



- SHINY SOUS LE CAPOT-



dreamRs

dreamRs est une société de conseil en datascience et en développement d'application Shiny :



Conseil et expertise en datascience : dreamRs met son expertise R au service de la datascience pour comprendre et expliquer vos données. Nos travaux de R&D nous permettent d'améliorer constamment la qualité de nos services grâce à une veille technologique et stratégique active.



Accompagnement et formation : de la montée en compétences au perfectionnement, dreamRs forme et accompagne vos équipes data au logiciel R.



Développement d'outils : afin d'aider vos équipes dans les tâches quotidiennes, nous développons des outils interactifs sous forme d'addin Rstudio, d'application Shiny ou de packages R adaptés à vos besoins.



Shiny sur le papier

C'est un package **R** développé par Rstudio (Winston Chang et Joe Cheng), qui permet de construire des **applications web** sans écrire une seule ligne de code HTML, CSS ou JavaScript.



Bootstrap

Shiny utilise le **framework bootstrap** (framework pour le développement de sites et d'applications web)



HTML

format de données conçu pour représenter des pages web



CSS

langage décrivant le style d'une page HTML



JavaScript

langage de programmation employé dans les pages web

Shiny dans le code

Une application Shiny est composée de deux scripts : **ui.R** et **server.R**

UI (User Interface)

Apparence de l'application

```
fluidPage(  
  headerPanel("Théorème Central Limite"),  
  sidebarLayout(  
    sidebarPanel(  
      sliderInput(  
        "n", "Nombre observations",  
        value = 1e4, min = 1e3, max = 1e5  
      ),  
      sliderInput(  
        "mu", "Moyenne",  
        value = 0, min = -50, max = 50  
      ),  
      sliderInput(  
        "sd", "Ecart-type",  
        value = 20, min = 1, max = 30  
      )  
    ),  
    mainPanel(  
      plotOutput(outputId = "plot")  
    )  
  )  
)
```

server

Sortie R à inclure dans l'application

```
function(input, output, session) {  
  output$plot <- renderPlot({  
    pop <- rnorm(n = input$n, mean = input$mu, sd = input$sd)  
    m_pop <- mean(pop)  
    sd_pop <- sd(pop)  
    pdens <- density(pop)  
    phist <- hist(pop, plot=FALSE)  
    hist(  
      pop, main = "Loi normale", xlab = "", freq = FALSE,  
      xlim = c(min(-100, pop), max(100, pop)),  
      ylim = c(0, max(pdens$y, phist$density)),  
      col = "lightblue", border = "white",  
      cex.main = 1.5, cex.axis = 1.5, cex.lab = 1.5  
    )  
    legend_pos = ifelse(input$mu > 0, "topleft", "toprig")  
    legend(  
      legend_pos, inset = 0.025,  
      legend = bquote(atop(mu == .(round(m_pop)),  
                           sigma == .(round(sd_pop)))),  
      bty = "n", cex = 1.5, text.col = "steelblue",  
      text.font = 2  
    )  
    lines(pdens, col = "steelblue", lwd = 3)  
  })  
}
```

Shiny dans le code (résultat)

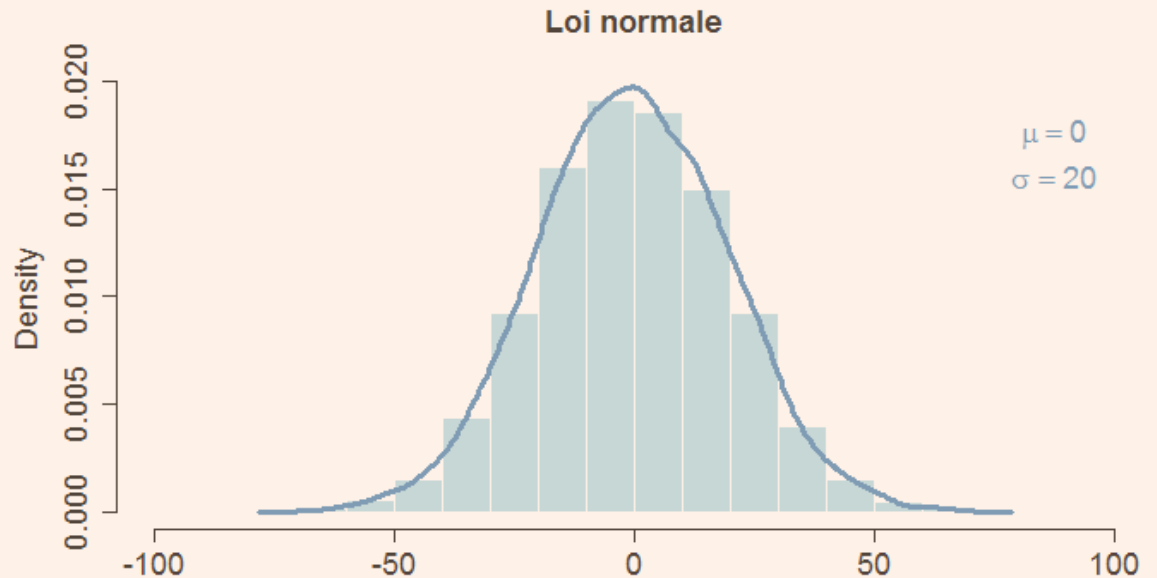
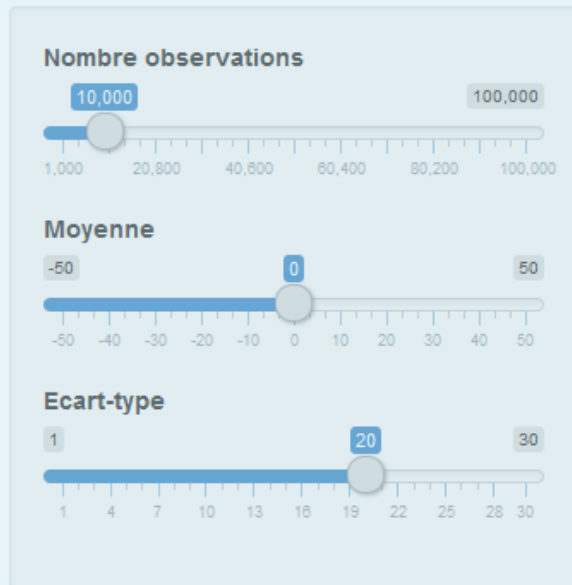
input



output



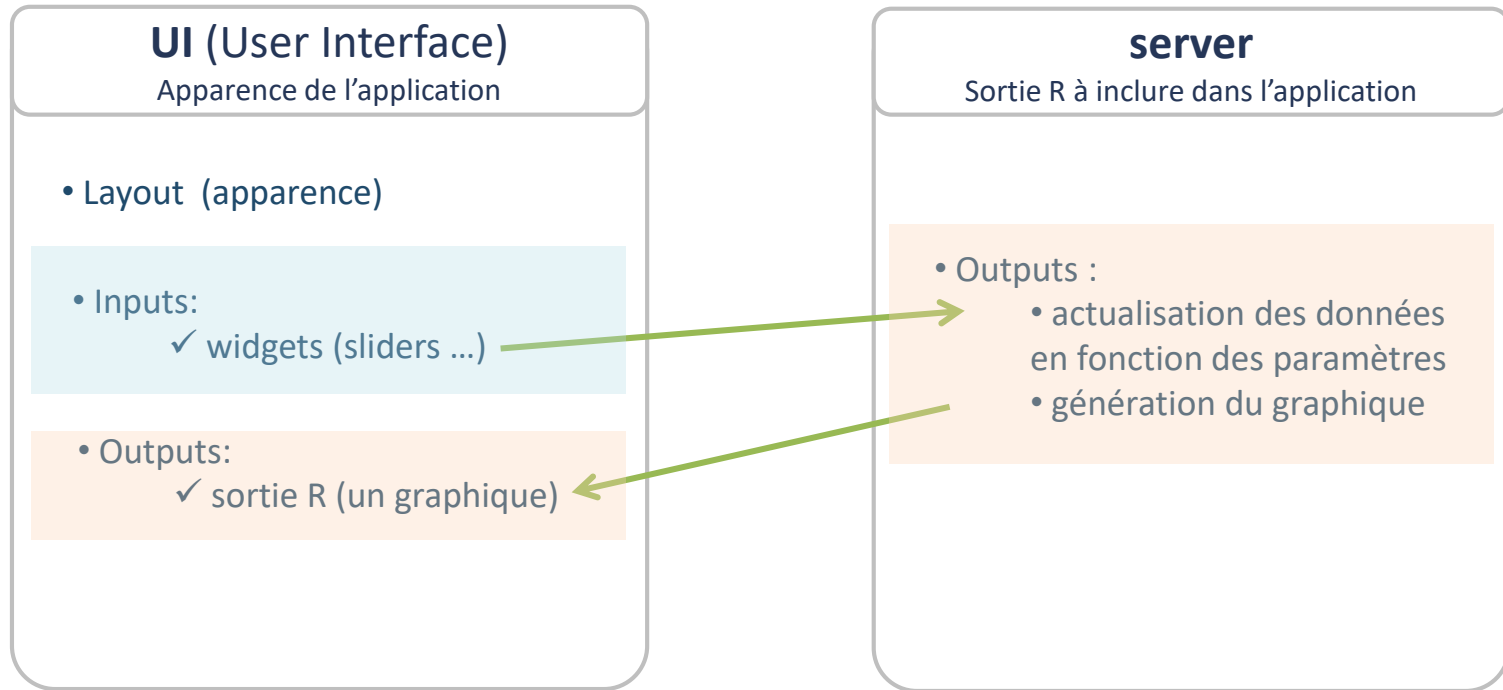
Théorème Central Limite





Comment cela fonctionne ?

On définit dans le UI l'apparence de l'application et les paramètres modifiables par l'utilisateur. Et dans le server le graphique que l'on veut générer pour l'utilisateur.



Fonctionnalités de base

1 Une interface par défaut



2 Des contrôles par défaut : menu déroulant, checkboxes, sliders, ...

Checkbox group

- ☒ Choice 1
- ☐ Choice 2
- ☐ Choice 3

Select box

Choice 1

Sliders



3 Des sorties R classiques (ou moins classiques) : tableaux, résultats de la console, graphiques (base, lattice, ggplot2, ...)

L'écosystème Shiny

Customisation :

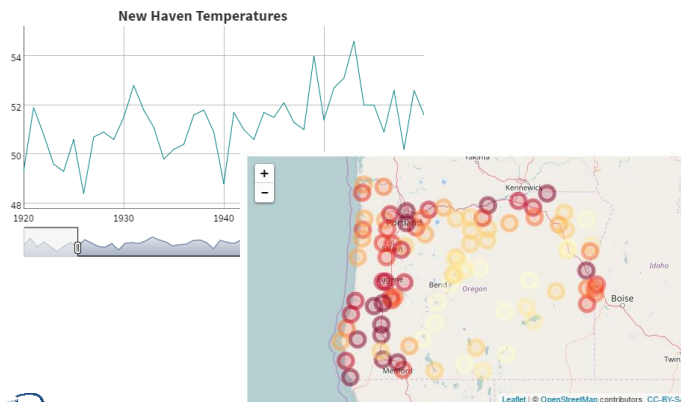
- Via shinyJS / shinyjqui
- Pur HTML et CSS
- En JavaScript en utilisant des fonctionnalités avancées de shiny

Apparence :

- shinydashboard
- shinythemes
- bsplus

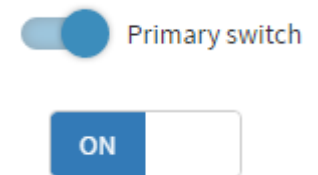


Visualisations : les htmlwidgets



Widgets et extensions :

- shinyWidgets
- shinyFiles
- shinytoastr
-



Make a choice :

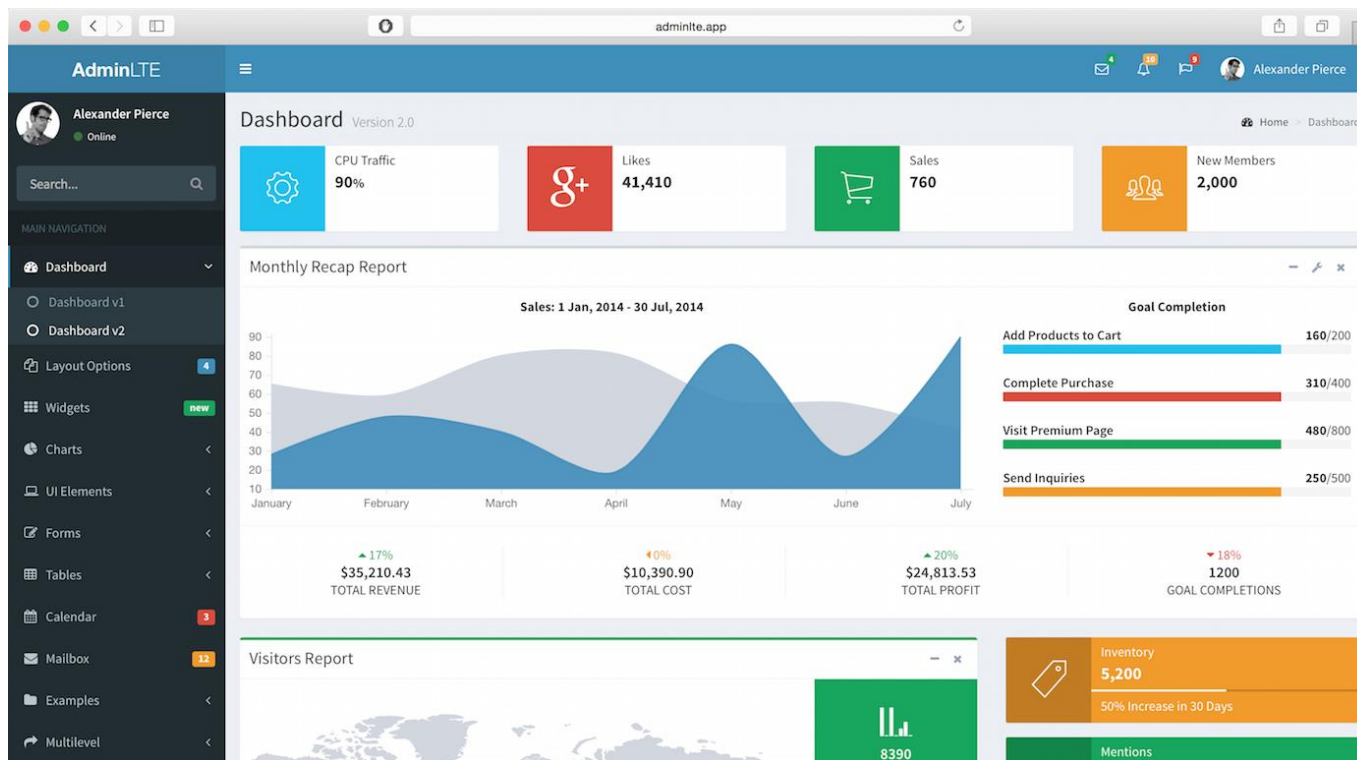


Apparence d'une application

Customisation de l'apparence : shinydashboard

shinydashboard est un package permettant d'utiliser une extension de bootstrap : AdminLTE.

AdminLTE est un framework permettant de construire des tableaux de bord facilement.



shinydashboard vs. shiny

Les principales différences avec shiny se situent dans la construction de l'interface, la partie server de l'application reste inchangée.

Header : titre de l'application

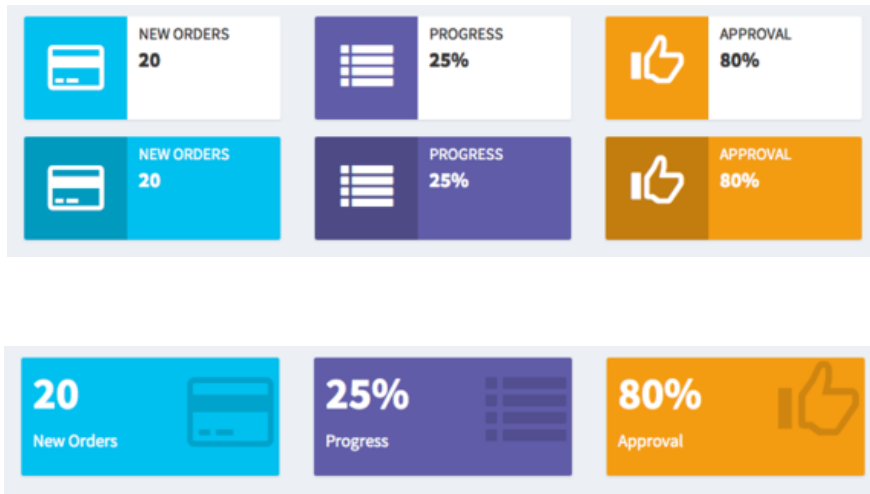
Sidebar :
menu de
l'application
pour naviguer
entre les
différents
onglets

Body :
contenu des onglets

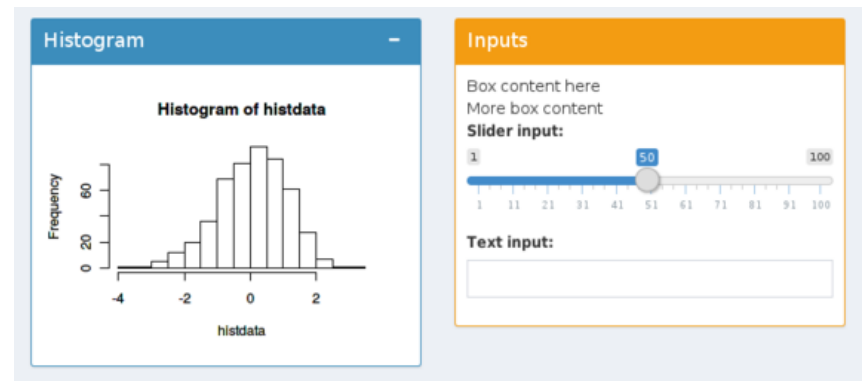
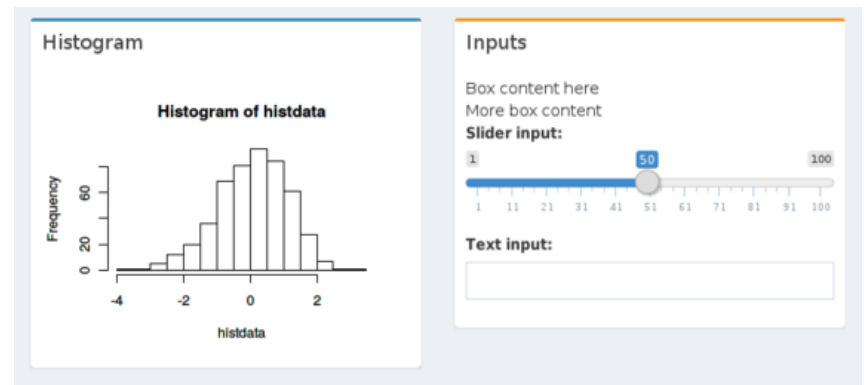
shinydashboard vs. shiny

shinydashboard contient de nombreuses fonctions pour placer des éléments dans une page sous forme de box.

Des indicateurs :

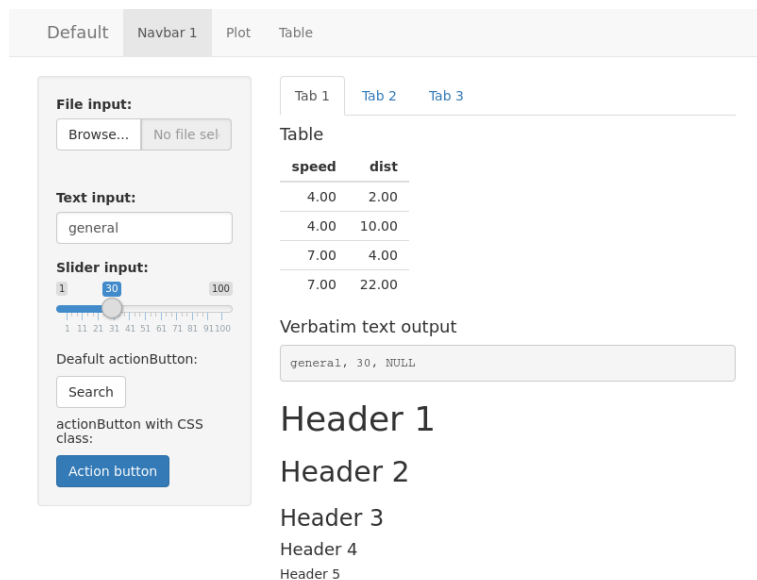


Des graphiques et autres sorties R :



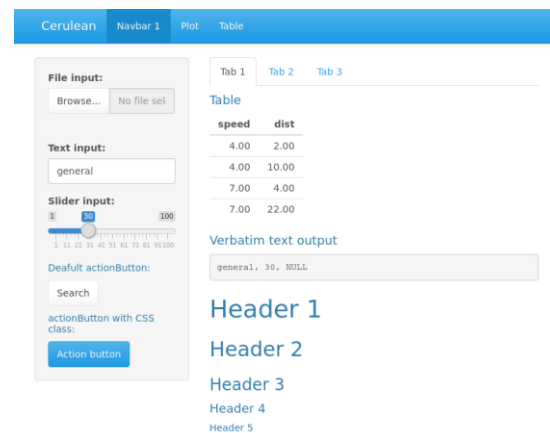
Customisation de l'apparence : shinythemes

shinydashboard est un package permettant d'utiliser des thèmes Bootstrap et ainsi facilement modifier l'apparence d'une application.



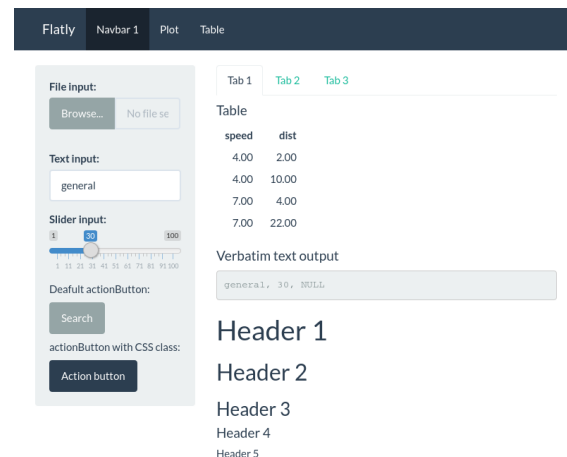
The image shows a Shiny dashboard with the 'Default' theme. The top navigation bar is light gray with tabs for 'Default', 'Navbar 1', 'Plot', and 'Table'. The 'Table' tab is active. The dashboard contains a sidebar with a 'File input', 'Text input' (containing 'general'), 'Slider input' (set to 30), and a 'Default actionButton' (labeled 'Search'). The main content area has a 'Table' with columns 'speed' and 'dist', a 'Verbatim text output' showing 'general, 30, NULL', and five headers labeled 'Header 1' through 'Header 5'.

speed	dist
4.00	2.00
4.00	10.00
7.00	4.00
7.00	22.00



The image shows a Shiny dashboard with the 'Cerulean' theme. The top navigation bar is blue with tabs for 'Cerulean', 'Navbar 1', 'Plot', and 'Table'. The 'Table' tab is active. The dashboard contains a sidebar with a 'File input', 'Text input' (containing 'general'), 'Slider input' (set to 30), and a 'Default actionButton' (labeled 'Search'). The main content area has a 'Table' with columns 'speed' and 'dist', a 'Verbatim text output' showing 'general, 30, NULL', and five headers labeled 'Header 1' through 'Header 5'.

speed	dist
4.00	2.00
4.00	10.00
7.00	4.00
7.00	22.00



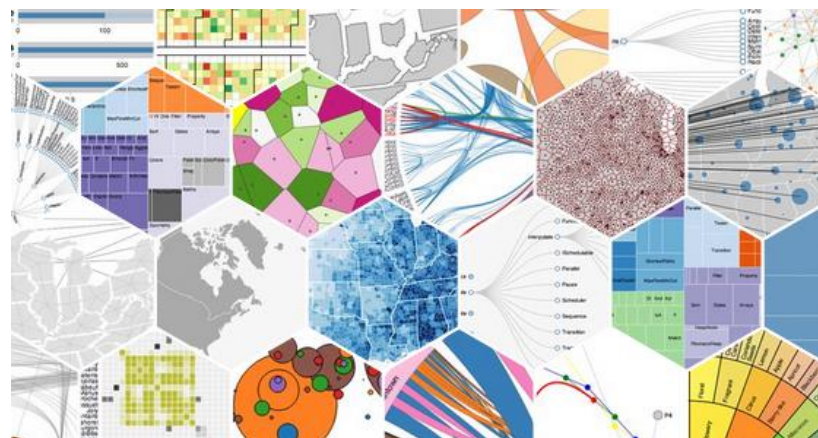
The image shows a Shiny dashboard with the 'Flatly' theme. The top navigation bar is dark gray with tabs for 'Flatly', 'Navbar 1', 'Plot', and 'Table'. The 'Table' tab is active. The dashboard contains a sidebar with a 'File input', 'Text input' (containing 'general'), 'Slider input' (set to 30), and a 'Default actionButton' (labeled 'Search'). The main content area has a 'Table' with columns 'speed' and 'dist', a 'Verbatim text output' showing 'general, 30, NULL', and five headers labeled 'Header 1' through 'Header 5'.

speed	dist
4.00	2.00
4.00	10.00
7.00	4.00
7.00	22.00

htmlwidgets

Les htmlwidgets

Les htmlwidgets sont un ensemble de packages permettant d'intégrer des visualisations interactives réalisées à l'aide de bibliothèques JavaScript dans des applications shiny (et même dans Rstudio).



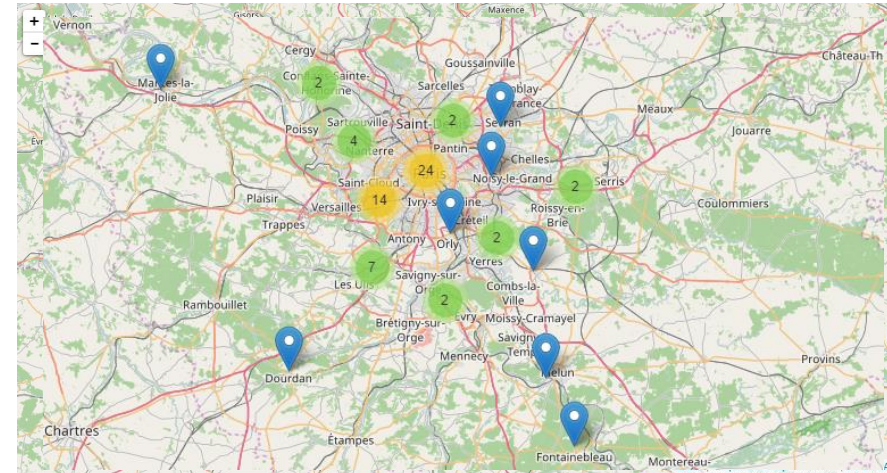
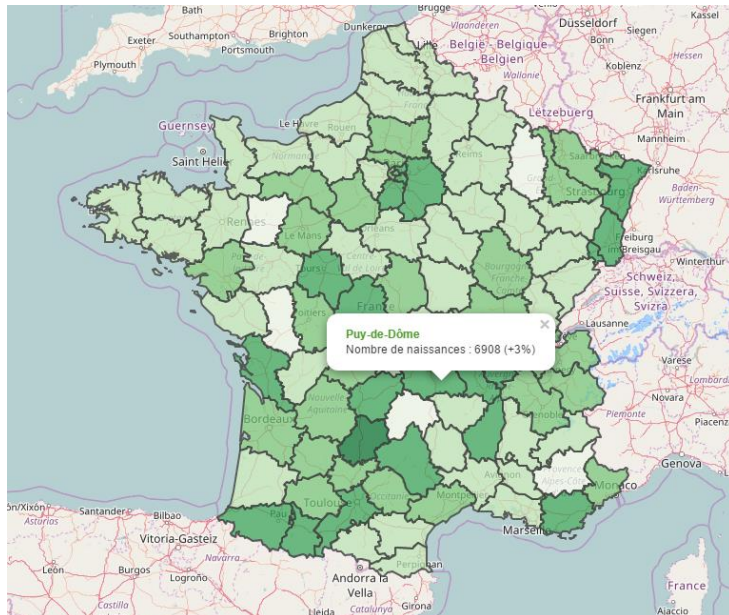
<http://www.htmlwidgets.org/>

Leaflet

Leaflet est un package permettant de réaliser des **cartes interactives** avec OpenStreetMap.

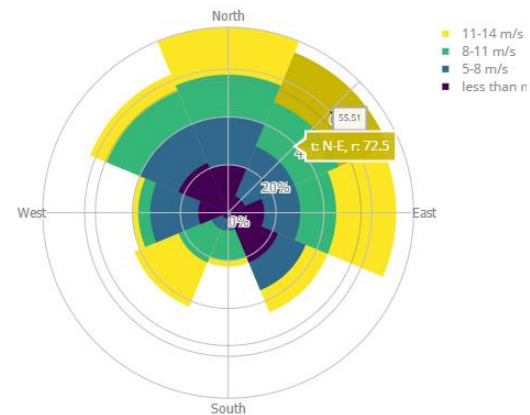
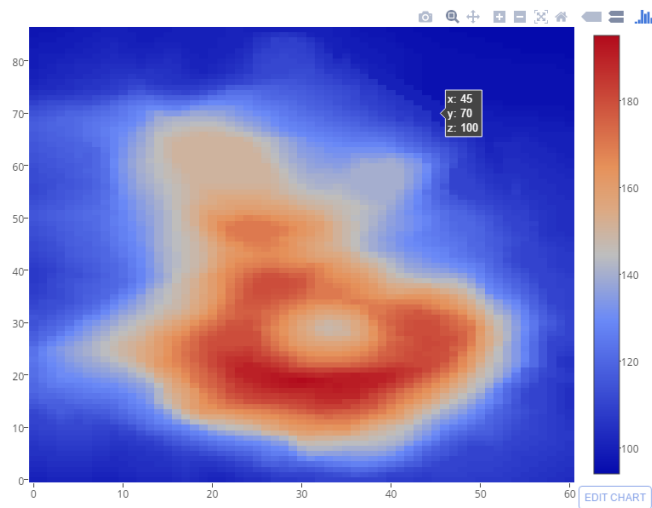
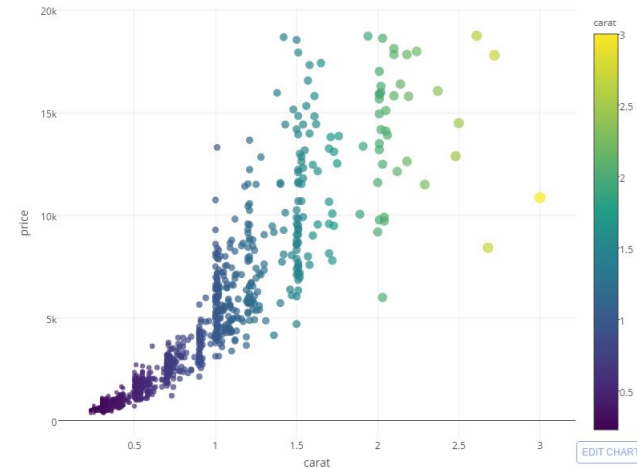
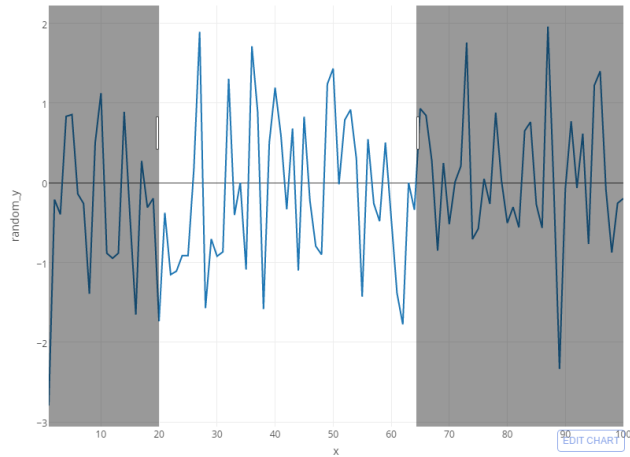
Ce package offre de nombreuses possibilités :

- Placers des points ou des polygones
- Zoomer
- Clics sur la carte
- ...



Plotly

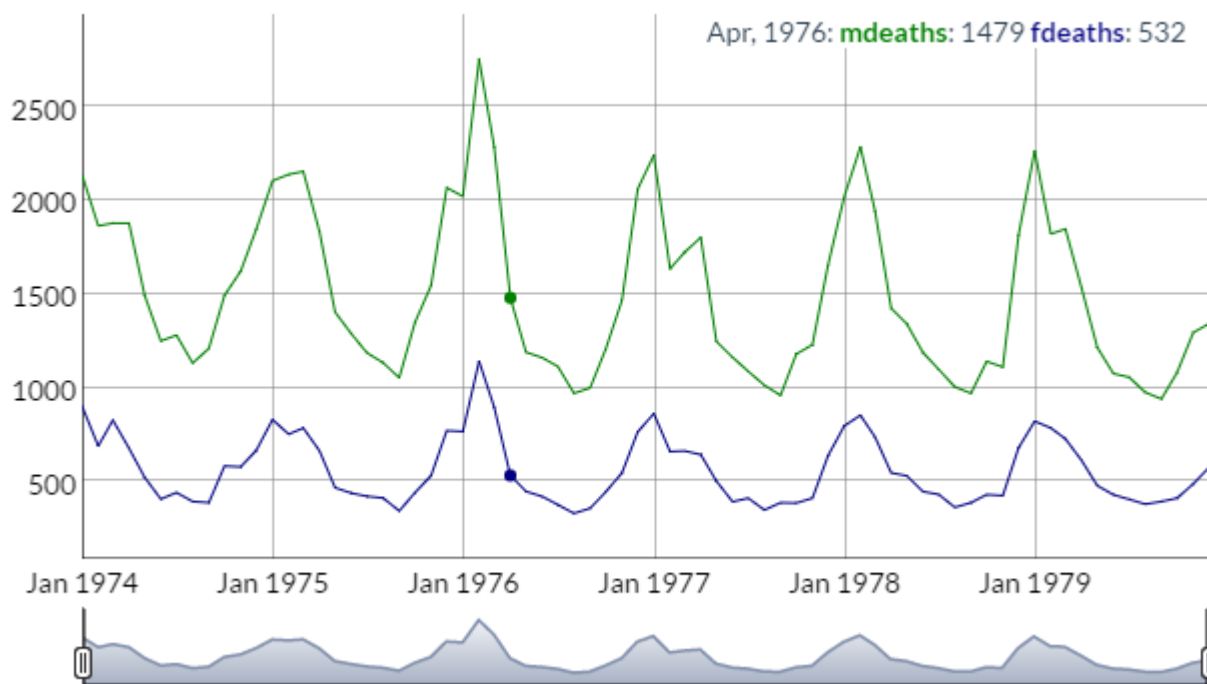
Plotly est un package de visualisation généraliste : de nombreux types de graphiques sont disponibles (courbes, barcharts, histogrammes,...).



<https://plot.ly/r/>

Dygraphs

Dygraphs est un package de visualisation de **séries temporelles**.

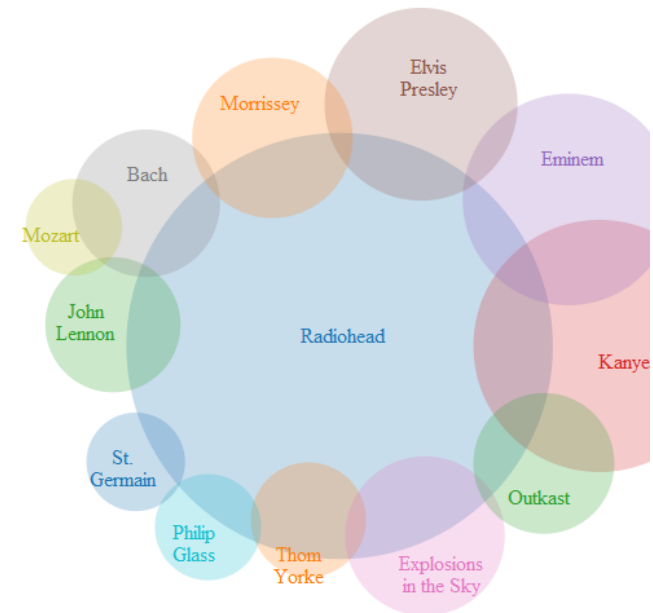
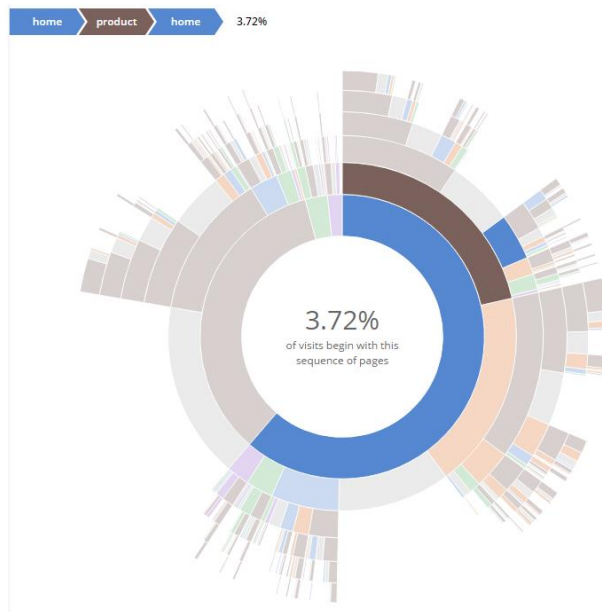
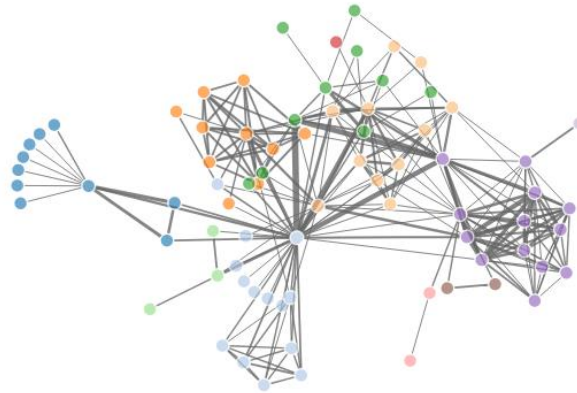


Il offre de nombreuses possibilités : zoom, sélection de zones sur le graphique, synchronisation de graphiques, ...

D3.js

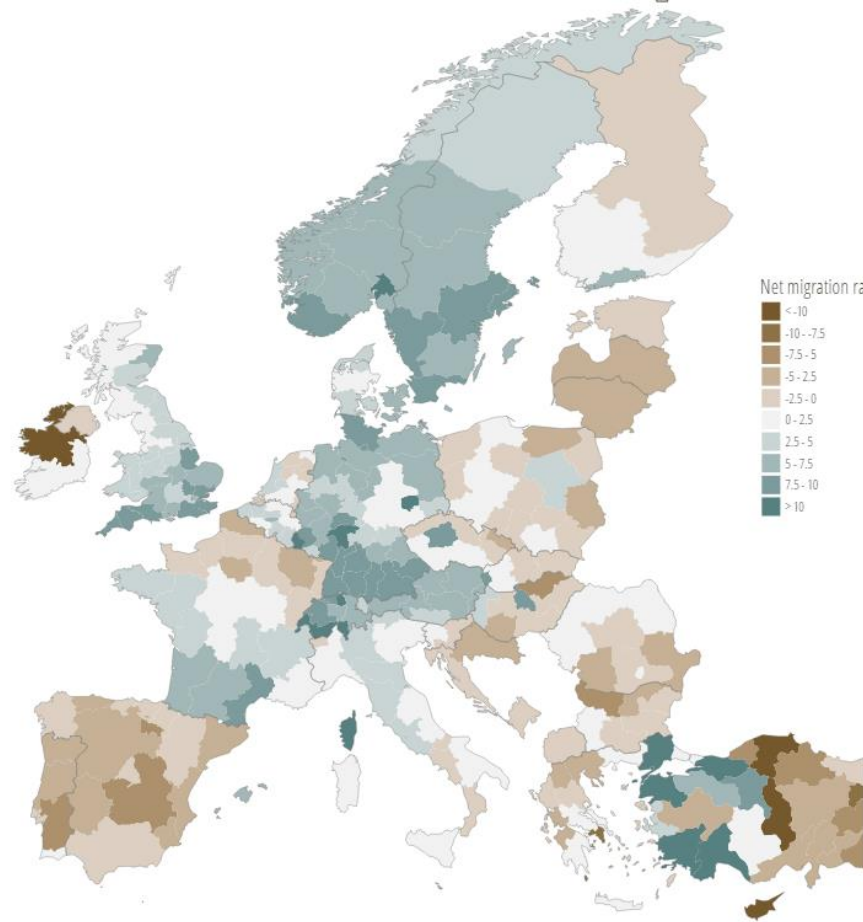
De nombreux htmlwidgets permettent de créer des graphiques en D3 :

- sunburstR
- D3partitionR
- networkD3
- ggiraph
- d3wordcloud
- d3VennR
- ...



ggiraph

ggiraph est un package permettant de rendre vos graphiques ggplot2 interactifs (avec un tooltip, la possibilité de zoomer, cliquer sur les points, ...).



Carte par Duc Quang Nguyen, disponible sur swissinfo.ch

htmlwidgets

Si aucun htmlwidget n'existe pour la librairie JS que vous souhaitez utiliser, vous pouvez écrire vous-même votre htmlwidget !

Mais il faut connaître un minimum le JavaScript car cela nécessite d'écrire des bindings pour que le code R communique avec le code JavaScript...

Extensions de shiny

shinyWidgets



shinyWidgets propose de nouveaux types d'inputs utilisables dans une applications shiny, ainsi que des éléments pour customiser celles-ci.

shinyWidgets Overview

Awesome checkbox Group

Checkboxes with status

☐ A ☒ B ☒ C

Value :

[1] "B" "C"

[Show code](#)

[More examples](#)

Awesome checkbox

☒ A single checkbox

Value :

[1] TRUE

[Show code](#)

[More examples](#)

Select Picker

With plain HTML

Badge danger Badge primary

Value :

[1] "Badge danger" "Badge primary"

[Show code](#)

[More examples](#)

Checkbox Group Buttons

Choices

Choice 1 Choice 2 Choice 3

Value :

[1] "Choice 2"

[Show code](#)

[More examples](#)

Material Design Switch

Primary switch

Value :

[1] TRUE

[Show code](#)

[More examples](#)

Bootstrap Switch

ON

Value :

[1] TRUE

[Show code](#)

[More examples](#)

Search field

Click search icon to update or hit 'Enter'

A placeholder x Q

Value :

[1] ""

[Show code](#)

```
searchInput(inputId = "Id009",
  label = "Click search icon to update or hit 'Enter'",
  placeholder = "A placeholder",
  btnSearch = icon("search"),
  btnReset = icon("remove"),
  width = "100%")
```



Autre extensions

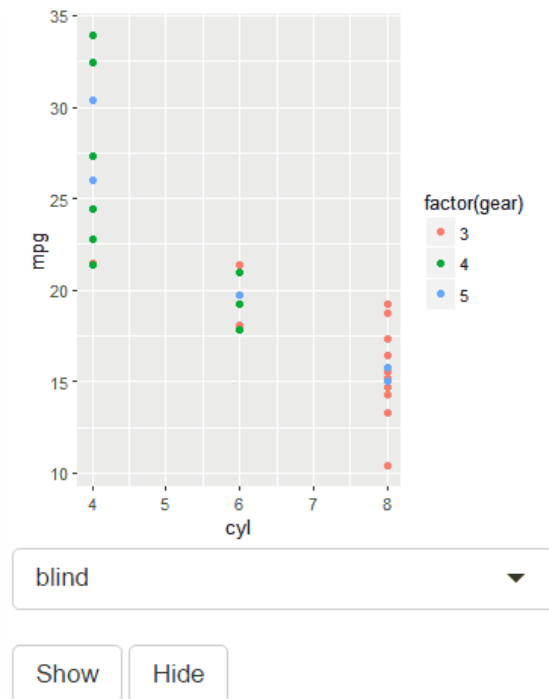
Il existe de nombreux packages permettant d'étendre les fonctionnalités de shiny :

- **shinyFiles** : import de fichier dans une application (côté serveur) (<https://github.com/thomasp85/shinyFiles>)
- **shinyFeedback** : validation des paramètres saisis par l'utilisateur (<https://github.com/merlinoa/shinyFeedback>)
- **shinytoastr** : affichage de notifications (<https://github.com/MangoTheCat/shinytoastr>)
- **bsplus** : fonctionnalités avancées de Bootstrap (<https://github.com/ijlyttle/bsplus>)
- **shinytest** : tester automatiquement le fonctionnement d'une application (<https://rstudio.github.io/shinytest/>)
- **miniUI** : pour la création d'addins, des mini apps utilisables directement dans Rstudio (<https://github.com/rstudio/miniUI>)

Customisation

shinyJS et shinyjui

Ces deux packages permettent des interactions dans des applications grâce à du **JavaScript** et du **jQuery**.



Source

Jan

Feb

Mar

Apr

May

Jun

Jul

Aug

Sep

Oct

Nov

Dec

Dest

Drag items here...

NULL

HTML, CSS et JS

Il est possible d'inclure du code HTML avec les fonctions « tags » de shiny, et donc d'inclure du CSS et du JavaScript facilement dans une application.

Ajouter du CSS :

```
tags$style(".obligatoire {color: red; font-weight: bold;}")
```

Ajouter du JavaScript :

```
tags$script(
  HTML(
    '$( "#id" ).hover(
      function() {
        $( this ).find(".control-label").append( $( "<span class =
\'obligatoire\'> Obligatoire !</span>" ) );
      }, function() {
        $( this ).find(".control-label").find( "span:last"
      ).remove();
    }
  );'
)
)
```

Résultat :

Faites un choix

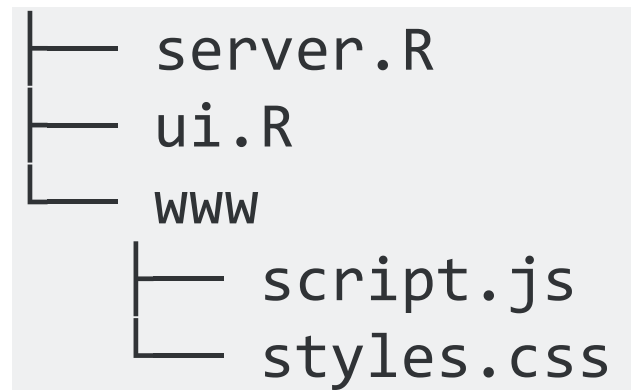
☐ Choix 1

☐ Choix 2

☐ Choix 3

HTML, CSS et JS

Il est aussi également possible d'**inclure des fichiers .CSS ou .JS**, pour cela il est nécessaire de les inclure dans un répertoire nommé « **www** » placé à la racine de l'application :



Puis de l'appeler de la manière suivante dans le ui :

```
tags$link(rel = "stylesheet", type = "text/css", href = "styles.css")
tags$script(src = "script.js")
```

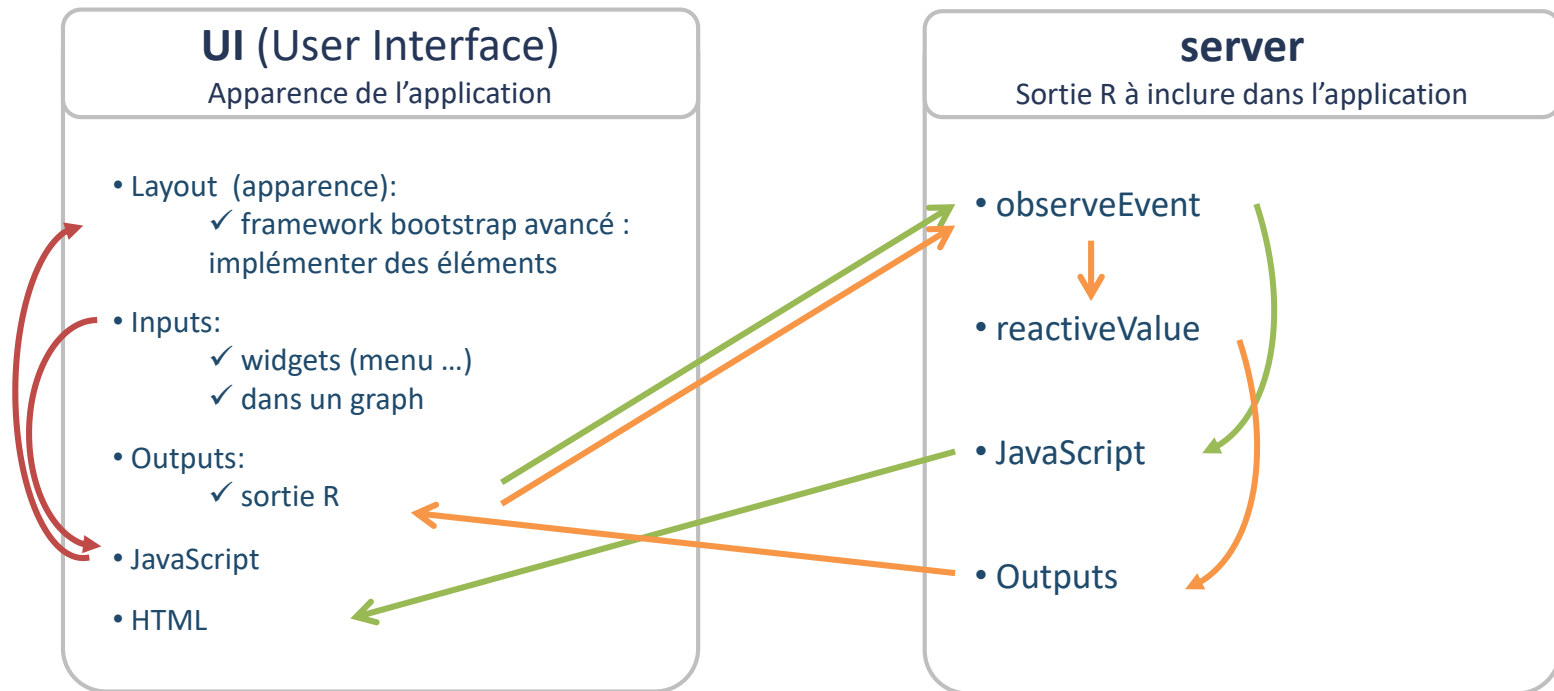


HTML, CSS et JS

Pimpé du Shiny, c'est un peu de savoir faire et de débrouille.

!!! Pimp my Shiny n'est pas un outil ou un package, c'est du shiny !!!

But : Créer des applications web interactives EN MIEUX, basées sur du code R
→ AVEC UTILISATION direct de code HTML, CSS, JavaScript



➤ création input via du code JavaScript (UI/graph)
➤ création des inputs en écrivant des bindings
JavaScript + du code HTML adéquats (UI)

➤ appeler du JavaScript via des events listener
(server)

HTML, CSS et JS

Le CSS permet de modifier l'apparence des éléments de la page et notamment leur couleur. Il peut être placé « inline » dans le code ou en passant par des « classes ».

Inline :

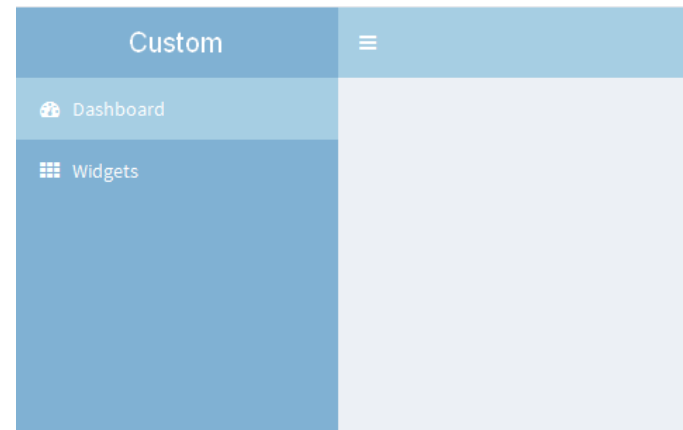
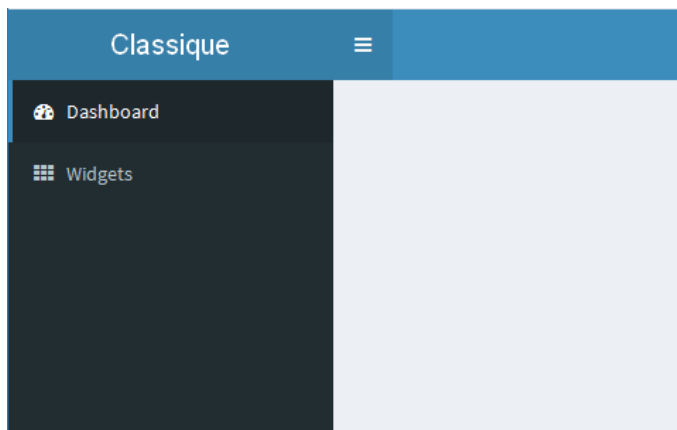
```
tags$h1("Titre")
```

↓
Titre

```
tags$h1("Titre", style = "color: steelblue;")
```

↓
Titre

Avec des classes :



Données

Mise à jour des données dans une application

Il existe 3 endroits possibles pour **charger des données** dans une application :

- **Dans le global.R** : les données sont chargées la première fois que l'application est lancée, ensuite elles sont partagées entre toutes les sessions des différents utilisateurs et ne sont rechargées quand cas de redémarrage explicite de l'application.
- **Dans la fonction server** : les données sont chargées chaque fois que l'application est lancée
- **Dans un contexte réactive** : les données sont chargées si besoin (action de l'utilisateur, besoin des données pour un output particulier)

La source de données peut être variée :

- **En local** : des Rdata ou RDS, la mise à jour se fait avec alors des batch ou manuellement
- **Dans une BDD** : vous interroger directement une base SQL, mongoDB, ElasticSearch, la fréquence de mise à jour dépendra de celle de votre BDD.

Mise à jour des données dans une application

Que cela soit en local ou dans une BDD **le chargement des données dans une application ne se fera qu'une seule fois**. Si vous souhaitez rafraîchir votre application en cours d'utilisation, il y a deux principales solutions :

- Ajouter un bouton « **Mise à jour** », vous pourrez rafraîchir les données dès que l'utilisateur cliquera dessus
- Utiliser la fonction « **reactiveTimer** » qui permet dans Shiny de réactualiser un code selon un laps de temps spécifié, l'actualisation sera alors automatique.

Démo



-MERCI-

Pour me contacter :
Mail : victor.perrier@dreamrs.fr
Twitter : [@pvictorr](https://twitter.com/pvictorr)
Web : <https://www.dreamrs.fr/>