Module 4: Creating Sidebar and Shiny Slider Input

contributed by

Debatosh Chakraborty
Project Research Assistant
R Team, FOSSEE, IIT Bombay

Objective

1. Adding sidebars to a page of a dashboard
2. Creating a one-way slider input
3. Converting the one-way slider to a two-way slider

Package Required

To get started, the first step is to load the required libraries “shiny”, “plotly” and “dplyr” using the following commands.

```
library(plotly)
library(shiny)
library(dplyr)
```

Dataset

airquality dataset available with base R

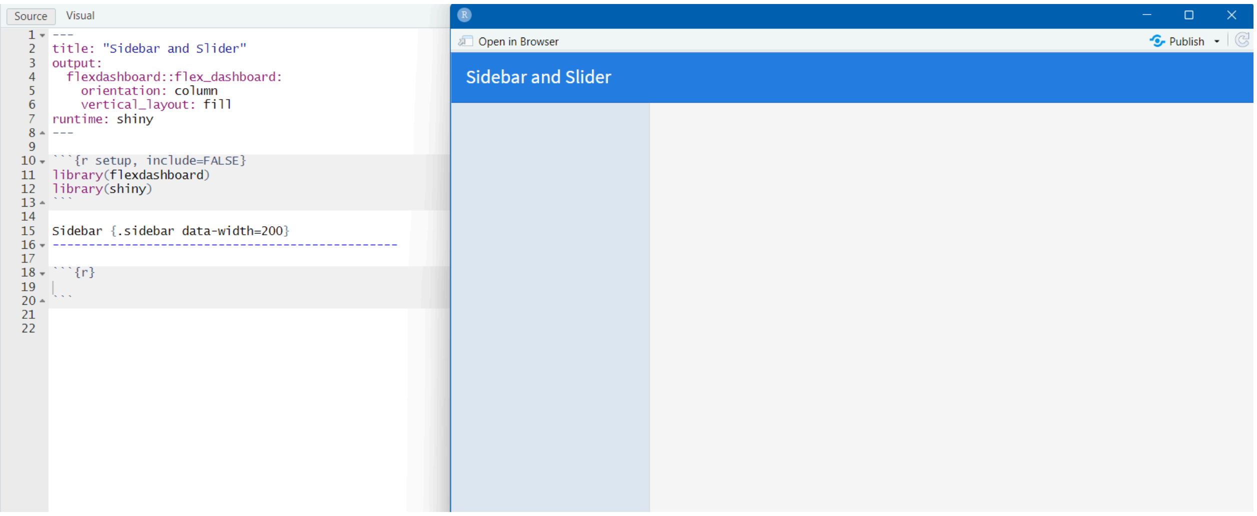
1. Adding sidebars to a page of a dashboard

- The sidebar in a dashboard contains input or filter controls.
- **Adding attribute ‘{.sidebar}’ with the name of a column creates a sidebar.**

Refer to Module 2 to create a dashboard of column orientation with one column named Sidebar**.**

A. **Step 1:** Type in **{.sidebar}** attribute beside the Sidebar name. (Figure 1)

B. **Step 2: Resize the sidebar:**
   a. Type ‘**{.sidebar data-width=200}**’ beside the Sidebar name.
   b. The ‘**data-width=200**’ attribute resizes the sidebar to a size of 200 px.
   c. **Unlike columns, <value> of the sidebar denotes the size in pixels.** (See Figure 2).



~~~
---
title: "Sidebar and Slider"
output:
  flexdashboard::flex_dashboard:
    orientation: column
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
```

Sidebar {.sidebar data-width=200}
------------------------------------------------

```{r}

```
~~~

Figure 1: Changing the width of the sidebar

2. Creating a one-way slider input

- **Sliders** are common dashboard input components used to **control values, ranges, or dates/times/years**.

  Note: Any input component can control any output component, **as long as their properties are compatible**.

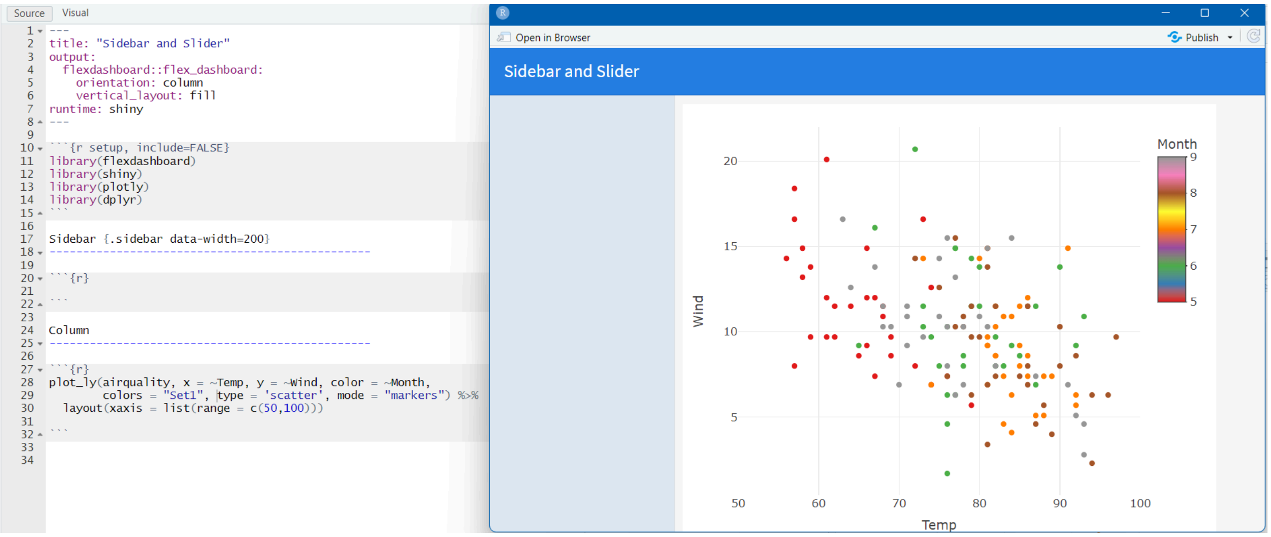
The steps to create a slider are as follows:

Step 1: Create a column that has a chart and a blank sidebar with width 200px.

Step 2: On the R code chunk under Chart, type the command to create a scatter plot of Temp vs Wind using Plotly.

~~~
```{r}
plot_ly(airquality, x = ~Temp, y = ~Wind, color = ~Month, colors = "Set1",
            type = 'scatter', mode = "markers") %>%
      layout(xaxis = list(range = c(0,100)))
```
~~~

Note: The same interactive plot can be created with GGPlotly.



~~~
---
title: "Sidebar and Slider"
output:
  flexdashboard::flex_dashboard:
    orientation: column
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
library(plotly)
library(dplyr)
```

Sidebar {.sidebar data-width=200}
-------------------------------------------------

```{r}

```

Column
-------------------------------------------------

```{r}
plot_ly(airquality, x = ~Temp, y = ~Wind, color = ~Month,
        colors = "Set1", type = 'scatter', mode = "markers") %>%
  layout(xaxis = list(range = c(50,100)))
```
~~~

Figure 2: Temp vs Wind plot of airquality dataset using Plotly.

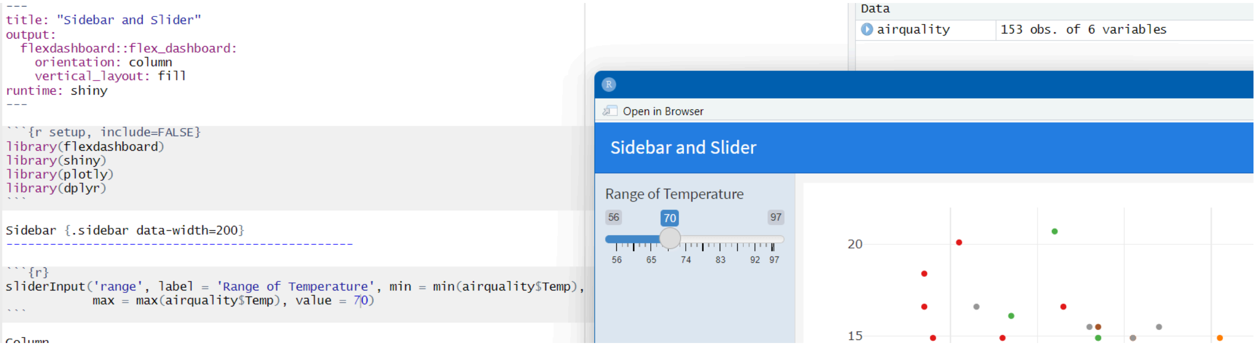
Step 3: Create a slider to control the range of x-axis:

a. Type the code in the coded chunk under the sidebar

~~~
```{r}
sliderInput('range', label = 'Range of Temperature', min =
min(airquality$Temp), max = max(airquality$Temp), value = 70)
```
~~~

b. Here, the function '**sliderInput**' will **create a slider** under the sidebar with the following properties: (Ref. Figure 3)

- **Input Id:** Unique **identifier/id** '**range**' of the slider.
- **Label**: Display the label of the slider on the dashboard.
- **min:** The **minimum value** of the slider.
- **max:** The **maximum value** of the slider.
- **value:** The **default value** of the slider when the dashboard is first opened.



~~~
---
title: "Sidebar and Slider"
output:
  flexdashboard::flex_dashboard:
    orientation: column
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
library(plotly)
library(dplyr)
```

Sidebar {.sidebar data-width=200}
-------------------------------------------------

```{r}
sliderInput('range', label = 'Range of Temperature', min = min(airquality$Temp),
            max = max(airquality$Temp), value = 70)
```

Column
~~~

```
Data
airquality        153 obs. of 6 variables
```

Figure 3: Creating a slider Input in the sidebar of the dashboard.

Note: At this stage, we can move the slider in the dashboard , but nothing changes on the graph because we have not included any functionality.

Step 4: Enclose the Plotly command written in Step 2 in **renderPlotly()** to ensure the plot will be used in a reactive context.

~~~
```{r}

renderPlotly(
plot_ly(airquality, x = ~Temp, y = ~Wind, color = ~Month, colors = "Set1",
            type = 'scatter', mode = "markers") %>%
      layout(xaxis = list(range = c(0,100)))
)
```
~~~

Step 5: Replace the upper range 100 in the code "`range = c(0,100)`" by **input$range.** '**range**' is the input id of the slider.

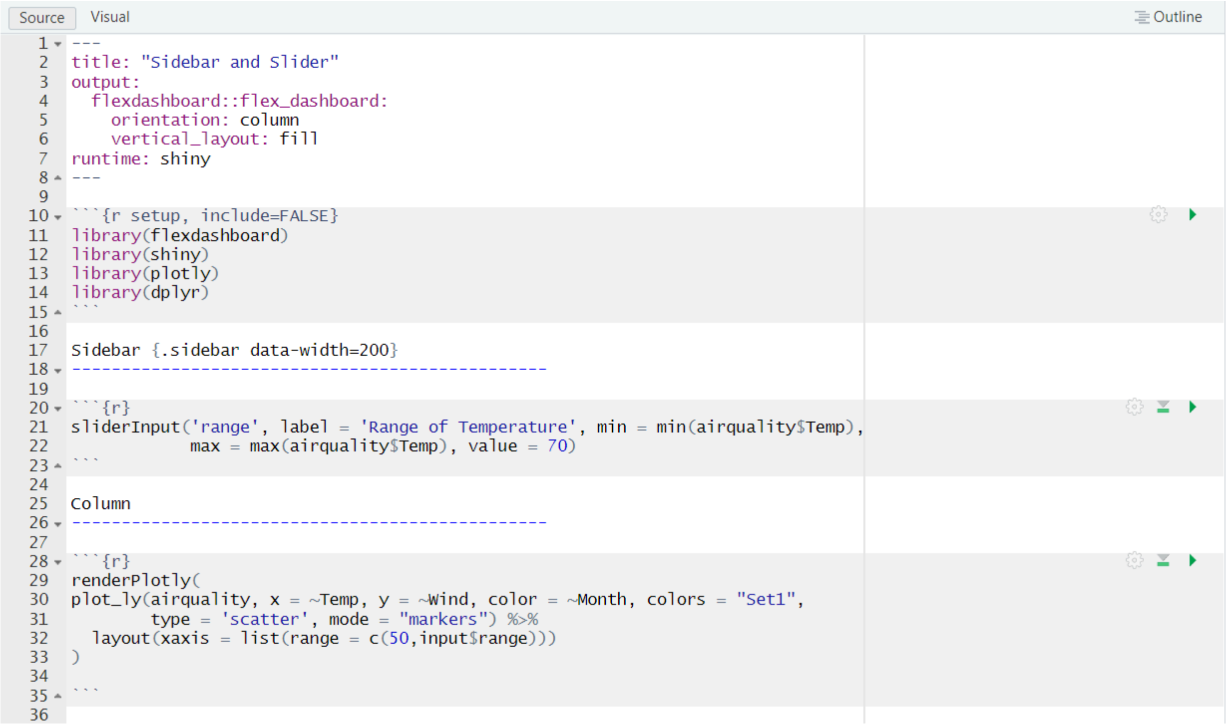
~~~
```{r}

renderPlotly(
plot_ly(airquality, x = ~Temp, y = ~Wind, color = ~Month, colors = "Set1",
            type = 'scatter', mode = "markers") %>%
      layout(xaxis = list(range = c(0,input$range)))
)

~~~
```

Note: Any dynamic control of any input to any output can be given by:

a. Substitute input for the output attribute that has to be controlled.
b. Specify 'input$' followed by input id with no spaces.


Source
Visual
Outline

```
---
title: "Sidebar and Slider"
output:
  flexdashboard::flex_dashboard:
    orientation: column
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
library(plotly)
library(dplyr)
```

Sidebar {.sidebar data-width=200}

```{r}
sliderInput('range', label = 'Range of Temperature', min = min(airquality$Temp),
            max = max(airquality$Temp), value = 70)
```

Column

```{r}
renderPlotly(
plot_ly(airquality, x = ~Temp, y = ~Wind, color = ~Month, colors = "Set1",
        type = 'scatter', mode = "markers") %>%
  layout(xaxis = list(range = c(50,input$range)))
)

```

```
```

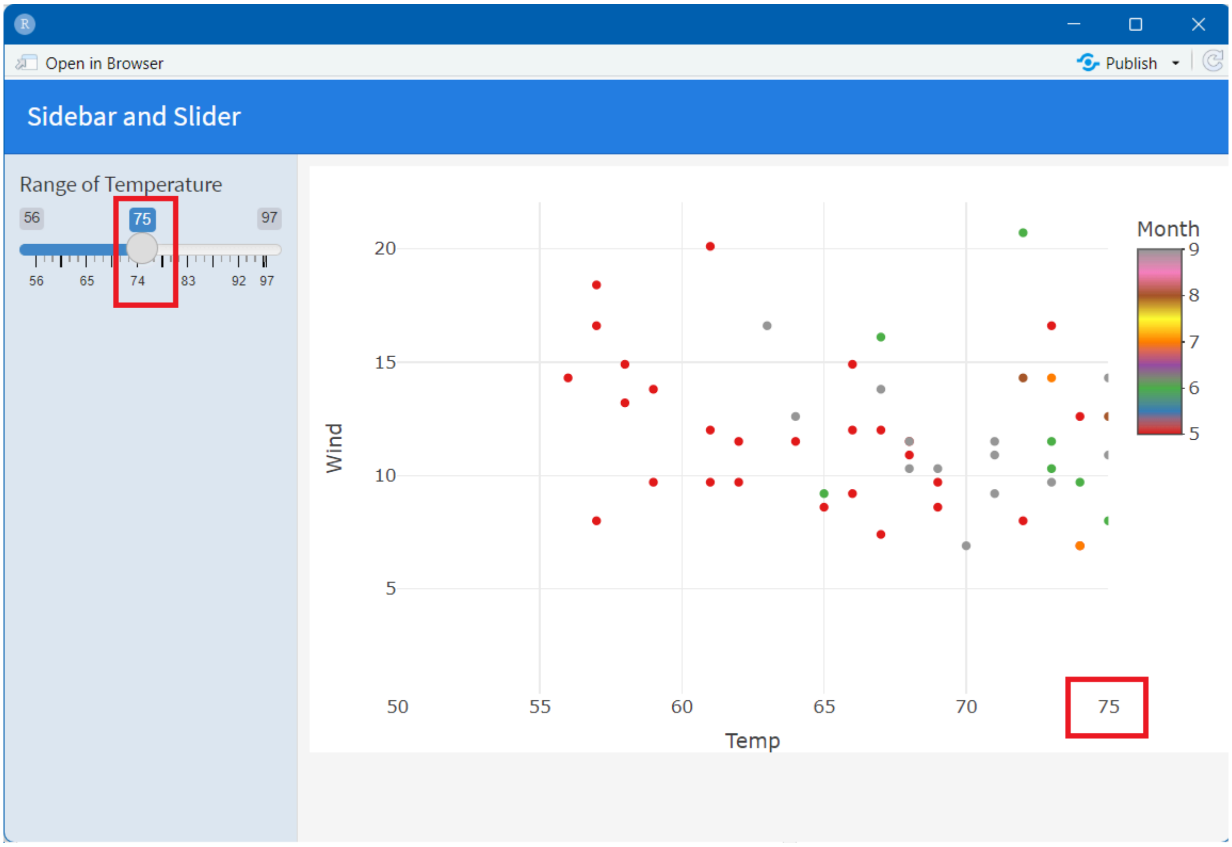


Figure 4: Controlling the range of the x-axis using the slider.

2. Converting the one-way to a two-way slider

- The two-way slider is used to **control both the upper and lower range simultaneously.**
- Change of code in sliderInput and renderPlotly function converts the slider. (Ref. Figure 5)

Step 1: Change the **value attribute** of the **sliderInput** function from a single value to a **vector of two values**, "c(20,80)" i.e. the lower and upper range. (Ref. Figure 5)

~~~
```{r}

sliderInput('range', label = 'Range of Temperature', min =
min(airquality$Temp), max = max(airquality$Temp), value = c(20,80))

```
~~~

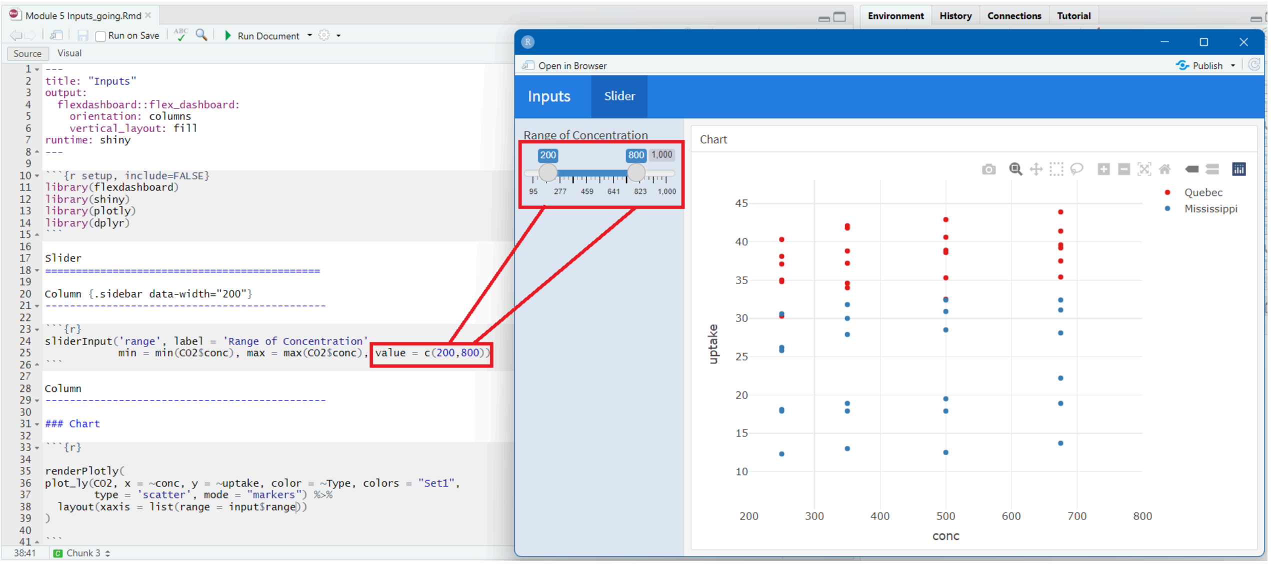
Note: SliderInput function for a two-way slider will always return a vector of two values.

Step 2: Replace the vector of range attribute **renderPlotly function** with **input$range** (Ref. Figure 5)

~~~
```{r}
~~~

~~~
renderPlotly(
plot_ly(CO2, x = ~conc, y = ~uptake, color = ~Type, colors = "Set1",
            type = 'scatter', mode = "markers") %>%
      layout(xaxis = list(range = input$range))
)

```
~~~



~~~
---
title: "Inputs"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
library(plotly)
library(dplyr)
~~~

Slider
===

Column {.sidebar data-width="200"}

```{r}
sliderInput('range', label = 'Range of Concentration',
            min = min(CO2$conc), max = max(CO2$conc), value = c(200,800))
```

Column

Chart

```{r}

renderPlotly(
plot_ly(CO2, x = ~conc, y = ~uptake, color = ~Type, colors = "Set1",
        type = 'scatter', mode = "markers") %>%
  layout(xaxis = list(range = input$range))
)

```
```

*Figure 5: Controlling both ends of the slider to input a range of values.*
```