Université Sorbonne Paris Nord

Programmation Statistique 2

Introduction

Davide Buscaldi

Contenus (selon le PPN)

- Développement/programmation de la gestion des données, de leur analyse statistique et de la restitution des résultats.
- Connexion à des sources de données et réalisation d’interfaces utilisateur
- Automatisation avancée de l’analyse statistique par langage de script
- Notions de test et validation d’application en lien avec les principales normes et guidelines

Plan

- Programmation web avec R et Shiny
  - Objectif: dessiner des pages web interactives avec des statistiques
  - Materiel:
    - https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/
    - https://towardsdatascience.com/creating-interactive-dashboards-in-r-shiny-using-python-scripts-as-the-backend-aed40e18fb3f
    - C.Beeley, “Web Application Development with R Using Shiny”
- R avancé (Machine Learning en R):
  - Objectif: appliquer des méthodes de machine learning en R pour effectuer des prédictions
  - On utilisera le package **caret**
    - http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_package_caret.pdf
    - http://topepo.github.io/caret/index.html

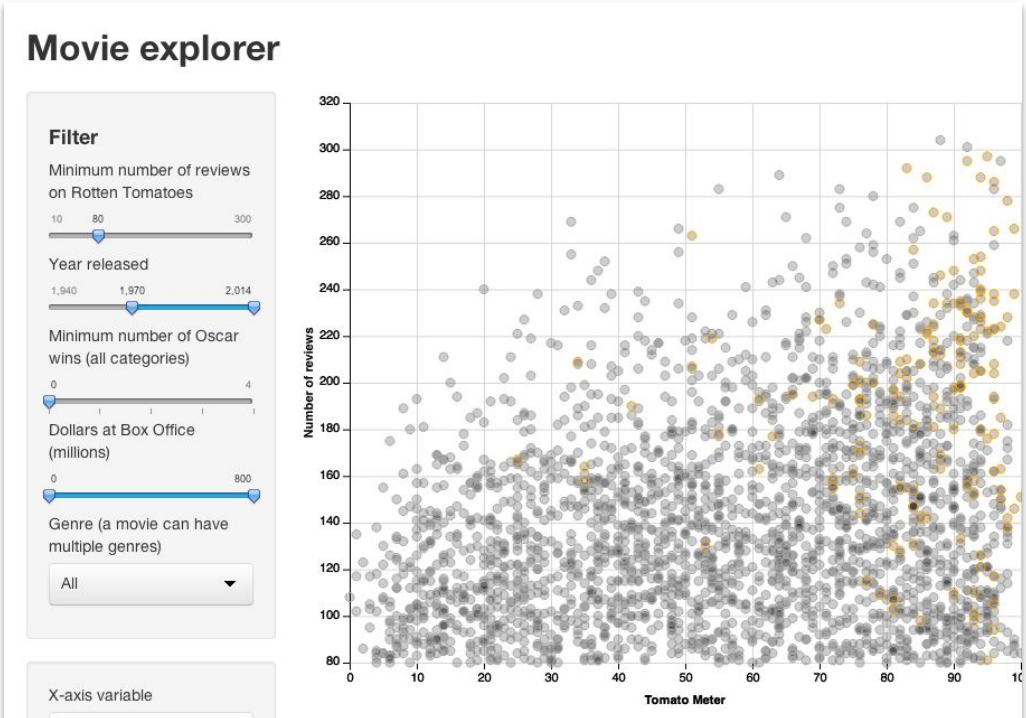
Evaluation

- 2 mini contrôles + 1 projet Shiny
- Contenu contrôle 1:
  - Interfaces utilisateur
  - Accès aux données
- Contenu contrôle 2:
  - Notions de test et validation
  - Intro Machine Learning

Introduction à Shiny

https://shiny.rstudio.com/

- Shiny est un paquet pour RStudio qui permet de développer des pages web interactives
- Exemple:



Shiny - Exemples interactifs

- https://shiny.rstudio.com/gallery/

Installation de Shiny

- Shiny n’est pas installé par défaut dans RStudio
- Il faut lancer RStudio avec droits d’administration et dans la console rentrer:
  - `install.packages(“shiny”)`
  - (attention aux guillemets)
- Une fois installé, on utilise shiny comme tous les autres paquets de R en invoquant la commande *library*
  - `library(shiny)`

Structure d’une application Shiny

- Shiny se base sur le paradigme de “programmation réactive”
  - réponse immédiate à un événement/changement de paramètre
- Une application Shiny est composée des parties suivantes:
  - L’interface utilisateur (UI - User Interface)
  - Une fonction “serveur”
  - Un appel à la fonction **shinyApp(...)**
- Tout cela dans un script nommé **app.R**

- Note: les versions antérieures de shiny utilisaient deux script nommés ui.R et server.R
  - Pour rétrocompatibilité c’est encore possible mais nous on le verra pas

Structure d’une application Shiny

- Voici la structure de base d’une application Shiny:

- Nous allons regarder un exemple avec la commande:
  - `runExample("01_hello")`

```r
library(shiny)

# Définition UI ----
ui <- fluidPage(
  ...code UI
)

# Définition logique serveur ----
server <- function(input, output) {
  ...code serveur
}

# Exécution ----
 shinyApp(ui = ui, server = server)
```

L’interface utilisateur

- Tout à l’heure on a vu l’utilisation de **fluidPage**:
  - fluidPage permet d’organiser le contenu de façon adaptive par rapport à la fenêtre du navigateur
- Exemple: cette fonction UI crée une page de type fluidPage et rajoute un box de texte pour le titre, un espace sidebar et un espace pour le contenu:
  - **sidebarLayout** prend deux paramètres: le sidebarPanel pour l’entrée, mainPanel pour la sortie

```
ui <- fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel("sidebar panel"),
    mainPanel("main panel")
  )
 )
```

Fonctions HTML

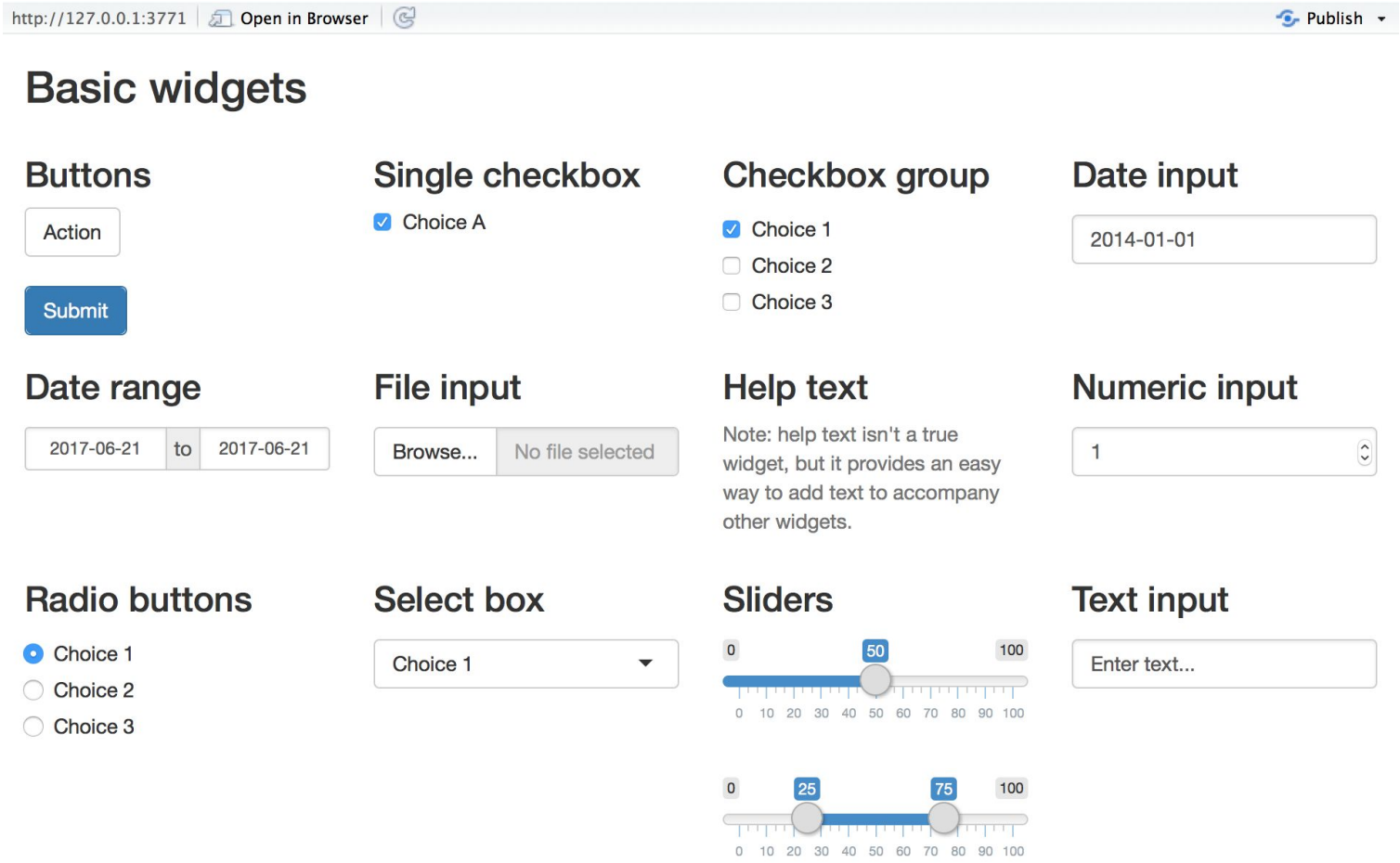
- Dans le code d'une section de l'interface on peut utiliser des fonctions qui créent des sections en format HTML 5:

Exemple:

h1("Ceci est un titre de niveau 1")

fonction Shiny	équivalent HTML	explication
p	<p>	Paragraphe de texte
h1 … h5	<h1>... <h5>	Titre de niveau 1...5
a	<a>	Lien hypertextuel
br	 	Interruption de ligne
div		Section avec format
span		Texte en ligne avec format
img	<img>	Image
strong		Gras
em		Italique
HTML		Insérer le code HTML qui suit

Les widgets


http://127.0.0.1:3771
Open in Browser
Publish
Basic widgets
Buttons
Action
Submit
Single checkbox
Choice A
Checkbox group
Choice 1
Choice 2
Choice 3
Date input
2014-01-01
Date range
2017-06-21
to
2017-06-21
File input
Browse...
No file selected
Help text
Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.
Numeric input
1
Radio buttons
Choice 1
Choice 2
Choice 3
Select box
Choice 1
Sliders
0
50
100
0 10 20 30 40 50 60 70 80 90 100
0
25
75
100
0 10 20 30 40 50 60 70 80 90 100
Text input
Enter text...

Les Widgets

- **actionButton**
- **checkboxGroupInput**
- **checkboxInput**
- **dateInput**
- **dateRangeInput** (calendrier)
- **fileInput** (chargement de fichiers)
- **helpText**
- **numericInput**
- **radioButtons**
- **selectInput** (choix multiple)
- **sliderInput**
- **submitButton**
- **textInput**

Attention! Chaque widget a besoin d'un nom et d'une label (mandatoire)

https://shiny.rstudio.com/tutorial/written-tutorial/lesson3/

Intégrer une sortie (output) à l'interface

- Jusqu'ici on s'est occupé de l'interface
- Comment lier l'interface avec les résultats d'un calcul, une graphique, etc.?
- Il faut rajouter à l'interface des objets "sortie"
  - Un peu comme des "trous" à remplir
- Shiny a les suivants objets "sortie":
  - **dataTableOutput** -> tableau "dynamique"
  - **htmlOutput** -> code HTML
  - **imageOutput** -> image
  - **plotOutput** -> graphique
  - **tableOutput** -> tableau statique
  - **textOutput** -> texte

https://shiny.rstudio.com/tutorial/written-tutorial/lesson4/

Exemple

```
ui <- fluidPage(
  titlePanel("censusVis"),

  sidebarLayout(
    sidebarPanel(
      helpText("Create demographic maps with
               information from the 2010 US Census."),

      selectInput("var",
                  label = "Choose a variable to display",
                  choices = c("Percent White",
                              "Percent Black",
                              "Percent Hispanic",
                              "Percent Asian"),
                  selected = "Percent White"),

      sliderInput("range",
                  label = "Range of interest:",
                  min = 0, max = 100, value = c(0, 100))
    ),

    mainPanel(
      textOutput("selected_var")
    )
  )
)
```

widget choix multiple nommé “var”

espace de sortie nommé “selected_var”

Le serveur

- Le serveur doit “calculer” les valeurs à rentrer dans la partie *output*
- Pour chaque type d’output, il existe une fonction *render* correspondante:
  - **renderDataTable** -> tableau dynamique
  - **renderImage** -> images
  - **renderPlot** -> plots, graphiques
  - **renderTable** -> data frame, matrices, tableaux
  - **renderText** -> chaînes de caractères
  - **renderUI** -> HTML ou objets Shiny de l’interface

- Important: les fonctions *render* prennent en paramètre une expression R entre accolades {}

Exemple

```
server <- function(input, output) {

  output$selected_var <- renderText({
    paste("You have selected", input$var)
  })

}
```

fonction de “rendering” côté serveur

valeur du contenu du widget nommé “var”

nom de l’espace de sortie dans l’UI

En synthèse

- Pour créer une application interactive Shiny il faut:
  - Définir l'interface utilisateur
    - Les entrées sont données par des **widgets**
    - Les espaces destinés à être remplis par des données sont décrits par des fonctions *output*
  - Côté serveur:
    - Utiliser une fonction **render*** pour générer la sortie
      - rentrer les expressions pour les render entre accolades {}
    - Lier les *render* aux sorties: pour chaque sortie il doit y avoir une render
    - La réactivité est créé en mettant les résultat des widgets dans une expression *render*