Основы программирования в R

Алла Тамбовцева

Введение в Shiny

Без инфографики и аналитических панелей – дэшбордов - сейчас никуда. Особенно полезны и интересны интерактивные аналитические панели – те, которые могут взаимодействовать с пользователем: реагировать на то, какой показатель для описания он выберет, какую цветовую гамму предпочтет, какие наблюдения его интересуют. В R возможность создавать такие панели есть, и для этого существует библиотека Shiny.

На самом деле Shiny – это не только библиотека, это целый проект. Он предлагает не только программные средства для создания аналитических панелей и приложений, но и ресурсы для хостинга и развития приложений онлайн. Кроме того, у проекта есть свой сайт http://shiny.rstudio.com и галерея с примерами дэшбордов.

Сначала установим библиотеку shiny для запуска готовых приложений shiny и создания своих:

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

Обратимся к библиотеке:

```
library(shiny)
```

Теперь в качестве первого знакомства запустим готовое приложение `"01_hello"`, которое предварительно загружено в библиотеку.

```
runExample("01_hello")
```

На это приложение можно смотреть как на шаблонный файл с кодом для своего дэшборда, своего приложения. После запуска открывается отдельное окно с дэшбордом, с которым можно поэкспериментировать, а также с кодом для его создания. Сначала посмотрим на сам дэшборд, а с кодом поработаем в следующем уроке.

Здесь у нас есть две части: гистограмма и «бегунок», который позволяет пользователю самому выбрать число столбцов в гистограмме. Можем попробовать!

Это все красиво, но логичным образом возникает вопрос: а как создать файл со своим приложением? Сейчас будем разбираться.

Так же как при создании проекта или сайта, файлы с кодом и данными для приложения должны храниться в отдельной папке. Удобнее создать ее в рабочей директории. Перед работой узнаем, какая директория является рабочей:

```
getwd()
```

```
## [1] "/Users/allat/Desktop"
```

В этой папке на следующем шаге мы создадим папку для приложения. Заходим в меню: *File — New file — Shiny Web App*.

Назовем приложение, а следовательно, и папку, *my-app*. Выберем настройки по умолчанию, один файл для кода (*single file*), позже обсудим, почему может возникнуть необходимость разделить его на две части.

Открылся шаблонный файл с примером кода для приложения, пример другой, но тоже с гистограммой. Первая и последняя строки кода в файле очень важны. Первая строка:

```
library(shiny)
```

Последняя строка:

```
shinyApp(ui = ui, server = server)
```

Без этих строк кода приложение не запустится, так как R не поймет, из какой библиотеки брать функции, и не получит явной команды запустить приложение. Запускаем – кликаем на кнопку *Run App*.

Опять дэшборд открывается в новом окне RStudio, однако на него можно посмотреть в браузере. Когда приложение запущено, в консоли RStudio отображается адрес локальной ссылки `http://127.0.0.1:7694`, которую можно скопировать в браузер. Тогда на дэшборд можно смотреть как на интерактивную html-страницу.

Итак, сами на дэшборд посмотрели, знаем, где файл лежит на компьютере (в папке *my-app* в рабочей директории) и где его искать в браузере. Как показать его другим? Опубликовать. Для этого вернемся в окно RStudio и нажмем кнопку *Publish*. Создадим аккаунт на сайте проекта Shiny.

Проект предлагает место на сервере, где можно хранить свои веб-приложения и делиться с другими. Бесплатный аккаунт позволяет создавать не более 5 активных приложений и на каждое приложение приходится не более 25 часов суммарного использования пользователями (например, 50 пользователей могут «играть» с нашей гистограммой в течение получаса каждый).

После создания аккаунта, выбираем его и нажимаем *Publish*. В консоли описываются все процессы и по завершению наше приложение, наш дэшборд, публикуется на сайте в открытом доступе.

Разбор примера с гистограммой

Гистограмму мы будем строить для одного из показателей из встроенного в R датафрейма `faithful`. Это тот самый датафрейм, который предлагается в шаблонном файле при создании нового приложения Shiny. Можем запросить описание этого датафрейма:

```
help(faithful)
```

Датафрейм содержит данные по извержениям гейзера *Old Faithful* в США и состоит из двух столбцов: `eruptions` – длительность извержения в минутах, `waiting` – время между извержениями в минутах.

Заметим, что раз мы работаем не просто с блоком кода для анализа данных или построения графика, а с приложением, все строки кода заключены в отдельные объекты. Этих объектов два. Первый из них отвечает за интерфейс для пользователя (`ui` от `user interface`). В него включаются все кнопки, «бегунки», выпадающие меню и детали, связанные с оформлением страницы. Второй объект — это функция, которая описывает действия, которые должны выполняться в зависимости от того, что выберет пользователь (`server`). В данном случае там содержится код для построения гистограммы.

Если знакомы с веб-дизайном, условно можно считать, что первая функция – это аналог файла HTML с разметкой страницы и файла CSS со стилем и оформлением этой страницы, а вторая функция – аналог файла с кодом JavaScript, который отвечает за выполнение определенных действий, активацию функций в зависимости от действий пользователя. Кстати, RStudio тоже предлагал разделить эти две функции и поместить в разные файлы, но мы выбрали один (*single file*) – помните?

Начнем с первого объекта.

```
ui <- fluidPage(
  # Application title
   titlePanel("Old Faithful Geyser Data"),
  # Sidebar with a slider input for number of bins
   sidebarLayout(
```

```
    sidebarPanel(
        sliderInput("bins", "Number of bins:",
                min = 1, max = 50, value = 30)
            ),
  # Show a plot of the generated distribution
   mainPanel(
    plotOutput("distPlot"
     )
    )
 )
```

Объект `ui` содержит целый набор вложенных друг в друга функций. К этим функциям можно относиться как к слоям. Первый слой `fluidPage` — главный, отвечает за создание страницы.

Что у нас есть на странице? Во-первых, область с заголовком, область с главной панелью, где отображается график или таблица, главный объект, который мы хотим показать, и область с боковой панелью, где обычно располагается меню. Поэтому внутри этой страницы мы добавляем панель с заголовком (`titlePanel`), боковую панель с меню (`sidebarLayout`) и основную панель `mainPanel`, в которой будет график. Внутри боковой панели размещается «бегунок» или слайдер для выбора числа столбцов (`sliderInput`).

Первый аргумент в `sliderInput()`, `"bins"` — это название переменной (`inputId`), в которой будет сохранено значение числа столбцов, которое выберет пользователь. Далее это значение будет использовано для обновления гистограммы. На втором месте стоит заголовок для слайдера (`label`) – текст, который мы видим над «бегунком» на странице. Далее указано максимальное и минимальное число столбцов, а также значение по умолчанию (`value`), то значение, на котором будет стоять «бегунок» при запуске приложения, то есть до того, как пользователь что-то выберет сам).

Последние строки этого объекта – строки кода для отображения графика на основной панели. Внутри `mainPanel` мы добавляем график, который называем `distPlot`, и помещаем его внутри функции `plotOutput` для отображения на странице.

Теперь рассмотрим вторую функцию.

```
server <- function(input, output) {
      output$distPlot <- renderPlot({
      # generate bins based on input$bins from ui.R
      x    <- faithful[, 2]
      bins <- seq(min(x), max(x), length.out = input$bins + 1)
      # draw the histogram with the specified number of bins
      hist(x, breaks = bins, col = 'darkgray', border = 'white')
   })
}
```

Функция `server()` имеет два аргумента, `input` и `output`. В `input` хранится ввод данных пользователем, то есть значение числа столбцов, на которое он поставит «бегунок». Из этого `input` мы извлечем значение `bins` (`input$bins`) и будем использовать его для построения гистограммы. В output хранится объект, который должен отображаться на выходе, в нашем случае график. Из output мы извлекаем объект `distPlot` (`output$distPlot`) и присваиваем ему результат функции `renderPlot()`, которая сгенерирует график.

Перейдем к самой гистограмме. Здесь гистограмма строится для второго столбца в датафрейме `faithful`, который уже загружен в R. Значения из этого столбца сохранены в вектор `x`.

На основе параметра `bins` (`input$bins` – число столбцов; оно было определено таким образом в функции `ui` выше) определяется число разбивок на нужное число столбцов – `breaks`, поскольку уже знакомая нам функция `hist()` принимает на вход именно такой специфический аргумент.

С остальными настройками все проще. Так как пользователь не выбирает цвет и прочие параметры, они фиксируются нами. Здесь это темно-серый цвет графика (`col = 'darkgray'`), и белый цвет границ столбцов (`border = 'white'`).

Модификация примера с гистограммой: выпадающее меню для выбора цвета

Начнем модифицировать панель с меню. В боковое меню добавим слой `selectInput` для выпадающего меню для выбора цвета гистограммы:

```
ui <- fluidPage(

   # Application title
   titlePanel("Old Faithful Geyser Data"),

   # Sidebar with a slider input for number of bins
   sidebarLayout(
      sidebarPanel(
         sliderInput("bins",
                     "Number of bins:",
                     min = 1,
                     max = 50,
                     value = 30),
         selectInput()
      ),

      # Show a plot of the generated distribution
      mainPanel(
         plotOutput("distPlot")
      )
   )
)
```

Назовем переменную, в которой будет сохраняться выбранное пользователем значение, `color`. Добавим заголовки для каждого меню: `Choose color:`. А потом добавим перечень опций выпадающего меню — вектор в аргументе `choices`:

```
ui <- fluidPage(

   # Application title
   titlePanel("Old Faithful Geyser Data"),

   # Sidebar with a slider input for number of bins
   sidebarLayout(
      sidebarPanel(
         sliderInput("bins",
                     "Number of bins:",
                     min = 1,
                     max = 50,
                     value = 30),
         selectInput("color", "Choose color: ",
                     choices = c("red", "green", "blue", "yellow"))
      ),

      # Show a plot of the generated distribution
```

```
      mainPanel(
         plotOutput("distPlot")
      )
   )
)
```

Интерфейс готов!

Теперь осталось учесть выбранный цвет в самой гистограмме — вместо фиксированного цвета поставить `input$color`, цвет `color` из ввода пользователя.

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

   output$distPlot <- renderPlot({
      # generate bins based on input$bins from ui.R
      x    <- faithful[, 2]
      bins <- seq(min(x), max(x), length.out = input$bins + 1)

      # draw the histogram with the specified number of bins
      hist(x, breaks = bins, col = input$color, border = 'white')
   })
}
```

Сохраняем изменения, запускаем приложение и радуемся!