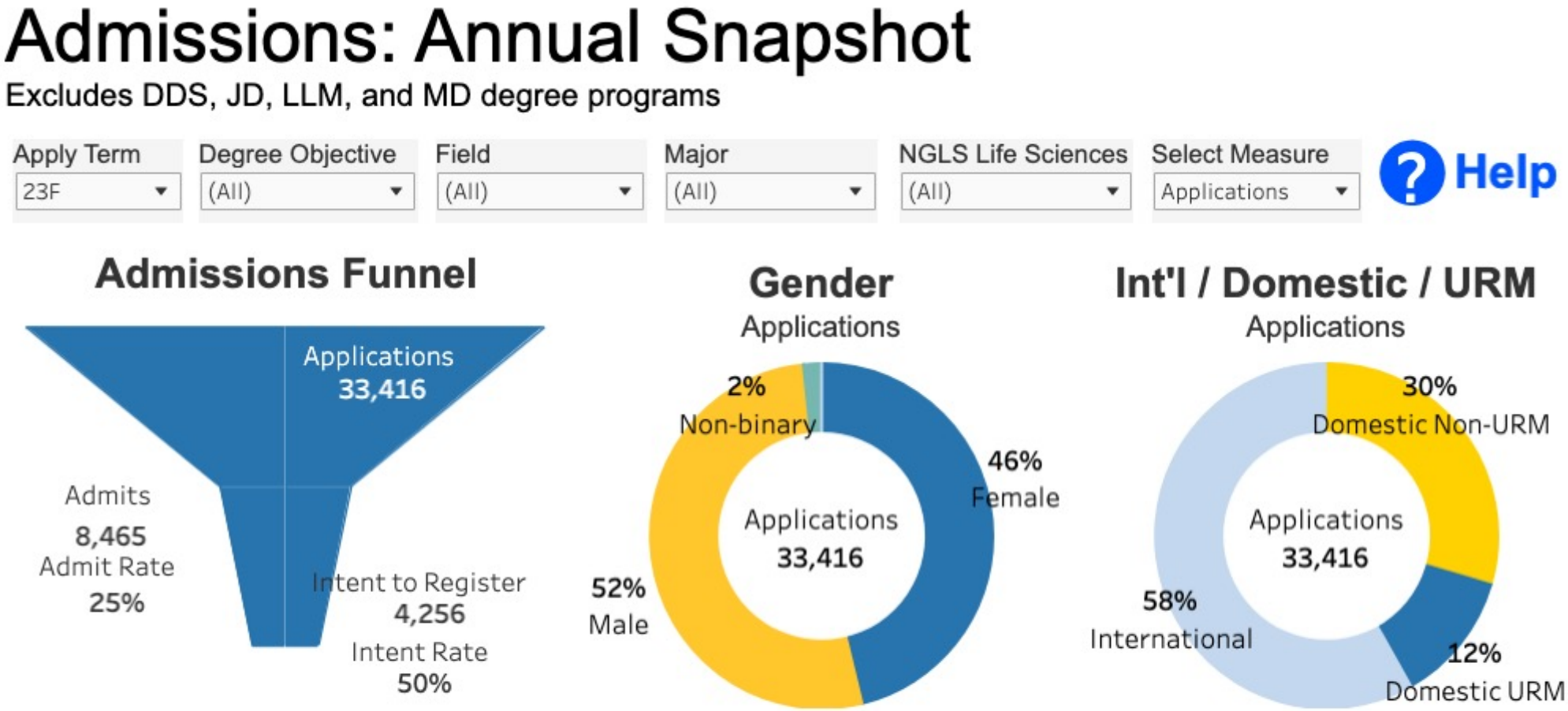
Contents

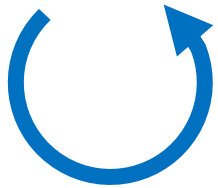
Motivation



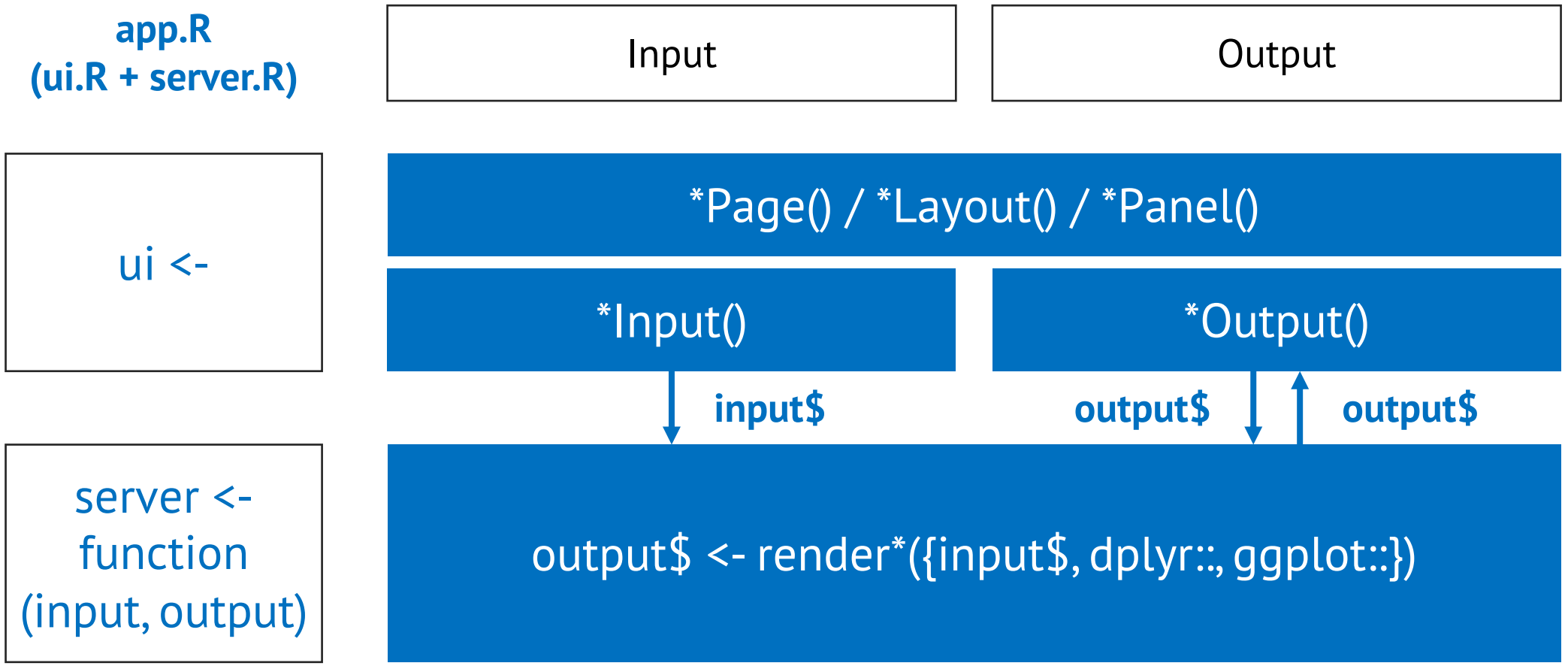
https://grad.ucla.edu/graduate-program-statistics/admissions/?t=Annualsnapshot

Shiny app Structure

	Input	Output
UI (Web page)	• Set page structure • Define input variables extracted from data	• Set output layout • Receive and display outputs generated in server.
Server (R session)	• Receive selected inputs (reactive values) from UI	• Perform calculations and visualizations • Return tables/graphs to UI



Shiny app Structure



shinyApp(ui, server) >> **click [Run App]**

Functions

UI

Server

Where

What

How

*Page / *Layout()

- fluidPage()
- slidebarLayout()
- splitLayout()
- verticalLayout()

*Panel()

- sidebarPanel()
- titlePanel()
- headerPanel()
- mainPanel()
- tabPanel()

*Input()

actionLink(inputId, label, icon, ...)

checkboxGroupInput(inputId, label, choices, selected, inline, width, choiceNames, choiceValues)

checkboxInput(inputId, label, value, width)

dateInput(inputId, label, value, min, max, format, startview, weekstart, language, width, autoclose, datesdisabled, daysofweekdisabled)

dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator, width, autoclose)

fileInput(inputId, label, multiple, accept, width, buttonLabel, placeholder)

numericInput(inputId, label, value, min, max, step, width)

*Output()

dataTableOutput(outputId)

imageOutput(outputId, width, height, click, dblclick, hover, brush, inline)

plotOutput(outputId, width, height, click, dblclick, hover, brush, inline)

verbatimTextOutput(outputId, placeholder)

tableOutput(outputId)

textOutput(outputId, container, inline)

uiOutput(outputId, inline, container, ...)
htmlOutput(outputId, inline, container, ...)

render*()

DT::**renderDataTable**(expr, options, searchDelay, callback, escape, env, quoted, outputArgs)

renderImage(expr, env, quoted, deleteFile, outputArgs)

renderPlot(expr, width, height, res, ..., alt, env, quoted, execOnResize, outputArgs)

renderPrint(expr, env, quoted, width, outputArgs)

renderTable(expr, striped, hover, bordered, spacing, width, align, rownames, colnames, digits, na, ..., env, quoted, outputArgs)

renderText(expr, env, quoted, outputArgs, sep)

renderUI(expr, env, quoted, outputArgs)

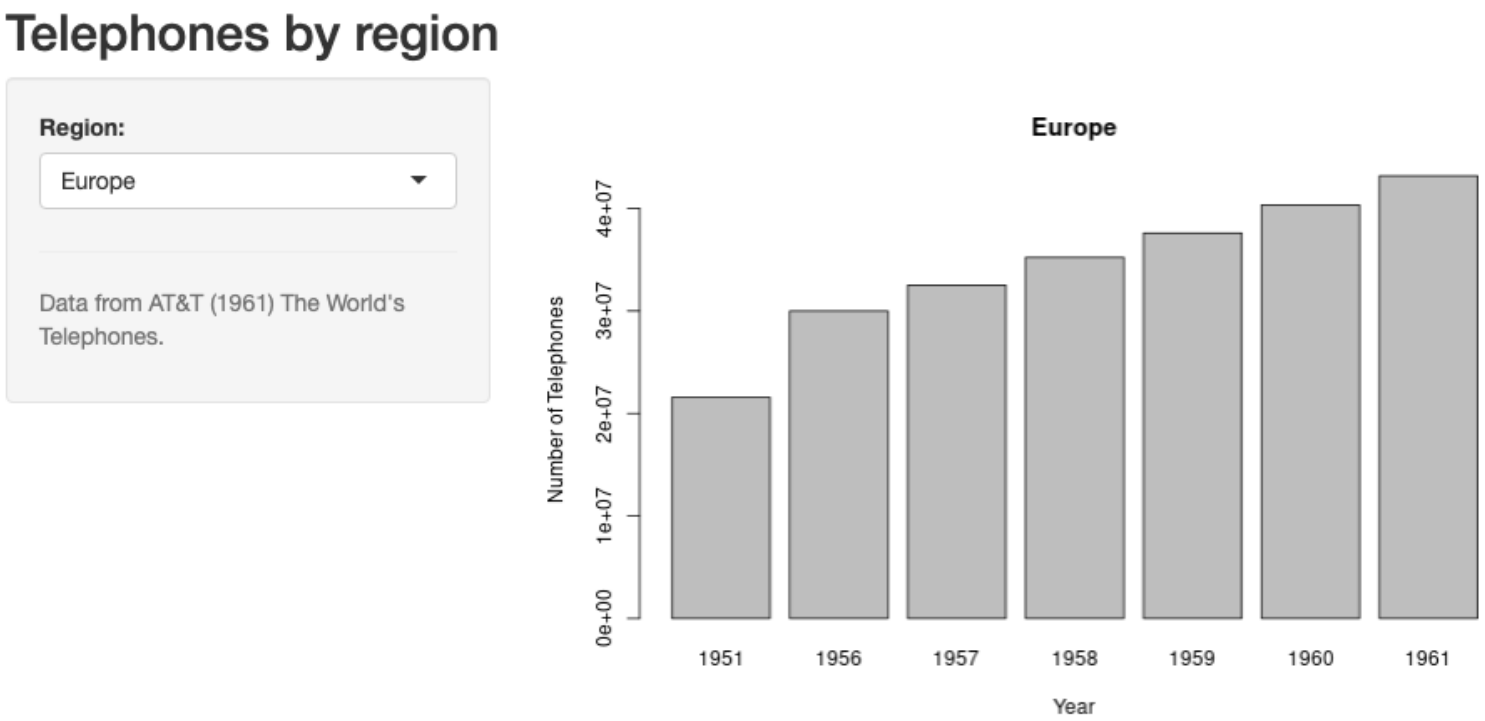
Cheat sheet: https://github.com/rstudio/cheatsheets/raw/main/shiny.pdf

Contents

- Motivation
- Shiny app Structure
- **Example**
  - **Shiny Gallery**
  - **Lecture Note**
  - **Last year's example**
- Efficient Processing with `reactive()`
- Beautiful Outputs with `bislib` package

Simple Example

Input	Output



https://shiny.posit.co/r/gallery/start-simple/telephones-by-region/

Simple Example

Region ▼

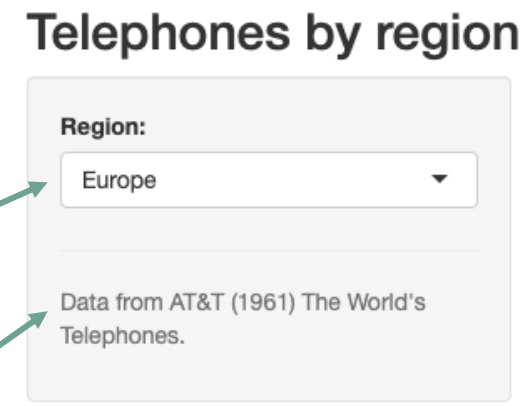
	N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
1951	45939	21574	2876	1815	1646	89	555
1956	60423	29990	4708	2568	2366	1411	733
1957	64721	32510	5230	2695	2526	1546	773
1958	68484	35218	6662	2845	2691	1663	836
1959	71799	37598	6856	3000	2868	1769	911
1960	76036	40341	8220	3145	3054	1905	1008
1961	79831	43173	9053	3338	3224	2005	1076

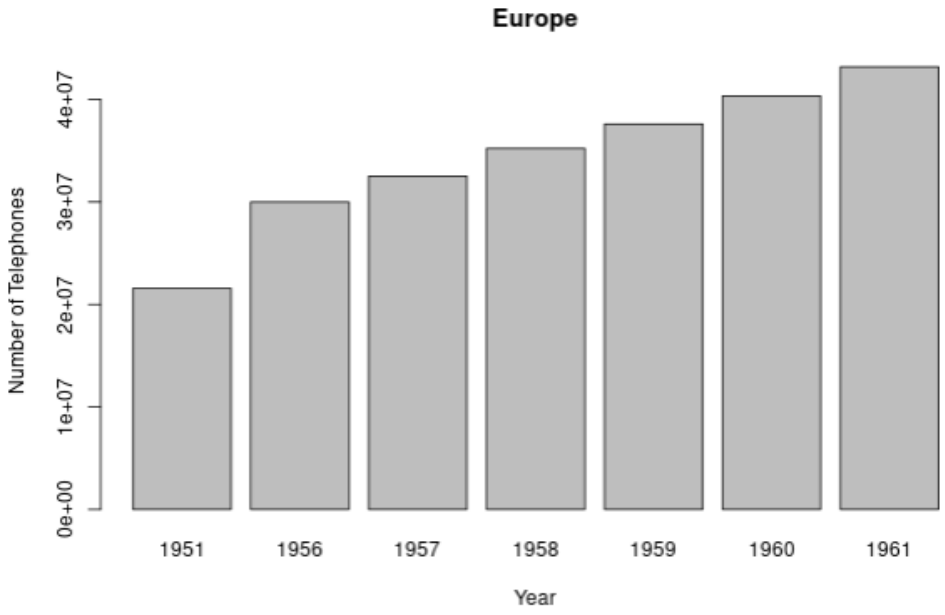
https://shiny.posit.co/r/gallery/start-simple/telephones-by-region/

What UI Does

ui <-	Input	Output

```
# Use a fluid Bootstrap layout
fluidPage(

  # Give the page a title
  titlePanel("Telephones by region"),

  # Generate a row with a sidebar
  sidebarLayout(

    # Define the sidebar with one input
    sidebarPanel(
      selectInput("region", "Region:",
                  choices=colnames(WorldPhones)),
      hr(),
      helpText("Data from AT&T (1961) The World's Telephones.")
    ),

    # Create a spot for the barplot
    mainPanel(
      plotOutput("phonePlot")
    )

  )
)
```

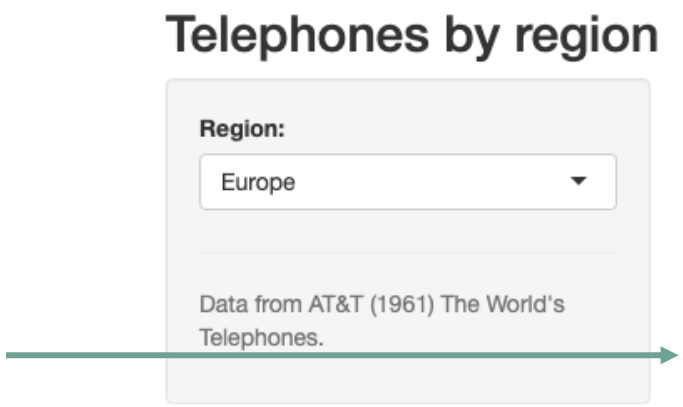


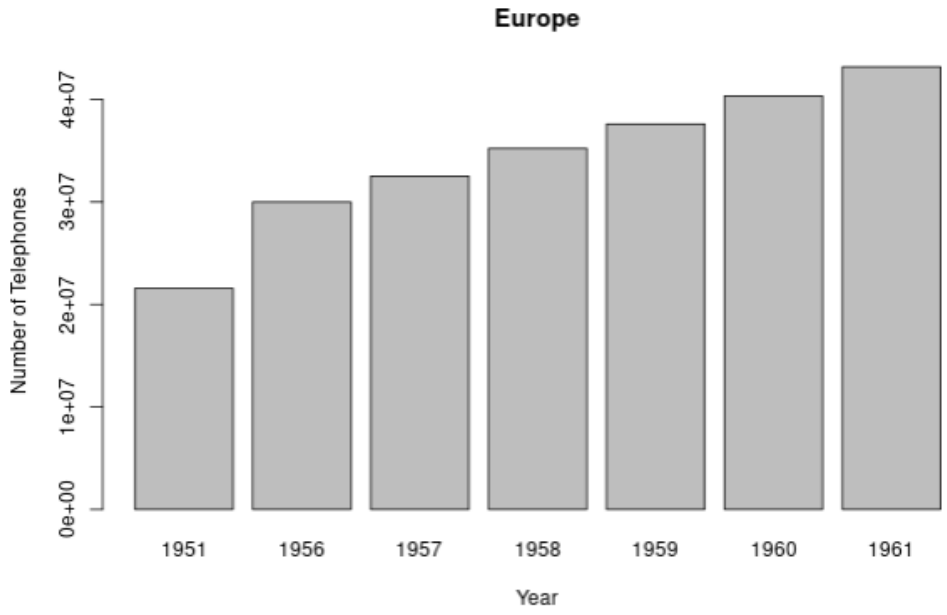


What Server Does

server <-	Input	Output

```
# Define a server for the Shiny app
function(input, output) {

  # Fill in the spot we created for a plot
  output$phonePlot <- renderPlot({

    # Render a barplot
    barplot(WorldPhones[,input$region]*1000,
            main=input$region,
            ylab="Number of Telephones",
            xlab="Year")
  })
}
```

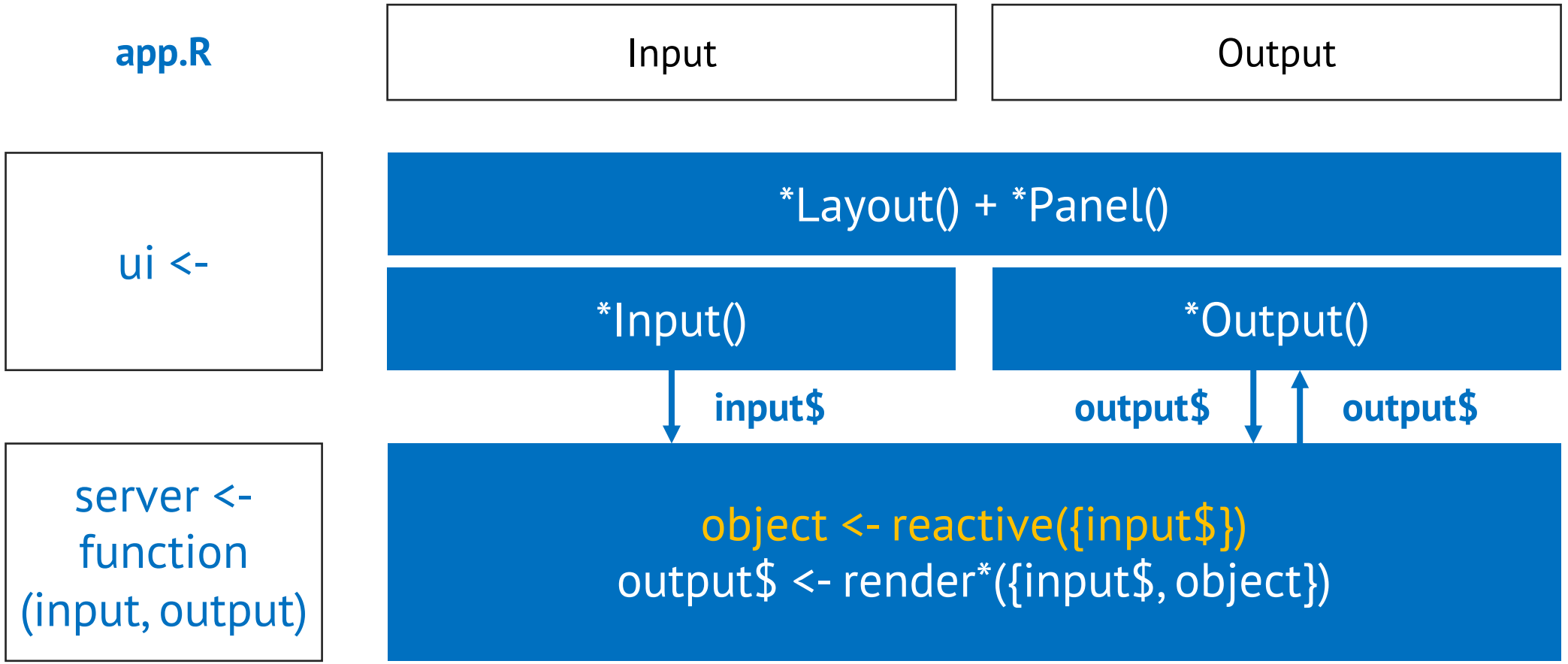




Other Examples

- Gallery
- Lecture note
- Jonathan’s work last year

Efficient Processing with `reactive()`



shinyApp(ui, server) >> **click [Run App]**

Efficient Processing with `reactive()`

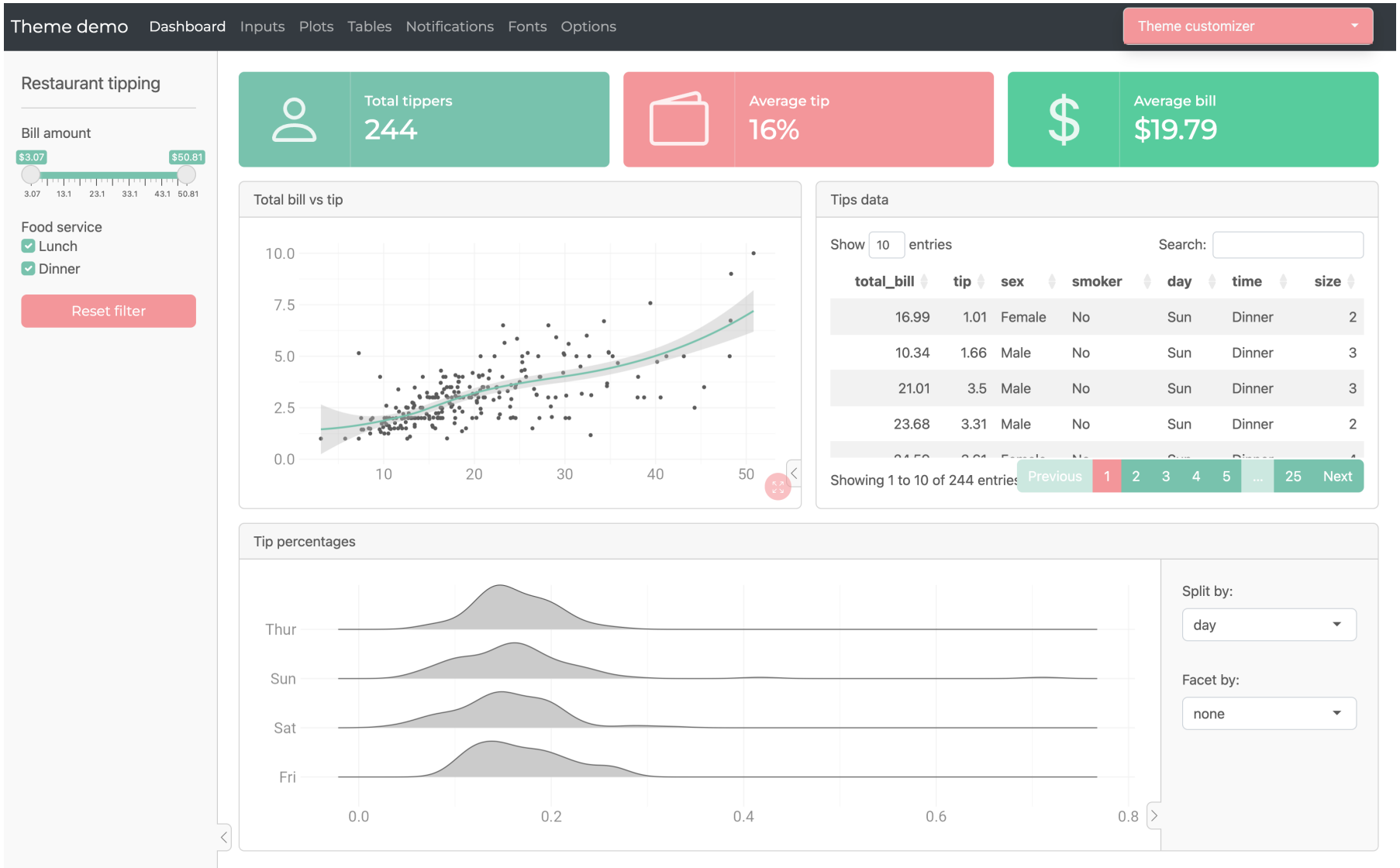
Not good (expensive)

```
server <- function(input, output) {
  output$distPlot <- renderPlot({
    # import data
    x = faithful[, 2]

    # binning
    bins <- seq(min(x),
                max(x),
                length.out = input$bins + 1)
    # plot
    hist(faithful[, 2],
         breaks = bins,
         col = input$color,
         border = 'white')
  })
}
```

Smart (efficient)

```
server <- function(input, output) {

  ## import data only one time
  x = faithful[, 2]

  ## Process only when input$bins changes
  bins_re <- reactive({
    bins <- seq(min(x),
                max(x),
                length.out = input$bins + 1)
    return (bins)
  })

  output$distPlot <- renderPlot({
    hist(faithful[, 2],
         breaks = bins_re(),
         col = input$color,
         border = 'white')
  })
}
```

bislib package



https://rstudio.github.io/bslib/articles/theming/