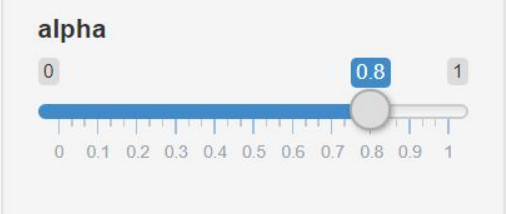
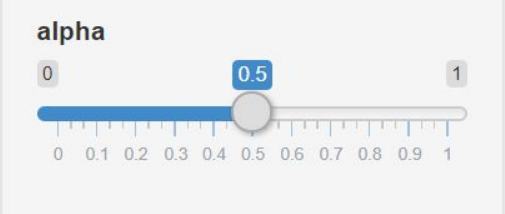
REACTIVITY IN SHINY

Omayma Said

```
sliderInput(inputId = "alpha_level",
            label = "alpha",
            min = 0, max = 1,
            value = 0.8)
```






```
input$alpha_level = 0.8
```

```
input$alpha_level = ?
```

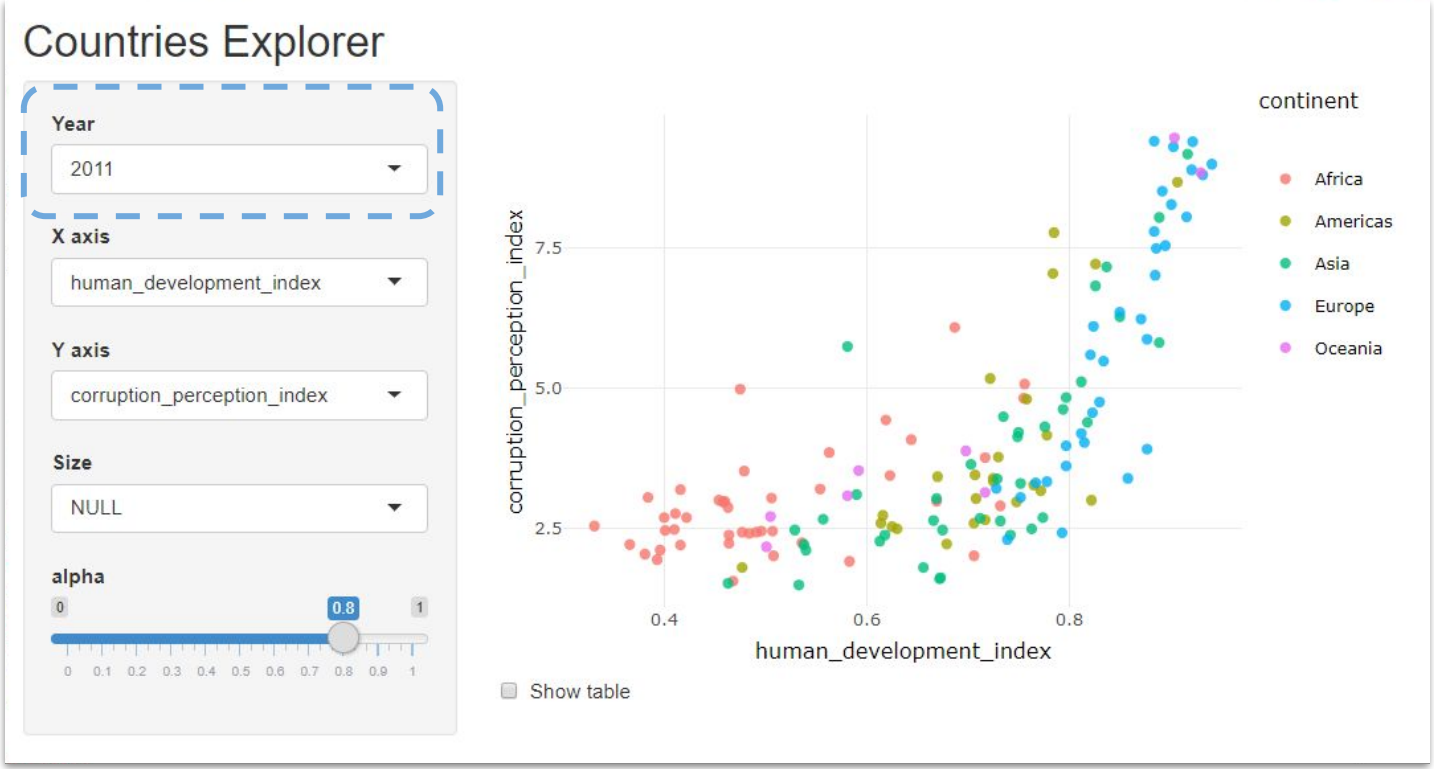
How many input values affect rendering this plot?

```
## create scatter plot ----------------------------------
output$countries_scatter <- renderPlot({
    ggplot(data = countries_data_2011,
           aes_string(x = input$x_axis, y = input$y_axis,
                      color = "continent",
                      size = input$point_size))+
      geom_point(alpha = input$alpha_level)+
      theme_minimal()
  })
```

How many input values affect rendering this plot?

```
## create scatter plot ----------------------------------
output$countries_scatter <- renderPlot({
    ggplot(data = countries_data_2011,
           aes_string(x = input$x_axis, y = input$y_axis,
                      color = "continent",
                      size = input$point_size))+
      geom_point(alpha = input$alpha_level)+
      theme_minimal()
  })
```

How would you plot a subset of data corresponding to a certain year?



1- Add a UI element for the user to select the year

```
## add select variable for year
selectInput(inputId = "year",
            label = "Year",
            choices = unique(countries_data$year),
            selected = 2011)
```

2- Filter the selected year and return a new dataframe as a reactive expression.

```
## filter data based on the selected year
countries_subset <- reactive({
   countries_data %>%
     filter(year == input$year)
  })
```

3- Use the new dataframe for plotting.

```
## create scatter plot
output$countries_scatter <- renderPlotly({
  p_scatter <- ggplot(data = countries_subset(),
                      aes_string(....))+
    geom_point(....)
    ....
    ggplotly(p_scatter)
})
```

SEVER

```
## show plot in main panel
plotlyOutput(outputId = "countries_scatter")
```

UI

EXERCISE

- Open **countries-09.R**
- Create a new reactive `countries_summary` as a new dataframe with median `gdp_per_capita` and median `life_exp` per continent.
- Modify DT::renderDataTable to use `countries_summary`.

EXERCISE

countries-10.R

corruption_perception_index

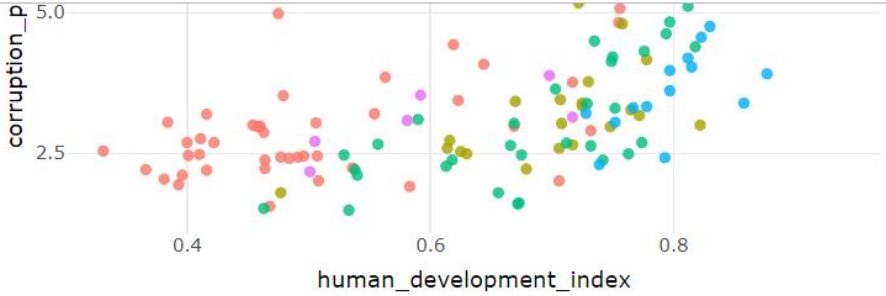
Size

NULL

alpha

0 0.8 1

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1



☑ Show table

Show 10 entries

Search:

continent	gdp_median	life_exp_median
Africa	1344.18	59.53
Americas	6986.05	73.95
Asia	5562.48	73.13
Europe	24985.25	80.44
Oceania	3942.05	70.99

Showing 1 to 5 of 5 entries

Previous 1 Next

ISOLATE

- Stop a reaction

EXERCISE

- Open **countries-10.R**
- Isolate `input$alpha_level` in the plot using `isolate()` and notice the effect.

eventReactive()

- Create a calculated value that only updates in response to an event *(e.g. button click)* .

eventReactive()

```
eventReactive(eventExpr, valueExpr, ...)
```

The expression that represents the event; this can be a simple reactive value like **input$click**, a call to a reactive expression like **dataset**().

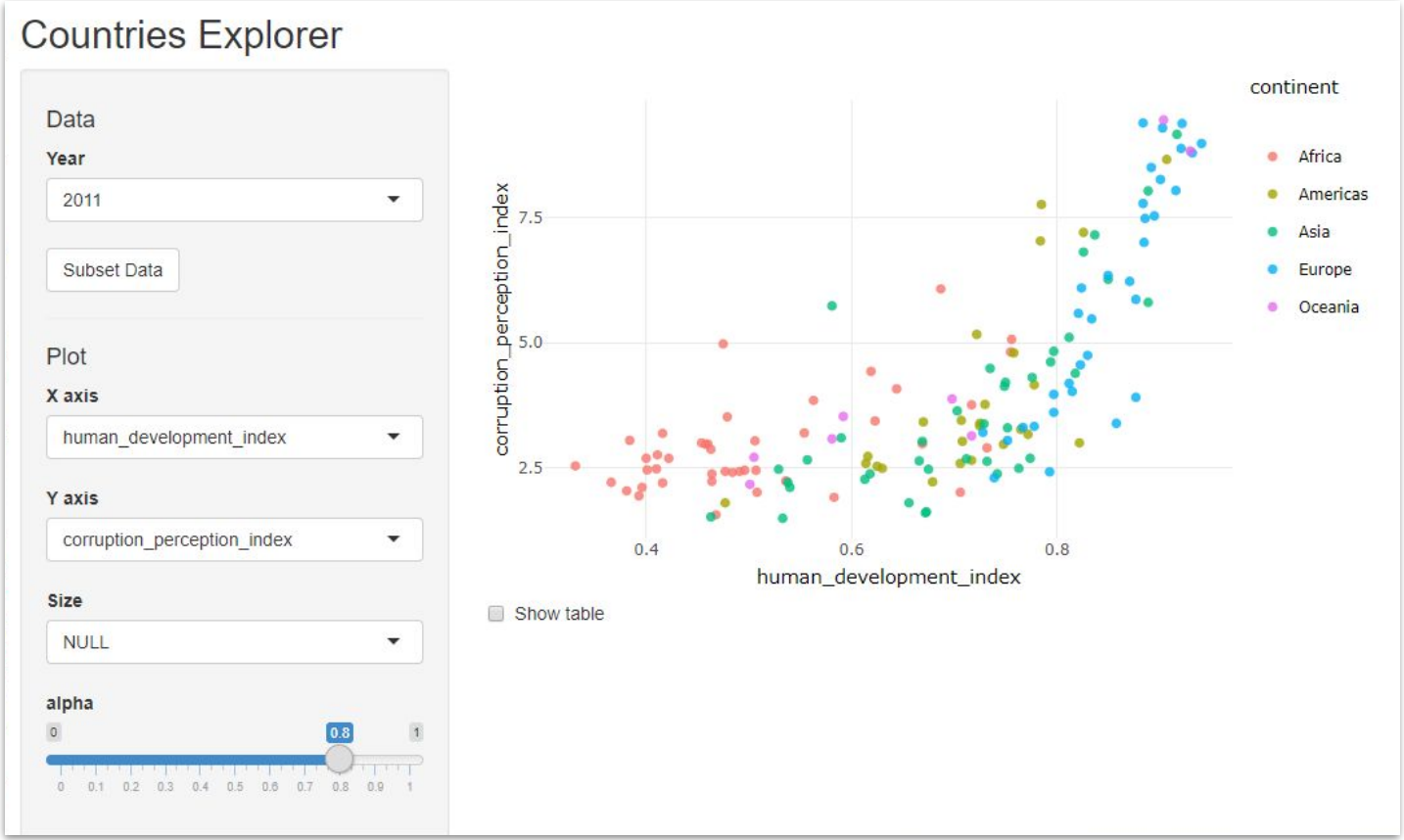
The expression that produces the return value of the eventReactive.

EXERCISE

- Open **countries-11.R**
- Add `actionButton` to trigger filtering the data.
- convert `countries_subset` to `eventReactive()` instead of `reactive()`
- make the `eventReactive()` triggered by the `actionButton` added in the UI.

EXERCISE

countries-12.R

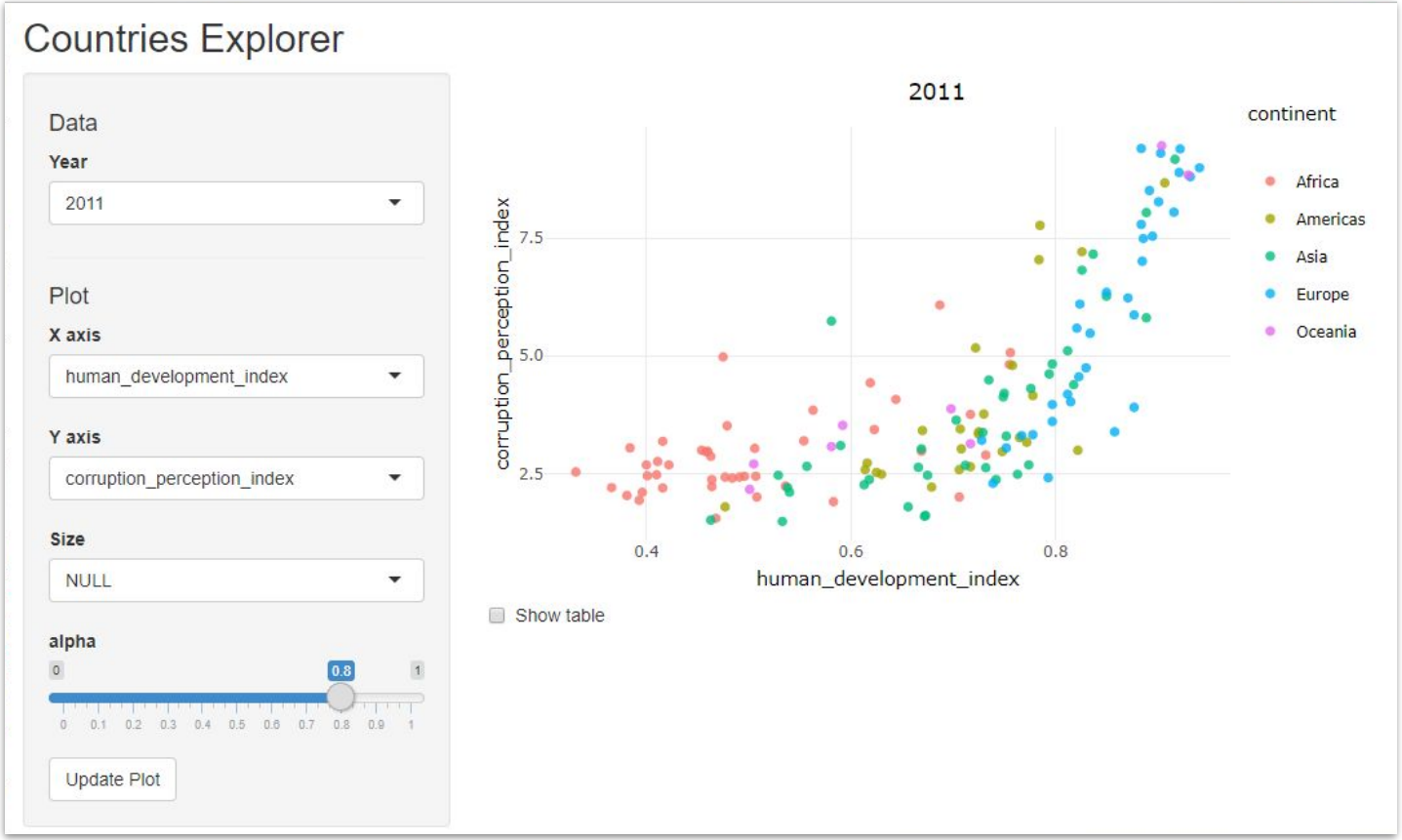


EXERCISE

- Open **countries-13.R**
- Add `actionButton` in the `SidePanel()`.
- Use the `actionButton` inside `renderPlotly()` to trigger plotting.

EXERCISE

countries-14.R



observeEvent()

- Perform an action in response to an event

observeEvent()

```
observeEvent(eventExpr, handlerExpr, ...)
```

The expression that represents the event; this can be a simple reactive value like **input$click**, a call to a reactive expression, etc.

A side-effect-producing action.

EXERCISE

- Open **countries-14.R**
- Add `actionButton` in the `SidePanel()`.
- Use the `actionButton` with `observeEvent()` to trigger saving `countries_subset()` to a .csv file.

EXERCISE

countries-15.R