Data Visualization with R
Shiny tutorial

Ariane Ducellier

University of Washington - Fall 2025

Simple layouts

Left-to-right and top-to-bottom. The elements reorder themselves when resizing the window.

```
flowLayout( ... )
```

Top-to-bottom

```
verticalLayout( ... )
```

Left-to-right with manually set widths

```
splitLayout(cellWidths = c("25%", "75%"),
            ... ),
```

Complete layouts

Side bar and main panel

```
fluidpage(
  sidebarLayout(sidebarPanel, mainPanel, position))
)
```

Rows and columns. The sum of the widths of the columns must be equal to 12.

```
fluidpage(
  fluidrow(
    column(width=4, ...),
    column(width=4, ...), ... ))
```

Tabs

We can show multiple frames on the screen and let the user select one. The processing of the data is only carried out for the currently selected tab.

```
tabsetPanel(
  tabPanel("Slider", sliderInput("slider",
                                 "Slider",
                                 min = 1,
                                 max = 100,
                                 value = 50)),
  tabPanel("Text input", textInput("text", "Text")),
  tabPanel("Table", tableOutput("table")
)
```

Complete layouts

Top level navigation bar and several tabs

```
navbarPage(title, tabPanel)
```

Left navigation bar and several tabs

```
fluidpage(
  navlistPanel(title, tabPanel)
)
```

Complete layouts

Combination of layouts

```
fluidPage(
  fluidRow(
    column(width=4, ...), column(width=8, ...)),
  splitLayout( ... ),
  verticalLayout( ... )
)
```

Functions in ui.R and server.R

ui.R		server.R
textOutput	$\leftrightarrow$	renderText
plotOutput	$\leftrightarrow$	renderPlot
tableOutput	$\leftrightarrow$	renderTable
dataTableOutput	$\leftrightarrow$	renderDataTable
leafletOutput	$\leftrightarrow$	renderLeaflet

The `leaflet` package allows us to produce maps shown with `leafletOutput` in the `ui.R` and created with `renderLeaflet` in the `server.R` file.

Tables - Basic Shiny

In `ui.R`:

```
tableOutput("textDisplay")
```

In `server.R`

```
output$textDisplay = renderTable({
  getMat = matrix(c( ... ), ncol = 2, byrow = TRUE)
  colnames(getMat) = c("Value", "Class")
  getMat
})
```

Tables - With package DT (DataTable)

In `ui.R`:

```
dataTableOutput("countryTable")
```

In `server.R`

```
output$countryTable = renderDataTable({
  dataTable(
    ... ,
    colnames = ... ,
    caption = ... ,
    filter = "top",
    options = list(pageLength = 15,
                   lengthMenu = c(10, 20, 50))
})
```

Hiding elements

Name the panels:

```
tabsetPanel(id = "theTabs",
  tabPanel( ... , value = "trend"),
  ...
)
```

Add a condition to show an UI element only if a tab is selected:

```
conditionalPanel(
  condition = "input.theTabs == 'trend'",
  checkboxInput( ... )
)
```

Reactive objects

In the `server.r` file, we filter the data using a reactive object:

```
theData = reactive({
   mapData %>%
      filter(year >= input$year)
})
```

- A reactive object changes when its input changes.
- When it runs, the output is cached.
- If it is called several times in an application, it will not run again if the inputs are unchanged.

Reactive user interfaces

In `ui.R`:

```
uiOutput("yearSelectorUI")
```

In `server.R`

```
output$yearSelectorUI = renderUI(
  selectedYears = ...
  selectInput( ... , selectedYears)
})
```

When the value in `selectedYears` change, the choice of years in the widget will also change.