ESCI upf. International Business

T3.1 | Interactive and dynamic data visualization

`htmlwidgets` and `shiny`

Marta Coronado Zamora
4 October 2024

Keep in touch

Theory lessons

Marta Coronado Zamora	Jose F. Sánchez
marta.coronado@prof.esci.upf.edu	jose.sanchez@prof.esci.upf.edu
@geneticament	@JFSanchezBioinf
Institut Botànic de Barcelona (CSIC-CMCNB)	Germans Trias i Pujol Research Institute (IGTP)

Practical lessons

Adrià Auladell
adria.auladell@ibe.upf-csic.es
Institut de Biologia Evolutiva (UPF-CSIC)

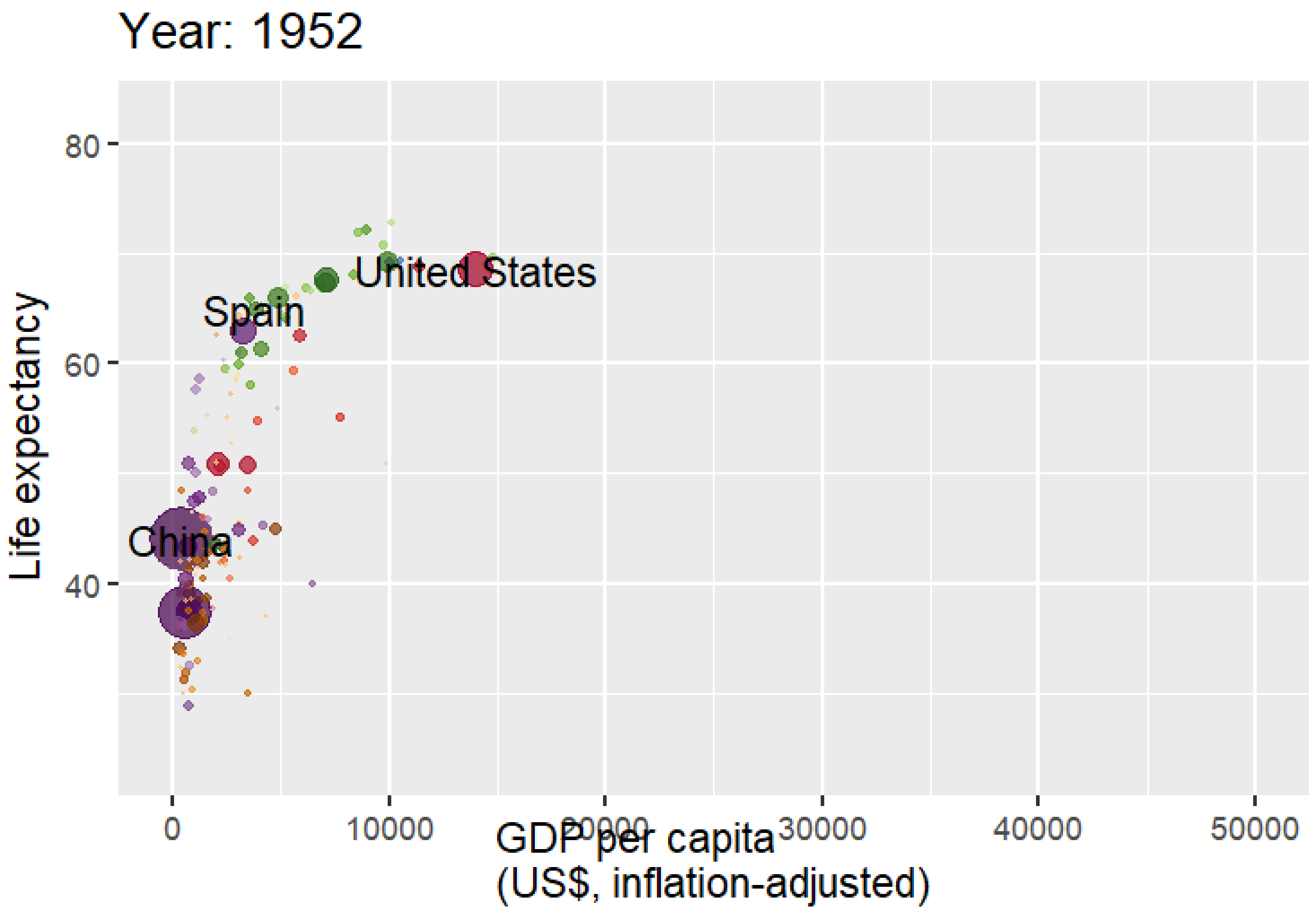
Content

- Interactive data visualization
- In R
  - `htmlwidgets`
  - `shiny`

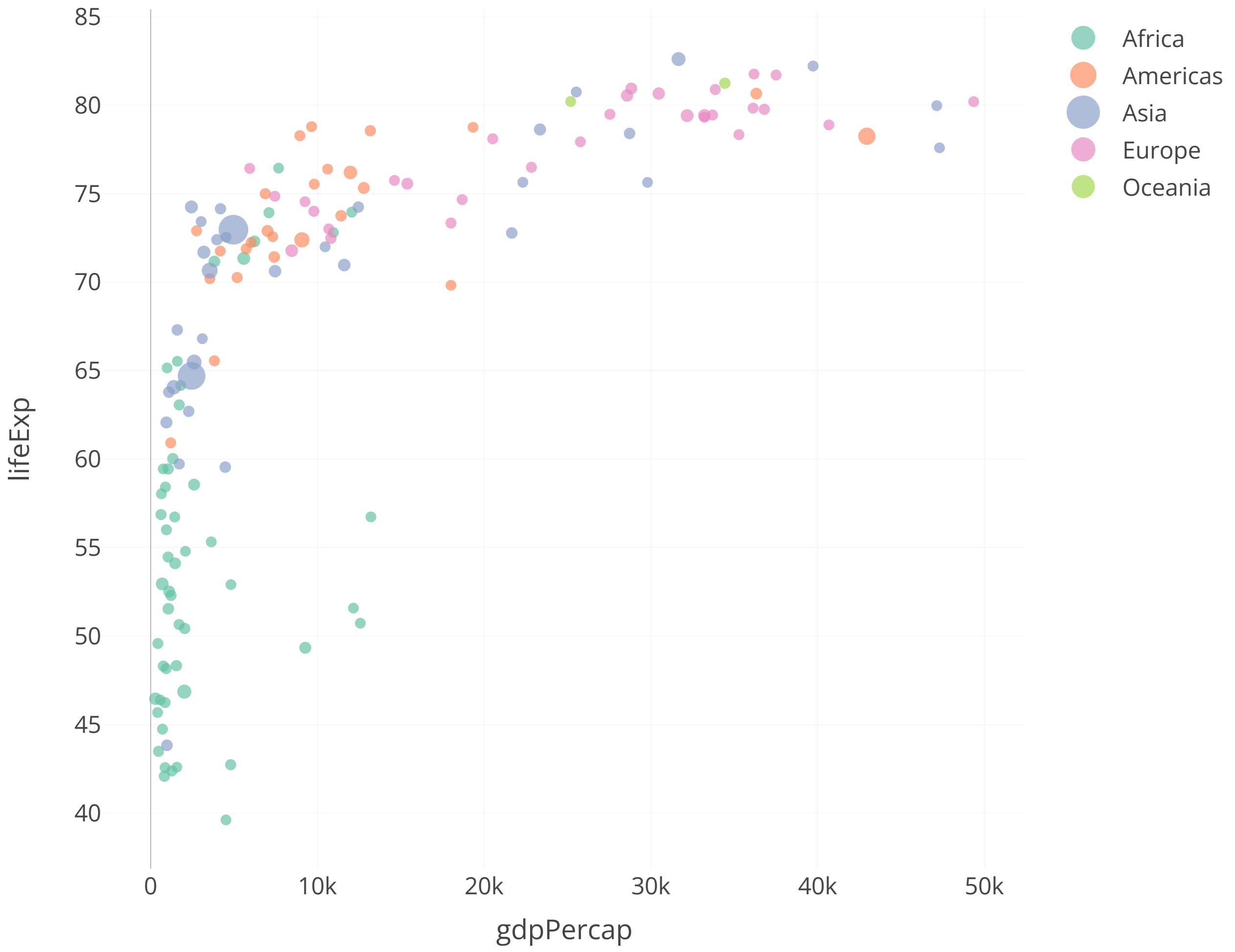
Get started!

Interactive and dynamic data visualization

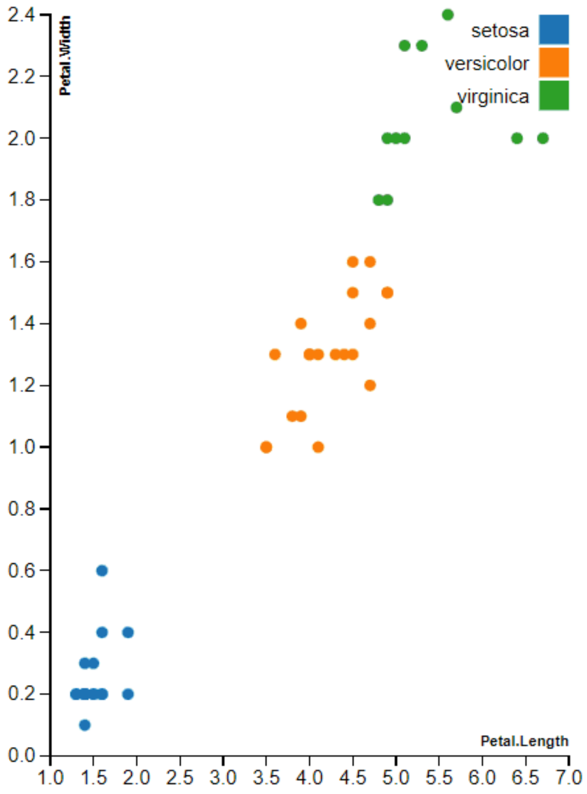
Example 1

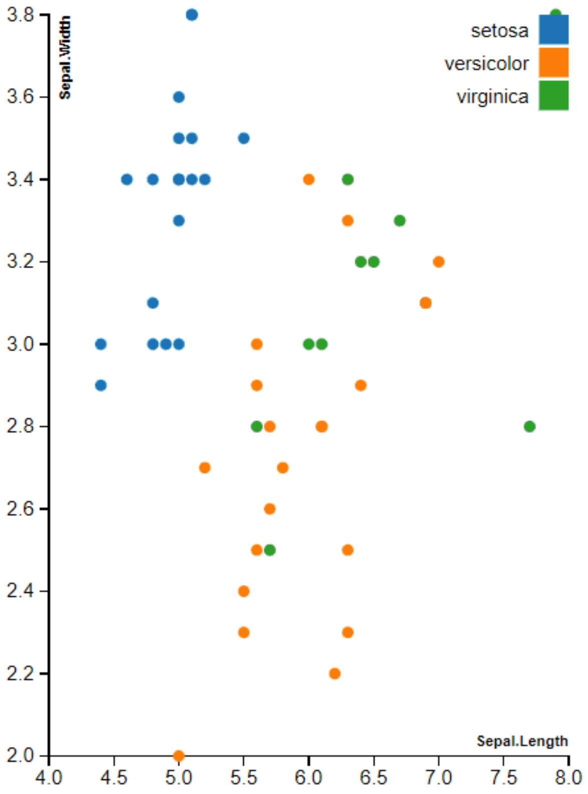

Year: 1952
Life expectancy
80
60
40
0
10000
20000
30000
40000
50000
GDP per capita
(US$, inflation-adjusted)
United States
Spain
China

Example 2


85
80
75
70
65
60
55
50
45
40
lifeExp
0
10k
20k
30k
40k
50k
gdpPercap
Africa
Americas
Asia
Europe
Oceania

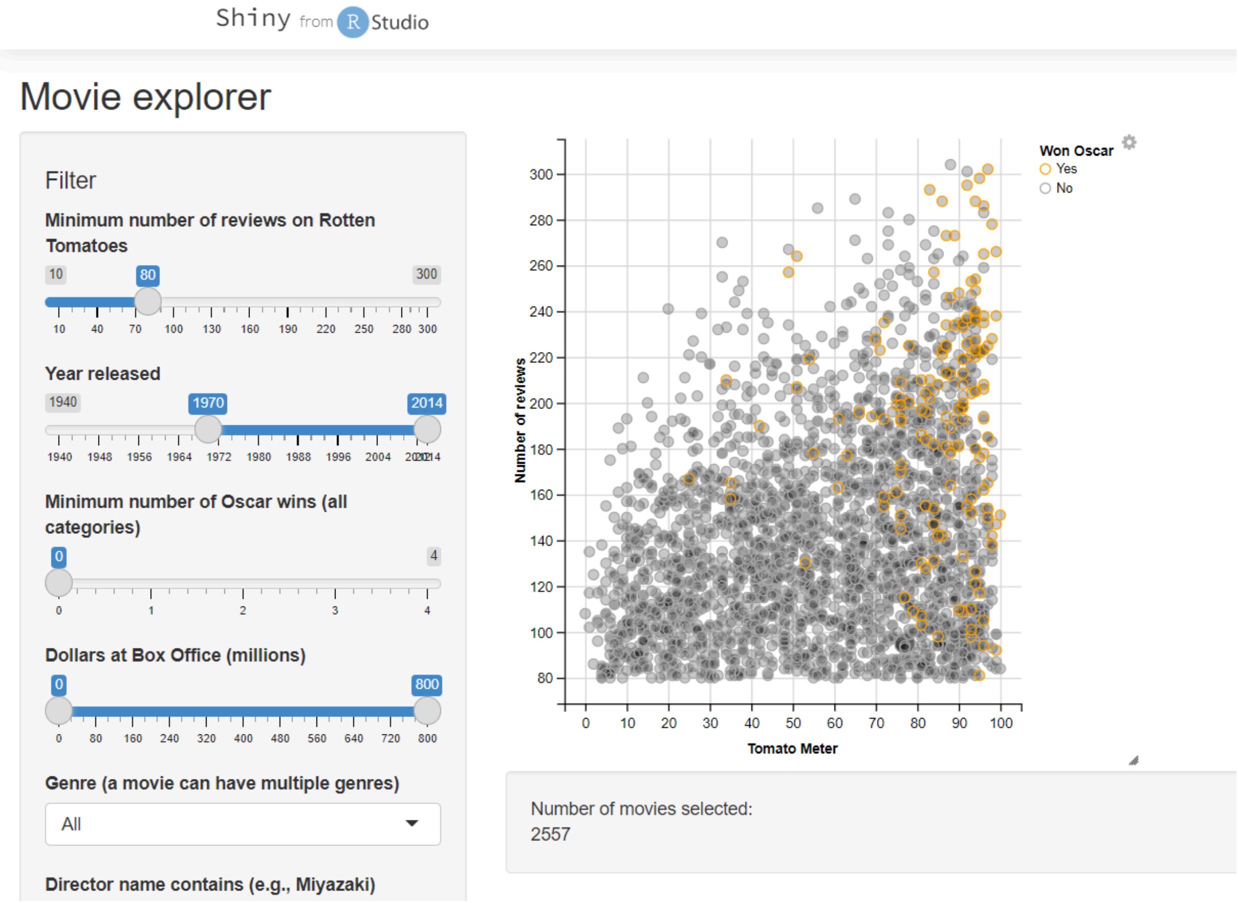
Example 3





Link

Example 4



Movie explorer

Filter

Minimum number of reviews on Rotten Tomatoes: 80 (range 10–300)

Year released: 1970 – 2014 (range 1940–2014)

Minimum number of Oscar wins (all categories): 0 (range 0–4)

Dollars at Box Office (millions): 0 – 800 (range 0–800)

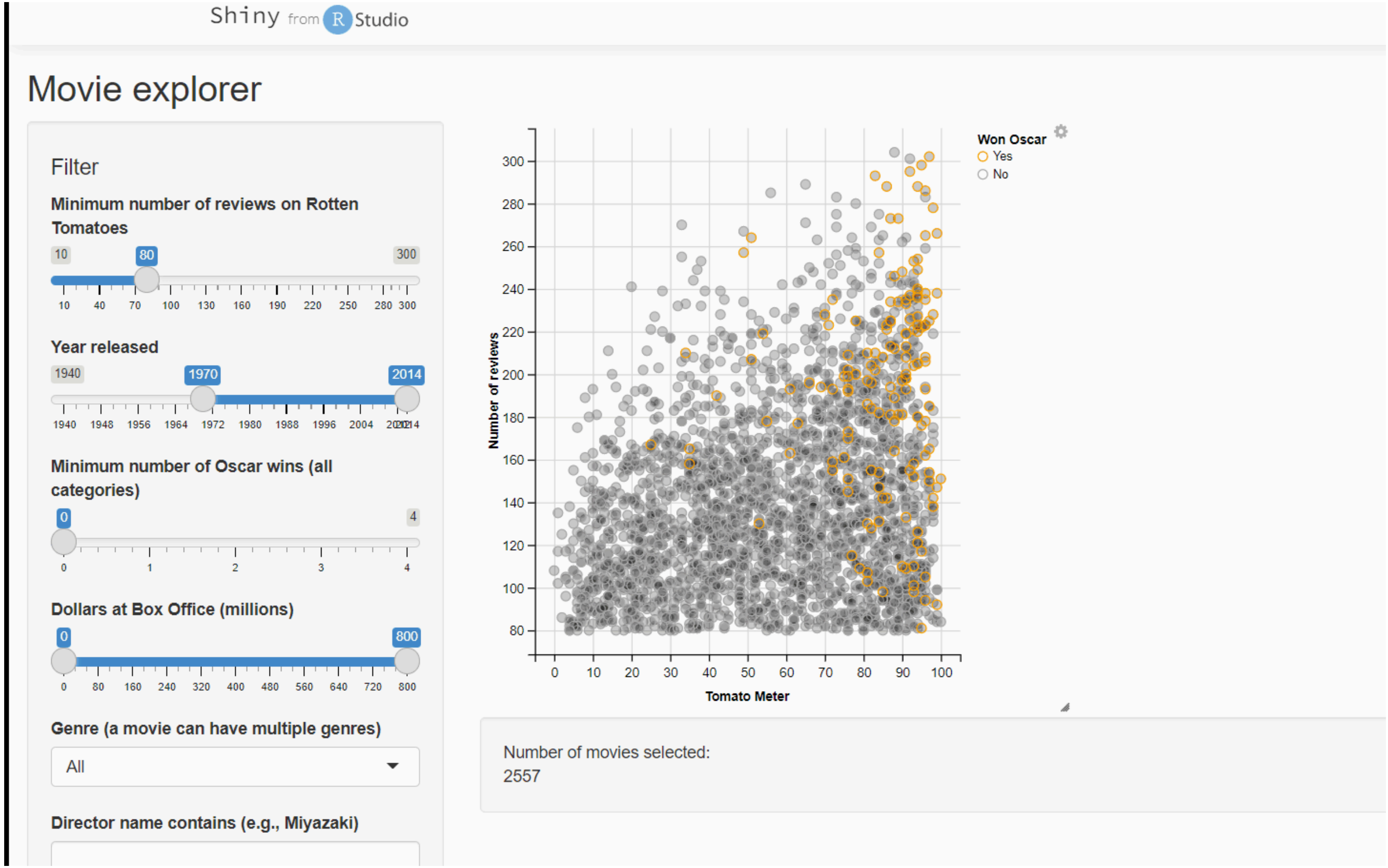
Genre (a movie can have multiple genres): All

Director name contains (e.g., Miyazaki)

Number of movies selected:
2557

Link

Example 4



Movie explorer

Filter

Minimum number of reviews on Rotten Tomatoes

Year released

Minimum number of Oscar wins (all categories)

Dollars at Box Office (millions)

Genre (a movie can have multiple genres)

All

Director name contains (e.g., Miyazaki)

Number of movies selected:
2557

✎ **Exercise | Describe the previous visualizations. Are they dynamic? And interactive? If so, what types of interaction do you identify?**

💬 Answer:

Types of interaction

- **Interactive plot area**
  - Action: hover, click, brush
  - Response: zoom, identify, link, add/remove
- **Interactive controls** (outside the plot)
  - Action: click, drag
  - Response: choose data set, variables, parameters and redraw

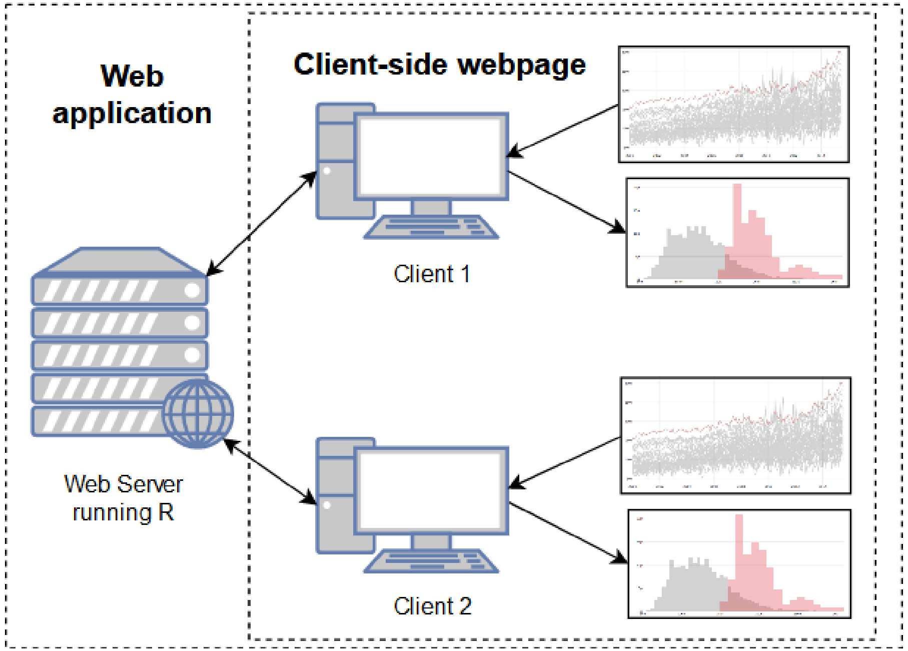
How interaction works

- **Basic framework**: web browsers
  `html + css + JavaScript`

- Options for interaction:

  - **Client-side**
    Static `html` page with `JavaScript` code
    Executed within the browser

  - **Server-side**
    Dynamic `html` page
    Executed in the host machine



In R

In recent years there has been a shift from using static R graphics to using interactive JavaScript web components for data analysis and data visualization: we need **R Bindings to JavaScript libraries**

- The `htmlwidgets` package

  Provides a framework for creating R bindings to `JavaScript` libraries. HTML Widgets can be:

  - Used at the R console for data analysis just like conventional R plots
  - Embedded within R Markdown documents
  - Incorporated into Shiny web applications
  - Saved as standalone web pages

  Executed in browser (client-side)
  Standalone `html` file

- `Shiny` applications

  Framework for interactive web applications in `R`
  Executed in host machine (server-side)
  Needs `R` running

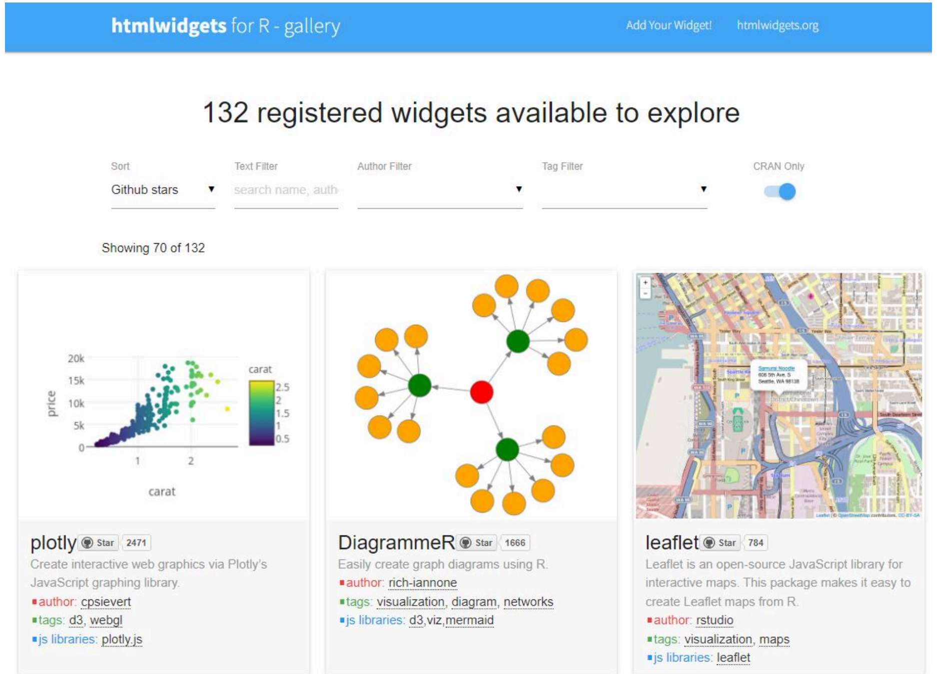
htmlwidgets

What are `htmlwidgets`?

Bridge between `R` and `JavaScript` libraries:

`R` package	`JavaScript` library
plotly	plotly.js (D3.js)
networkD3	D3.js
dygraphs	Dygraphs.js
leaflet	leaflet.js
…	…

Full list of `htmlwidgets`



132 registered widgets available to explore

Sort: Github stars ▾

Text Filter: search name, auth

Author Filter: ▾

Tag Filter: ▾

CRAN Only

Showing 70 of 132

plotly Star 2471

Create interactive web graphics via Plotly's JavaScript graphing library.

- author: cpsievert
- tags: d3, webgl
- js libraries: plotly.js

DiagrammeR Star 1666

Easily create graph diagrams using R.

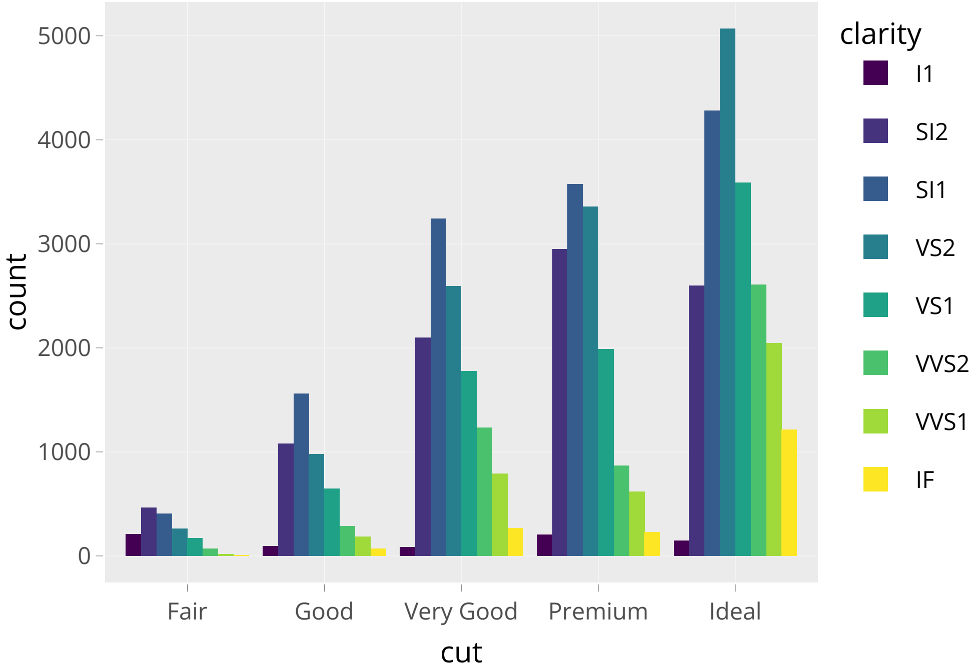
- author: rich-iannone
- tags: visualization, diagram, networks
- js libraries: d3,viz,mermaid

leaflet Star 784

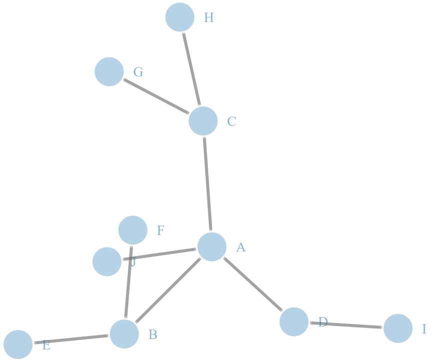
Leaflet is an open-source JavaScript library for interactive maps. This package makes it easy to create Leaflet maps from R.

- author: rstudio
- tags: visualization, maps
- js libraries: leaflet

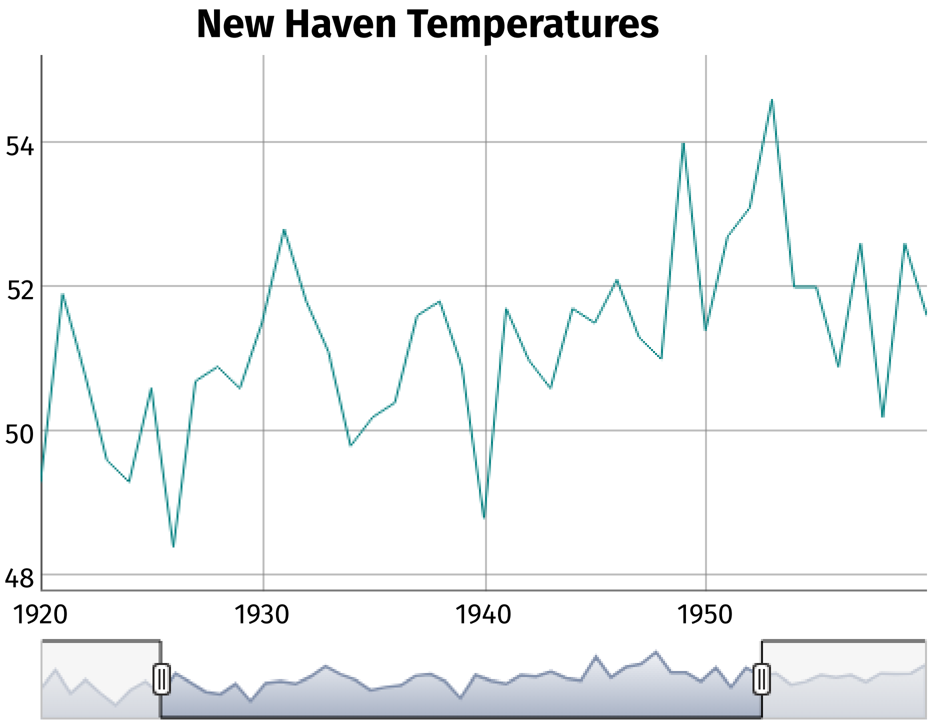
Example: `plotly`



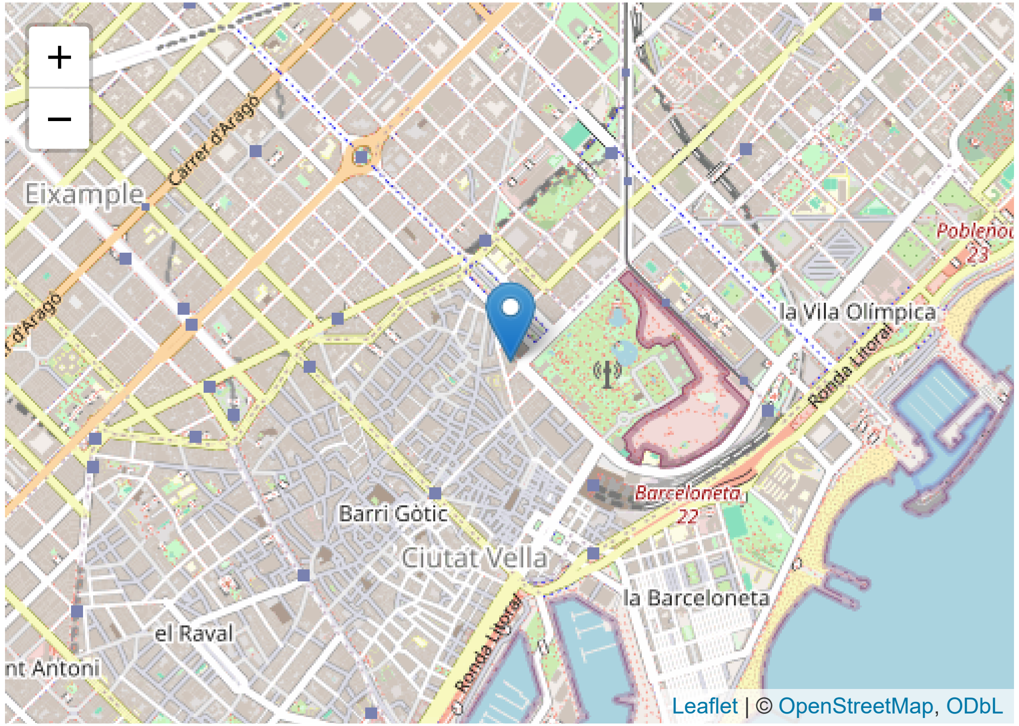
Example: `networkD3`



Example: `dygraphs`



Example: `leaflet`



Exercise | Tranform the following graphics made with `ggplot2` into `plotly` interactive versions. What kind of interaction does `plotly` add?

```
p1 ← ggplot(iris, aes(Sepal.Length, Petal.Length, shape = Species, colour = Species)) +
  geom_point()

p2 ← ggplot(iris, aes(Species, Petal.Length, fill = Species)) +
  geom_boxplot()
```

Answer:

Shiny

What is Shiny?

- Framework to create dynamic, reactive html pages without `html`, `css` or `JavaScript`
  - There isn't any static html output
  - Needs an R session running
- Creates web applications
  - Interactive visualizations
  - Much more

Gallery of `Shiny` apps

Syntax

```r
library(shiny)

# Web page
ui <- fluidPage()

# Running R session
server <- function(input, output){}

# Connection ui + server
shinyApp(ui = ui, server = server)
```

Two basic files: `ui.R` and `server.R` (or have them both in `app.R`)

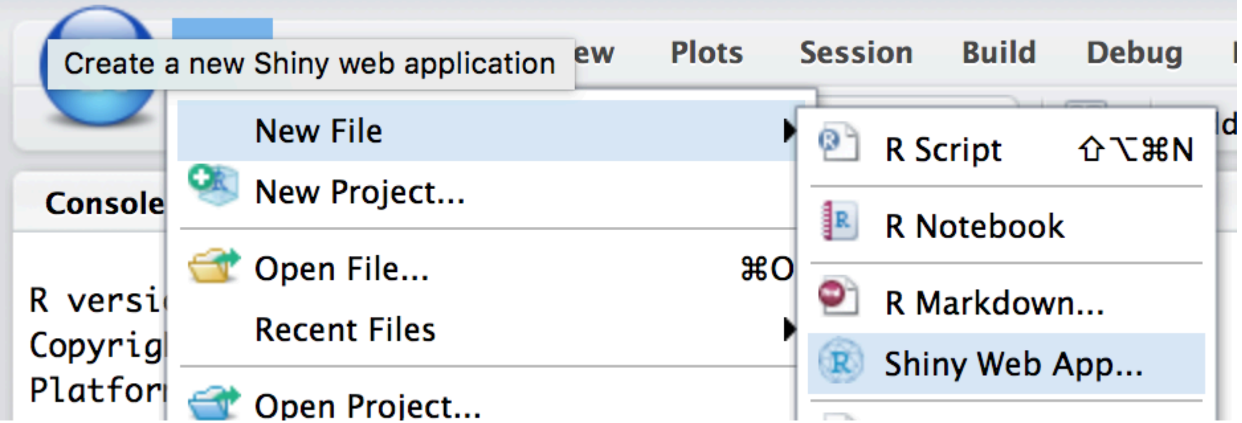
- `ui`
  - html specifications (R functions)
  - input functions
  - output functions
- `server`
  - instructions to build and rebuild R objects
  - refers to input and output in `ui`

Let's create a Shiny app!

First, install the `shiny` package:

```r
install.packages("shiny")
library(shiny)
```

Create a new `Shiny Web App` in RStudio:

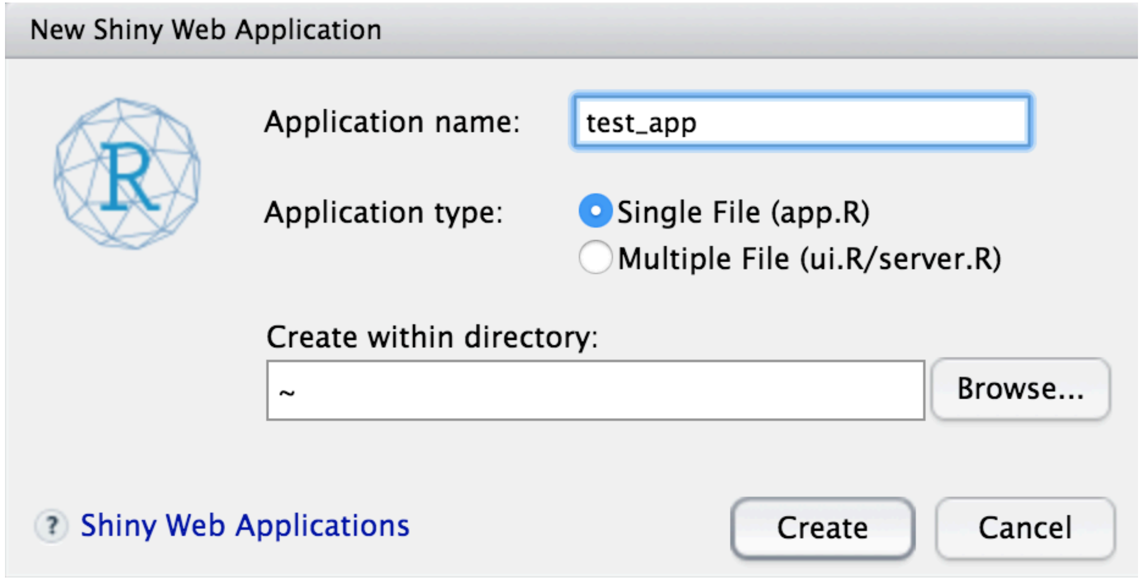


Let's create a Shiny app!

First, install the `shiny` package:

```r
install.packages("shiny")
library(shiny)
```

As an application name you can type `test_app` and create a Single File (`app.R`) and save it in a Directory:

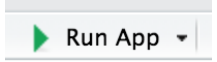


New Shiny Web Application

Application name: test_app

Application type: ● Single File (app.R) ○ Multiple File (ui.R/server.R)

Create within directory: ~

Run the app! 

Parts of the UI

UI is *de facto* an HTML file.

- In building `myui` (or `ui.R` file) what we really do is to construct an HTML file with R functions.
- By default, uses bootstrap, the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Parts of the UI:

1. HTML tags

2. Layout

3. Input (control widgets)

4. Output

UI: HTML tags

- You can build UI by using HTML tags
- Use `names(tags)` to see all available tags

Exercise | Modify your `app.R` adding the following code inside `fluidPage()` and run the app again.

```
ui <- fluidPage(

titlePanel(HTML('<h1>Old Faithful Geyser Data</h1>
        <hr><br>
        <p> I`m a paragraph showing how to write
        <strong>bold</strong> and <em>italics</em>.<br>
        This is a <code>block of code</code>.<br>
        I can also put <a href="http://www.google.com">a link to google</a>.<br>
        And I can also add images!<br>
        <center><img src="https://ih0.redbubble.net/image.543360195.2115/pp,550x550.jpg" width=10%></center
        <h2>The app</h2>')),

    # Sidebar with a slider input for number of
    # ...
```

Which HTML tags are you able to identify? What are they used for?

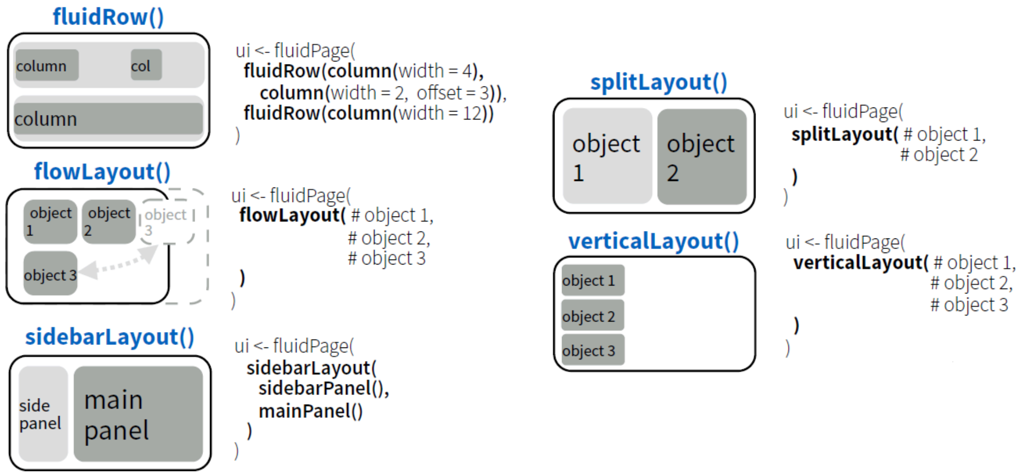
Answer:

UI: Layout

- **Panels**
  - Panel functions are used to put a group of elements together into a single ‘panel’.
  - There are several panel functions defined in shiny:

`absolutePanel()`, `conditionalPanel()`, `fixedPanel()`, `headerPanel()`, `inputPanel()`, `mainPanel()`, `navlistPanel()`, `sidebarPanel()`, `tabPanel()`, `tabsetPanel()`, `titlePanel()`, `wellPanel()`,

- **Layouts**
  - Layout functions are used to organize panels and elements into an existing layout.
  - There are several layout functions defined in shiny:



```
ui <- fluidPage(
 fluidRow(column(width = 4),
   column(width = 2,  offset = 3)),
 fluidRow(column(width = 12))
)
```

```
ui <- fluidPage(
 flowLayout( # object 1,
             # object 2,
             # object 3
 )
)
```

```
ui <- fluidPage(
  sidebarLayout(
   sidebarPanel(),
   mainPanel()
  )
)
```

```
ui <- fluidPage(
 splitLayout( # object 1,
              # object 2
 )
)
```

```
ui <- fluidPage(
 verticalLayout( # object 1,
                 # object 2,
                 # object 3
 )
)
```

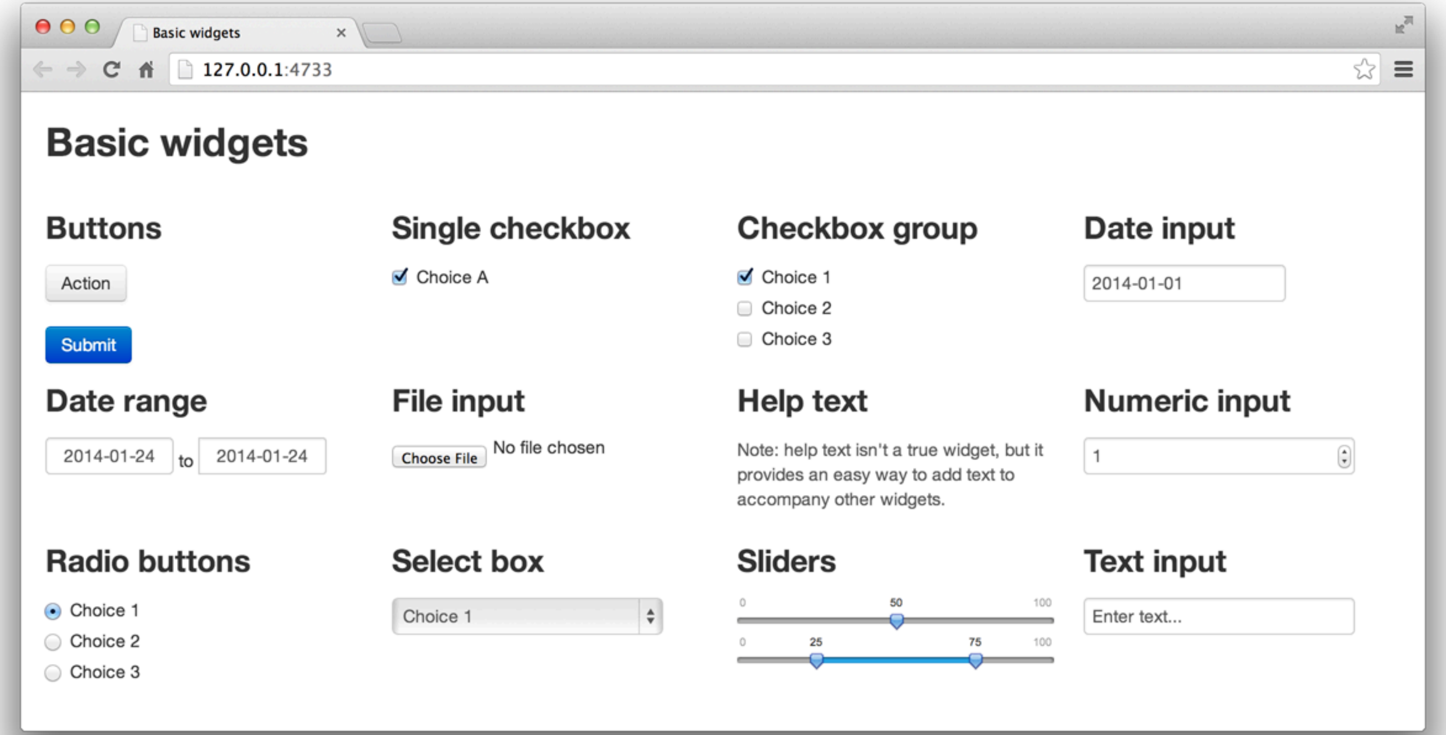
UI: Layout

Exercise | Which layouts and panels are used in the `app.R`?

Answer:

UI: Inputs (control widgets)

- Wigets are web elements that users can interact with.
- The standard Shiny widgets are:



Basic widgets

Buttons

Action

Submit

Single checkbox

☑ Choice A

Checkbox group

☑ Choice 1
□ Choice 2
□ Choice 3

Date input

2014-01-01

Date range

2014-01-24 to 2014-01-24

File input

Choose File No file chosen

Help text

Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Numeric input

1

Radio buttons

● Choice 1
○ Choice 2
○ Choice 3

Select box

Choice 1

Sliders

Text input

Enter text...

UI: Inputs (control widgets)

- Increase counter: `actionButton`, `actionLink`
- TRUE/FALSE: `checkboxInput`
- Date: `dateInput`, `dateRangeInput`
- File: `fileInput`
- Number: `numericInput`
- Text: `textInput`, `passwordInput`
- Select elements: `radioButtons`, `selectInput`, `sliderInput`
- Trigger reaction: `submitButton`

✎ **Exercise | Which widget(s) are used in the `app.R`?**

💬 Answer:

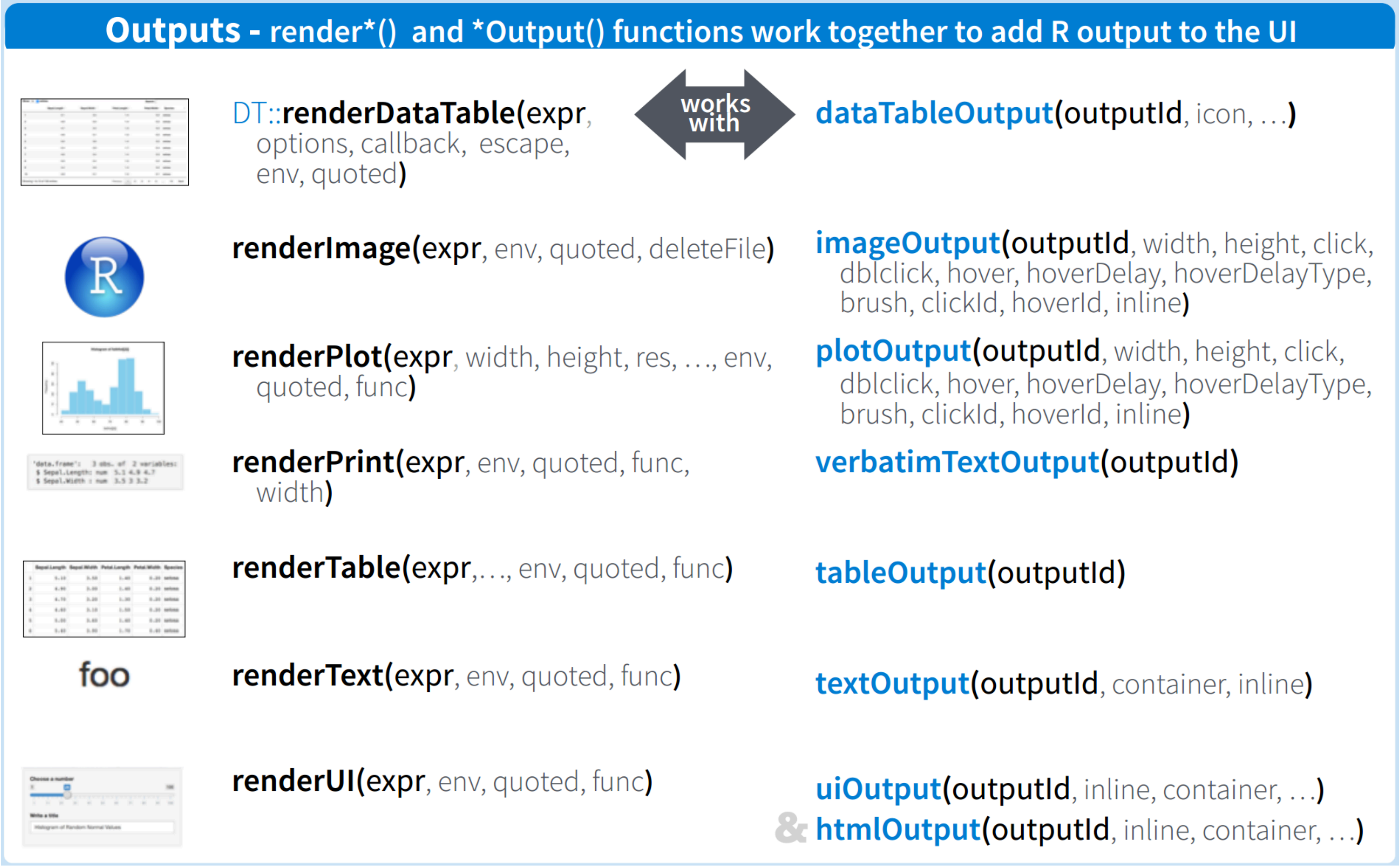
UI: Output

- The output will be updated automatically when an input widget changes: **reactivity**
  - when an user manipulates the app, Shiny reruns parts of `server.R` to create an updated output

Types of outputs:

- Plot: `plotOutput`
- Image: `imageOutput`
- Text: `textOutput`
- Code: `verbatimTextOutput`
- Table: `tableOutput`

UI and server: Output



Outputs - render*() and *Output() functions work together to add R output to the UI

DT::**renderDataTable(expr**, options, callback, escape, env, quoted**)** — **dataTableOutput(outputId**, icon, …**)**

renderImage(expr, env, quoted, deleteFile**)** — **imageOutput(outputId**, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline**)**

renderPlot(expr, width, height, res, …, env, quoted, func**)** — **plotOutput(outputId**, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline**)**

```
'data.frame':   3 obs. of  2 variables:
 $ Sepal.Length: num  5.1 4.9 4.7
 $ Sepal.Width : num  3.5 3 3.2
```

renderPrint(expr, env, quoted, func, width**)** — **verbatimTextOutput(outputId)**

renderTable(expr,…, env, quoted, func**)** — **tableOutput(outputId)**

foo

renderText(expr, env, quoted, func**)** — **textOutput(outputId**, container, inline**)**

renderUI(expr, env, quoted, func**)** — **uiOutput(outputId**, inline, container, …**)**
& **htmlOutput(outputId**, inline, container, …**)**

Shiny app summary

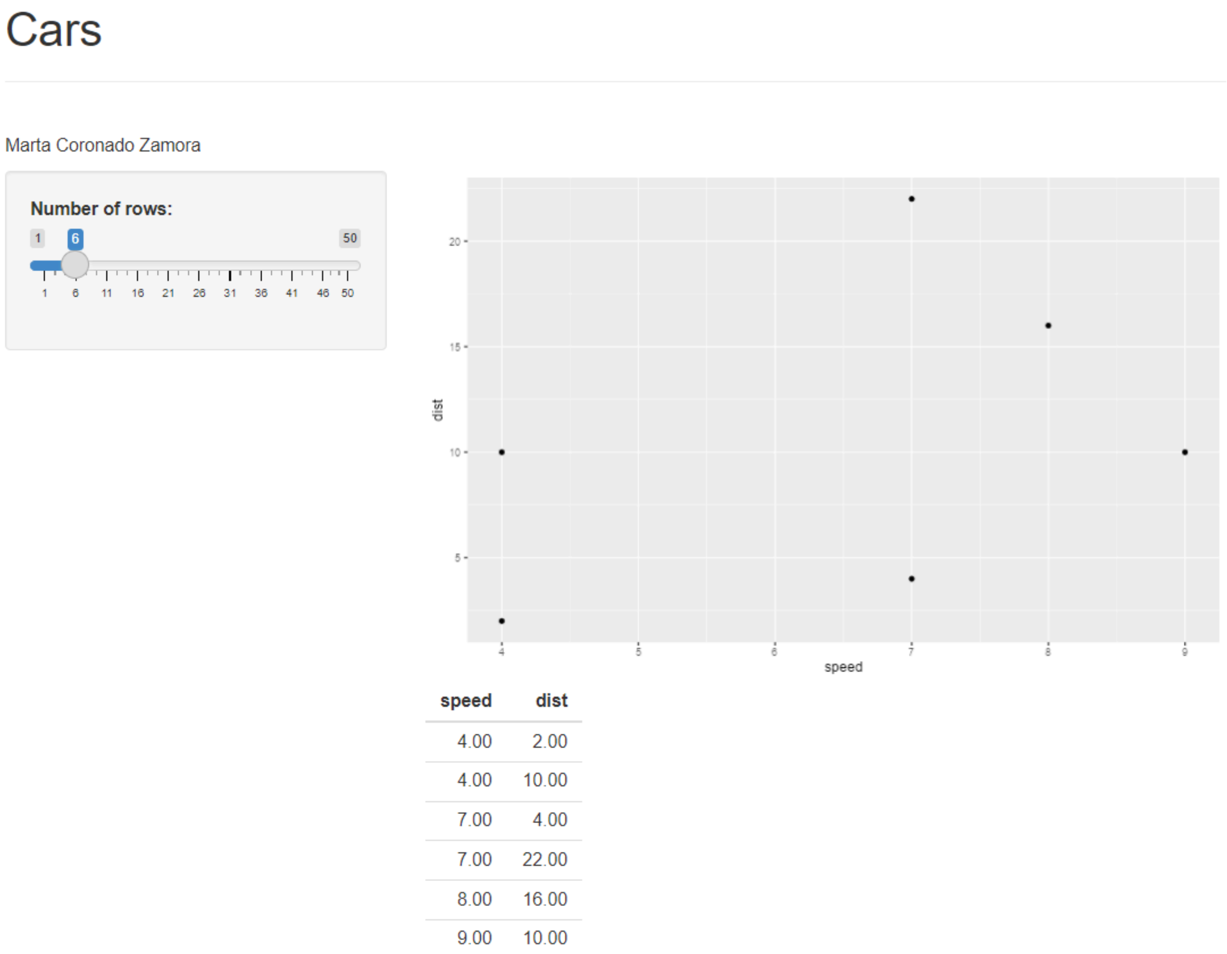
✎ **Exercise | Create a shiny app named 'Cars' with the following requirements:**

1. **Layout**: sidebar layout.
2. A **slider** in the sidebar panel with `inputId` "nrows" and `label` "Number of rows:", which controls how many rows of the data set `datasets::cars` to use in the following analysis. The minimum value is 1, maximum value is 50 and default value is 10.
3. In the main panel, create a **scatterplot** with `x` axis 'speed' and `y` axis 'dist' on the top and a table showing the data on the bottom, using `outputId` "`carsPlot`" and "`carsTable`" respectively.
4. Use HTML tags to format the UI: indicating an app title and your name.

🔑 You can use a subset of your data in ggplot as `cars[1:input$nrows,]`

💬 Answer:

Shiny app summary



Cars

Marta Coronado Zamora

Number of rows: 6

speed	dist
4.00	2.00
4.00	10.00
7.00	4.00
7.00	22.00
8.00	16.00
9.00	10.00

Upload `T3.1_slides.Rmd` with the completed exercises (text included) to aul@-ESCI