

# Introdução ao Shiny



Setembro de 2022

# O que é o Shiny?

Shiny é um framework em linguagem R para a criação de aplicativos web. Por não exigir conhecimento prévio de HTML, CSS e JavaScript, ele democratiza o acesso a essa área de desenvolvimento, permitindo a criação de aplicativos bonitos e complexos a partir de um script R.

Um aplicativo Shiny pode ser reduzido a vários elementos:

- uma página web: ele será acessado por um navegador, possuirá um endereço (URL) e será constituído por HTML, CSS e JavaScript.
- um aplicativo web: permitirá que quem estiver acessando interaja com as visualizações apresentadas.
- um código (ou uma coleção de códigos) em linguagem R: construídos, sobretudo, com o pacote `{shiny}`.

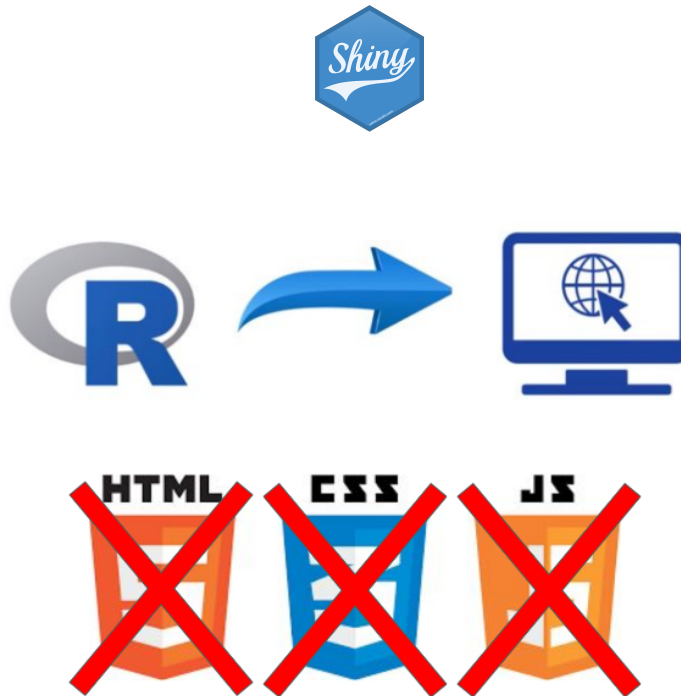
# HTML, CSS, JS... pra que serve?

Se você não conhece essas linguagens, uma boa maneira de entender o papel de cada uma delas na construção de um site é pensar em uma casa.

- Podemos pensar o HTML como a estrutura física da casa: chão, paredes, colunas, teto, encanamento, fiação etc.
- O CSS é o responsável pela aparência: pintura, pisos, azulejos, decoração em geral.
- O JavaScript traz funcionalidades a cada cômodo: pia, vaso sanitário, geladeira, cama, televisão e por aí vai.

# Shiny: programando em HTML sem saber HTML

Com o Shiny, podemos produzir aplicativos web em HTML, CSS e JavaScript sem saber programar nessas linguagens. E melhor: sem sair do R!



Fonte: [rstudio.com/shiny/](https://rstudio.com/shiny/)

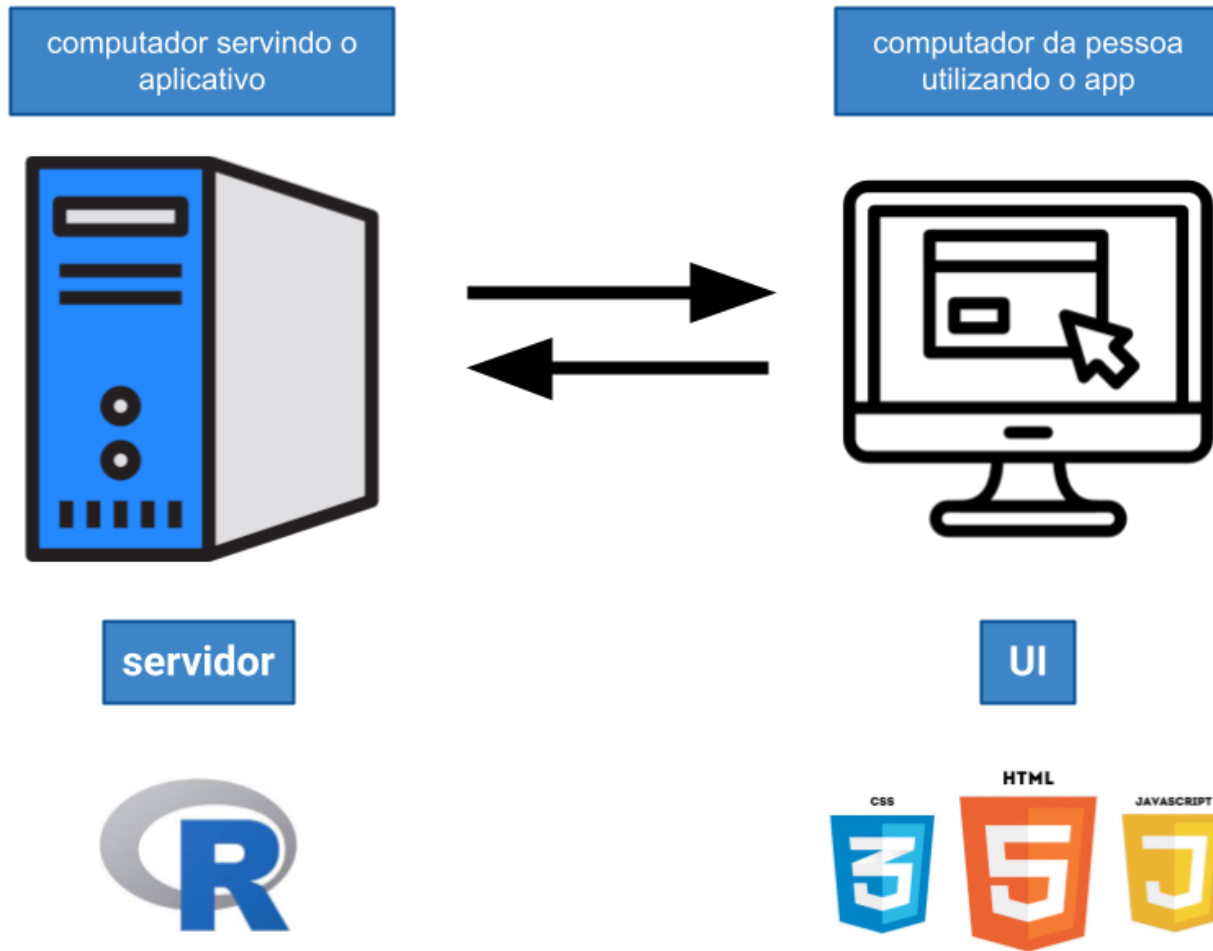
# Componentes básicos

Um aplicativo Shiny tem dois componentes básicos: a **interface de usuário** e o **servidor**.

O primeiro componente se refere à construção do código HTML que compõe o app. Podemos pensar na programação desse código HTML como a construção daquilo que será mostrado na tela, a cara do seu app, a interface de usuário ou **UI** (sigla para o termo *user interface*, em inglês).

O segundo componente se refere àquilo que não será visto por quem utilizar o app: o **servidor**. O lado do servidor (*server side* ou simplesmente *server*, em inglês) contém toda a lógica para a construção das saídas apresentadas na UI.

# Juntando tudo...



## Exemplo de Shiny app

# Estrutura de um código Shiny

O código de qualquer aplicativo em Shiny terá a estrutura abaixo:

- Um objeto chamado `ui`.
- Uma função chamada `server`.
- Uma chamada da função `shinyApp()`.

```
library(shiny)

ui <- fluidPage("Olá, mundo!")

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

}

shinyApp(ui, server)
```



# Rodando um aplicativo

Normalmente, enquanto estamos desenvolvendo um aplicativo Shiny, queremos testá-lo localmente para verificar se tudo funciona corretamente, se está ficando bonito ou simplesmente para gastar alguns minutos apreciando a nossa obra de arte. Testar localmente significa que **o seu próprio computador fará as vezes de servidor**, embora isso não signifique que seu app ficará disponível na internet.

Quando servimos um app localmente, isto é, quando *rodamos um app*, ganhamos um endereço que será acessível apenas do nosso computador. A partir desse endereço, podemos testar nosso app no navegador, como se ele já estivesse em produção.

No RStudio, podemos rodar nossos apps:

- rodando o script que contém o nosso app (atalho: CTRL + SHIFT + ENTER);
- clicando no botão **Run App**;
- rodando no console a função `runApp("caminho/ate/app.R")`.

# Atividade

Vamos criar e rodar o exemplo minimal do slide anterior.



Ao RStudio: 01-ola-mundo.R

Adicionando interatividade

# Inputs e Outputs

Uma das principais tarefas no desenvolvimento de um Shiny app é a definição e construção dos inputs e outputs. São esses elementos que nos permitem interagir com o app.

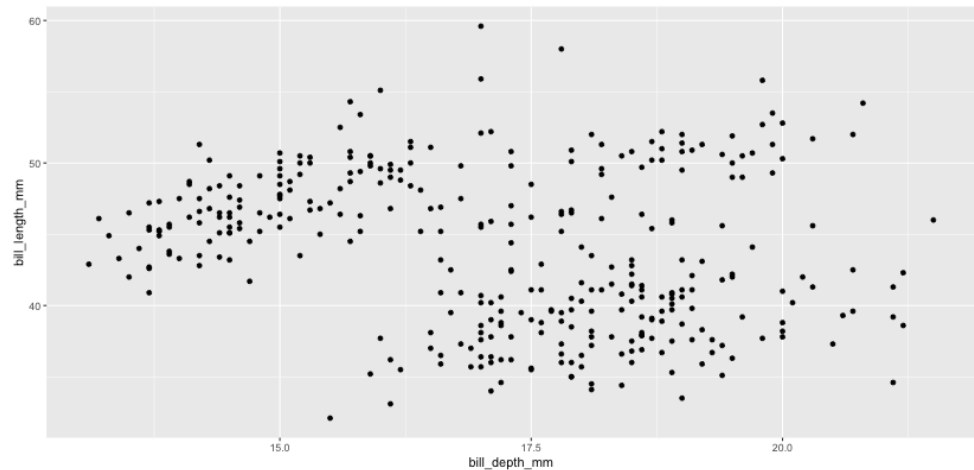
## Palmer penguins 🐧🐧🐧

**Variável eixo x**

bill\_depth\_mm ▼

**Variável eixo y**

bill\_length\_mm ▼



# Outputs: tabelas, gráficos e muito mais!

Outputs representam as *saídas* do nosso aplicativo, isto é, tudo que queremos que nosso código R retorne para o usuário. Essas saídas podem ser tabelas, gráficos, mapas, texto, imagens ou qualquer outro elemento em HTML.

Os outputs são definidos na UI e criados no server. Cada tipo de output é definido por uma função do tipo `*Output()`. Veja as principais funções dessa família:

Função	Saída
<code>plotOutput()</code>	Gráficos
<code>tableOutput()</code>	Tabelas
<code>textOutput()</code>	Textos

# Funções render

Para criar um output, precisamos das funções do tipo `render*()`. Essas funções são responsáveis por conectar as nossas visualizações criadas pelo R com o código HTML do UI. Na grande maioria dos casos, teremos o par `visualizacaoOutput()` `renderVisualizacao()`.

Veja a seguir as principais funções `render*()` e como elas se comunicam com as funções `*Output()`.

<b>*Output()</b>	<b>render*()</b>
<code>plotOutput()</code>	<code>renderPlot()</code>
<code>tableOutput()</code>	<code>renderTable()</code>
<code>textOutput()</code>	<code>renderText()</code>

# Acessando outputs no server

O argumento `outputId` das funções `_Output()` é utilizado para nos referirmos aos outputs dentro do server. Todos os outputs criados ficarão dentro da lista `output`.

```
library(shiny)

ui <- fluidPage(
  "Histograma da variável mpg",
  plotOutput(outputId = "histograma")
)

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

  output$histograma <- renderPlot({
    hist(mtcars$mpg)
  })

}

shinyApp(ui, server)
```

# Atividade

Vamos criar e rodar um shiny app com um gráfico como output.



Ao RStudio: 02-output.R



# Inputs: dê controle ao usuário

Inputs representam as entradas do nosso aplicativo, isto é, a maneira como informações são transmitidas entre a pessoa usando o app e o servidor. Essas informações podem ser valores, textos, datas, arquivos ou até mesmo cliques em um botão.

Para facilitar a escolha desses valores, o pacote `shiny` possibilita diversas opções de *widgets*, a depender do tipo de valor a ser passado.

Você pode conferir a lista de widgets do pacote `shiny` [nesta página](#). Repare que no campo `Current Value(s)` é mostrado qual valor será levado para dentro da função `server` em cada caso.

# Criando inputs

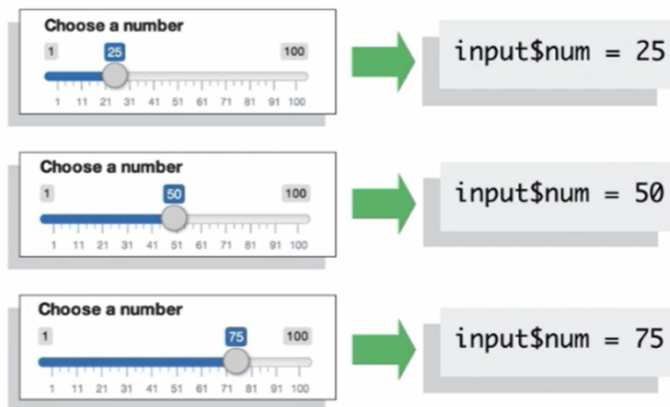
Para criar esses widgets utilizamos as famílias de funções `*Input()` ou `*Button`.

De forma análoga ao `outputId` das funções `*Output()`, todas essas funções possuem `inputId` como primeiro argumento, que recebe uma string e será utilizado para acessar cada input dentro da função `server`.

Isso implica que **dois inputs não podem ter o mesmo `inputId`**. Apenas o primeiro input funcionará caso você crie dois ou mais `inputId` repetidos.

# Acessando os inputs no server

Para acessar os inputs dentro da função server, utilizamos a lista `input`. Essa lista guardará todos os inputs criados no UI.



```
sliderInput(inputId = "num",...)
```

↓  
`input$num`

- `input$num` pode ser usado no server para deixar as visualizações dinâmicas.

Fonte: [rstudio.com/shiny/](https://rstudio.com/shiny/)

# Atividade

Vamos colocar um seletor de variáveis no exemplo anterior para permitir que o usuário escolha qual variável será exibida no histograma.



Ao RStudio: 03-output-input.R

# Shinyapps.io

O [shinyapps.io](https://shinyapps.io) é um serviço do RStudio para hospedagem de Shiny apps.

A conta gratuita permite você ter até 5 aplicações e 25 horas mensais de uso (um aplicativo utilizado por 1 hora consome 1 hora do seu plano, 2 aplicativos utilizados simultaneamente por 1 hora consomem 2 horas do seu plano).

Criada uma conta, você poderá subir o seu app para o shinyapps.io diretamente do RStudio. Para isso, você precisará apenas conectar a sua conta com o RStudio.

Neste vídeo, mostramos como conectar o shinyapps.io com o RStudio.

# Atividade

Vamos conectar o nosso RStudio com o shinyapps.io e subir um app para lá.



Ao RStudio: [shinyapps/04-dois-inputs-outputs.R](#)

# Referências e material extra

## Tutoriais

- [Slides completos](#)
- [Tutorial de Shiny do Garrett Grolmund](#)
- [Mastering Shiny](#)

## Galeria de Exemplos

- [Galeria do Shiny](#)
- [Site Show me Shiny](#)