

EAMC 2020

Desenvolvimento de Web Apps via Shiny R, O Ambiente Shiny

Rafael Silva Pereira
Orientador: Dr Fábio Porto

Laboratorio Nacional de Computação Científica

17/07/2019

O ambiente Shiny

- ▶ O ambiente shiny é utilizado na linguagem R para desenvolvimento de aplicações Web
- ▶ Este é dividido em tres partes, estas são:
 - ▶ Funções
 - ▶ Interface Gráfica
 - ▶ Servidor

Funções

- ▶ A região de funções pode ser escrita tanto fora da região do servidor quanto dentro.
- ▶ As funções aqui definidas são funções que não se utilizam das propriedades da biblioteca shiny, e serão utilizadas por outras partes do shiny pela sua aplicação.
- ▶ É possível implementar todas as funções em questão em arquivos separados e utilizar o comando source para incluí-las no arquivo principal.

Interface Grafica (UI)

- ▶ A interface gráfica do ambiente shiny é construída através de HTML, CSS, e javascript.
- ▶ Apesar disto não é necessário o conhecimento de nenhum destes para este módulo.
- ▶ Diversos wrappers para diferentes objetos de interação existem dentro do pacote e assim criação utilizando-se estes wrappers é possível criação do ambiente gráfico com pouca dificuldade.
- ▶ Aqueles que possuem conhecimento das linguagens acima podem incluir código destes dentro do ambiente UI para maior customização da interface gráfica.

Servidor

- ▶ O modulo servidor contera toda a parte de controle de sua aplicação.
- ▶ Neste será incluída o código que utiliza-se das entradas e manipula as saidas de sua aplicação.
- ▶ Na parte do servidor é onde definimos funções reativas, uma propriedade da biblioteca que permite sua escalabilidade.
- ▶ Utilizando- se shinyjs é possível integrar javascript neste processo.

Exemplos de UI

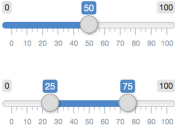
Buttons <input type="button" value="Action"/> <input type="button" value="Submit"/>	Single checkbox <input checked="" type="checkbox"/> Choice A	Checkbox group <input checked="" type="checkbox"/> Choice 1 <input type="checkbox"/> Choice 2 <input type="checkbox"/> Choice 3	Date input <input type="text" value="2014-01-01"/>
Date range <input type="text" value="2017-06-21"/> to <input type="text" value="2017-06-21"/>	File input <input type="button" value="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 <input type="text" value="1"/>
Radio buttons <input checked="" type="radio"/> Choice 1 <input type="radio"/> Choice 2 <input type="radio"/> Choice 3	Select box <input type="text" value="Choice 1"/>	Sliders 	Text input <input type="text" value="Enter text..."/>

Figura 1: Exemplos de tipos de input

Exemplos de UI: Outputs

dataTableOutput	DataTable
htmlOutput	HTML
imageOutput	imagem
plotOutput	Grafico
tableOutput	Tabela
textOutput	texto
uiOutput	UI Dinamica
verbatimTextOutput	Texto

Tabela 1: Tipos de output

Reatividade

- ▶ O conceito de reatividade em shiny é tratado da seguinte forma:
- ▶ Considere uma função F que depende de atributos a_1, a_2, a_3
- ▶ Esta função é chamada por uma função G que utiliza-se de a_1, a_2, a_3, a_4
- ▶ Na primeira execução de G , F é executado também. onde F e G guardam seus retornos.
- ▶ Caso G seja chamado novamente sem alteração de a_1, a_2, a_3, a_4 , então a função G não será executada, apenas retornará o ultimo retorno salvo.
- ▶ Caso apenas o atributo a_4 tenha sido modificado, então G executará, mas a chamada da função F apenas retornara seu ultimo estado.
- ▶ Apenas na modificação de atributos que modificam o retorno de ambas as funções, tanto F quanto G serão reexecutados.
- ▶ Assim reduzimos as tarefas a serem computadas para o minimo necessário para integridade de resultados.

Funções Reativas

- ▶ Funções reativas se diferenciam de funções comuns no sentido que estas funcionam da seguinte forma:
 - ▶ A função F depende de atributos a,b,c
 - ▶ Na primeira chamada é executada.
 - ▶ Ela vincula a sua chamada seu retorno
 - ▶ Caso seja rechamada sem a mudança destes parametros, retorna o retorno vinculado.
 - ▶ Caso um dos parametros mude, ela será reexecutada
- ▶ Sua sintaxe é executada como `Funcao < - reactive()`

Eventos Reativos

- ▶ Os eventos reativos se diferem de funções reativas da seguinte forma?
- ▶ Um evento reativo é vinculado a mudança de valores de um conjunto de atributos determinados pelo desenvolvedor.
- ▶ Assim se executa-se uma função que depende de parâmetros a,b,c podemos modificar os 3 e esta função não será executada sem o atributo vinculado ao evento ter mudança
- ▶ Normalmente vinculamos um evento reativo a um botão por exemplo, tornando ele significativamente útil quando esperamos que o usuário queira modificar diversos parâmetros para que ocorra a execução de uma função.
- ▶ A sintaxe deste é escrita como:
- ▶ `Evento < -eventReactive(input$botao,`

Exemplo

Vejamos o arquivo Exemplo1.R

Primeiro Desafio, Visualização de informações do dataset iris

- ▶ Aqui paramos brevemente a teoria para explorar uma aplicação simples
- ▶ Buscamos construir uma aplicação minimalística para explorar o conteúdo de um dataset específico chamado iris.
- ▶ O objetivo deste é criar uma aplicação em que o usuário definira uma coluna para representar o eixo x, uma para o eixo y, onde as opções são as colunas contidas no dataset.
- ▶ Ao modificar uma coluna o programa deve gerar um gráfico visualizando um atributo contra outro por pontos.

Ui output, criando inputs dinâmicos

- ▶ Caso algum de nossos inputs tiver que modificar seus valores validos dado uma condição utilizamos inputs dinamicos.
- ▶ Estes são construidos da seguinte forma.
- ▶ Em UI
 - ▶ `uiOutput(" Saida1")`
- ▶ Em server
 - ▶ `output$Saida1 < -renderUI()`

Segundo Desafio

- ▶ Expanda a aplicação acima, utilizando inputs dinâmicos para que o usuário possa escolher entre o dataset iris e mtcars

Inputs de arquivos

- ▶ Inputs de arquivos são tipos de inputs que esperam um arquivo de formato específico.
- ▶ Estes permitem que um usuário entregue seu próprio conjunto de dados a ser analisados.
- ▶ Devido a isto sua aplicação provavelmente vai requerer outputs reativos para que os inputs se adaptem a informação do arquivo, dependendo do tipo de serviço que quer prover
- ▶ Discutimos a seguir algumas coisas a se considerar quando trabalhamos com inputs de arquivos.

Inputs de arquivos

- ▶ Este é declarado da forma:
- ▶ `fileInput(id="Arquivo1","Choose file",accept=c(".csv",".txt"))`
- ▶ id se refere a um identificador do html para referenciar este input, ele deve ser unico.
- ▶ O segundo item se refere ao texto escrito abaixo dele
- ▶ accept indica os tipos de formatos que ele espera do arquivo.
- ▶ Este arquivo tem os seguintes atributos:
 - ▶ name
 - ▶ size
 - ▶ type
 - ▶ datapath
- ▶ E estes são acessados como:
 - ▶ `input$Arquivo1$name`
 - ▶ `input$Arquivo1$size`
 - ▶ `input$Arquivo1$type`
 - ▶ `input$Arquivo1$datapath`

Exemplo

Vejamos o arquivo Exemplo2.R

Exercicio

- ▶ Considere o problema proposto para a aplicação do iris do início da aula.
- ▶ Construa a mesma aplicação aonde agora o usuário deve entregar um dataset com formato csv. e deve ter a opção de gerar gráficos x por y de qualquer conjunto de atributos do dataset.

TabPanels

Tabpanels permitem criar varias abas para sua aplicação, aonde cada aba tem uma função diferente em sua aplicação.

Estas diferentes abas podem conter por exemplo etapas sequenciais em um processo de análise, ou diferentes tecnicas aplicaveis ao teu conjunto de dados.

Estas são escritas da seguinte forma:

```
tabsetPanel("Principal",  
  tabPanel("Plot", plotOutput("plot")),  
  tabPanel("Summary", verbatimTextOutput("summary")),  
  tabPanel("Table", tableOutput("table"))  
)
```

Exemplo

Vejamos o arquivo Exemplo3.R

Exercicio

- ▶ Considere o projeto criado anteriormente
- ▶ Insira um modulo em que podemos visualizar ou o dataframe puro. ou uma tabela estatistica que para variaveis numericas retorna minimo media maximo, e para categoricas retorna numero de palavras unicas,frequencia da moda, frequencia media de repetição de palavras.
- ▶ Uma opção deve existir apenas nesta seção para o usuario escolher qual das duas opções quer ver.