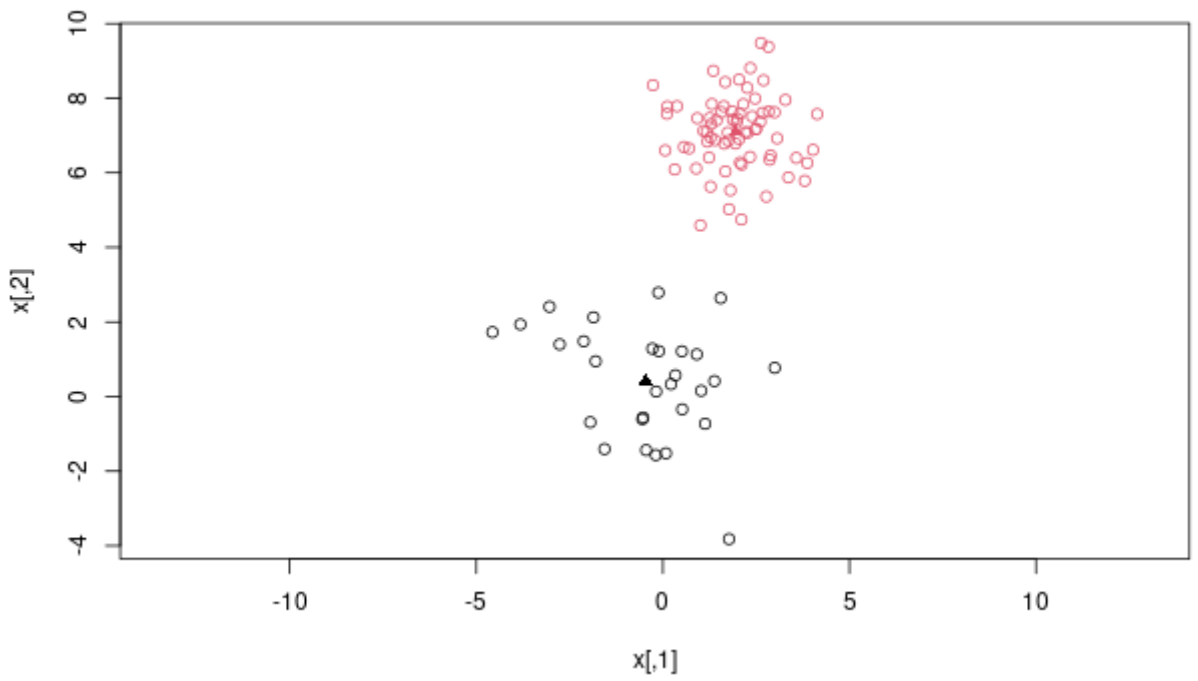
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Application Kmeans - Partie UI

```
library(shiny)
library(MASS)
fluidPage(
  sidebarLayout(
    sidebarPanel(
      sliderInput("n", label = "Choix de la taille d'échantillon",min=10,max=1000, value
      sliderInput("k", label = "Choix du nombre de classes",min=1,max=9, value = 2),
      sliderInput("ninit", label = "Choix du nombre d'initialisations",min=1,max=100, val
      selectInput('algo','Choix de l\'algorithme',choices=c("Hartigan-Wong", "Lloyd", "Fo
      actionButton("update" ,"Rafraichir", icon("refresh")),
      checkboxInput('centers','Afficher les barycentres',value=TRUE)),
    mainPanel(
      plotOutput("kmplot")
    )))
```

Application Kmeans - Partie Serveur

```
library(shiny)

function(input, output, session) {
  output$kmplot <- renderPlot({
    n=100;k=2;ninit=10;algo="Lloyd"
    z=rbinom(n,1,0.3)
    x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    plot(x,col=km$cluster,asp=1)
    if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}  })
}
```

Activation du bouton Rafraichir

```
library(shiny)
function(input, output, session) {
  output$kmplot <- renderPlot({
    input$update
    n=100;k=2;ninit=10;algo="Lloyd"
    z=rbinom(n,1,0.3)
    x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    plot(x,col=km$cluster,asp=1)
    if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
  })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

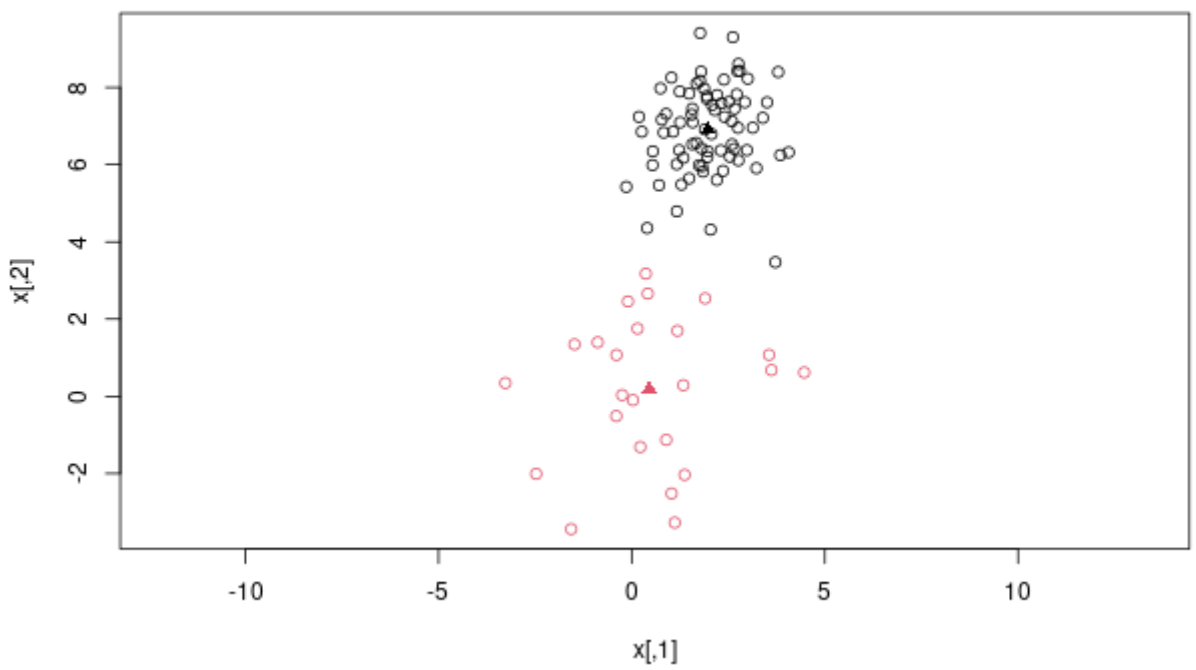
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Puis prise en compte des mises à jour

```
library(shiny)
function(input, output, session) {
  output$kmplot <- renderPlot({
    input$update
    n=input$n;k=input$k;ninit=input$ninit;algo=input$algo
    z=rbinom(n,1,0.3)
    x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    plot(x,col=km$cluster,asp=1)
    if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
  })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

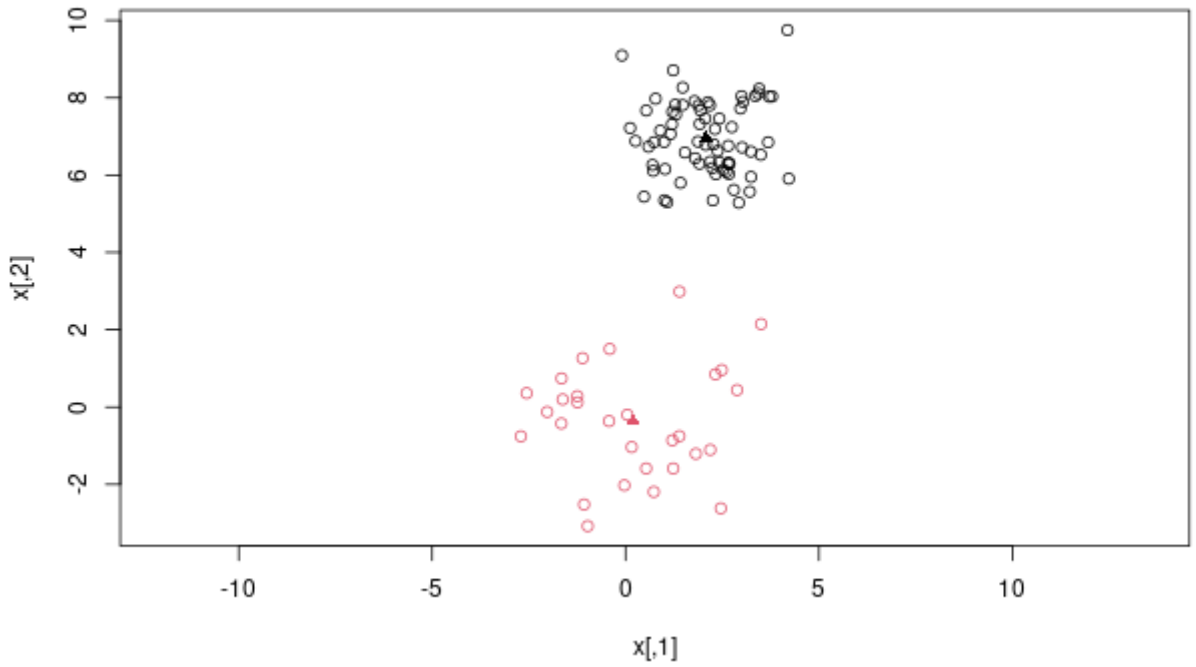
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Ne mettre à jour qu'avec le bouton Rafraichir

```
library(shiny)
function(input, output, session) {
  output$kmplot <- renderPlot({
    input$update
    n=isolate(input$n);k=isolate(input$k);ninit=isolate(input$ninit);algo=isolate(input$a
    z=rbinom(n,1,0.3)
    x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    plot(x,col=km$cluster,asp=1)
    if (isolate(input$centers)) {if (input$centers) {points(km$centers,col=sort(unique(km
  })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

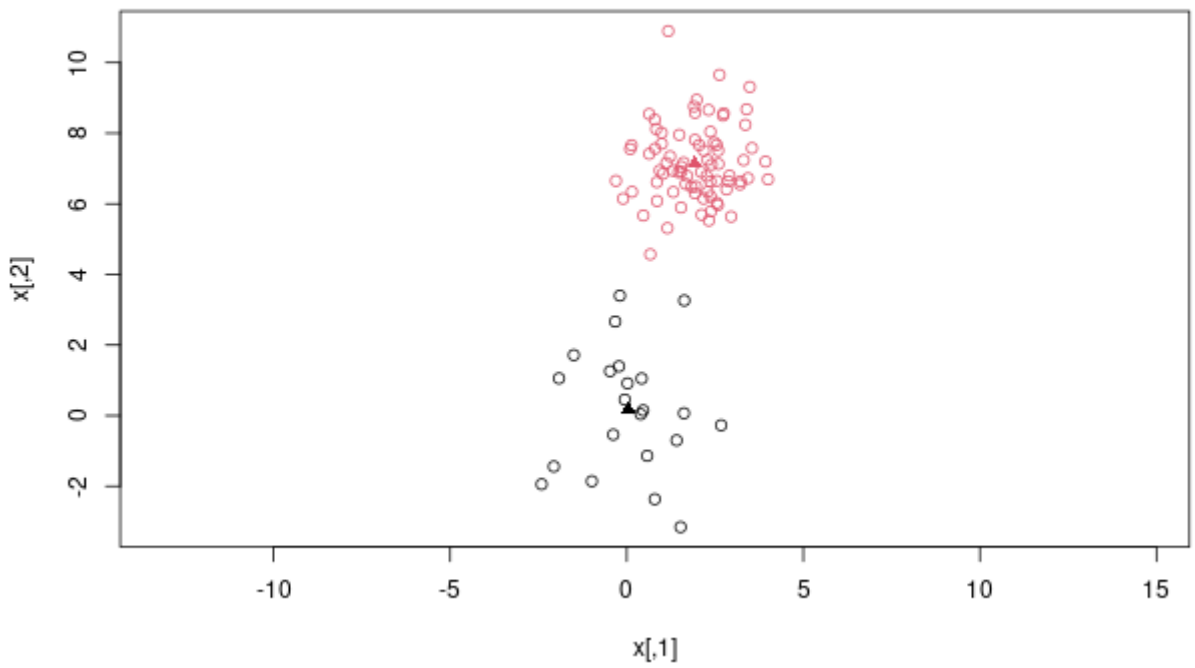
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Gestion plus fine de la réactivité - Mise a jour plus adaptee des observations

Avec la fonction reactive

```
library(shiny)
function(input, output, session) {
  data=reactive({
    input$update; n=input$n
    z=rbinom(n,1,0.3)
    x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    })
  km=reactive({
    x=data();k=input$k;ninit=input$ninit;algo=input$algo
    km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    })
  output$kmplot <- renderPlot({
    x=data();km=km();
    plot(x,col=km$cluster,asp=1)
    if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}    })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

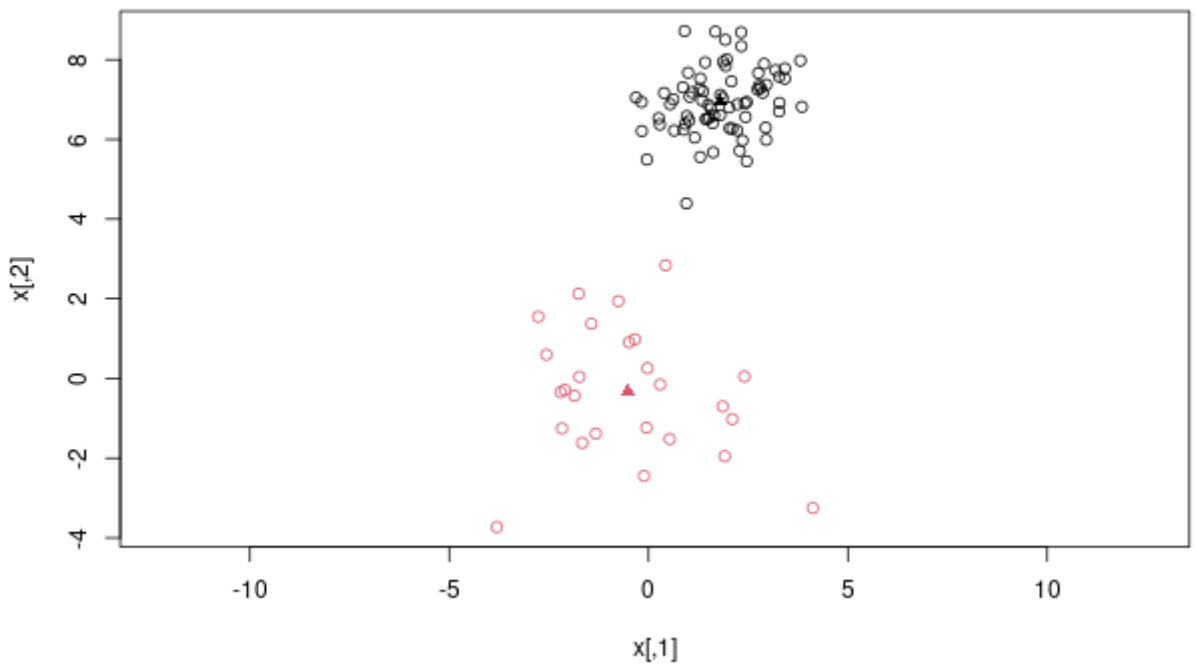
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Avec la fonction eventReactive

```
library(shiny)
function(input, output, session) {
        data=eventReactive({input$update; input$n},{
          n=input$n
          z=rbinom(n,1,0.3)
          x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
        })
        km=eventReactive({input$update;input$n;input$k;input$ninit;input$algo},{
          x=data();k=input$k;ninit=input$ninit;algo=input$algo
          kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
        })
        output$kmplot <- renderPlot({
          x=data();km=km();
          plot(x,col=km$cluster,asp=1)
          if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
        })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

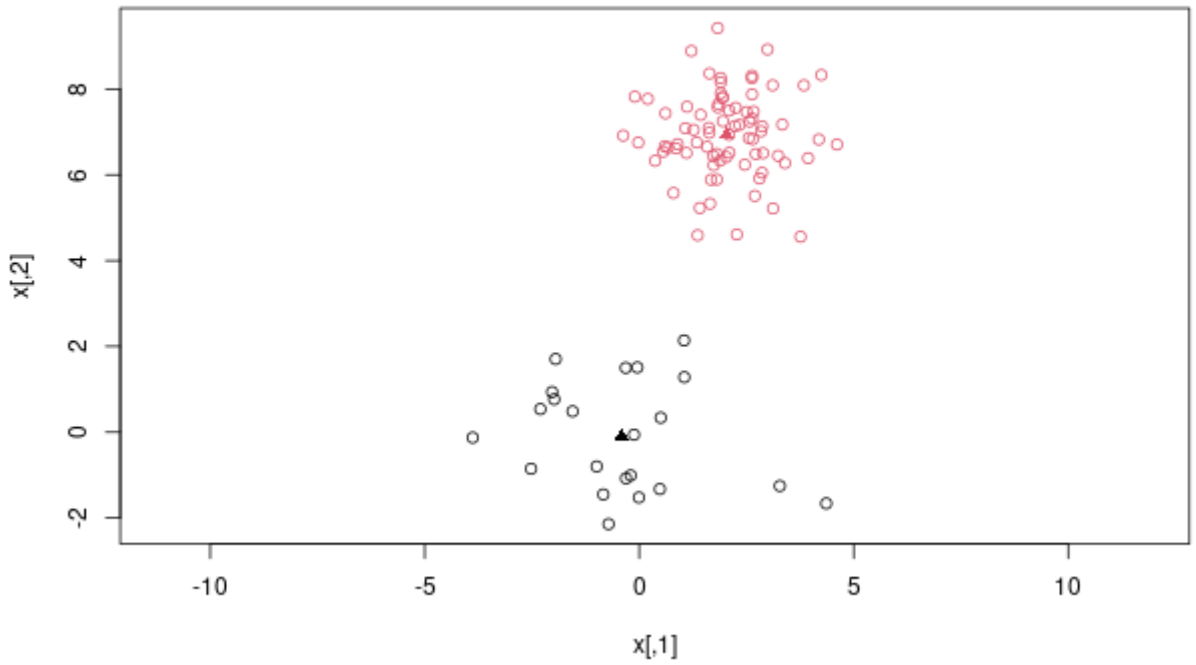
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Avec reactiveValues et observe

```
library(shiny)
function(input, output, session) {
    values=reactiveValues(x=NULL,km=NULL)
    observe({
      input$update; n=input$n
      z=rbinom(n,1,0.3)
      values$x=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    })
    observe({
      x=values$x;k=input$k;ninit=input$ninit;algo=input$algo
      values$km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    })
    output$kmplot <- renderPlot({
      x=values$x;km=values$km;
      plot(x,col=km$cluster,asp=1)
      if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
    })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

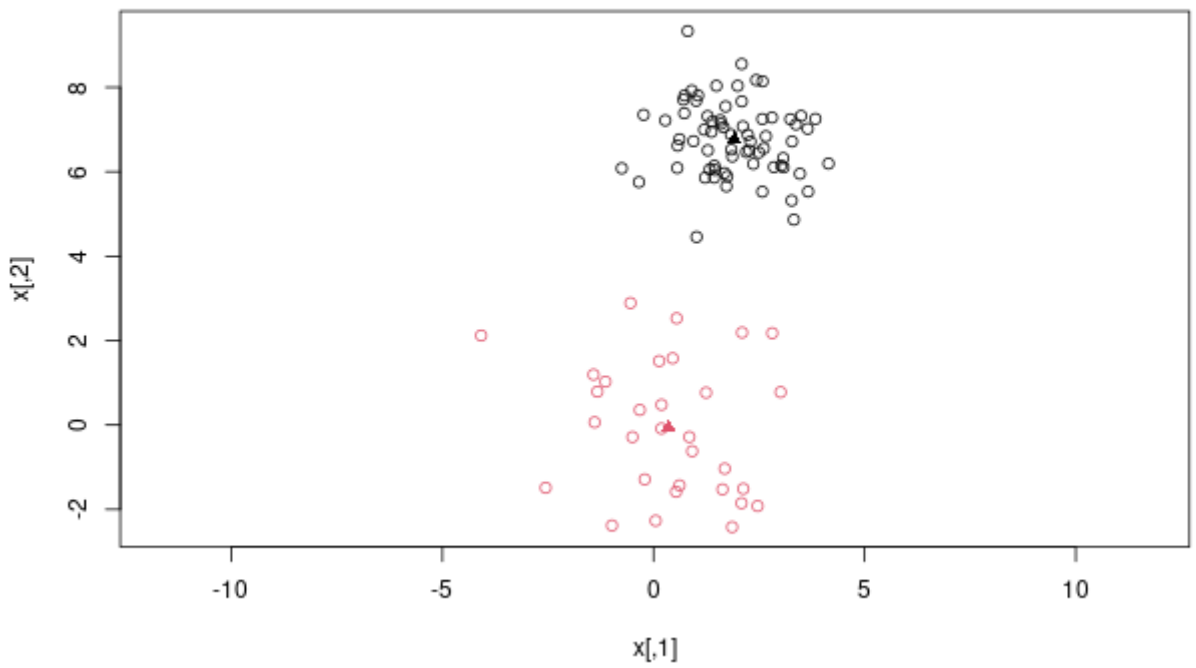
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Avec reactiveValues et observeEvent

```
library(shiny)
function(input, output, session) {
  values <- reactiveValues(x=NULL,km=NULL)
  observeEvent({input$update;input$n},
               {
               z=rbinom(input$n,1,0.3)
               values$x=z*mvrnorm(input$n,c(0,0),3*diag(2))+(1-z)*mvrnorm(input$n,c(2,7
               })
  observeEvent({input$update; input$n; input$k;input$ninit;input$algo},
               {
               values$km=kmeans(values$x,centers=input$k,nstart=input$ninit,algorithm=i
               })
  output$kmplot <- renderPlot({
      x=values$x;km=values$km;
      plot(x,col=km$cluster,asp=1)
      if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
    })
}
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

1 2 9

1 2 3 4 5 6 7 8 9

Choix du nombre d'initialisations

1 10 100

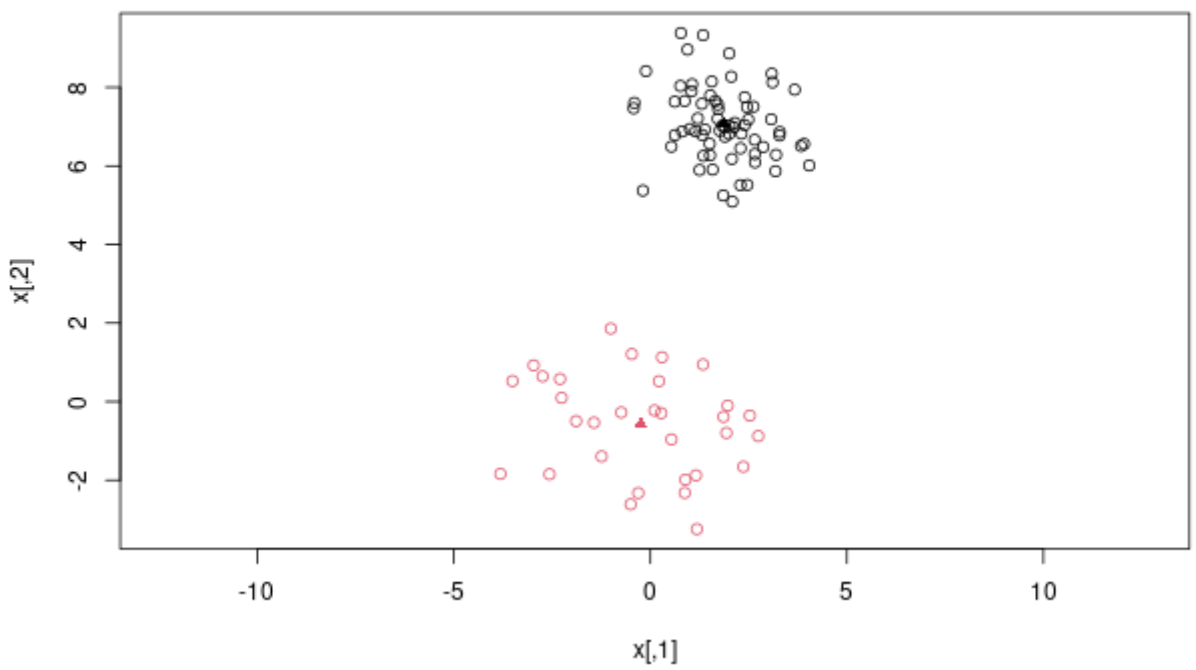
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Widget dépendant d’un autre…

Partie UI: Remplacer le sliderInput par uiOutput

```
library(shiny)

fluidPage(
    sidebarLayout(
      sidebarPanel(
        sliderInput("n", label = "Choix de la taille d'échantillon",min=10,max=1000, valu
        uiOutput('kc'),
        sliderInput("ninit", label = "Choix du nombre d'initialisations",min=1,max=100, v
        selectInput('algo','Choix de l\'algorithme',choices=c("Hartigan-Wong", "Lloyd", "
        actionButton("update" ,"Rafraichir", icon("refresh")),
        checkboxInput('centers','Afficher les barycentres',value=TRUE)),
      mainPanel(
        plotOutput("kmplot")
      )))
```

Partie Serveur:

```
library(shiny)
function(input, output, session) {
    values=reactiveValues(data=NULL,km=NULL)
    observe({val=ifelse(is.null(input$k),2,min(input$k,input$n-1))
      output$kc=renderUI(sliderInput("k", label = "Choix du nombre de classes",min=1,max=
    })
    observe({input$update; n=input$n; z=rbinom(n,1,0.3)
      values$data=z*mvrnorm(n,c(0,0),3*diag(2))+(1-z)*mvrnorm(n,c(2,7),1*diag(2))
    })
    observe({x=values$data;k=min(input$k,input$n-1);ninit=input$ninit;algo=input$algo
      values$km=kmeans(x,centers=k,nstart=ninit,algorithm=algo,iter.max = 100)
    })
    output$kmplot <- renderPlot({
      input$update;x=values$data;km=values$km;
      plot(x,col=km$cluster,asp=1)
      if (input$centers) {points(km$centers,col=sort(unique(km$cluster)),pch=17)}
    })
 }
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

2 99

1 11 21 31 41 51 61 71 81 91 99

Choix du nombre d'initialisations

1 10 100

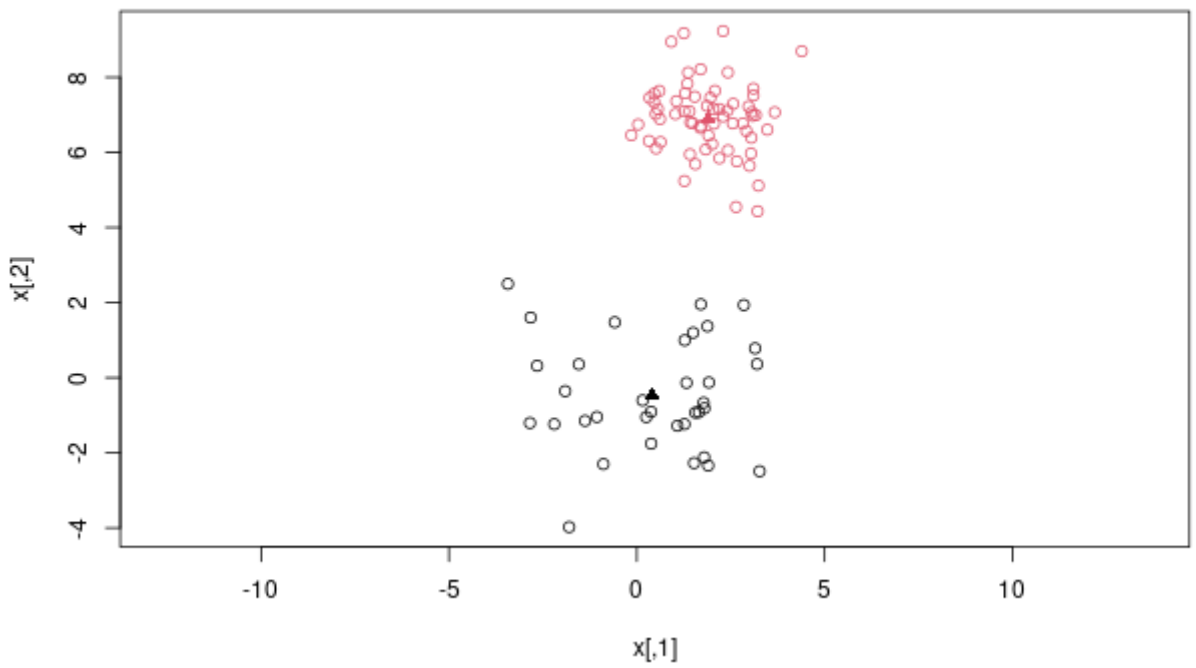
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Enregistrer les observations

Enregistrer les observations

- On rajoute un bouton de téléchargement des observations au niveau du fichier ui.R via la commande `downloadButton("Tele_Obs", "Telecharger les observations")` ou `"Tele_Obs"` est le nom que vous avez choisi pour la fonction qui compilera le .Rmd et `"Telecharger les observations"` est le message qui apparaitra sur le bouton.

- On rajoute une commande pour effectuer le telechargement, comme par exemple ici au format **.RData**, au niveau du fichier server.R.

```
output$Tele_Obs <- downloadHandler(
     filename = 'data.RData',
     content = function(file) {
       x=values$data; km=values$km
       save(list=c('x','km'), file=file)
     }
   )
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

2 99

1 11 21 31 41 51 61 71 81 91 99

Choix du nombre d'initialisations

1 10 100

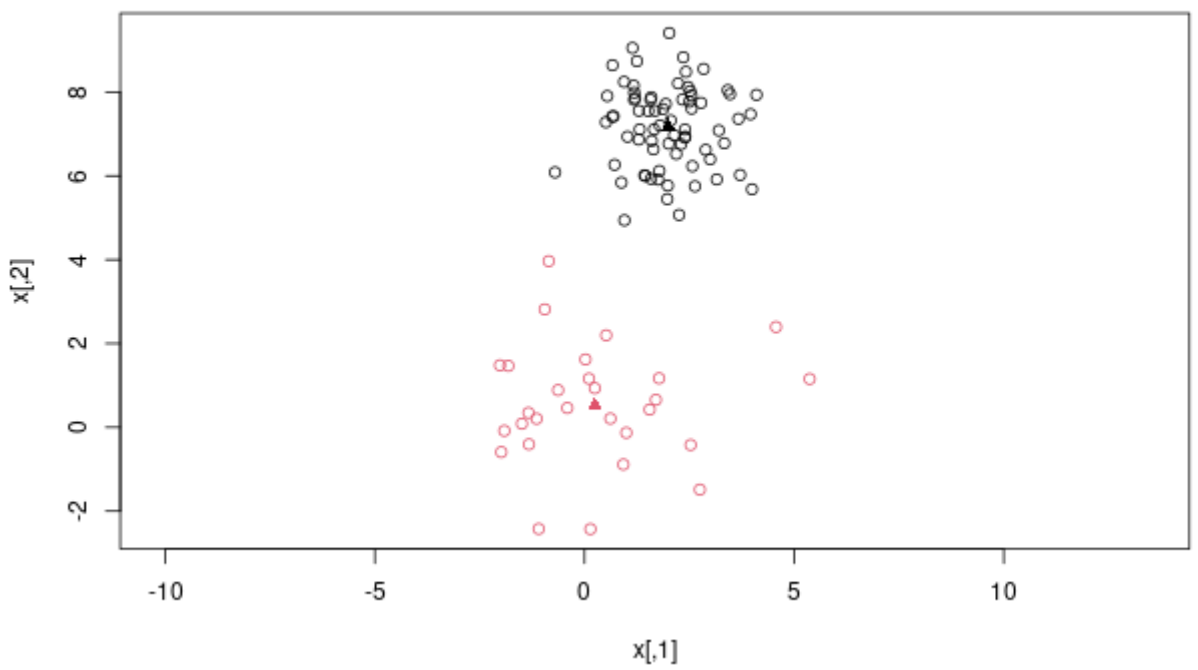
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Telecharger les observations (session/83c568485bfccddf31f63dc4a0b0ec8b/download/Tele_Obs?v

Charger des observations au format txt

Cote interface utilisateur

Les points clés

- Utiliser la commande **`fileInput('upload',...)`** ou **`'upload'`** est le nom de l'input a personnaliser.

- On utilisera les options

  - **accept** pour spécifier un type de fichier
  - **placeholder** et **buttonLabel** pour la mise en forme du texte

```
library(shiny); library(DT)
ui <- fluidPage(titlePanel('Charger des observations au format .txt'),
    sidebarLayout(
    sidebarPanel(fileInput('upload', NULL, accept = c(".txt"),buttonLabel='Parcourir',
    placeholder = 'Aucun fichier selectionne'),
    checkboxInput('header','Le fichier contient une entete',value=TRUE)),
    mainPanel(dataTableOutput("DT"),
  )))
```

Cote serveur

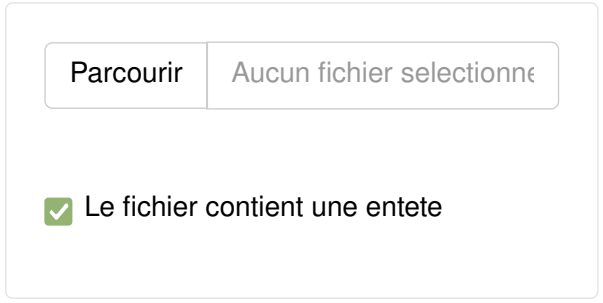
- Utiliser **`input$upload$datapath`** et **`input$upload$name`** pour avoir respectivement le chemin d'acces et le nom du fichier.

- Utiliser la fonction **`req`** peut etre utile pour ne pas evaluer les lignes qui suivent lorsqu'une valeur n'est pas encore attribuee à une variable.

```
server <- function(input, output, session) {
  values=reactiveValues(data=NULL)
  observe({req(input$upload)
    name=sub('.txt$', '', basename(input$upload$name))
    values$data=read.table(input$upload$datapath,header=isolate(input$header))
  })
  output$DT <- renderDT({
    req(values$data)
    values$data},
    options=list(pageLength=7)
  )
}
```

Resultat

Charger des observations au format .txt



Charger des observations au format txt - quelques améliorations

Cote interface utilisateur

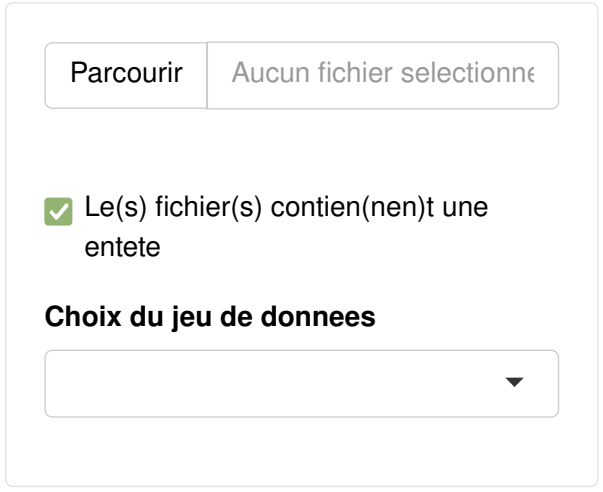
```
library(shiny); library(DT)
ui <- fluidPage(titlePanel('Charger des observations au format .txt'),
    sidebarLayout(
    sidebarPanel(fileInput("upload", NULL, accept = c(".txt"),multiple=TRUE,
    buttonLabel='Parcourir',placeholder = 'Aucun fichier selectionne'),
    checkboxInput('header','Le(s) fichier(s) contien(nen)t une entete',value=TRUE),
    uiOutput('data_select'),
),
    mainPanel(
    dataTableOutput("DT"),
  )))
```

Cote serveur

```
server <- function(input, output, session) {
  values=reactiveValues(Liste=NULL)
  e=environment()
  output$data_select=renderUI({selectInput("data", "Choix du jeu de donnees", choices=val
  observe({req(input$upload)
    for (j in 1:length(input$upload$name))
    {
    name=sub(".txt$", "", basename((input$upload$name)[j]))
    assign(name,read.table((input$upload$datapath)[j],header=isolate(input$header)),envir
    values$Liste=c(isolate(values$Liste),name)
    }
  })
  output$DT <- renderDT({
    req(input$data)
    get(input$data)},
    options=list(pageLength=7)
  )
}
```

Resultat

Charger des observations au format .txt



Charger des environnements de travail

Comment faire?

- Creer 2 menus deroulants pour selectionner l'environnement de travail puis la variable
- Utiliser les fonctions **load**, **assign**, **new_env** et **get** pour gerer le chargement et une mise en memoire ordonnee

Cote interface utilisateur

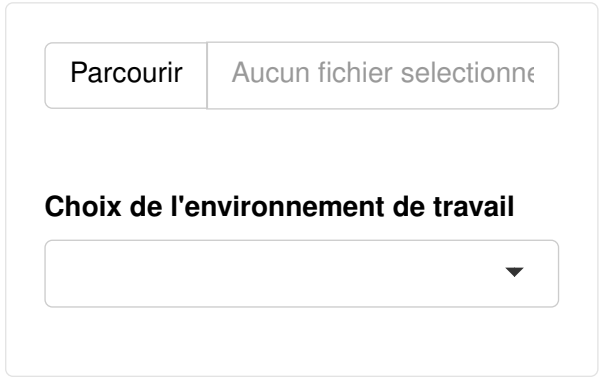
```
library(shiny); library(DT)
  ui <- fluidPage(titlePanel('Charger un environnement de travail'),
    sidebarLayout(
    sidebarPanel(
    fileInput("upload", NULL, accept = c(".RData"),multiple=TRUE,buttonLabel='Parcourir',
    uiOutput('envir_select'),
    uiOutput('data_select')),
    mainPanel(dataTableOutput("DT"))
  ))
```

Cote serveur

```
server <- function(input, output, session) {
    values=reactiveValues(Liste=NULL);e=environment()
    output$envir_select=renderUI({selectInput("envir", "Choix de l\'environnement de trav
    output$data_select=renderUI({req(input$envir)
    selectInput("data", "Choix de la variable", choices=ls(envir=get(input$envir,envir=e)
    observe({req(input$upload)
    for (j in 1:length(input$upload$name))
      {
        name=sub(".RData$", "", basename((input$upload$name)[j]))
        assign(name,new.env(),envir=e)
        load((input$upload$datapath)[j],envir=get(name,envir=e))
        values$Liste=c(isolate(values$Liste),name)
      }
    })
    output$DT <- renderDT({req(input$data)
      as.matrix(get(input$data,envir=get(isolate(input$envir),envir=e)))},
      options=list(pageLength=7)
    )
}
```

Resultat

Charger un environnement de travail



Charger des observations fournies par R

Comment faire?

- Creer 2 menus deroulants pour selectionner le jeu d'observations puis la variable
- Utiliser les fonctions **data**, **attach**, **new_env** et **get** pour gerer le chargement et une mise en memoire ordonnee

Cote interface utilisateur

```
library(shiny); library(DT)
ui <- fluidPage(
    titlePanel('Charger des observations fournies par R'),
    sidebarLayout(
    sidebarPanel(
    selectInput("df", "Choix du jeu d'observations fourni par R", choices=(data()[[3]])[,
    uiOutput('data_select')),
    mainPanel(dataTableOutput("DT")),
  ))
```

Cote serveur

```
server <- function(input, output, session) {
    output$data_select=renderUI({req(input$df)
      selectInput("data", "Choix des variables", choices=colnames(get(input$df)),selected
    output$DT <- renderDT({
      req(isolate(input$df));
      eval(parse(text=paste('data(',isolate(input$df),')')))
      if (!is.null(colnames(get(input$df)))) { req(input$data)
        eval(parse(text=paste('attach(data.frame(',isolate(input$df),'))')))
      X=NULL
      for (j in 1:length(input$data)) {X=cbind(X,get(input$data[j]))}
      colnames(X)=input$data}
      else {X=get(input$df)}
      as.matrix(X)},
      options=list(pageLength=7,scrollX = TRUE)
    )}
```

Resultat

Charger des observations fournies par R

Choix du jeu d'observations fourni par R

trees

Choix des variables

Girth Height Volume

Show 7 **entries** **Search:**

Girth	Height	Volume
8.3	70	10.3
8.6	65	10.3
8.8	63	10.2
10.5	72	16.4
10.7	81	18.8
10.8	83	19.7
11	66	15.6

Showing 1 to 7 of 31 entries Previous 1 2 3 4 5 Next

Mettre a jour des widgets

Comment faire?

Utiliser les fonctions update.... comme par exemple

updateActionButton

updateCheckboxInput

updateDateRangeInput

updateRadioButtons

updateSliderInput

updateTextAreaInput

updateCheckboxGroupInput

updateDateInput

updateNumericInput

updateSelectInput (updateSelectizeInput)

updateTextInput

Les input du package **shinyWidgets** peuvent etre mis a jour de la meme maniere.

Mise en ligne de l’application

Mise en ligne de l'application

- Creer un compte sur [https://www.shinyapps.io/ (https://www.shinyapps.io/)] et choisir un nom de compte pour les applications.

- installer rsconnect

```
install.packages('rsconnect')
```

- autoriser la connexion au compte

```
rsconnect::setAccountInfo(name='laurentdelsol2',
              token='B2049E2AD4AE6A732C64924450E328C7',
              secret='LwC3odn8j4kWW4T3YUhNE1o3AbpCV4lrqo6Q1N7B')
```

- Deployer l'application shiny

```
library(rsconnect)
rsconnect::deployApp('C:/Cours/Shiny/Exemples/Shiny_App/Linear_Reg')
```

Rendre le code accessible

C’est tres simple. Vous avez simplement a ajouter un fichier nomme DESCRIPTION (https://drive.google.com/file/d/1UachdZkq5ayTtf9DUk_rOPb2wrOe2q7a/view?usp=sharing) (sans extension) dans le repertoire de votre application contenant ce qui suit.

```
Title: LLN_and_CLT
Author: Laurent DELSOL
AuthorUrl: https://idpoisson.fr/delsol/
License: GPL-3
DisplayMode: Showcase
Tags: getting-started
Type: Shiny
```

Pour aller un peu plus loin...

Compiler un document Markdown

- On commence par preparer un fichier .Rmd avec des parametres qui seront ceux choisis via l'application (voir l'exemple Kmeans.Rmd (https://drive.google.com/file/d/1m1rl-pSDSJr4MaiwuV3oQ-sDRk83-LAE/view?usp=sharing))
- On rajoute un bouton de compilation du fichier .Rmd au niveau du fichier ui.R via la commande `downloadButton("Gen.report", "Generer le compte-rendu")` ou `"Gen.report"` est le nom que vous avez choisi pour la fonction qui compilera le .Rmd et `"Generer le compte-rendu"` est le message qui apparaitra sur le bouton.
- On rajoute une commande pour compiler le fichier .Rmd au niveau du fichier server.R (voir page suivante).

Compiler un document Markdown

Code a ajouter dans la partie serveur.

```
output$Gen.report <- downloadHandler(
         filename = "Kmeans.html",
         content = function(file) {
           tempReport <- file.path(tempdir(), "Kmeans.Rmd")
           file.copy("Kmeans.Rmd", tempReport, overwrite = TRUE)
           params <- list(x=values$data,algo=input$algo,k=input$k,centers=input$centers,r
           rmarkdown::render(tempReport, output_file = file,
                             params = params,
                             envir = new.env(parent = globalenv())
           )
         }
       )
```

Resultat

Choix de la taille d'échantillon

10 100 1,000

10 109 208 307 406 505 604 703 802 901 1,000

Choix du nombre de classes

2 99

1 11 21 31 41 51 61 71 81 91 99

Choix du nombre d'initialisations

1 10 100

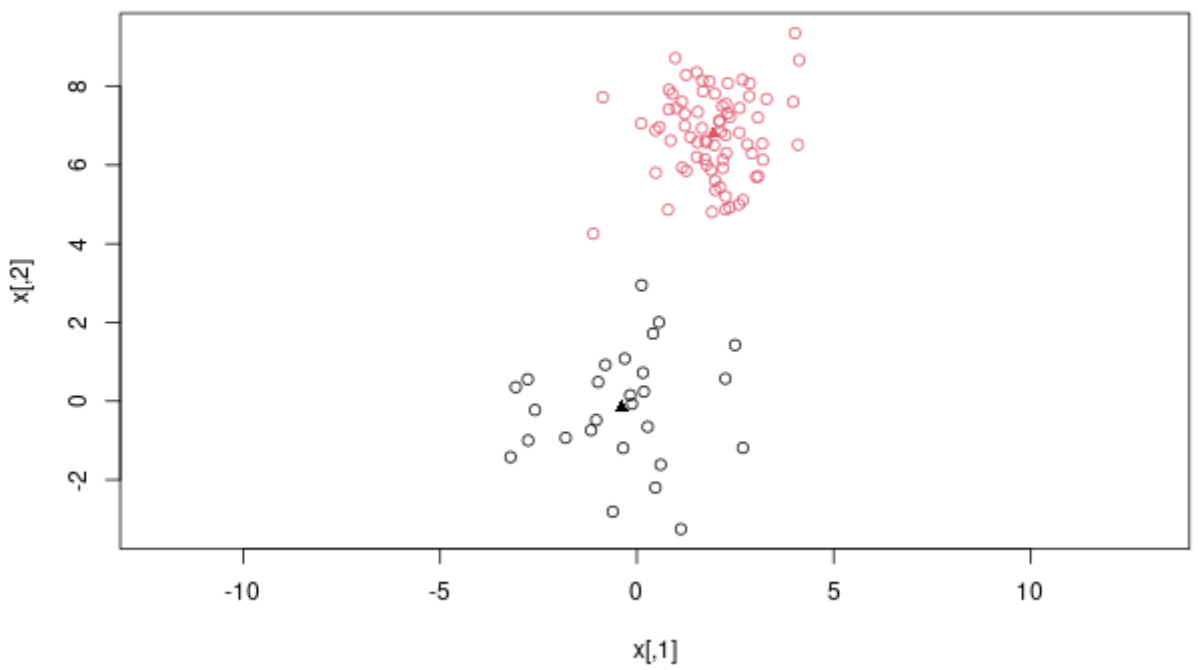
1 11 21 31 41 51 61 71 81 91 100

Choix de l'algorithme

Hartigan-Wong

Rafraichir

Afficher les barycentres



Telecharger les observations (session/b2e86822d39faab968d3b8f88e2257b9/download/Tele_Obs

Generer le compte-rendu (session/b2e86822d39faab968d3b8f88e2257b9/download/Gen.report?w

/

Ecrire sur un document partage

Configuration de la connexion

- Installer **googledrive** et **googlsheets4**
- Taper ensuite les commandes suivantes pour configurer la connexion à votre compte google, pensez à cocher la case **Afficher, modifier, créer et supprimer des fichiers dans Google Drive.**

```
# Charger les packages
library(googledrive)
library(googlesheets4)
# Configurer les options pour enregistrer la clef d'identification dans un fichier .secre
options(
  gargle_oauth_email = TRUE,
  gargle_oauth_cache = "/home/laurent/Bureau/Cours/NTSR/.secrets"
)
# Selectionner le compte a utliser, pensez a permettre l'affichage, la modification, la c
drive_auth()
```

Creer, lire et modifier des fichiers partages

Pour creer un fichier partage nomme **Fichier_partage** ayant 2 feuilles **F1** et **F2**:

```
gs4_create(name = "Fichier_partage", sheets = c('F1','F2'))
```

Pour recuperer l'identifiant du fichier

```
fichier_id <- drive_get("Fichier_partage")$id
```

Pour lire la feuille F2 du fichier

```
F2 <- read_sheet(ss = fichier_id, sheet = "F2")
```

Pour remplir/completer la feuille F1 du fichier partage

- `sheet_write(data = trees,ss = fichier_id,sheet = "F1")`
  Attention, cela remplace ce qui est ecrit avant.

- `sheet_append(data = data.frame(Girth=17,Height=74,Volume=43),ss = fichier_id,sheet = "F1")`

Exemple d'utilisation: lire, ajouter, retirer, mettre à jour

- Creer une application nommee **Googlesheets**
- Puis configurer la connexion

```
setwd("~/Bureau/Shiny_App/Googlesheets")
library(shiny);library(googledrive);library(googlesheets4);
options(
  gargle_oauth_email = TRUE,
  gargle_oauth_cache = "Googlesheets/.secrets"
)
drive_auth()
```

Partie Interface utilsateur

```
library(shiny);library(googledrive);library(googlesheets4);library(DT)
sheet_id <- drive_get("Etu")$id
shinyUI(fluidPage(titlePanel('Document partage'),
                  sidebarLayout(
                    sidebarPanel(
                      textInput("Name", 'NOM', value=NULL),
                      textInput("Surname", 'Prenom', value=NULL),
                      textInput('Email','Adresse e-mail',value=NULL),
                      actionButton('Add','Ajouter'),
                      actionButton('Update','Mettre a jour'),
                      actionButton('Remove','Retirer')
                    ),
                    mainPanel(
                      dataTableOutput("DT")
                    )
                  )
))
```

Partie Serveur

```
function(input, output, session) {
  values=reactiveValues(DP=read_sheet(ss = sheet_id, sheet = "Feuille 1"))
  observeEvent({input$Update},{values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")})
  observeEvent({input$Remove},{req(input$DT_rows_selected)
    values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")[-input$DT_rows_selected,]
    sheet_write(data=values$DP,ss = sheet_id,sheet = "Feuille 1")
  })
  observeEvent({input$Add},{
    req(input$Name,input$Surname,input$Email)
    data = data.frame('Name'=input$Name,'Surname'=input$Surname,'Email'=input$Email)
    if (nrow(values$DP)==0) {sheet_write(data=data,ss = sheet_id,sheet = "Feuille 1")}
    else {sheet_append(data=data,ss = sheet_id,sheet = "Feuille 1")}
    values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")}
  )
  output$DT <- renderDT({values$DP},options=list(pageLength=30, selection = 'multiple'))
}
```

Resultat

Affichage et activation conditionnels des boutons

Partie UI
On remplace les lignes definissant les boutons **Add** et **Remove** par

```
uiOutput('Ad'),
uiOutput('Re')
```

Partie Serveur
On ajoute les lignes suivantes pour afficher ou activer les boutons **Add** et **Remove**

```
output$Ad=renderUI({req(input$Name,input$Surname,input$Email)
                      actionButton('Add','Ajouter')})
output$Re=renderUI({
                     actionButton('Remove','Retirer',disabled=is.null(input$DT_rows_select
                    })
```

Resultat

Envoyer un message pop-up a l'utilisateur

Envoyer un message pop-up a l'utilisateur

On peut utiliser le package **shinyalert** et la fonction du même nom.

La syntaxe est la suivante:
`shinyalert('Titre','Texte',type='warning',inputId='Id')`

- type peut etre "warning", "error", "success" ou "info". On pourra meme indiquer "input" pour une solliciter une reponse de l'utilisateur.
- on peut afficher ou non les boutons 'OK' et 'Cancel' avec les options **showCancelButton** et **showConfirmButton**
- inputId peut etre utilisee pour fermer l'alerte via la fonction **closeAlert** ou prendre en compte la reponse complementaire de l'utilisateur.
- voir toutes les options avec `?shinyalert`

Un exemple

```
library(shiny);library(shinyalert)
shinyApp(
  ui <- fluidPage(titlePanel('Enigme'),
                      "Qu'est-ce qui commence par E, fini par E, et contient une lettre ?
                      textInput("Reponse", 'Indiquez votre reponse en minuscules', value=
                      actionButton('Val','Valider'),
                      actionButton('Ind','Indice')
  ),
  server <- function(input, output, session) {
  observeEvent({input$Val},{
              if(input$Reponse=='enveloppe') {shinyalert("Félicitations! Vous avez GAGNE
              else {shinyalert('Vous faites erreur!','Vous n\'avez pas encore trouve la
  )
    observeEvent({input$Ind},{shinyalert('La solution est en general timbree...',type='in
    }
)
```

Resultat

Enigme

Qu'est-ce qui commence par E, fini par E, et contient une lettre ?
Indiquez votre reponse en minuscules

Valider Indice

Un exemple ameliore

```
library(shiny);library(shinyalert)
shinyApp(
  ui <- fluidPage(titlePanel('Enigme'),
                      "Qu'est-ce qui commence par E, fini par E, et contient une lettre ?
                      textInput("Reponse", 'Indiquez votre reponse en minuscules. Vous n\
                      uiOutput('V'),
                      actionButton('Ind','Indice')
  ),
  server <- function(input, output, session) {
  values=reactiveValues(count=0)
  output$V=renderUI(actionButton('Val','Valider',disabled=values$count>=3))
  observeEvent({input$Val},{values$count=values$count+1
               if(input$Reponse=='enveloppe') {shinyalert("Félicitations! Vous avez GAGNE
               else {if (values$count<3) {shinyalert(paste('Vous faites erreur! Vous n\'a
                     else {shinyalert("Vous avez PERDU!",'Vous n\'avez pas trouve la solu
    observeEvent({input$Ind},{shinyalert('Indice','La solution est en general timbree...'
    }
)
```

Resultat

Enigme

Qu'est-ce qui commence par E, fini par E, et contient une lettre ?
Indiquez votre reponse en minuscules. Vous n'avez que trois essais.

Valider

Indice

Solliciter une confirmation via un pop-up

On peut utiliser le package **shinyWidgets** via les fonctions **confirmSweetAlert** et **ask_confirmation**.

La syntaxe est par exemple la suivante:

```
confirmSweetAlert(inputId='Id',title = 'Titre',text =
'Texte',type = 'question')
ask_confirmation(inputId='Id',title = 'Titre',text = 'Texte',type
= 'question')
```

- type peut etre “info”, “success”, “warning”, “error” ou “question”.
- inputId peut etre utilisee pour fermer l’alerte via la fonction **closeSweetAlert** ou prendre en compte la reponse de l’utilisateur via la commande `input$Id`.
- voir toutes les options avec `?confirmSweetAlert` et `?ask_confirmation`

Remarque: Utiliser alors les fonctions **show_alert**, et **sendSweetAlert** (au lieu de **shinyalert**) pour afficher simplement des pop-up.

Retour à l'exemple du document partage

Partie UI
On ajoute le chargement de la librairie via la commande `library(shinyWidgets)`.

Partie Serveur
On ajoute/change les lignes suivantes:

```
observeEvent({input$Remove},{ask_confirmation(inputId='confirm',title='Attention!',text='
observeEvent({input$confirm},{
      if (input$confirm==TRUE)
          {
          req(input$DT_rows_selected)
          values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")[-input$DT_rows_selecte
          sheet_write(data=values$DP,ss = sheet_id,sheet = "Feuille 1")
          }
})
```

Resultat

Solliciter une reponse via message pop-up

On peut utiliser le package **shinyWidgets** et la fonction **inputSweetAlert**.

La syntaxe est par exemple la suivante:

```
inputSweetAlert(inputId='Id',title = 'Titre',text = 'Texte',type
= 'question',input ="password" )
```

- type peut etre “info”, “success”, “warning”, “error” ou “question”.
- input peut etre “text”, “password”, “textarea”, “radio”, “checkbox”, “select”, “email”, ou “url”
- inputId peut etre utilisee pour fermer l’alerte via la fonction **closeSweetAlert** ou prendre en compte la reponse de l’utilisateur.
- voir toutes les options avec `?inputSweetAlert`

Retour a l'exemple precedent

Objectif: ne permettre qu'aux administrateurs de retirer des lignes du document partage.

Changer dans la partie serveur les lignes suivantes

```
observeEvent({input$Remove},{inputSweetAlert(inputId='Admin',title='Attention!',text='Seu
observeEvent({input$Admin},{
  if (input$Admin=='sesame') {
    req(input$DT_rows_selected)
    values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")[-input$DT_rows_selected,]
    sheet_write(data=values$DP,ss = sheet_id,sheet = "Feuille 1")
  }
})
```

Resultat

Retour a l'exemple precedent

Changer ceci pour que les administrateurs n'aient a rentrer leur code qu'une fois.

```
values=reactiveValues(DP=read_sheet(ss = sheet_id, sheet = "Feuille 1"),Adminmode=FALSE)
observeEvent({input$Update},{values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")})
observeEvent({input$Remove},{
if (!values$Adminmode)
    {inputSweetAlert(inputId='Admin',title='Attention!',text='Seuls les administrateurs p
    })
observeEvent({input$Admin},{if (input$Admin=='sesame') {values$Adminmode=TRUE}})
observeEvent({if (values$Adminmode) {input$Remove} else {input$Admin}},
{
if (values$Adminmode) {
      req(input$DT_rows_selected)
      values$DP=read_sheet(ss = sheet_id, sheet = "Feuille 1")[-input$DT_rows_selected,]
      sheet_write(data=values$DP,ss = sheet_id,sheet = "Feuille 1")
    }
})
```

Resultat

Faire apparaitre/cacher des panneaux ou des onglets

Faire apparaitre/cacher des panneaux

On peut utiliser la fonction **conditionalPanel** via la syntaxe suivante:
`conditionalPanel(condition,contenu)`

Voici un exemple avec l'appli de base, Partie UI

```
library(shiny)
shinyUI(fluidPage(
      titlePanel("Old Faithful Geyser Data"),
      sidebarLayout(
          sidebarPanel(
            selectInput(inputId = "Graph",label = 'Quel graphique choisissez vous?', choi
            conditionalPanel(condition="input.Graph=='Histogramme'",
                             sliderInput("bins","Nombre de classes:",min = 1,max= 50,valu
            conditionalPanel(condition="input.Graph=='Estimateur a noyau'",
                             sliderInput("h","Valeur du parametre de lissage:",min = 0.1,
            width=5),
          mainPanel(plotOutput("distPlot"),width=7)
))
```

Faire apparaitre/cacher des panneaux

Partie Serveur

```
function(input, output) {
        output$distPlot <- renderPlot({
        x<-faithful[, 2]
        if (any(input$Graph=='Histogramme')){
          bins <- seq(min(x), max(x), length.out = input$bins + 1)
          hist(x, breaks = bins, col = 'darkgray', border = 'white',
               xlab = 'Waiting time to next eruption (in mins)',
               main = 'Histogram of waiting times')
          }
        if (any(input$Graph=='Estimateur a noyau')){
          h <- input$h
          plot(density(x,bw=h),
               xlab = 'Waiting time to next eruption (in mins)',
               main = 'Kernel estimator of the density function')
        }
      })
}
```

Resultat

Old Faithful Geyser Data

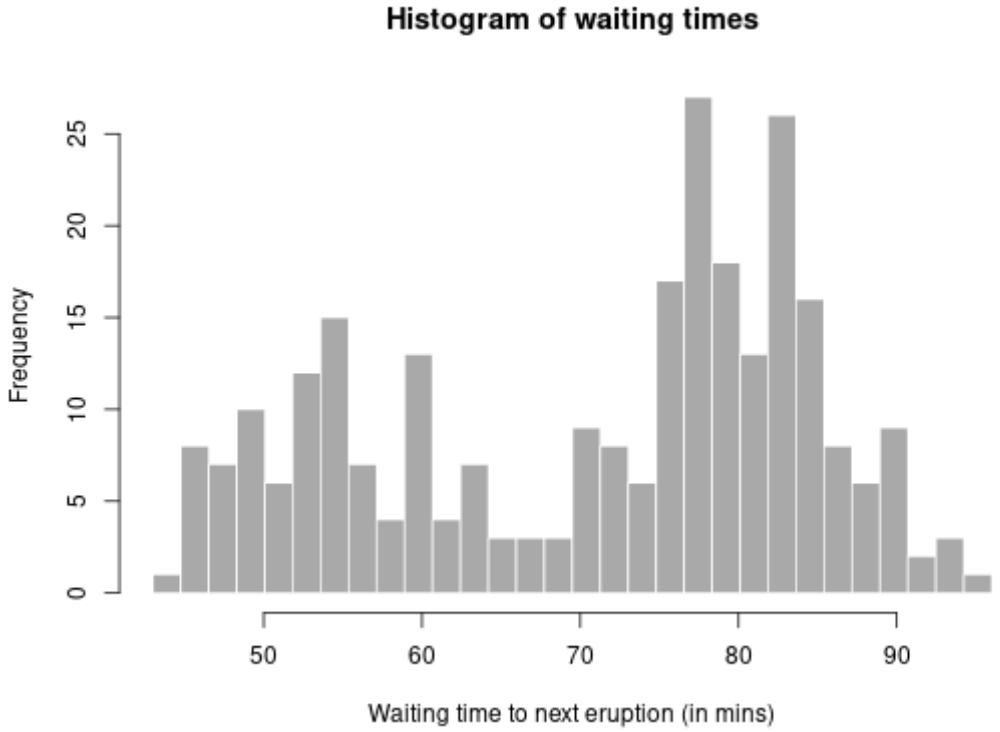
Quel graphique choisissez vous?

Histogramme

Nombre de classes:

1 30 50

1 6 11 16 21 26 31 36 41 46 50



Faire apparaitre/cacher des onglets

Pour ce qui est des onglets, on peut utiliser les fonctions **hideTab**, **showTab** et **updateTabsetPanel** (**updateNavbarPage**, ou **updateNavlistPanel**) via par exemples les syntaxes suivantes:

```
showTab(inputId, target, select = FALSE)
hideTab(inputId, target)
updateTabsetPanel(session,inputId,selected = target)
```

ou

- **inputId** correspond au nom du **tabsetPanel** (ou **navlistPanel** ou **navbarPage**)
- **target** est le nom de l'onglet considere
- **select** precise si l'onglet doit devenir l'onglet actif

On pourra aussi utiliser **insertTab** (**prependTab**, **appendTab**, **removeTab**)

Voici un exemple avec l'appli de base, Partie UI

```
library(shiny);library(shinyalert)
shinyUI(navbarPage("Old Faithful Geyser Data",id="Main",
            tabPanel('Onglets actifs','Quel(s) onglet(s) rendre accessibles:',
                   checkboxInput('H','Histogramme',value=FALSE),
                   checkboxInput('KE','Estimateur a noyau',value=FALSE),
                   actionButton('VH','Voir l\'histogramme'),
                   actionButton('VKE','Voir l\'estimateur a noyau')),
            tabPanel("Histogramme",
                   sidebarLayout(
                       sidebarPanel(sliderInput("bins","Nombre de classes:",min = 1,max=
                       mainPanel(plotOutput("distPlotH"))
            )),
            tabPanel('Estimateur a noyau',
                 sidebarLayout(
                     sidebarPanel(sliderInput("h","Valeur du parametre de lissage:",min =
                     mainPanel(plotOutput("distPlotKE"))
            ))
          )
```

Voici un exemple..., Partie Serveur

```
function(input, output,session) {
      hideTab('Main','Histogramme');hideTab('Main','Estimateur a noyau')
      observeEvent({input$VH},
      {if (!input$H) {shinyalert('Information!','L\'onglet Histogramme est maintenant acc
                     updateCheckboxInput(session,'H', value = TRUE)}
                     updateNavbarPage(session,'Main',selected='Histogramme')})
      observeEvent({input$VKE},
      {if (!input$KE) {shinyalert('Information!','L\'onglet Estimateur a noyau est mainte
                      updateCheckboxInput(session, 'KE', value = TRUE)}
                      updateNavbarPage(session,'Main',selected='Estimateur a noyau')})
      observeEvent({input$H},{ifelse(input$H,showTab('Main','Histogramme'),hideTab('Main'
      observeEvent({input$KE},{ifelse(input$KE,showTab('Main','Estimateur a noyau'),hideT
      output$distPlotH <- renderPlot({x<-faithful[, 2];bins <- seq(min(x), max(x), length
          hist(x, breaks = bins, col = 'darkgray', border = 'white',xlab = 'Waiting time
      output$distPlotKE <- renderPlot({x<-faithful[, 2];h <- input$h
          plot(density(x,bw=h),xlab = 'Waiting time to next eruption (in mins)',main = 'K
      })
}
```

Resultat

Quel(s) onglet(s) rendre accessibles:

- [ ] Histogramme
- [ ] Estimateur a noyau

Voir l'histogramme | Voir l'estimateur a noyau

Graphiques plus interactifs encore avec plotly

Graphiques plus interactifs encore avec plotly

On pourra utiliser les fonctions **plotlyOutput** et **renderPlotly**.

Exemple, partie UI:

```
library(shiny); library(sm); library(plotly)
shinyUI(fluidPage(titlePanel("Old Faithful Geyser Data"),
                  sidebarLayout(
                    sidebarPanel('Quel graphique choisissez-vous d\'afficher?',
                      checkboxInput('SP','Nuage de points',value=FALSE),
                      checkboxInput('KE','Nuage de points',value=FALSE)),
                             mainPanel(
                               conditionalPanel(condition="input.SP",plotlyOutput("scatP
                               conditionalPanel(condition="input.KE",plotlyOutput("distP
                   )))
```

Graphiques plus interactifs encore avec plotly

Partie Serveur

```
function(input, output,session) {
    output$distPlot <- renderPlotly({
      x<-faithful; attach(x)
      w=seq(min(waiting),max(waiting),len=100)
      e=seq(min(eruptions),max(eruptions),len=100)
      RES=sm.density(cbind(waiting,eruptions),eval.points=cbind(w,e), eval.grid=TRUE,meth
      Est=matrix(RES,100,100)
      DT=data.frame(w,e,RES)
      plot_ly(DT,x=~w,y=~e,z = ~Est, type = "surface")
    })
    output$scatPlot <- renderPlotly({
      x<-faithful; attach(x)
      plot_ly(x=~waiting,y=~eruptions, type = "scatter",mode='markers')
    })
  }
```

Resultat

Old Faithful Geyser Data

Quel graphique choisissez-vous d'afficher?

- [ ] Nuage de points
- [ ] Nuage de points

Exercices

Application Jeu de Fibonacci

Le cahier des charges

Utiliser la fonction suivante pour generer une application shiny permettant de faire le tour de magie en laissant a l'utiliateur le choix des 2 premiers nombres et du nombre de lignes a remplir

```
Fibonacci_game=function(x,y,K,plot=FALSE)
{
  num=rep(NA,K);r=rep(NA,K-1);num[1:2]=c(x,y);r[1]=y/x
  for (j in 3:K) {num[j]=sum(num[j-(1:2)]);r[j-1]=num[j]/num[j-1]}
  if (K>=10)
  {
    ratio=num[10]/num[9]
    total=sum(num[1:10]);estim=num[7]*11
  }
  if (plot) { r1=data.frame(z=(3:(K)),max=pmax(r[-1],r[-(K-1)]),min=pmin(r[-1],r[-(K-1)])
  plot_ly(r1,x=~z,y=~max, type = 'scatter', mode = 'lines') %>%
    add_lines(x=~z,y=~min)}
  return(list('num'=num,'ratio'=ratio,'total'=total,'estim'=estim,'r'=r ))
}
```

Personnalisation

Les etapes

- creer une page avec une mise en forme **sidebarLayout** ayant pour titre le nom de l'application
- ajouter trois **numericInput** (dans le panneau lateral) pour permettre le choix des valeurs initiales et du nombre de lignes a calculer
- au niveau du panneau principal, generer un **tableOutput** pour afficher le tableau des resultats
- ajouter 2 **textOutput** (dans le panneau lateral) pour afficher les valeurs cles.

Remarques:
- on pourra utiliser wellPanel avec les options `style = "background: white",bgcolor='white'` pour avoir un panneau clair pour les resultats evoques au point 4
- on pourra utiliser `HTML('<b> texte </b>')` pour afficher un texte en gras

Resultat souhaite

Jeu de Fibonacci

Choix du premier nombre:

0

Choix du second nombre:

1

Choix du nombre n de lignes:

10

Total des lignes 1 à 10 / Ligne 7 :
11

Ligne n / Ligne n-1 :
1.61904761904762

Ligne	Resultat
1	0
2	1
3	1
4	2
5	3
6	5
7	8
8	13
9	21
10	34

Application Regression Lineaire

Une premiere ebauche: UI

```
library(shiny)
shinyUI(fluidPage(
  titlePanel("Linear regression with R"),
  sidebarLayout(
    sidebarPanel(
       selectInput(inputId='X_law', label='Faites votre choix pour la loi de X', choices=
       sliderInput("n","Sample size:",min = 1,max = 100,value = 30),
       sliderInput("a","A:",min = -10,max = 10,value = 3),
       sliderInput("b","B:",min = -10,max = 10,value = -3),
       sliderInput("s","S:",min =0,max = 10,value =1,step= .01)
    ),
    mainPanel(plotOutput("Scatterplot"))
  )
))
```

Une premiere ebauche: Serveur

```
library(shiny)
shinyServer(function(input, output) {
         output$Scatterplot <- renderPlot({
           n=input$n
           if (input$X_law=='N(0,1)') {x=rnorm(n)}
           if (input$X_law=='Exp(1)') {x=rexp(n)}
           if (input$X_law=='U(0,1)') {x=runif(n)}
           e=rnorm(n)
    y=input$a*x+input$b+input$s*e
    plot(x,y,main=paste('cor=',cor(x,y),sep=''))
    res=lm(y~x)
    abline(res,col='red')
  })

})
```

Resultat

Linear regression with R

Faites votre choix pour la loi de X

N(0,1)

Sample size:

1 30 100

1 11 21 31 41 51 61 71 81 91 100

A:

-10 3 10

-10 -8 -6 -4 -2 0 2 4 6 8 10

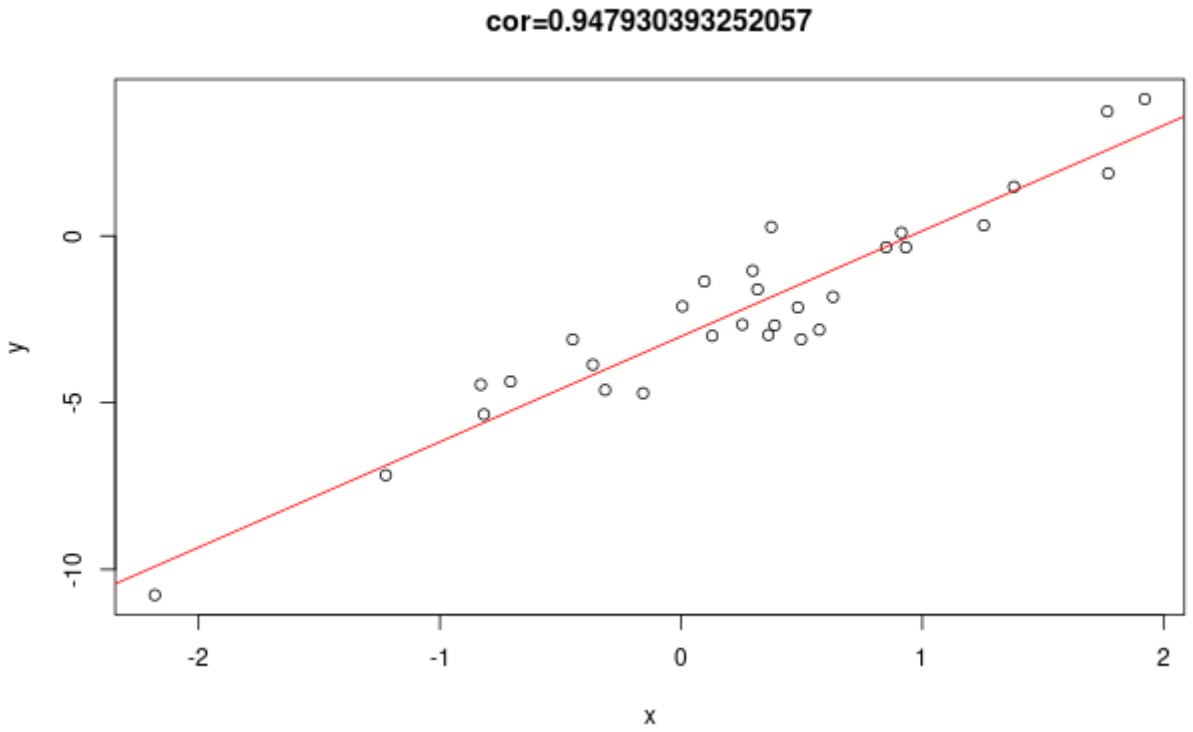
B:

-10 -3 10

-10 -8 -6 -4 -2 0 2 4 6 8 10

S:

0 1 10



Personnalisation

Pour s'entrainer

- ajouter une barre de defilement pour choisir un point x_0 ou realiser une estimation
- ajouter la creation et le telechargement d'un rapport RMarkdown a partir des parametres selectionnes via le fichier Modeles_lineaires.Rmd (https://drive.google.com/file/d/1YoAFFSUdGCSW0nqwfXaYY6VvfFrEIqzC/view?usp=sharing)
- ajouter une page permettant de charger un jeu de donnees R et de choisir les variables a selectionner pour faire la regression.
- Ajouter une page permettant de charger des donnees au format txt.

Resultat souhaite

Choose X's distribution

N(0,1)

Sample size:

1 30 100

1 11 21 31 41 51 61 71 81 91 100

A:

-10 3 10

-10 -8 -6 -4 -2 0 2 4 6 8 10

B:

-10 -3 10

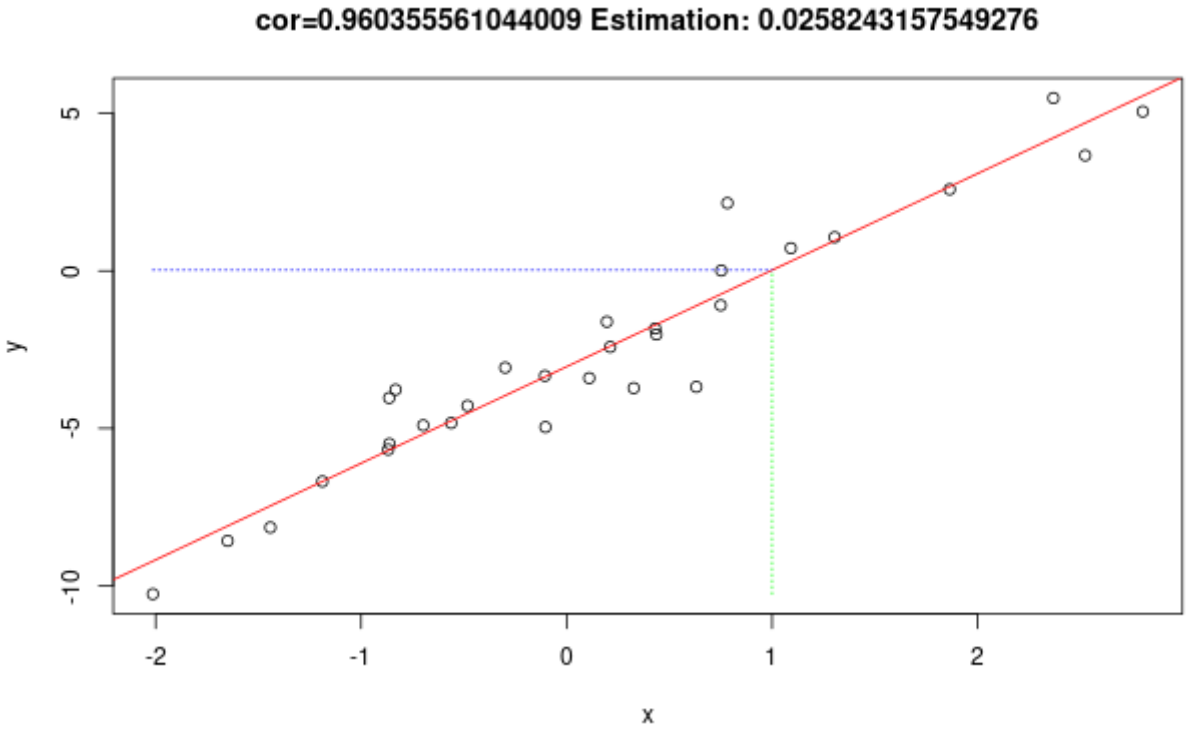
-10 -8 -6 -4 -2 0 2 4 6 8 10

S:

0 1 10

0 1 2 3 4 5 6 7 8 9 10

Eval.point :



Generate report (session/aaf26b6cefb1e1a960f024f531c2e329/download/report?w=1d1e7590d38

Application Enigmes

Appli de base Partie UI

```
library(shiny);library(shinyalert);library(shinyWidgets)
fluidPage(
    navbarPage('Escape Game',id='Main',
               tabPanel('Jeu d\'enigmes',
                        'Emprisone, dans un cachot, votre gardien accepte de vous liberer
                        radioGroupButtons( inputId ='Niveau', label = 'Quel niveau choisi
                        actionButton('Go','Relever le defi!')
               ),
               tabPanel('Enigme 1',"Qu'est-ce qui commence par E, fini par E, et contient
                        textInput("Reponse1", 'Indiquez votre reponse (un mot) en minuscu
                        actionButton('Val1','Valider'),
                        actionButton('Ind1','Indice'))
    ))
```

Appli de base Partie Server

```
function(input, output, session) {
    Reponse1='';
    observeEvent({},{
      if(Reponse1=='enveloppe') {
      sendSweetAlert(title="Félicitations!",text="Vous avez trouve la solution de cette p
      }
      else {
      sendSweetAlert(title="Aie!",text='Vous n\'avez pas trouve la solution de cette prem
      }
    })
    observeEvent({},{sendSweetAlert(title='Indice',text='La solution est en general timbr
}
```

Resultat

Emprisone, dans un cachot, votre gardien accepte de vous liberer si vous prouvez votre intelligence en repondant correctement a une serie d'enigmes... Pas le droit a l'erreur si vous voulez quitter votre prison... Alors reflechissez avant de repondre.

Quel niveau choisissez vous?

✔ Facile | Normal

Relever le defi!

Personnalisation

Pour s'entrainer

- corriger ce qu'il faut pour rendre l'application interactive (effet des boutons d'action, valeur de Reponse1)
- creer les onglets correspondant aux Enigmes 2 et 3
- mettre a jour la partie server pour prendre en compte ce qui a ete ajoute
- rendre l'affichage des indices conditionne a la valeur du niveau choisie
- rendre l'affichage des onglets sequentiel (on n'affiche pas l'enigme 1 avant que le bouton Go ait ete presse, ni l'enigme 2 avant que la 1 ait ete resolue…). On pourra utiliser les fonctions **hideTab**, **showTab**, **updateTabsetPanel**…
- desactiver les boutons de validation lorsque une reponse a ete fournie. On pourra utiliser la fonction **updateActionButton**.

Resultat souhaite

Emprisone, dans un cachot, votre gardien accepte de vous liberer si vous prouvez votre intelligence en repondant correctement a une serie d'enigmes... Pas le droit a l'erreur si vous voulez quitter votre prison... Alors reflechissez avant de repondre.

Quel niveau choisissez vous?

✔ Facile | Normal

Relever le defi!