

RR2018

LES ADDINS RSTUDIO

Par Fanny & Victor



Rennes
5 juillet 2018

dreamRs

La data science avec un grand R



Conseil et expertise
en datascience



Accompagnement
et formation R



Développement
d'outils

ADDINS

Les 5 W's + 1BIG H !



WHAT?

Extension de RStudio



Faire des
graphiques



Modifier ou
créer des
lignes de code



Modifier des
objets



Créer des
objets, des
fichiers, des
répertoires



Aussi loin que
votre
imagination
vous portera ...

WHY?

Accélérer des tâches répétitives
et
Simplifier des manipulations complexes

Fonctions pour interagir avec les scripts via {rstudioapi}

Ajout d'éléments, modification du code



Shiny gadgets via {miniUI}

Interface pour interagir avec l'utilisateur

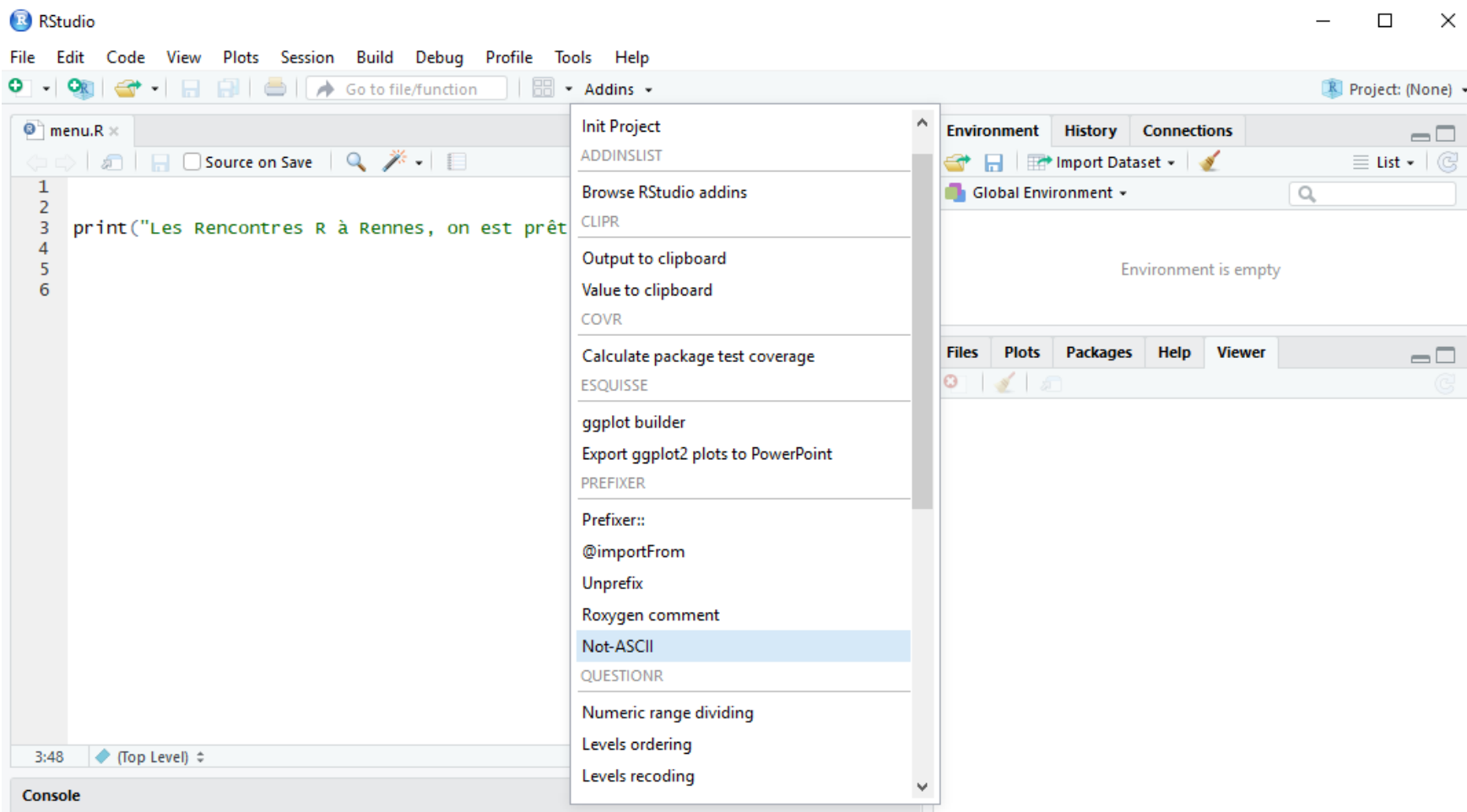


WHERE?



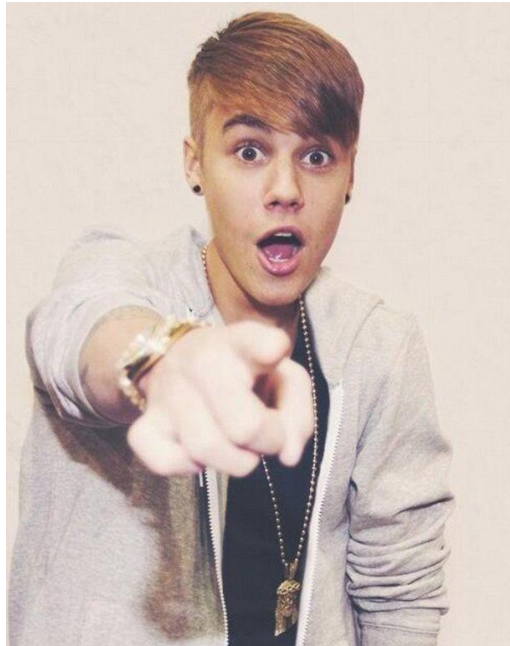
WHERE?

Le menu des addins et les raccourcis claviers



WHO?

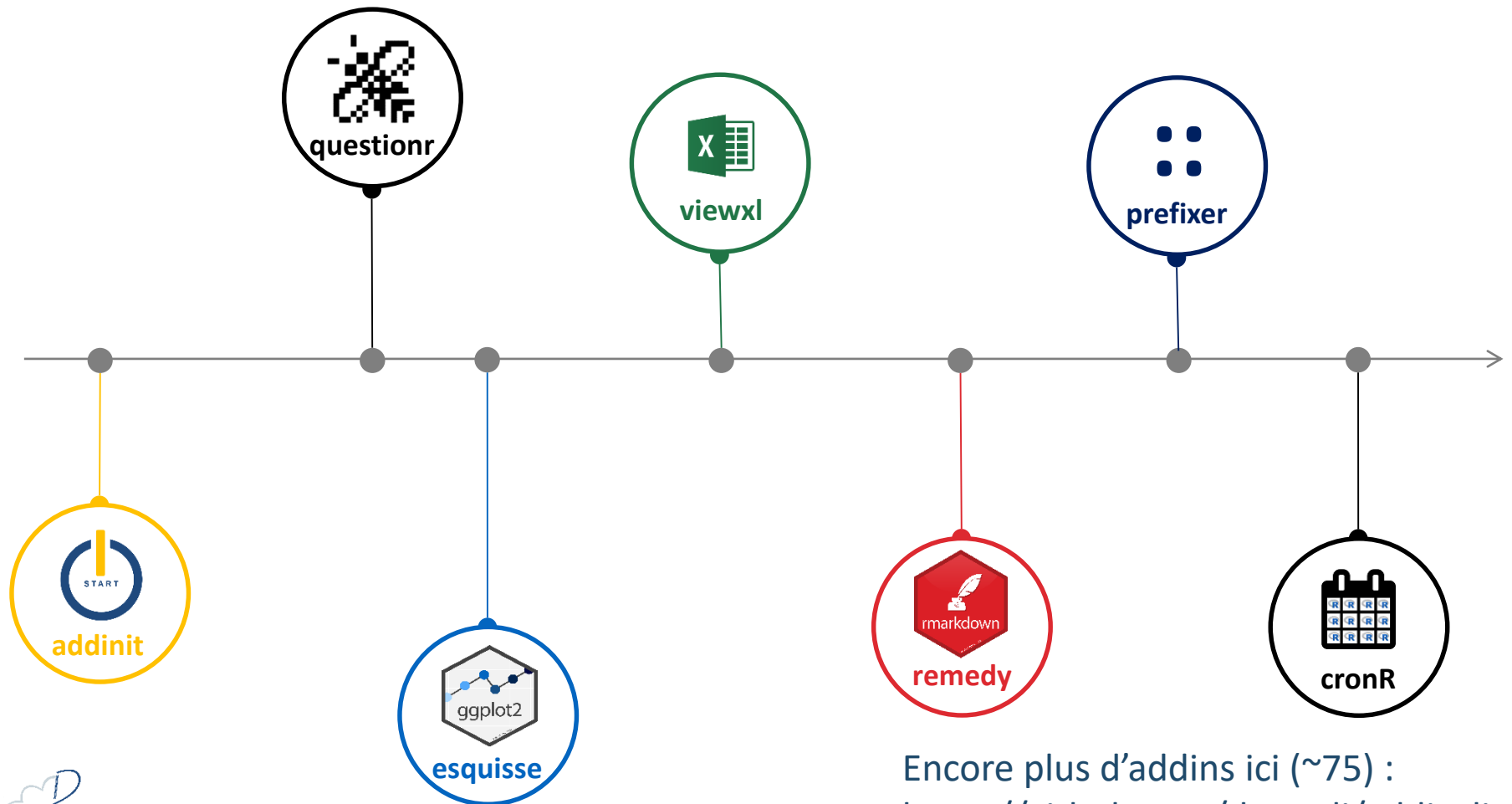
**Tous les utilisateurs de Rstudio
(développeurs, analystes, ...)**



Contrairement aux applications Shiny souvent destinés aux utilisateurs finaux, les gadgets Shiny sont des outils pour interagir avec l'environnement Rstudio de l'utilisateur.

WHEN?

À toutes étapes d'un projet



Encore plus d'addins ici (~75) :
<https://github.com/daattali/addinslist>

{addinit}



initialisation de projets RStudio normés.

Help

Init Project

Cancel

Create folders

Folders :

☒ scripts

☒ datas

☐ funs

☒ inputs

☒ outputs

☐ logs

Others :

folder1;folder2

Create folders !

Create scripts

Where :

./scripts

Script's name :

01_import

R

By :

Victor

Title :

Import des données

Packages to load :

data.table, sf

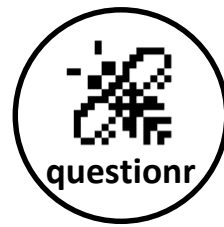
☐ Config Script

Add script

Organize your project

Organize your Shiny app

{questionr}



Icut : découpage de variables continues

Cancel

Découpage interactif

Done

Statistiques de `iris$Sepal.Length` :

	1st quartile	Median	Mean	3rd quartile	Max	NA	
Min	4.3	5.1	5.8	5.8433	6.4	7.9	0

Méthode

Quantile

Nombre d'intervalles

4

Breaks

4.3,5.1,5.8,6.4,7.9

☒ Intervalles fermés à droite (`right`)

☐ Inclure la valeur extrême (`include.lowest`)

☐ Ajouter les valeurs extrêmes si nécessaire

Variable originale

Frequency

iris\$Sepal.Length

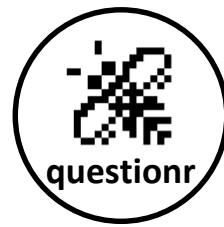
Variable et paramètres

Découpage en classes

Code et résultat

Julien Barnier : <https://github.com/juba/questionr>

{questionr}



lorder : réordonnement des levels d'un factor

CancelInteractive levels orderingDone

⛶ Autre

⛶ Ouvrier specialise


⛶ Ouvrier qualifie


⛶ Employe


⛶ Technicien

⛶ Profession intermediaire

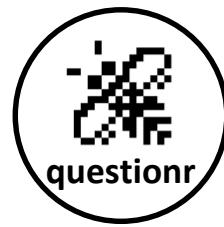
⛶ Cadre


Variable and settings


Ordering


Code and result


{questionr}





Irec : recodage des levels d'un factor

CancelInteractive recodingDone

Ouvrier specialise →	<input type="text" value="Ouvrier"/>
Ouvrier qualifie →	<input type="text" value="Ouvrier"/>
Technicien →	<input type="text" value="Technicien"/>
Profession intermediaire →	<input type="text" value="Intermediaire"/>
Cadre →	<input type="text" value="Cadre"/>
Employe →	<input type="text" value="Employe"/>
Autre →	<input type="text" value="NA"/>
NA →	<input type="text" value="Missing"/>

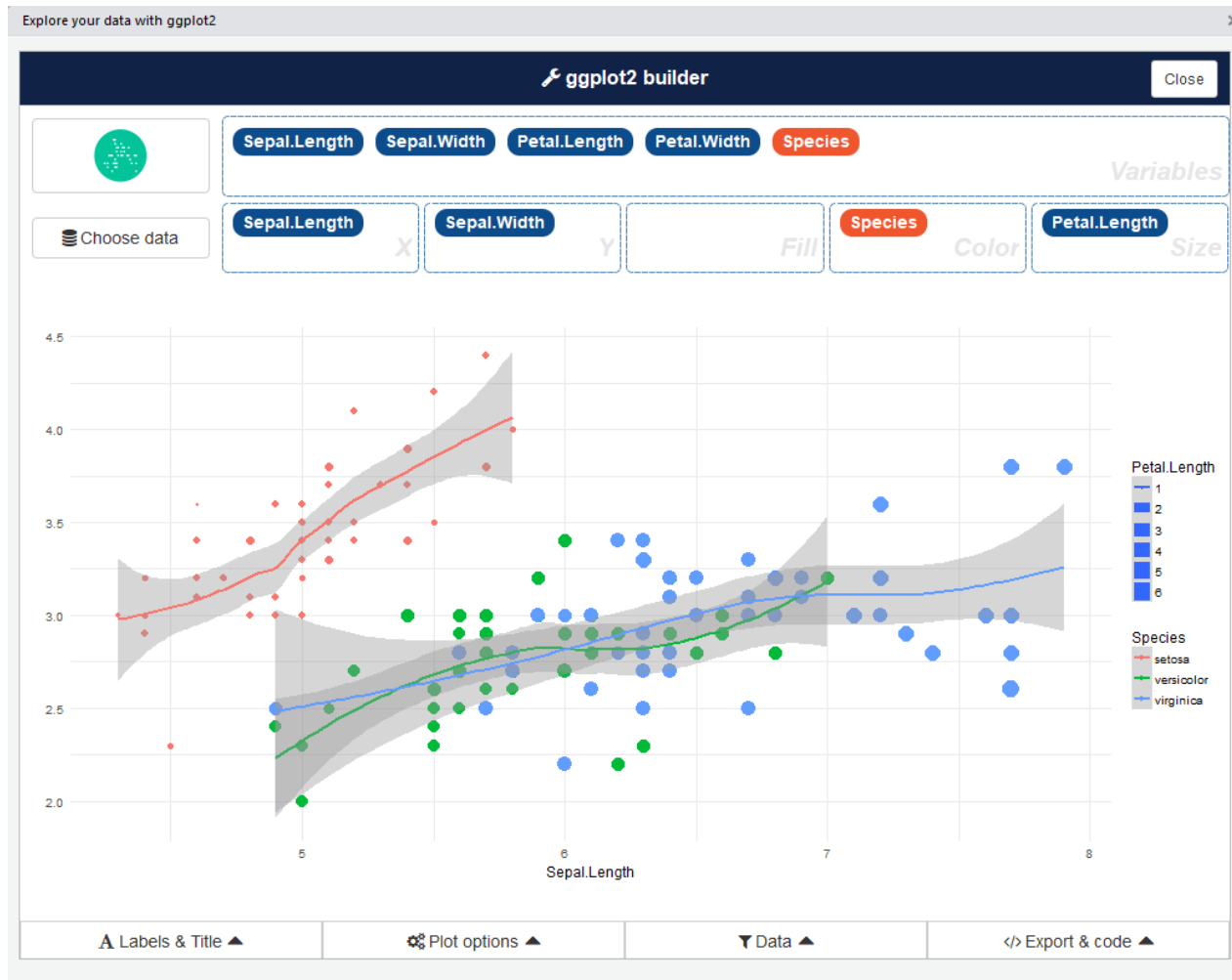

Variable and settings


Recoding


Code and result

{esquisse}

Création interactive de graphiques avec ggplot2



dreamRs : <https://github.com/dreamRs/esquisse>

{prefixer}



Ajout du namespace devant les fonctions et autres outils pour faire des packages.

The screenshot shows the RStudio IDE with a script editor on the left and a console on the right. The script editor contains the following R code:

```
1  
2  
3 fread_dir <- function(path, pattern = "\\*.csv$") {  
4   paths <- list.files(path = path, pattern = pattern, full.names = TRUE)  
5   files <- lapply(paths, fread)  
6   files <- setNames(files, paths)  
7   rbindlist(l = files, idcol = "path")  
8 }  
9  
10
```

The console on the right shows the output of the script, which is a data frame with 10 rows and 2 columns. The first column is named 'path' and the second column is named 'fread'. The output is as follows:

```
  path      fread  
1  /.../.../...  
2  /.../.../...  
3  /.../.../...  
4  /.../.../...  
5  /.../.../...  
6  /.../.../...  
7  /.../.../...  
8  /.../.../...  
9  /.../.../...  
10 /.../.../...
```

dreamRs : <https://github.com/dreamRs/prefixer>

HOW?

2 exemples concrets

Fonctions pour interagir avec les scripts via {rstudioapi}

Réorganisation des appels à library()



Shiny gadgets via {miniUI} (et {shiny} et {leaflet})

Géolocalisation de coordonnées



Code des exemples disponible ici :

<https://github.com/dreamRs/rr2018addins>



Un addin est une fonction R

Réorganisation des appels à library() dans un script d'analyse

```
reorder_library <- function() {

  # Recupère le contenu du script courant
  script <- rstudioapi::getActiveDocumentContext()$contents

  # Identifie les lignes avec des appels à library()
  indice_library <- stringr::str_which(
    string = script,
    pattern = "library\\\[[:alnum:]+\]"
  )

  # on les remplace par une section avec le nom du package
  rng_library <- Map(c, Map(c, indice_library, 1), Map(c, indice_library, 80))
  rstudioapi::modifyRange(
    location = rng_library, # position des library() dans le script
    text = stringr::str_replace(
      string = script[indice_library],
      pattern = "library\\\[[:alnum:]+\]",
      replacement = "# \\1 ----"
    )
  )

  # on les place les appels à library() en début de script
  rstudioapi::insertText(
    location = c(1, 1), # début du script
    text = stringr::str_c(c(
      "\n# Packages ----",
      script[indice_library]
    ), collapse = "\n")
  )
  return(invisible())
}
```

1 Ecrire une fonction



En exécutant la fonction, on modifie le script de l'utilisateur :

Avant

Les appels à library() sont disséminés dans le code

```
library(readr)
mes_data <- read_csv(file = "mon_csv.csv")
```

```
library(dplyr)
mes_data <- mes_data %>%
  group_by(colonne) %>%
  summarise(indicateur = sum(valeur))
```

```
library(ggplot2)
ggplot(data = mes_data) +
  geom_col(aes(colonne, indicateur))
```

Après

Les appels à library() sont organisés en début de script

```
# Packages ----
library(readr)
library(dplyr)
library(ggplot2)
```

```
# readr ---
mes_data <- read_csv(file = "mon_csv.csv")
```

```
# dplyr ----
mes_data <- mes_data %>%
  group_by(colonne) %>%
  summarise(indicateur = sum(valeur))
```

```
# ggplot2 ----
ggplot(data = mes_data) +
  geom_col(aes(colonne, indicateur))
```

HOW?

Et ensuite ?



2 Créer un package contenant la fonction R

(cela se fait en 6 minutes, coucou Diane 😊)

3 Enregistrer l'addin dans RStudio

pour cela il faut créer un fichier « .dcf » :

- Le fichier doit être situé dans ici inst/rstudio/addins.dcf (et doit s'appeler addins.dcf)
- Il doit contenir les informations suivantes :

```
DESCRIPTION
addins.dcf
README.md
reorder_library.R
geocode_addin.R

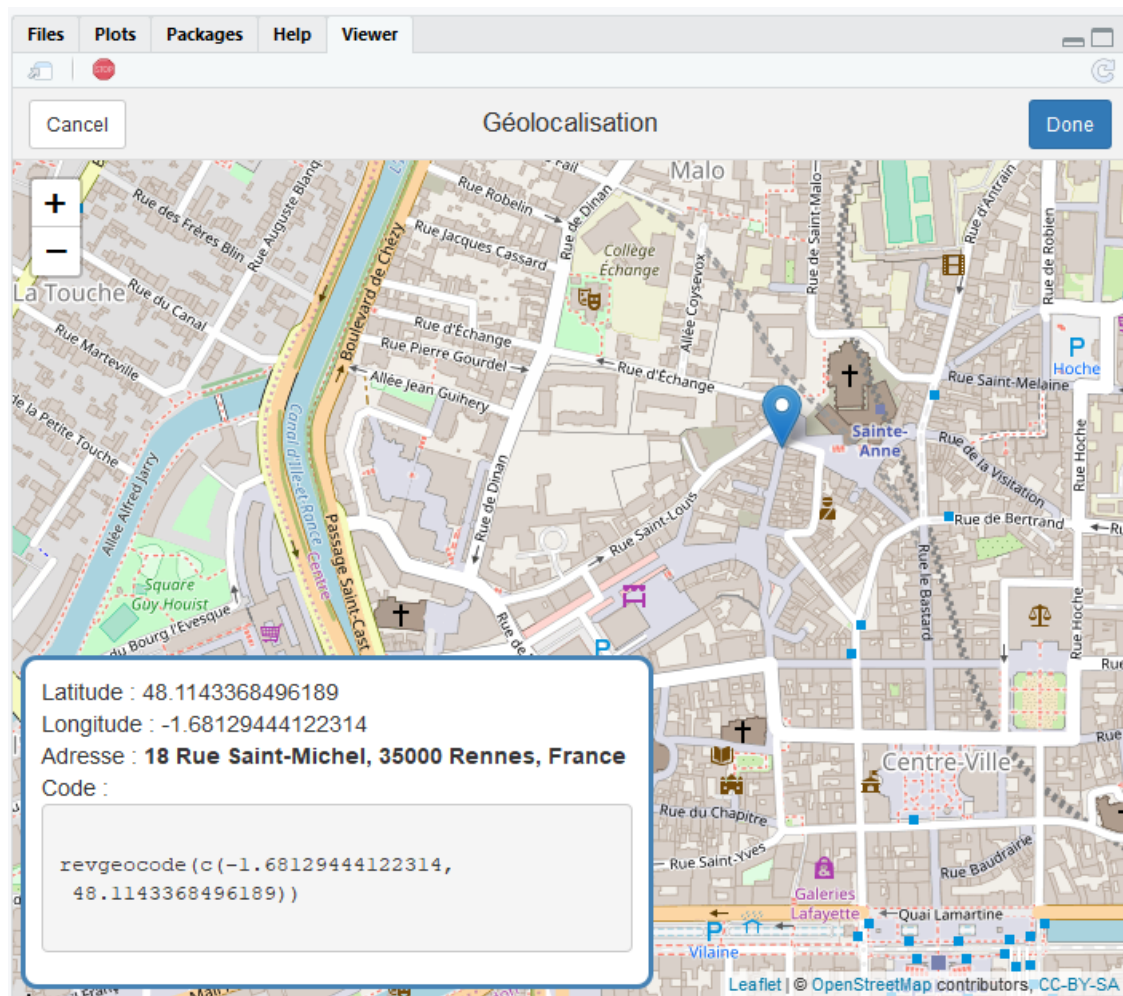
1 Name: Réorganise library()
2 Description: Positionne les appels à library() en début de script.
3 Binding: reorder_library
4 Interactive: false
5
```

Sous forme de mini app



Les addins peuvent être développés sous forme de petites applications Shiny grâce au package {miniUI}.

Exemple : géolocalisation de coordonnées à partir d'une carte Open Street Map.



1 interface (UI)



Utilisation de {miniUI} pour développer l'interface

Application avec un titre, une carte occupant toute la page et un panneau de résultat

```
ui <- miniPage(  
  gadgetTitleBar("Géolocalisation"),  
  miniContentPanel(  
    padding = 0,  
    leafletOutput(outputId = "carte", height = "100%"),  
    absolutePanel(  
      bottom = 5, left = 5,  
      uiOutput(outputId = "res")  
    )  
  )  
)
```



Même manière de procéder que pour une application Shiny

Définition des outputs dans la fonction server

```
server <- function(input, output, session) {  
  
  output$carte <- renderLeaflet({  
    leaflet() %>%  
      addTiles() %>%  
      setView(lng = -1.66, lat = 48.1, zoom = 11)  
  })  
  
  localisation <- reactiveValues(lng = NULL, lat = NULL, adresse = NULL)  
  
  observeEvent(input$carte_click, {  
    leafletProxy(mapId = "carte") %>%  
      removeMarker(layerId = "points") %>%  
      addMarkers(lng = input$carte_click$lng, lat = input$carte_click$lat, layerId = "points")  
    localisation$lng <- input$carte_click$lng  
    localisation$lat <- input$carte_click$lat  
    geoloc <- revgeocode(location = c(input$carte_click$lng, input$carte_click$lat))  
    localisation$adresse <- geoloc  
  })  
  
  ...  
}
```

3 Lancement de l'app



Trois options sont disponibles pour lancer un Shiny gadget:

- Dans le Viewer Rstudio (fenêtre en bas à droite par défaut)
- Dans une fenêtre de dialogue en surimpression de Rstudio
- Dans le navigateur par défaut de l'utilisateur

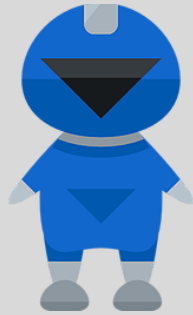
Lancement d'un Shiny gadget

Exemple d'affichage dans le Viewer RStudio

```
viewer <- paneViewer(300)  
runGadget(ui, server, viewer = viewer)
```

Merci ☺

Des questions ?



victor.perrier@dreamrs.fr
[@_pvictorr](#)



fanny.meyer@dreamrs.fr
[@_mfaan](#)

<https://www.dreamrs.fr/>
[@dreamrs_fr](#)