Основы программирования в R

Пояснения к коду для дэшборда Shiny с построением графиков средствами `ggplot2`

Алла Тамбовцева, НИУ ВШЭ

Содержание

Постановка задачи

Задача. Создать дэшборд для визуализации данных по медалям, заработанным той или иной страной на Олимпийских играх. Пользователь должен иметь возможность выбрать сезон (Зимние Олимпийские игры и Летние Олимпийские игры), год, страну и тип графика для динамики числа медалей (*line plot* или *area plot*). На странице должны находиться следующие элементы:

- на боковой панели: выпадающее меню для выбора сезона; выпадающее меню для выбора года (зависит от сезона); выпадающее меню для выбора страны; радиокнопки для выбора типа графика динамики числа медалей;
- на основной странице: первые пять строк в таблице с данными с учетом выбранного сезона, года и страны; столбиковая диаграмма с числом золотых, серебряных и бронзовых медалей в выбранный год в выбранной стране; график динамики числа медалей для всех медалей сразу (с группировкой, у каждого типа медалей свой цвет) с учетом выбранного типа визуализации.

Подготовка к работе

Загрузим библиотеки и данные:

```
library(shiny)
library(tidyverse)
```

```
winter <- read.csv("/Users/allat/Desktop/winter.csv")
summer <- read.csv("/Users/allat/Desktop/summer.csv")

# объединим в один датафрейм - пригодится
games <- rbind.data.frame(summer, winter)
```

Одна строка в датафреймах выше соответствует одному спортсмену. Полное описание данных можно посмотреть здесь.

Front end

Начнем с кода, который задает внешний вид дэшборда. Давайте разобьем этот код на части. В первой части, до `mainPanel()`, мы создаем заголовок дэшборда с помощью `titlePanel()` и задаем содержимое боковой панели в `sidebarLayout()` и `sidebarPanel()`.

```
ui <- fluidPage(

   titlePanel("Olympic Games"),

   sidebarLayout(
      sidebarPanel(
         selectInput("season",
                     "Season:",
                     choices = c("Summer", "Winter")),
         uiOutput("special"),
         selectInput("cnt",
                     "Country",
                     choices = unique(games$Country)),
         radioButtons("graph",
                      "Graph",
                     choices = c("Area plot", "Line plot"),
                     selected = "Area plot")
      ),
```

На боковой панели будут находиться уже знакомые нам выпадающие меню `selectInput()` и радиокнопки для выбора ответа `radioButtons()`. В первом выпадающем меню, которому мы присвоили имя `season`, пользователь будет выбирать сезон (Летняя олимпиада или Зимняя), во втором выпадающем меню, которому мы присвоили имя `cnt`, пользователь будет выбирать страну из уникальных значений стран в датафрейме `games`. Далее пользователь будет делать выбор графика (значение по умолчанию записано в `selected`) с помощью радиокнопок, которым мы присвоили имя `graph`.

Помимо уже встречавшихся нам элементов меню на боковой панели будет находиться нечто `uiOutput("special")`. Функцией `uiOutput()` мы зарезервировали место на панели для объекта, которому присвоили имя `special` и вид которого зависит от того, что пользователь выбрал ранее. В данном случае мы хотим, чтобы на панели появлялось выпадающее меню для выбора года, но чтобы перечень лет зависел от того, какой сезон указал пользователь. Очевидно, что бессмысленно предлагать пользователю среди прочих вариантов годы, в которые проводились Зимние Олимпийские игры, если он выбрал Летние игры. Так как вид такого выпадающего меню для года зависит от ввода пользователя, описывать это меню с названием `special` мы будем внутри функции `server()`.

Во второй части, с `mainPanel()` мы создаем одну строку `fluidRow()` для выдачи таблицы с названием `table`, которая генерируется с помощью функции `dataTableOutput` из модуля `DT` библиотеки `shiny`. Потом создаем еще одну строку с двумя столбцами по 6 сантиметров, в которых мы разместим графики с названиями `distPlot` и `distPlot2`.

```
     mainPanel(
         fluidRow(DT::dataTableOutput("table")),
         fluidRow(column(6, plotOutput("distPlot")),
                  column(6, plotOutput("distPlot2")))
     )
   )
)
```

Back end

Теперь перейдем к коду для функции `server()`, которая обеспечивает работу нашего дэшборда. Код сложный. Начнем с первой части:

```
output$special <- renderUI({
    switch(input$season,
```

```
          "Winter" = selectInput("year", "Year:",
                                 choices = unique(winter$Year)),
          "Summer" = selectInput("year", "Year:",
                                 choices = unique(summer$Year)))
  })
```

Здесь мы описываем тот самый объект `special`, для которого мы зарезервировали место на боковой панели на странице. Для этого мы используем функцию `renderUI()` для генерации элемента интерфейса, а затем с помощью функции `switch()` объясняем, что вид этого элемента интерфейса зависит от значения, выбранного в меню с названием `season`. Если выбрано значение `Winter`, то на боковую панель добавляется выпадающее меню с уникальными значениями лет из датафрейма `winter`, а если — `Summer`, то из `summer`.

Перейдем ко второй части:

```
output$distPlot <- renderPlot({

   if (input$season == 'Winter'){
     games <- winter
   }else{
     games <- summer
   }

 dat <- games %>% filter(Year == req(input$year),
                         Country == req(input$cnt))

 ggplot(data = dat, aes(x = Medal, fill = Medal)) + geom_bar() +
   scale_fill_manual(values = c("#cd7f32", "lightgray", "gold")) +
   theme_dark()
 })
```

Здесь мы описываем первый график `distPlot`, который должен отображаться на странице. Создаем область для него с помощью `renderPlot()`. Далее с помощью конструкции *if-else* переопределяем значение `games` в зависимости от того, что выбрал пользователь, чтобы график строился по данным за определенный сезон. Потом отбираем только те строки, которые соответствуют только выбранному году и выбранной стране. Так как функции `dplyr` работают без `$`, чтобы R понял нас правильно, код с `$` мы поместили внутрь функции `req()` для запросов. Осталось построить столбиковую диаграмму с помощью `ggplot()`, тут уже все знакомо.

В третьем блоке кода мы описываем таблицу, которая должна отображаться в верхней части таблицы:

```
output$table <- DT::renderDataTable(DT::datatable({
     if (input$season == 'Winter'){
       games <- winter
     }else{
       games <- summer
     }
     dat <- games %>% filter(Year == req(input$year),
                             Country == req(input$cnt)) %>%
       select(-c(Year, Country))
     head(dat, 5)
     }))
```

С помощью функции `renderDataTable()` из модуля `DT` мы создаем область для таблицы, с помощью функции `datatable()` — саму таблицу. Опять переопределяем `games` в зависимости от того, какой сезон выбрал пользователь (код с *if-else* из блока выше неприменим, он написан в рамках другой функции и

в пределах этой функции невидим), и отфильтровываем строки, соответствующие выбранному году и стране. Просим показать на странице только первые пять строк с помощью `head()`.

Последняя часть кода самая длинная. Она описывает второй график `distPlot2` на странице, вид которого зависит от того, какой тип визуализации (`input$graph`) предпочел пользователь.

```
output$distPlot2 <- renderPlot({
     if (input$season == 'Winter'){
       games <- winter
     }else{
       games <- summer
     }
     dat <- games %>% filter(Country == req(input$cnt),
                             Year >= as.numeric(input$year) - 15,
                             Year <= as.numeric(input$year) + 15)
     counts = dat %>% group_by(Year, Medal) %>% tally

     if (input$graph == 'Area plot'){
       ggplot(data = counts,
              aes(x = Year, y = n,
                  group = Medal, fill = Medal)) +
         geom_area() +
         scale_fill_manual(values = c("#cd7f32", "lightgray", "gold")) +
         theme_dark()
     }else{
       ggplot(data = counts, aes(x = Year, y = n,
                                 group = Medal, color = Medal)) +
         geom_line(lwd = 2) +
         scale_color_manual(values = c("#cd7f32", "lightgray", "gold")) +
         theme_dark()
     }
   })
```

В самом начале опять переопределяем `games` и отфильтровываем строки, соответствующие выбранной стране. Только с годом теперь мы поступаем по-другому: так как график типа *line plot* или *area plot* отображает динамику показателя, мы отбираем данные не за конкретный год, а за период «выбранный год $\pm$ 15 лет». Далее полученные данные мы группируем по году и типу медалей и считаем число наблюдений в каждой группе с помощью `tally`. Результат сохраняем в датафрейм `counts`.

А дальше все знакомо — строим графики на основе датафрейма `counts` с помощью `ggplot()`. Только код для графиков, в свою очередь, тоже заключен в конструкцию *if-else*: либо мы строим *line plot*, либо мы строим *area plot*, глядя на то, что выбрал пользователь.