Flexdashboard Tutorial

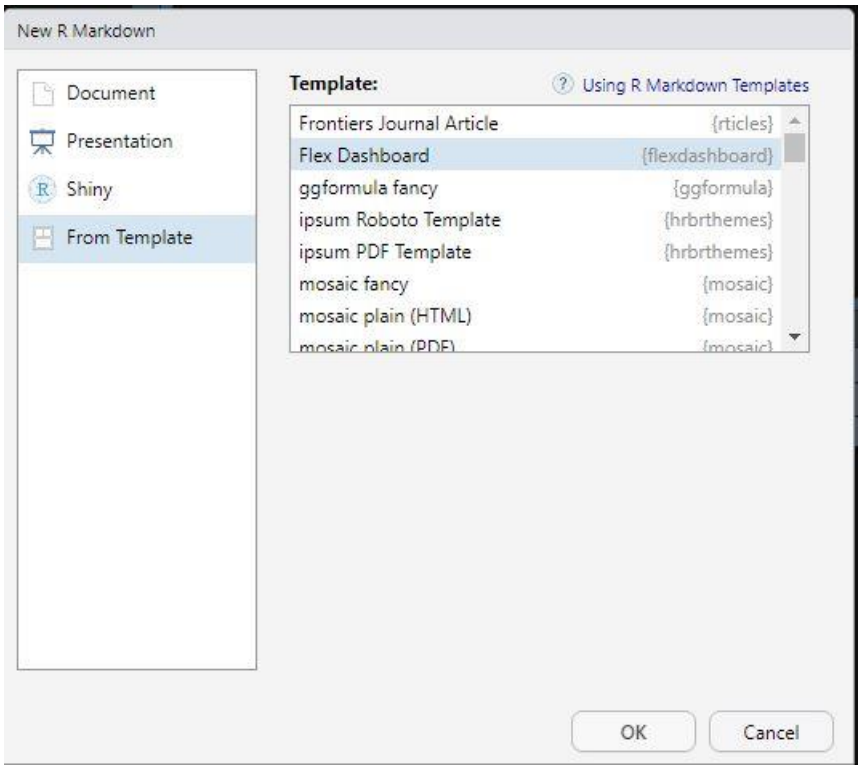
Premier pas

Les tableaux de bord ou "Dashboard" constituent des outils graphiques qui sont très souvent utilisés pour répresenter de façon simple et parlante la donnée. C'est un moyen de visualisation très puissant qui permet de regrouper un ensemble de données très variées, et réside au coeur des métiers de la donnée.
En tant qu'étudiants en MSc Data Management au sein de l'école Paris School of Business, il est bien évidemment important pour nous, et même essentiel de pouvoir présenter les informations clefs de façon claire et compréhensible de tous. Il existe un multitude d'outils et de logiciels pour çela, et le package R, flexdashboard en est un exemple parfait.

Avant toute chose, il est nécessaire d'installer le package "flexdashboard":

```
install.packages("flexdashboard")
library(flexdashboard)
```

Ensuite, pour créer un flexdashboard, il faut créer un document R avec le format output : "flexdashboard::flex_dashboard".
Il est également possible de le faire sur RStudio en utilisant la boîte de diaogue : File –> New File –> Rmarkdown –> From Templates –> Flex Dashboard.



Cela va permettre de créer un fichier template dashboard de ce genre :

```
---
title: "Test dashboard"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
---

```{r setup, include=FALSE}
library(flexdashboard)
```

Column {data-width=650}

Chart A

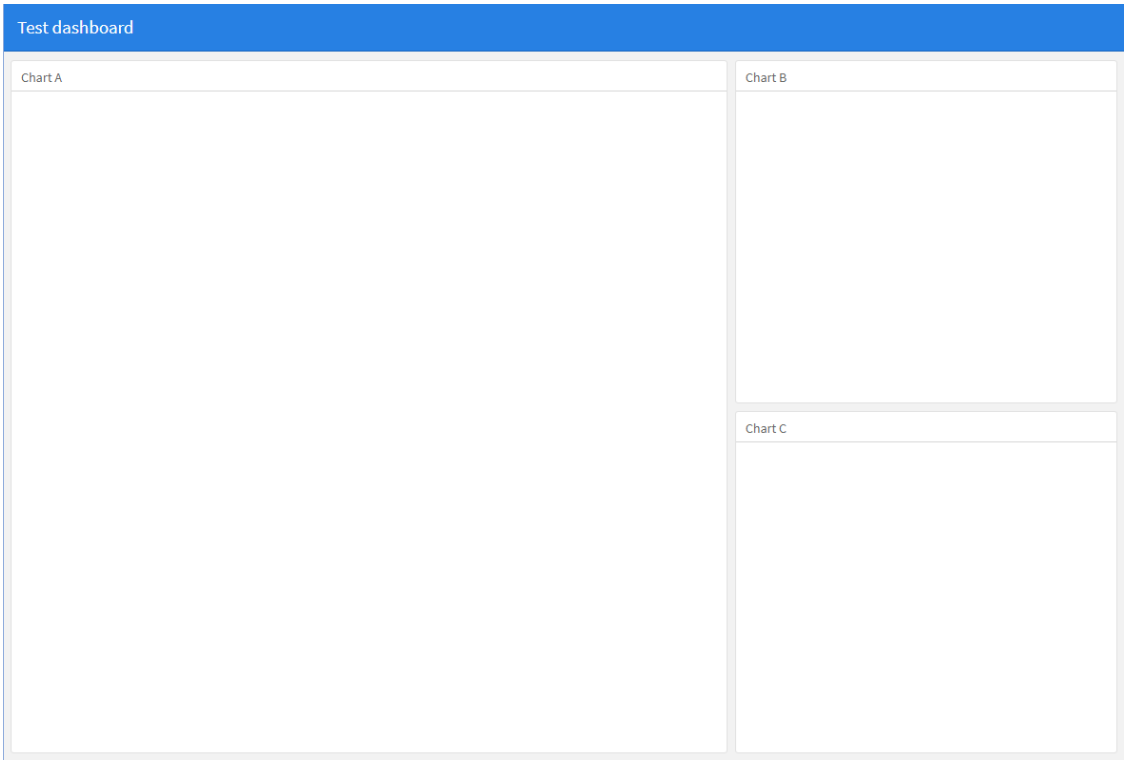
```{r}

```

Column {data-width=350}

Chart B

```{r}

```

Chart C

```{r}

```
```

Appuyez sur le bouton Knit afin d'afficher le rendu:



Bravo !! Vous venez de créer votre premier dashboard interactif. Il est vrai qu'il peut paraître un peu simpliste, mais le squelette de votre futur dashboard est bel et bien là ! Essayons de voir les différentes fonctionnalités que vous offre `flexdashboard`pour structurer et garnir votre tableau de bord.

Les basiques du packages

La disposition

Flexdashboard va vous permettre de personnaliser vos visualisations avec énormément de liberté. A commencer par la façon dont vous voulez disposer vos graphiques

En colonnes, ou en lignes

Comme vous avez pu le constater plus haut, par défaut les dashboards sur markdown se présente en colonnes, et les graphiques sont empilés verticalement dans chacune des colonnes.

Toutefois vous avez également la possibilité de choisir d'orienter vos tableaux de boards en lignes. POur cela il vous suffit de le spécifier dans l'en tête en modifiant l'option orientation par **rows**.

```
---
title: "Column Orientation"
output: flexdashboard::flex_dashboard
---

Column
-------------------------------------

### Chart 1

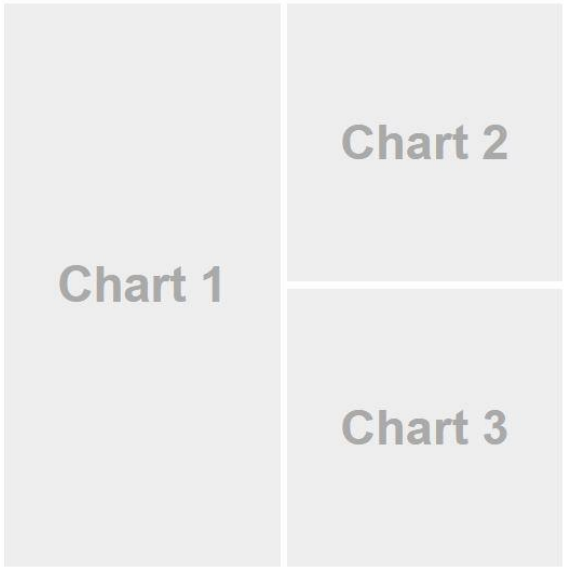
```{r}
```

Column

Chart 2

```{r}
```

Chart 3

```{r}
```
```




```
---
title: "Row Orientation"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
---

Row
-------------------------------------

### Chart 1

```{r}
```

Row

Chart 2

```{r}
```

Chart 3

```{r}
```
```

Défiler pour naviguer

Les graphiques flexdashboard sont générés de façon à ce qu'ils remplissent automatiquement la hateur du navigateur. Vous vous rendrez rapidement compte que si vous avez un grand nombre de graphique l'affichage ne va plus du tout être optimal. Flexdashboard vous donne la possibité de pouvoir défiler grâce à l'option vertical_layout dans l'en-tête, plutôt que d'avoir à tout insérer sur la même page. Cela permet de conserver la hauteur naturelle de vos graphiques, et de défiler la page si nécessaire pour naviguer à travers les graphs.

```
---
title: "Chart Stack (Scrolling)"
output:
  flexdashboard::flex_dashboard:
    vertical_layout: scroll
---

### Chart 1

```{r}
```

Chart 2

```{r}
```

Chart 3

```{r}
```
```

Organiser en onglets

Une autre astuce d'affichage que vous offre ce package est la possibilité d'organiser vos graphiques dans des tabsets.
Cela peut paraître bien pratiquer si vous souhaiter représenter plusieurs éléments dans une même ligne ou colonne, et vous évitera de vous perdre dans en essayant d'adapter tout sur un seul et même écran.
Dans de nombreux cas, les onglets sont une meilleure solution que le *vertical_layout: scroll* pour afficher un grand nombre de composants, et rendent la navigation plus simple.

Pour mettre en page une ligne ou une colonne sous forme de tabset, il vous suffit d'ajouter l'attribut *{.tabset}* dans l'en-tête de section :

```
---
title: "Tabset Column"
output: flexdashboard::flex_dashboard
---

Column
-------------------------------------

### Chart 1

```{r}
```

Column {.tabset}

Chart 2

```{r}
```

Chart 3

```{r}
```
```

Ajuster la taille

La grande diversité des graphiques que vous pourrez exploiter vont parfois vous obliger à revoir les dimensions de ces derniers afin de mettre en avant certains éléments, ou bien pour ajuster et corriger l'affichage d'autres.
Vous pouvez modifier le dimensionnement par défaut en appliquant les attribut *data-widht* et *data-height* aux lignes, aux colonnes ou même directement aux graphique concernés. Ces attributs permettent d'ajuster les dimensions (de 0 à 1000) des éléments sur la page en fonction de vos préférences.

````
---
title: "Chart Sizing"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
---

Row {data-height=650}
-------------------------------------

### Chart 1

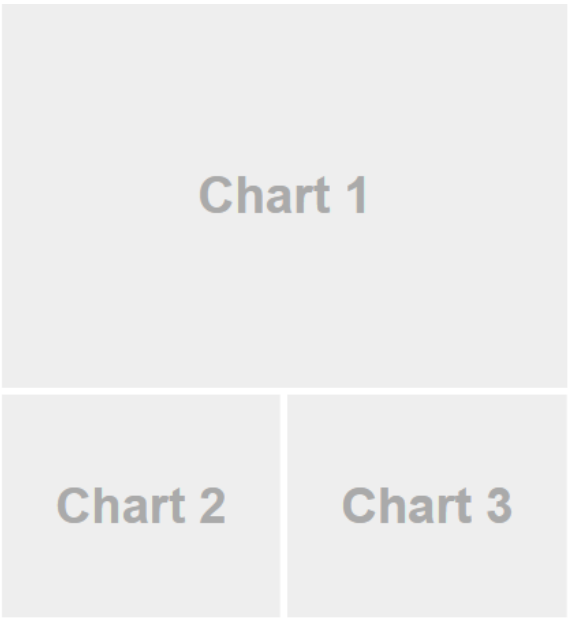
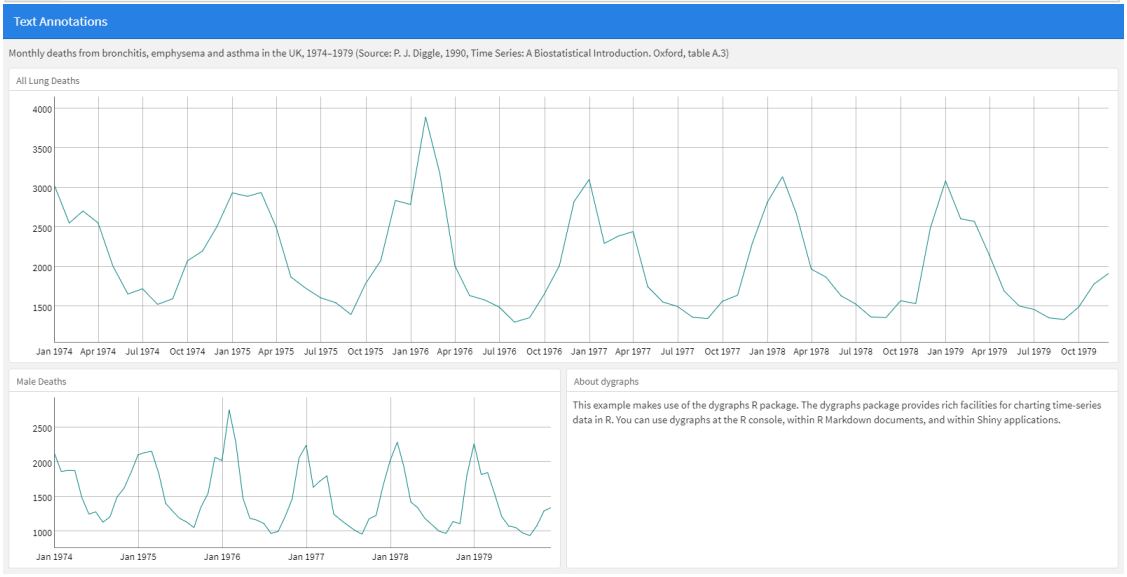
```{r}
```

Row {data-height=350}
-------------------------------------

### Chart 2

```{r}
```

### Chart 3

```{r}
```
````



Organiser en pages

Si vous souhaitez inclure plus d'une poignée de graphiques dans un dashboard, vous avez la possiblité de le diviser en plusieurs pages. Pour définir une page il vous suffit d'utiliser un en-tête de démarquage comme cela *(==========)*.
Chaque page aura son propre onglet de navigation dans la barre supérieure.

```
---
title: "Multiple Pages"
output: flexdashboard::flex_dashboard
---

Page 1
=====================================

### Chart 1

```{r}
```

Chart 2

```{r}
```

Page 2
=====================================

Chart 1

```{r}
```

Chart 2

```{r}
```
```

Les composants

Texte et annotations

Si vous voulez inclure des remarques ou des explications supplémentaires dans votre tableau de bord, vous pouvez le faire de différentes façon. Vous pouvez inclure du contenu en haut de la page avant l'introduction des sections du dashboard ou alors définir des sections d'annotations dans le dashboard même.

```
---
title: "Text Annotations"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
---

Monthly deaths from bronchitis, emphysema and asthma in the
UK, 1974-1979 (Source: P. J. Diggle, 1990, Time Series: A
Biostatistical Introduction. Oxford, table A.3)

```{r setup, include=FALSE}
library(dygraphs)
```

Row {data-height=600}

All Lung Deaths

```{r}
dygraph(ldeaths)
```

Row {data-height=400}

Male Deaths

```{r}
dygraph(mdeaths)
```

About dygraphs

This example makes use of the dygraphs R package. The dygraphs
package provides rich facilities for charting time-series data
in R. You can use dygraphs at the R console, within R Markdown
documents, and within Shiny applications.
```



HTML Widgets

Le frameword htmlwidgets va permettre d'améliorer le visuel des graphique de votre dashboard. Les graphiques basés sur htmlwidgets sont parfaits pour une utilisation avec flexdashboard car ils peuvent se redimensionner de manière dynamique et s'intégreront parfaitement dans votre dashboard. Les widgets html incluent :

1. **Leaflet**, une bibliothèque permettant de créer des cartes dynamiques.

2. **dygraphs**, fournit des fonctionnalités riches pour la création de séries de données chronologiques intéractives.

3. **plotly**, qui grâce à son interface ggplotly permet de convertir facilement vos graphiques ggplot2 en une version Web intéractive.

4. **rbokeh**, une interface vers Bokeh, un puissant framework pour la création de tracés Web

Graphiques R

Vous pouvez également utiliser n'importe quel graphique créé avec des graphiques R standard, avec flexdashboard.
Dans les tableaux de bord dynamique (shiny), ces graphiques sont automatiquement dimensionnés pour s'adapter à leurs emplacements.

Des tables

Vous avez aussi la possibilité d'inclure des données tabulaires dans flexdashboards de deux façons : * En tant que simple affichage tabulaire. * En tant que DataTable qui inclut le tri, le filtrage et la pagination.

Les boîtes de valeurs

Parmis tout les affichages possibles, on retrouve aussi des valeurs simples incluent dans des boîtes de valeur.
Vous pouvez utiliser la fonction valueBox afin d'afficher des valeurs unique avec un titre et une icône si nécessaire.

```
Row
-----------------------------------------------------------------------

### Articles per Day

```{r}
articles <- computeArticles()
valueBox(articles, icon = "fa-pencil")
```

Comments per Day

```{r}
comments <- computeComments()
valueBox(comments, icon = "fa-comments")
```

Spam per Day

```{r}
spam <- computeSpam()
valueBox(spam,
         icon = "fa-trash",
         color = ifelse(spam > 10, "warning", "primary"))
```
```

Des jauges

Les jauges permettent d'afficher les valeurs sur un compteur dans une plage spécifiée.

```
Row
-----------------------------------------------------------------------

### Contact Rate

```{r}
rate <- computeContactRate()
gauge(rate, min = 0, max = 100, symbol = '%', gaugeSectors(
  success = c(80, 100), warning = c(40, 79), danger = c(0, 39)
))
```

Average Rating

```{r}
rating <- computeAverageRating()
gauge(rating, min = 0, max = 50, gaugeSectors(
  success = c(41, 50), warning = c(21, 40), danger = c(0, 20)
))
```

Cancellations

```{r}
cancellations <- computeCancellations()
gauge(cancellations, min = 0, max = 10, gaugeSectors(
  success = c(0, 2), warning = c(3, 6), danger = c(7, 10)
))
```
```

Optimiser ses dashboards avec Shiny

A quoi ça sert ?

En utilisant Shiny avec flexdashboard, vous pouvez créer des tableaux de bord qui permettent aux utilisteurs de modifier les paramètres sous-jacents et de voir les résultats de façon immédiate.Cela se fait en ajoutant *runtime: shiny* à un flexdashboard standard, puis en ajoutant un ou plusieurs contrôles d'entrée et / ou expressions réactives qui pilotent dynamiquement l'apparence des composants dans le dashboard.
L'utilisation de Shiny avec flexdashboard transforme un rapport R Markdown statique en un document interactif. Il est important de noter que les documents interactifs doivent être déployés sur un serveur Shiny pour être largement partagés (alors que les documents

statiques R Markdown sont des pages Web autonomes qui peuvent être jointes à des e-mails ou servies à partir de n'importe quel serveur Web standard).

Pour commencer avec Shiny

Les étapes requises pour ajouter des composants Shiny à un tableau de bord flex sont les suivantes:

1. Ajouter *runtime: shiny* aux options déclarées en haut du document (avant-propos YAML).

2. Ajoutez l'attribut *{.sidebar}* à la première colonne du tableau de bord pour en faire un hôte pour les contrôles d'entrée Shiny.

3.Ajoutez des entrées et sorties Shiny selon vos besoins.

4.Lorsque vous incluez des graphiques, assurez-vous de les encapsuler dans un call à renderPlot. Ceci est important non seulement pour répondre de manière dynamique aux modifications, mais également pour s'assurer qu'ils sont automatiquement redimensionnés lorsque leur emplacement change.

Un exemple simple

````
---
title: "Old Faithful Eruptions"
output: flexdashboard::flex_dashboard
runtime: shiny
---

```{r global, include=FALSE}
# load data in 'global' chunk so it can be shared by all users of the dashboard
library(datasets)
data(faithful)
```

Column {.sidebar}
-----------------------------------------------------------------------

Waiting time between eruptions and the duration of the eruption for the
Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

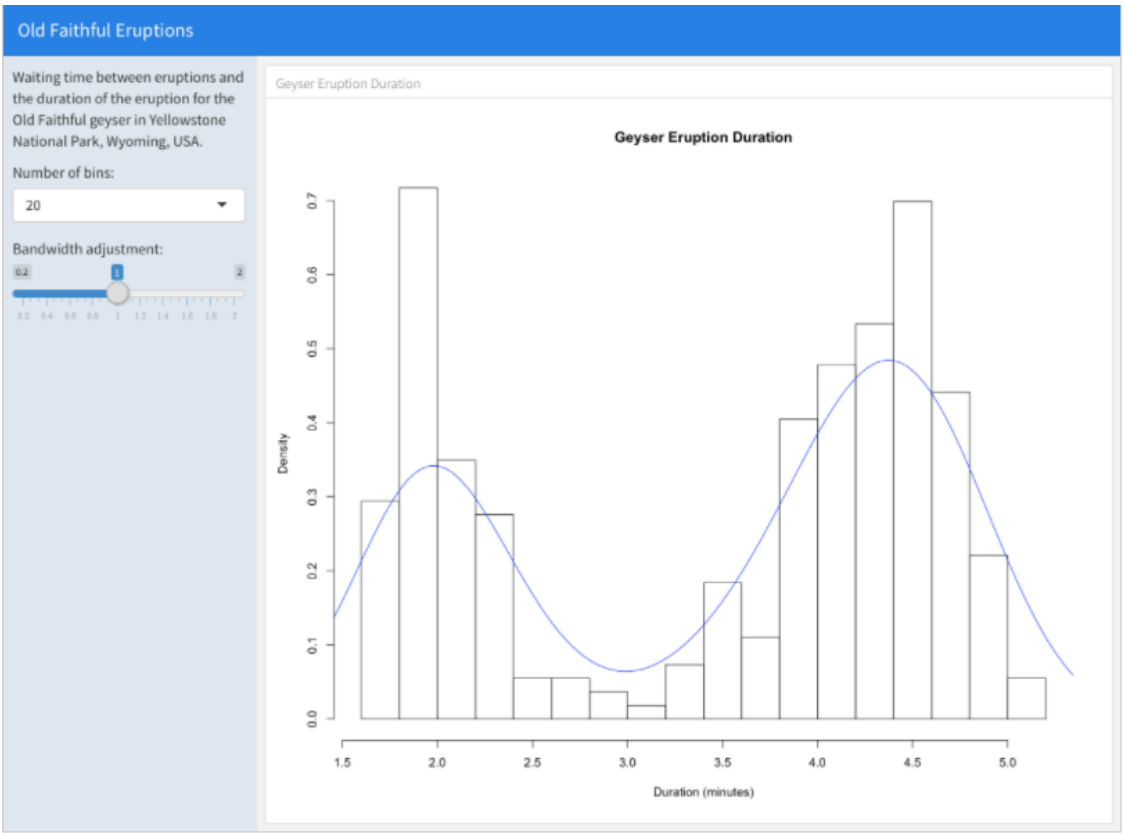
```{r}
selectInput("n_breaks", label = "Number of bins:",
            choices = c(10, 20, 35, 50), selected = 20)

sliderInput("bw_adjust", label = "Bandwidth adjustment:",
            min = 0.2, max = 2, value = 1, step = 0.2)
```

Column
-----------------------------------------------------------------------

### Geyser Eruption Duration

```{r}
renderPlot({
  hist(faithful$eruptions, probability = TRUE, breaks = as.numeric(input$n_breaks),
       xlab = "Duration (minutes)", main = "Geyser Eruption Duration")

  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
```
````



Vous avez maintenant tout les prérequis pour commencer à réaliser vos premiers dashboard intéractifs. C'est un outil non négligeable qui fera que vous vous démarquerez en montrant de façon claire et simple des données aussi hétérogènes soient-elles.
C'est en forgeant que l'on devient forgeron, alors à vos commandes.

Bibliographie

- http://rstudio.github.io/leaflet/
- https://delladata.fr/dashboard-r/
- https://rmarkdown.rstudio.com/flexdashboard/shiny.html#advanced