Dashboards com R II

Trabalhando com htmlwidgets

Novembro de 2023

Motivação

`htmlwidgets` são bibliotecas de visualização JavaScript encapsuladas em pacotes de R. Elas nos permitem usar diversas ferramentas JavaScript diretamente do R, adicionando algumas poucas linhas de código em nosso script.

Além de diversos recursos interativos nativos, esssas bibliotecas permitem a captura de certos **eventos**, que são transformados em inputs dentro Shiny e podem ser utilizados para gerar ou modificar resultados.

Nesta aula, vamos ver como acessar esses eventos em alguns `htmlwidgets` e como podemos usurfluir de todas as possibilidades dessas bibliotecas mesmo quando não estão diretamente implementadas no pacote R.

Eventos

A implementação dos pacotes que dão acesso aos `htmlwidgets` a partir do R costumam dar suporte ao Shiny. Percebemos isso primeiro pela existência das funções `*Output()` e `render*()` que possibilitam a integração dessas bibliotecas com o Shiny de maneira simples. Mas também notamos essa integração pela captura de **eventos**.

Eventos são ações realizadas pela pessoa usando o app que podem ser capturadas pelo navegador e transformadas em informação útil para a aplicação. Exemplos de eventos são:

- clique do mouse
- posição do ponteiro
- hover sobre uma área

A depender da implementação do pacote R, essas informações são passadas de volta para o Shiny como um item da lista `input` e podem ser utilizadas para gerar novos comportamentos no app.

Eventos no plotly

Eventos no `plotly` podem ser acessados a partir da função `event_data()`. Essa função guarda valores reativos com informação de diversas ações realizadas no gráfico, entre elas

- cliques realizados no gráfico
- área do gráfico selecionada
- seleção de itens do gráfico

Para selecionar um tipo de evento, basta passar seu nome como argumento:

```
event_data("plotly_click")
event_data("plotly_selected")
```

Para saber mais veja o `help(event_data)` ou leia a documentação oficial.

Eventos no highcharts

Para acessarmos eventos em um highchart, precisamos adicionar no código que gera os gráficos as funções `hc_add_event_point()` ou `hc_add_event_series`(). Dessa forma, a informação é adiciona à lista `input` em valores com a seguinte nomenclatura `outputId_eventType`.

Assim, para um highchart com `outputId = "grafico"`, teremos

```
input$grafico_click
input$grafico_mouseOver
```

Para saber mais, acesso a documentação do pacote.

Eventos no echarts

O echarts disponibiliza automaticamente diversos eventos na lista `input` com a seguinte nomenclatura: `outputId_eventType`. Alguns deles são

- `input$outputId_clicked_data`: retorna os dados de um elemento clicado.
- `input$outputId_clicked_serie`: retorna a série de um elemento clicado.
- `input$outputId_mouseover_data`: retorna os dados de um elemento indicado pelo mouse (cursor do mouse em cima do elemento).

Veja a lista completa na documentação do pacote.

Eventos no leaflet

Mapas feitos com a biblioteca leaflet enviam automaticamente valores provenientes de eventos para a lista `input`. O seguinte padrão de nome é utilizado: `input$outputId_tipoObjeto_nomeEvento`.

Por exemplo, se um mapa com `outputId = "mapa"` tiver um círculo, ao clicar no círculo o `input$mapa_shape_click` será atualizado. Antes do primeiro clique, o valor de `input$mapa_shape_click` é `NULL`.

O leaflet possui os seguintes tipos de objeto (`tipoObjeto`): `marker`, `shape`, `geojson` e `topojson`. E os seguintes tipos de eventos (`nomeEvento`): `click`, `mouseover` and `mouseout`.

Ao realizar uma ação, o valor do `input` correspondente passa a ser uma lista com os seguintes valores:

- `lat`: a latitude do objeto
- `lng`: a longitude do objeto
- `id`: o `layerId`, se disponível

Eventos no leaflet

Além dos eventos associados a um objeto, o leaflet também disponibiliza os seguintes valores

- `input$outputId_click`: que é atualizado sempre que o mapa base é clicado (o clique não é em um elemento, e sim diretamente no mapa). O valor é uma lista com a latitude e a longitude.
- `input$outputId_bounds`: que contém a latitude/longitude dos limites do mapa atualmente visível na tela.
- `input$outputId_zoom`: que contém um inteiro indicando o nível do zoom aplicado ao mapa.
- `input$outputId_center`: que contém a latitude/longitude do centro do mapa atualmente visível na tela.

Eventos no reactable

Em uma `reactable`, você pode ativar a seleção de linhas e receber a informação de quais linhas estão selecionadas dentro do Shiny.

Para isso, utilize o parâmetro `selection` na função `reactable()` que permite a opção `single` ou `multiple`, para permitir a seleção de apenas uma linha ou de várias linhas, respectivamente.

Para acessar quais linhas estão selecionadas, basta utilizar a função `getReactableState("outputId", name = "selected")`. Você receberá um vetor com o índice das linhas selecionadas.

A partir dessa função, você também pode recuperar o número da página, o número de linhas da página e o número de páginas da tabela. Basta trocar o parâmetro `name` respectivamente por `"page"`, `"pageSize"` ou `"pages"`.

Saiba mais na documentação do pacote reactable.

Eventos no DT

O pacote DT disponibiliza automaticamente diversos eventos na lista `input`. Alguns deles são

- `input$outputId_rows_selected`: linhas selecionadas
- `input$outputId_columns_selected`: colunas selecionadas
- `input$outputId_cell_clicked`: valor, linha e coluna da célula clicada
- `input$tableId_search`: a string atualmente no campo de busca global da tabela

Saiba mais acessando a documentação do pacote.

Indo além do pacote em R

Embora os pacotes que vimos até agora sejam excelentes alternativas para acessarmos os `htmlwidgets` dentro do R, eles não trazem a implementação de todas as possibilidades das bibliotecas JS em forma de parâmetros em suas funções.

Muitas funcionalidades precisam ser passadas através do parâmetro `...`, que permite passarmos argumentos não definidos na função. Esses argumentos serão passados para a biblioteca JS e, se a sintaxe estiver correta, gerarão o efeito desejado.

Mas se esses argumentos não são definidos nas funções, como saber quais funcionalidades cada elemento permite? Para saber isso, precisamos estudar as documentações das próprias bibliotecas JS.

Documentação dos htmlwidgets

A documentação de bibliotecas JS geralmente compreende uma lista de parâmetros disponíveis, uma descrição do que o parâmetro faz, os possíveis valores que ele pode receber e, se tivermos sorte, alguns exemplos de como utilizá-lo.

A seguir listamos a documentação das bibliotecas que vimos até agora:

- plotly
- highcharts
- echarts
- leaflet
- reactable
- DT

Funções JS

Alguns parâmetros dos htmlwidgets permitem como argumento funções JS que desempenham alguma tarefa dentro do widget, geralmente de formação de texto e valores.

Algumas personalizações só são possíveis a partir dessas funções JS, exigindo que saibamos pelo menos um pouco de JS.

Essas funções podem ser escritas como um string e passadas nesses argumentos embrulhadas pela função `htmlwidgets::JS()`.

```
htmlwidgets::JS(
  "function(x) {
    var res 'O resultado é:' + x;
    return(res)
  }"
)
```

Atividade

Vamos ao RStudio fazer alguns exemplos de interação com htmlwidgets.

Referências e material extra

- Eventos no plotly
- Eventos no highcharts
- Eventos no echarts
- Eventos no leaflet
- Eventos em uma tabela reactable
- Eventos em uma tablea DT

Referências e material extra

Documentação das bibliotecas JS

- plotly
- highcharts
- echarts
- leaflet
- reactable
- DT