SAÉ 3.01 Rapport de probas/stats

Jules CHIRON, Matis RODIER, Thomas GODINEAU | INF2 FI A

12 janvier 2024

Table des matières

Introduction

Le but de cette SAÉ est de réaliser une **application web** de ticketing en PHP. Chaque utilisateur de cette plateforme peut créer un compte, se connecter et créer un ticket qui sera pris en charge par un technicien. Dans la partie probas/stats de ce projet, nous avons réalisé une page web qui permet de visualiser des statistiques sur les tickets et les connexions en utilisant le module **Shiny** de **R**.

La **première partie** de ce rapport contient une présentation du module **Shiny**. La **deuxième partie** contient une présentation de notre code et de son contenu. Enfin, la **troisième partie** contient une présentation du déploiement de notre application.

1 Présentation

Le langage **R** est un langage de programmation apparu en 1993. Ce langage a pour but de réaliser des statistiques et des graphiques en étudiant des données.

Le module **Shiny** est un module de **R** qui a commencé à être développé en 2012. Ce module permet de créer des applications web en **R**. Il permet de créer des pages web qui contiennent des graphiques et des statistiques et qui disposent d'une interface graphique permettant aux utilisateurs de choisir les paramètres des données qu'ils souhaitent étudier.

Le module **Shiny** peut être installé directement depuis la console **R** avec la commande suivante :

```
install.packages("shiny")
```

2 Notre page

Notre page Shiny contient 3 principaux éléments :

- L'interface utilisateur (*ui*)
- La partie serveur (fonction) (*server*)
- ShinyApp (fonction)

*Le code de notre page ainsi que les fichiers csv utilisés se trouvent dans le dossier **src**/ fourni avec ce rapport.*

Déroulement du programme

Nous commençons notre programme en chargeant la bibliothèque **shiny** avec la fonction `library(shiny)`. Puis on règle le port par défaut de l'application pour qu'elle soit toujours accessible sur le même port (*port 3000*) avec la fonction `options(shiny.port = 3000)`.

Nous ajoutons ensuite les éléments *ui* et *server*. Puis nous lançons l'application avec la fonction `shinyApp(ui, server)`.

Interface utilisateur

*La Figure 1 représente la partie **ui** de notre page.*

L'élément **ui** correspond à une **fluidPage** qui contient l'interface utilisateur de notre application. Notre **fluidPage** contient 3 éléments :

- Un **titlePanel** : titre
- Deux **sidebarLayout** : choix des *paramètres* et représentation des *résultats*

L'élément **titlePanel** permet de mettre un titre à notre page, on l'a appelée ici 'Statistiques'. Nos **sidebarLayout** contiennent chacun un **sidebarPanel** et un **mainPanel**.

Le premier **sidebarLayout** sert à choisir les paramètres pour la première statistique (le pourcentage de tickets selon leur status). Son **sidebarPanel** contient un **titlePanel** permettant de donner un titre au panel, ainsi que deux **inputs** (*élément de choix*) :

- Le premier input est un *selectInput* qui permet de choisir entre les différents status de ticket (open (*ouvert*), in_progress (*en cours*), closed (*fermé*))
- Le deuxième input est un *numericInput* qui permet de choisir le nombre de tickets à étudier (entre 1 et 40)

Son **mainPanel** contient deux outputs (*sorties*) :

- Le premier est de type *verbatimTextOutput* > la sortie sera sous forme de texte
- Le deuxième est de type *plotOutput* > la sortie sera un graphique

Le deuxième **sidebarLayout** sert aux choix des paramètres de la seconde statistique (le pourcentage de connexions réussies). Son **sidebarPanel** contient un **titlePanel** et un seul input :

- L'input est un *numericInput* qui permet de choisir le nombre de connexions que l'on souhaite étudier.

Son **mainPanel** contient deux *outputs* :

- Les deux sont de type *plotOutput* > il y aura donc deux graphiques.

Partie serveur

La Figure 2 représente différents outputs définis dans la partie ***serveur*** *de notre programme.*

Notre fonction **server** commence par récupérer les données des fichiers csv (*tickets.csv* et *connexions.csv*). Nous avons créé ces deux fichiers en s'inspirant des données présentes dans la base de données de l'application.
Nous paramétrons ensuite les représentations de nos différents *outputs*. Ils sont listés dans l'ordre décrit dans la partie précédente :

- Le premier est une zone de texte contenant le pourcentage de ticket selon le type choisis dans le *selectInput*.
- Le deuxième est un graphique en camembert dont chaque partie représente le pourcentage de chaque status de ticket.
- Le troisième est encore un graphique en camembert dont chaque partie représente le pourcentage de connexions réussies ou échouées.
- Le quatrième graphique est la représentation du pourcentage de connexions réussies sous forme de points.

3 Déploiement

Nous avons d'abord souhaité héberger notre serveur Shiny sur le **Raspberry Pi** fourni pour la SAÉ. Cependant, il s'est avéré impossible de lancer une application Shiny sur ce serveur. Nous avons essayer d'installer **R** et le **module Shiny** depuis les dépôts officiels mais nous avions une *erreur de bus*. Nous avons donc installé **R** par compilation depuis les sources. Cependant, nous avions toujours une erreur lors du lancement de l'application. Nous ne pouvions même pas lancé les application de test (*01_hello, 02_text ...*) fournies **par Shiny**.

Finalement, nous avons décider de déployer notre application Shiny sur un serveur **shinyApps.io**. Ce site est un site officiel de **Shiny** qui permet d'héberger des applications Shiny. Nous avons créé un compte gratuit qui permet d'héberger jusqu'à 5 applications. Cependant, une application ne peut être *active* (chargée sur un navigateur) que 25 heures par mois. Étant donné que cette page ne sert que dans ce projet (il est très peu probable qu'elle soit chargée très longtemps), nous avons choisi cette solution.

Notre application est donc accessible depuis l'adresse suivante :

`https://Boucanier.shinyapps.io/proba`

Nous avons intégré notre application dans un **iframe** sur notre site web. Elle est consultable depuis la page **Statistiques** (*stats.php*) de la plateforme, accessible pour **l'administrateur système**.

Si nous avions hébergé notre application sur le Raspberry Pi, nous aurions pu l'intégrer en mettant un **iframe** avec l'adresse **localhost:3000** (cf 2).

4 Annexes

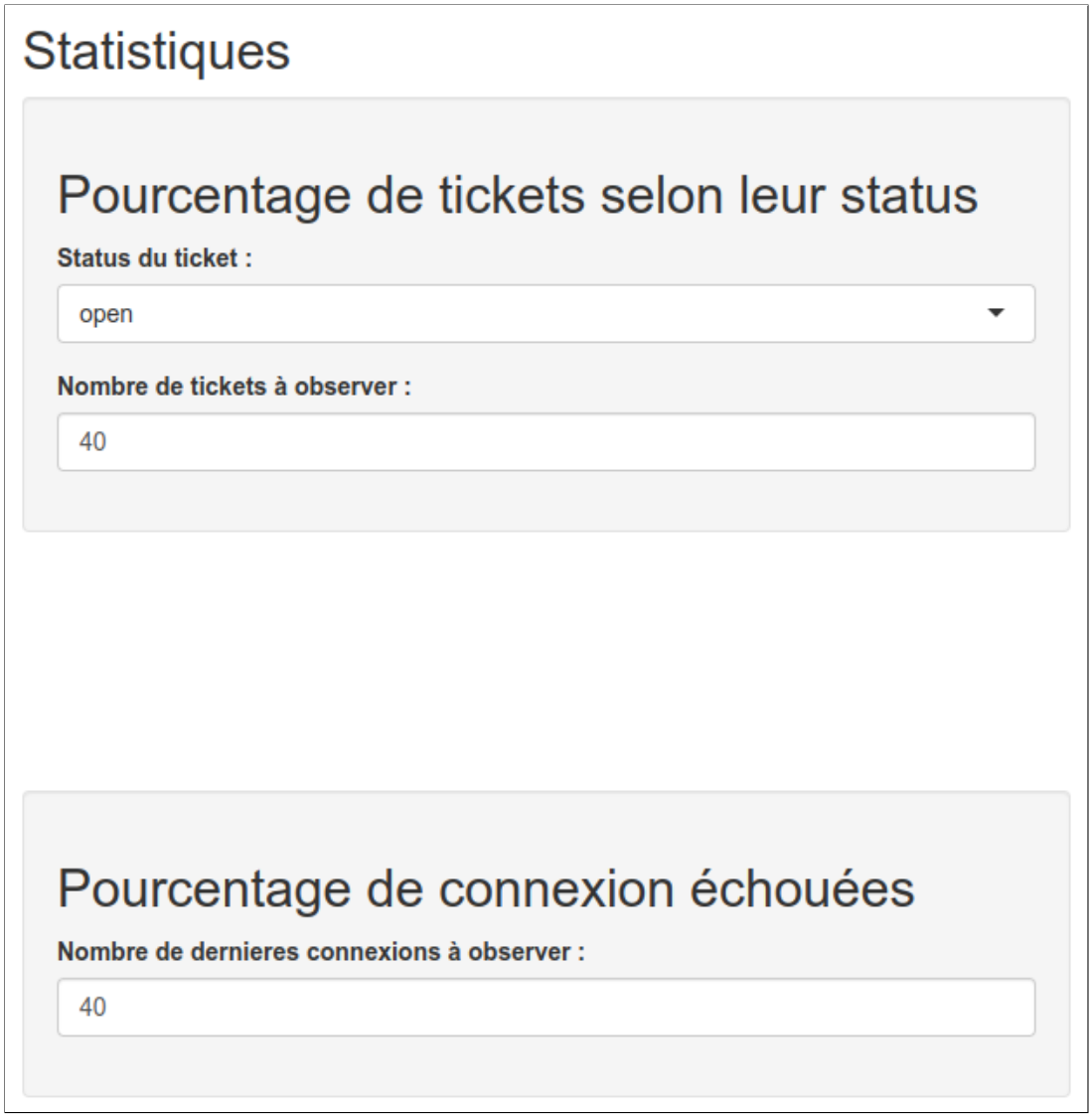

FIGURE 1 – Partie Interface Utilisateur de notre page

FIGURE 2 – Différents *outputs* de notre application