Aplicações em Shiny

Benilton Carvalho & Guilherme Ludwig

Aplicações na web com interfaces para bancos de dados

- `Shiny` (https://shiny.rstudio.com/) é um pacote que permite a produção de *web apps* interativos no R. Você pode criar aplicações interativas em uma página, incluir o aplicativo em documentos de `Rmarkdown`, criar `dashboards` (http://rstudio.github.io/shinydashboard/) e outras soluções.
- Na prática, aplicativos em `shiny` podem precisar salvar ou carregar dados em sessões, ou gerar *logs* de informação (de fato, o R estará sendo executado em um computador remoto); se você hospedar seu aplicativo em um serviço como shinyapps.io, poderá perder seus dados caso o servidor mude. Neste caso, você precisará levar em consideração estruturas de armazenamento de dados permanentes.

Aplicativos Shiny, visão geral

Um aplicativo `shiny` terá dois componentes:

1. A `ui` (de *user interface*), usualmente um objeto resultante da função `fluidPage`, que em seu nível mais baixo configura o layout da página.

2. O `server`, que a grosso modo, executa os comandos do R, com base em informação obtida na `ui`. O server devera ser uma função com argumentos `input`, `output` que respectivamente

    - Obtém *input* do usuário.
    - Geram *output* para a aplicação.

O aplicativo é executado com uma chamada de `shinyApp(ui, server)`.

Aplicativos Shiny, exemplo específico

Considere os dados `islands`, disponível no R, com as áreas de massas de terra com mais de dez mil milhas quadradas.

```
library(tidyverse)
library(shiny)
islands %>% sort(decreasing = TRUE) %>% head(10)
```

```
##          Asia        Africa North America South America    Antarctica
##         16988         11506          9390          6795          5500
##        Europe     Australia     Greenland    New Guinea        Borneo
##          3745          2968           840           306           280
```

```
islands %>% sort(decreasing = TRUE) %>% tail(10)
```

```
##        Melville     Southampton     New Britain     Spitsbergen          K
##              16              16              15              15
##          Taiwan          Hainan Prince of Wales           Timor       Vanc
##              14              13              13              13
```

Histograma interativo: ui

```
# exemplo minimal: inicia a fluidPlage
ui <- fluidPage(

  titlePanel("Histograma Interativo"), # titlePanel() gera título

  sidebarPanel( # Página lateral, com interação
    # Input: Slider com o número de caixas
    sliderInput(inputId = "caixas",
                label = "Número de caixas:",
                min = 2,
                max = 30,
                value = 10)
  ),

  mainPanel( # Conteúdo da página principal
    # o conteúdo será um gráfico...
    plotOutput(outputId = "grafico") # Etiqueta no output
  )

)
```

Histograma interativo: server

```
# exemplo minimal: gera a sessão; shinyApp(ui, server) liga
# automaticamente as funções input (e.g. sliderInput())
# e output (e.g. plotOutput()) na página anterior
server <- function(input, output) {

  output$grafico <- renderPlot({ # Coloca resultados no output!

    # Extrai caixas do input!
    hist(log(islands), breaks = input$caixas,
         col = "grey", border = "white",
         xlab = "Área das Ilhas")

    })

}
```

Histograma interativo: resultados

Em tese, basta adicionar `runtime: shiny` no campo de YAML do documento Rmarkdown, e rodar:

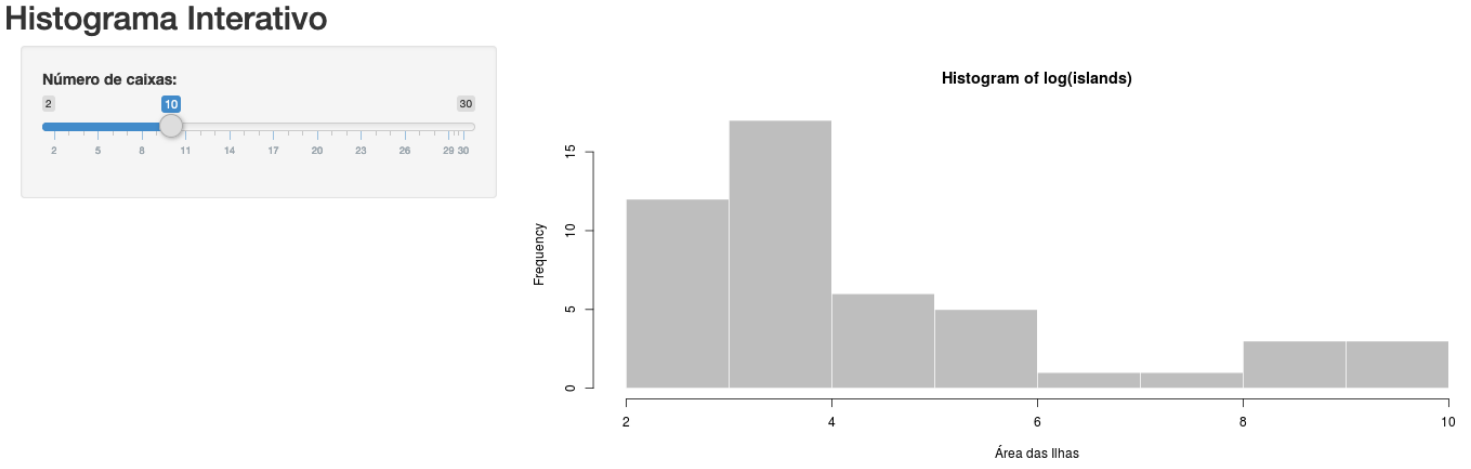
```
shinyApp(ui, server)
```

Isso gera o app na minha sessão de Rstudio. Porém, para embutir o app em uma apresentação, devo colocá-lo rodando em algum lugar. Por exemplo:

https://gvludwig.shinyapps.io/minimalExample/

Vejam também: https://github.com/yihui/knitr/issues/481

Exemplo


Histograma Interativo
Número de caixas:
2
10
30
2 5 8 11 14 17 20 23 26 29 30
Histogram of log(islands)
Frequency
0
5
10
15
2
4
6
8
10
Área das Ilhas

Construindo ui.R

O exemplo minimal `fluidPage(titlePanel(), sidebarPanel(), mainPanel())` pode ser modificado, mas o número de opções é bastante extenso. Há uma lista de opções em http://shiny.rstudio.com/articles/layout-guide.html

- Inputs comuns:
  - `numericInput()`, `textInput()` geram campos onde o usuário insere números/texto manualmente
  - `selectInput()`, `checkboxInput()`, `sliderInput()` geram campos onde o usuário pode escolher opções de input.
  - `fileInput()` para inserir arquivos (por exemplo, CSV).
  - `actionButton()` cria um botão para atualizar resultados.
- Outputs comuns:
  - `plotOutput()` (no server: `renderPlot()`).
  - `tableOutput()` (no server: `renderTable()`).
  - `textOutput()` (no server: `renderText()`).

Construindo server.R

Além das funções de `render*()`, o bloco mais importante é o chamado `reactive()`.

- Toda expressão que estiver envolta de um `reactive()` será avaliada de novo, caso o `input` mude. Ela pode ser evocada dentro de um `render*()` pelo nome, como uma função.
- Caso contrário, ela só será avaliada no começo da sessão.

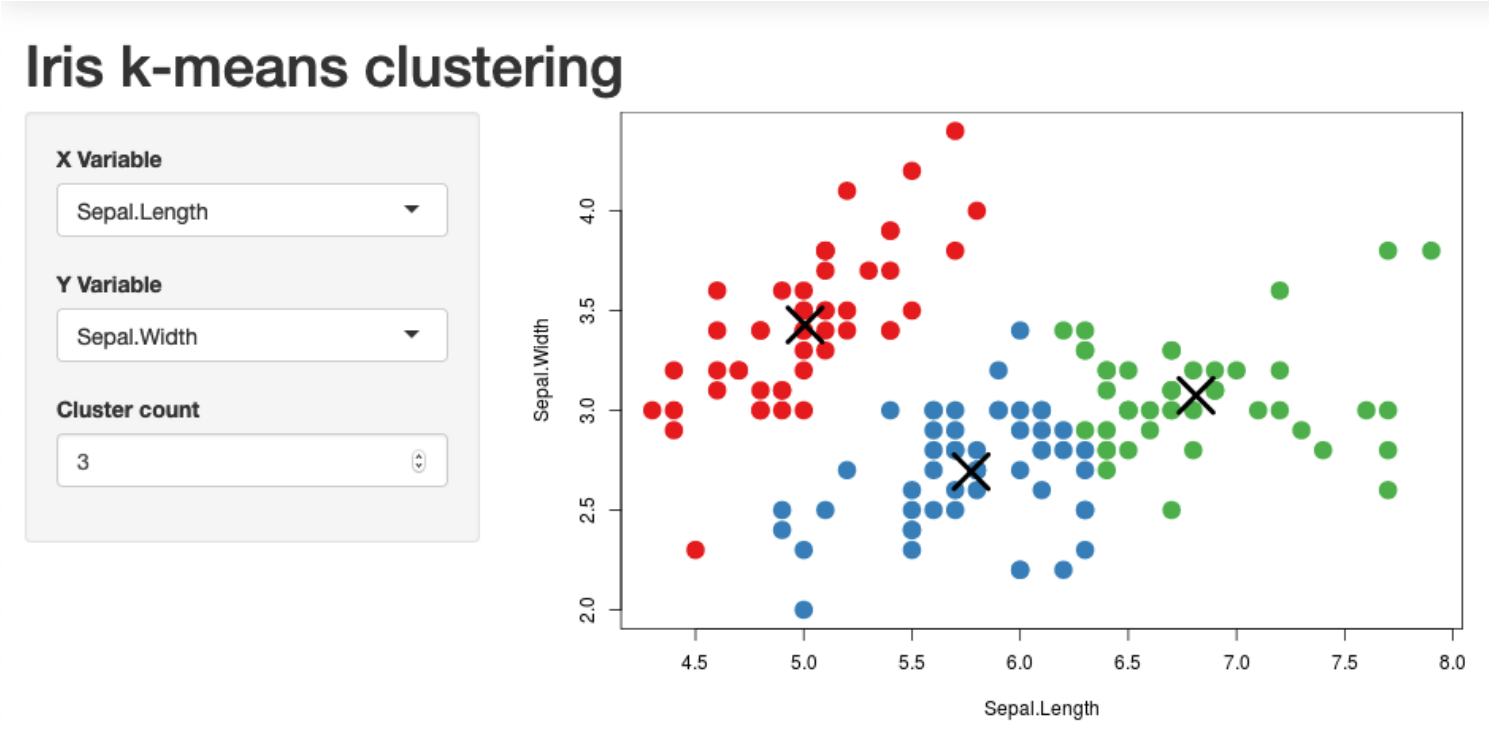
Isso é particularmente importante se vocês precisarem realizar novamente algo como um ajuste de modelo, etc.

- `observeEvent()` pode ser usado para monitorar inputs e executar código condicionalmente.

Um exemplo de reactive() do Rstudio:

https://shiny.rstudio.com/gallery/kmeans-example.html

Exemplo: reactive()



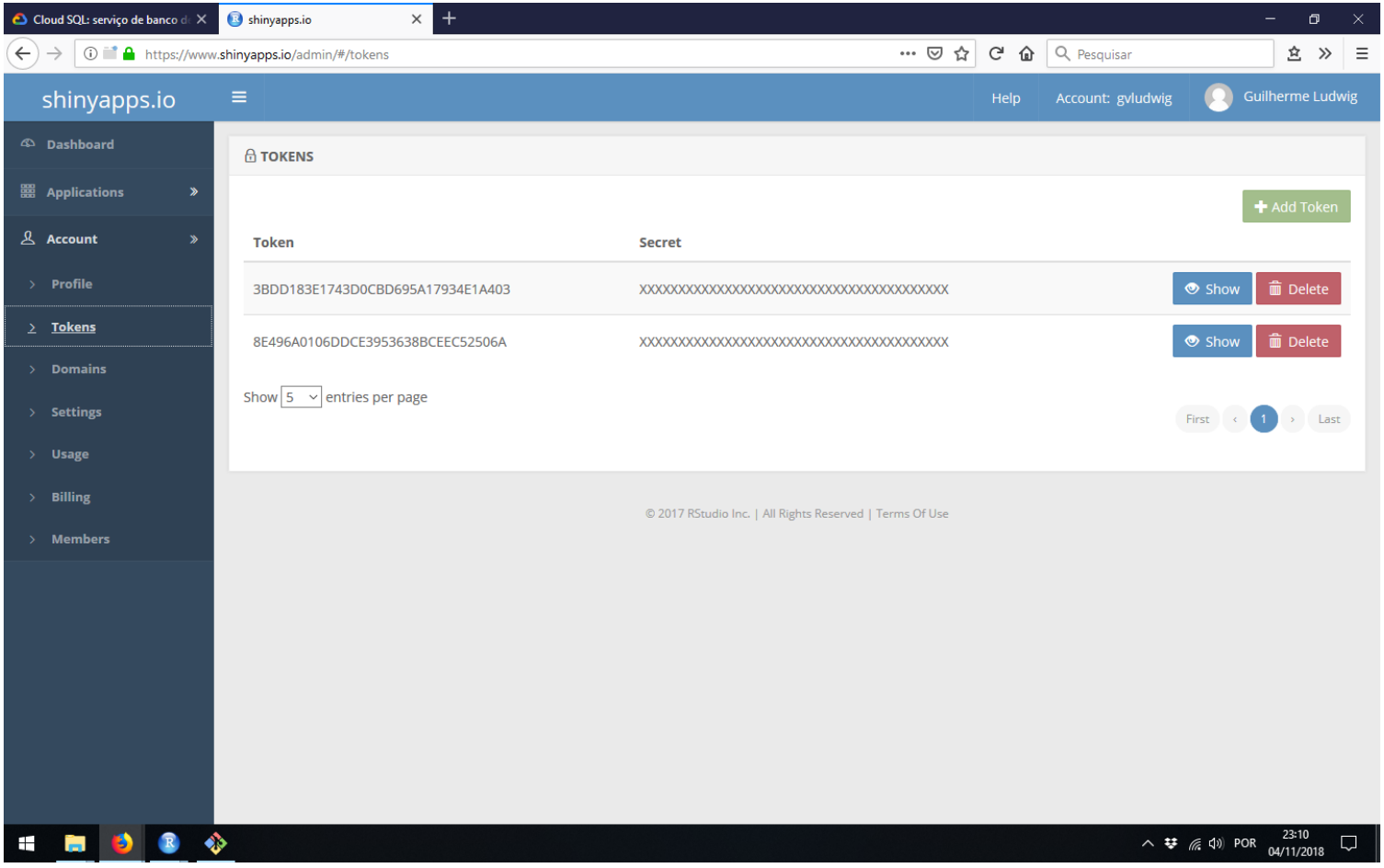
Hospedando sua aplicação online

O jeito mais fácil de hospedar sua aplicação é usando a plataforma shinyapps.io. É preciso instalar o pacote `rsconnect`, e salvar a aplicação num script `app.R` (você pode usar "File" -> "New File" -> "Shiny Web App...").

```
library(rsconnect)
```

Você também precisará registrar uma conta na plataforma shinyapps.io. Na página principal, você poderá ver suas instâncias criadas, bem como gerar tokens para submeter sua aplicação (usando o comando "publish", do Rstudio).

Hospedando sua aplicação online



Coletando dados

- Um aplicativo Shiny é útil para representações interativas (veja https://shiny.rstudio.com/gallery/), mas também pode ser usado para coletar dados.
- Um serviço como `shinyapps.io` não oferece muito espaço, e as sessões podem ser ligadas/desligadas (e os dados da sessão, perdidos). Idealmente, sua aplicação deve ser capaz de guardar os dados coletados em um banco permanente.

Diagrama de bases de dados

Method	Data type	Local storage	Remote storage	R package
Local file system	Arbitrary data	YES		-
Dropbox	Arbitrary data		YES	rdrop2
Amazon S3	Arbitrary data		YES	aws.s3
SQLite	Structured data	YES		RSQLite
MySQL	Structured data	YES	YES	RMySQL [1]
Google Sheets	Structured data		YES	googlesheets
MongoDB	Semi-structured data	YES	YES	mongolite

Fonte: https://shiny.rstudio.com/articles/persistent-data-storage.html.

Note 1: `RMySQL` está sendo substituído por `RMariaDB`.

Conectando a uma base de dados MySQL

A sintaxe da linguagem SQL é, salvo algumas diferenças de implementação, sempre a mesma, independente da base de dados. Portanto, o que vocês sabem sobre `SQLite` também é útil para `MySQL`. A principal vantagem de `MySQL` é a possibilidade de acesso remoto (usando, por exemplo, algo como https://cloud.google.com/sql/).

```
library(DBI)
library(RMariaDB)
conn <- dbConnect(RMariaDB::MariaDB(),
                  host = "mydb.mycompany.com",
                  user = "gvludwig", password = "*******")
dbGetQuery(conn, "SELECT * FROM table") # etc.
dbWriteTable(conn, "table", someData) # etc.
dbDisconnect(conn)
```

Acessando DBs no Shiny

Em geral, o acesso a DB ocorre no componente `server` do aplicativo Shiny. Como não tenho um servidor, vou criar uma base em SQLite mesmo. Primeiro, faço um `ui` simples:

```
library(shiny)
library(RSQLite)
# ui é basicamente a mesma
ui <- fluidPage(
  sidebarPanel(
    numericInput(inputId = "nota",
                 label = "Nota que espero:",
                 min = 0,
                 max = 10,
                 value = 7),
    actionButton("submeter", "Submeter")
  ),
  mainPanel(
    textOutput("saida")
  )
)
```

Acessando DBs no Shiny

Devo criar um banco de dados para guardar os resultados (neste momento, está vazio)

```
conn <- dbConnect(SQLite(), "dataShiny.db")
dbExecute(conn, "CREATE TABLE notas(valor int)")
dbDisconnect(conn)
```

Acessando DBs no Shiny

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

  formData <- reactive({
    data.frame(valor = as.integer(input$nota))
  })

  observeEvent(input$submeter, {
    saveData(formData())
  })

  output$saida <- renderText({
    input$submeter # Atualiza quando submeter é clicado
    if(length(loadData()$valor) == 0){
      return(NA_integer_)
    } else {
      return(mean(loadData()$valor))
    }
  })
}
```

Acessando DBs no Shiny

```
saveData <- function(data) {
  db <- dbConnect(SQLite(), "dataShiny.db")
  teste <- dbGetQuery(db, "SELECT * FROM notas")
  if(nrow(teste) >= 1){
    dbWriteTable(db, "notas", data, append = TRUE)
  } else {
    dbWriteTable(db, "notas", data, overwrite = TRUE)
  }
  dbDisconnect(db)
}

loadData <- function() {
  db <- dbConnect(SQLite(), "dataShiny.db")
  data <- dbGetQuery(db, "SELECT * FROM notas")
  dbDisconnect(db)
  data
}
```

Executando...

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