Interactive data visualization and reporting

Dr. Çetinkaya-Rundel

2018-04-18

Project tips

https://www.youtube.com/watch?v=vGUNqq3jVLg

Outline

- Review
- Stop-trigger-delay
- Shiny in Rmd
- Deploying your app
- Your turn!

Review

Is there something wrong with this? If so, what? Fix it in `review/mult_3.R`.

```
ui <- fluidPage(
  titlePanel("Multiply by 3"),
  sidebarLayout(
    sidebarPanel( sliderInput("x", "Select x", min = 1, max = 50, value = 30) ),
    mainPanel( textOutput("x_updated") )
  )
)

server <- function(input, output) {
  mult_3           <- function(x) { x * 3 }
  current_x        <- reactive({ mult_3(input$x) })
  output$x_updated <- renderText({ current_x })
}
```

See `review/whats_wrong.R`.
There are a bunch of small mistakes.
See how many you can catch and fix.

Is there something wrong with this? If so, what? Fix it in `review/mult_3.R`.

```
ui <- fluidPage(
  titlePanel("Multiply by 3"),
  sidebarLayout(
    sidebarPanel( sliderInput("x", "Select x", min = 1, max = 50, value = 30) ),
    mainPanel( textOutput("x_updated") )
  )
)

server <- function(input, output) {
  mult_3           <- function(x) { x * 3 }
  current_x        <- reactive({ mult_3(input$x) })
  output$x_updated <- renderText({ current_x })
}
```

Is there something wrong with this? If so, what? Fix it in `review/add_2.R`.

```
ui <- fluidPage(
  titlePanel("Add 2"),
  sidebarLayout(
    sidebarPanel( sliderInput("x", "Select x", min = 1, max = 50, value = 30) ),
    mainPanel( textOutput("x_updated") )
  )
)

server <- function(input, output) {
  add_2             <- function(x) { x + 2 }
  current_x         <- add_2(input$x)
  output$x_updated <- renderText({ current_x })
}
```

Stop - delay - trigger

Stop with `isolate()`

- Wrap an expression with `isolate()` to suppress its reactivity
- This will stop the currently executing reactive expression/observer/output from being notified when the isolated expression changes

Only update the alpha level
when other inputs of the plot change

`movies/movies-06.R`

Delay with `eventReactive()`

- Calculate a value only in response to a given event with `eventReactive()`
- Two main arguments (the event to react to and the value to calculate in response to this event):

```
eventReactive(eventExpr, valueExpr, …)
```

simple reactive value - `input$click`,
call to reactive expression - `df()`,
or complex expression inside `{}`

the expression that produces the return value when `eventExpr` is invalidated

Remove the functionality for selecting types,
instead randomly sample a
user defined number of movies,
but only sample and update outputs
when an action button is pushed

`movies/movies-07.R`

Update the previous app
so that a sample with a default sample size
is taken and plotted upon launch

`movies/movies-08.R`

Trigger with `observeEvent()`

- Trigger a reaction (as opposed to calculate/recalculate a value) with `observeEvent()`
- Also two main arguments:

```
observeEvent(eventExpr, handlerExpr, …)
```

simple reactive value - `input$click`,
call to reactive expression - `df()`,
or complex expression inside `{}`

expression to call whenever
`eventExpr` is invalidated

Add a button to write a csv
of the current random sample

`movies/movies-09.R`

Stop-delay-trigger

- `isolate()` is used to stop a reaction
- `eventReactive()` is used to create a calculated value that only updates in response to an event
- `observeEvent()` is used to perform an action in response to an event

Shiny in Rmd

You can embed a Shiny app
in an R Markdown file.

`movies/movies-10.Rmd`

Deploying your app

Deploying your app

- Start with shinyapps.io
- Easy to use, secure, and scalable
- Comes with built in metrics
- Free tier available!

Your turn!

Go to our final project on RStudio Cloud.

Create a folder called Shiny.

In that folder create an R script called **`app.R`**.

Create a simple data visualization app for using your project.

There is no Shiny app requirement for the project,
this is just for practice.