BUILDING WEB APPLICATIONS IN R WITH SHINY

Interface builder functions

tags

```
> names(tags)
  [1] "a"          "abbr"       "address"    "area"       "article"
  [6] "aside"      "audio"      "b"          "base"       "bdi"
 [11] "bdo"        "blockquote" "body"       "br"         "button"
 [16] "canvas"     "caption"    "cite"       "code"       "col"
 [21] "colgroup"   "command"    "data"       "datalist"   "dd"
 [26] "del"        "details"    "dfn"        "div"        "dl"
 [31] "dt"         "em"         "embed"      "eventsource" "fieldset"
 [36] "figcaption" "figure"     "footer"     "form"       "h1"
 [41] "h2"         "h3"         "h4"         "h5"         "h6"
 [46] "head"                                 "hr"         "html"
 [51] "i"                                    "input"      "ins"
 [56] "kbd"                                  "legend"     "li"
 [61] "link"       "mark"       "map"        "menu"       "meta"
 [66] "meter"      "nav"        "noscript"   "object"     "ol"
 [71] "optgroup"   "option"     "output"     "p"          "param"
 [76] "pre"        "progress"   "q"          "ruby"       "rp"
 [81] "rt"         "s"          "samp"       "script"     "section"
 [86] "select"     "small"      "source"     "span"       "strong"
 [91] "style"      "sub"        "summary"    "sup"        "table"
 [96] "tbody"      "td"         "textarea"   "tfoot"      "th"
[101] "thead"      "time"       "title"      "tr"         "track"
[106] "u"          "ul"         "var"        "video"      "wbr"
```

```
<i> some text </i>
```

tag → HTML

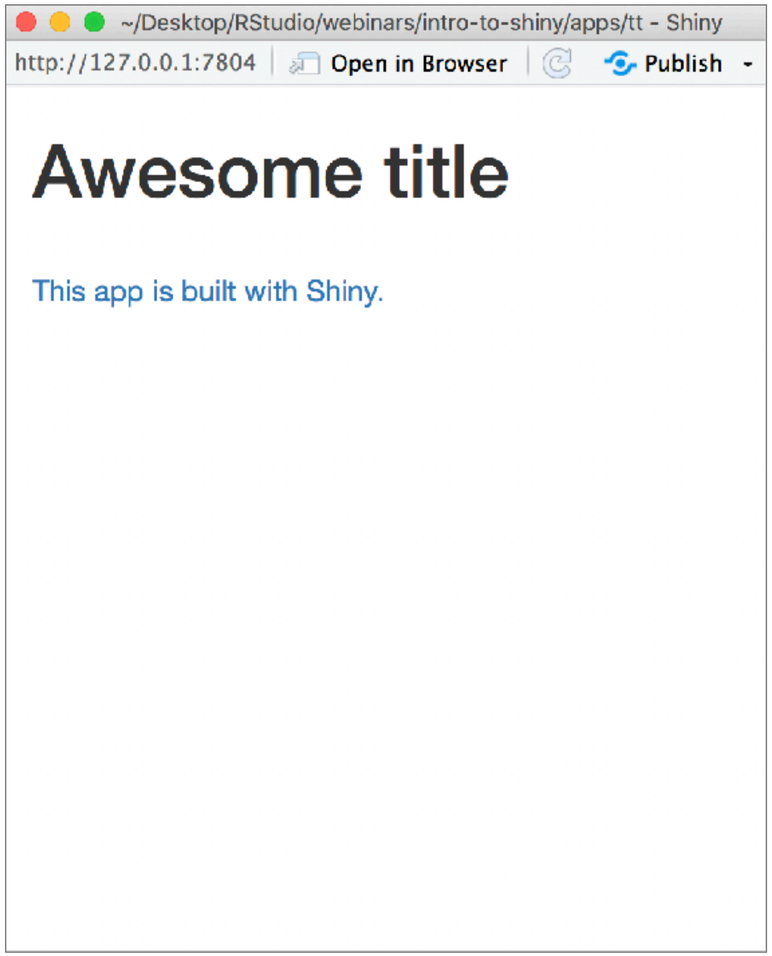
```
> tags$b("This is my first app")
<b>This is my first app</b>
```

Header tags

```
library(shiny)

# Define UI with tags
ui <- fluidPage(
  tags$h1("First level heading"),
  tags$h2("Second level heading"),
  tags$h3("Third level heading")
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```

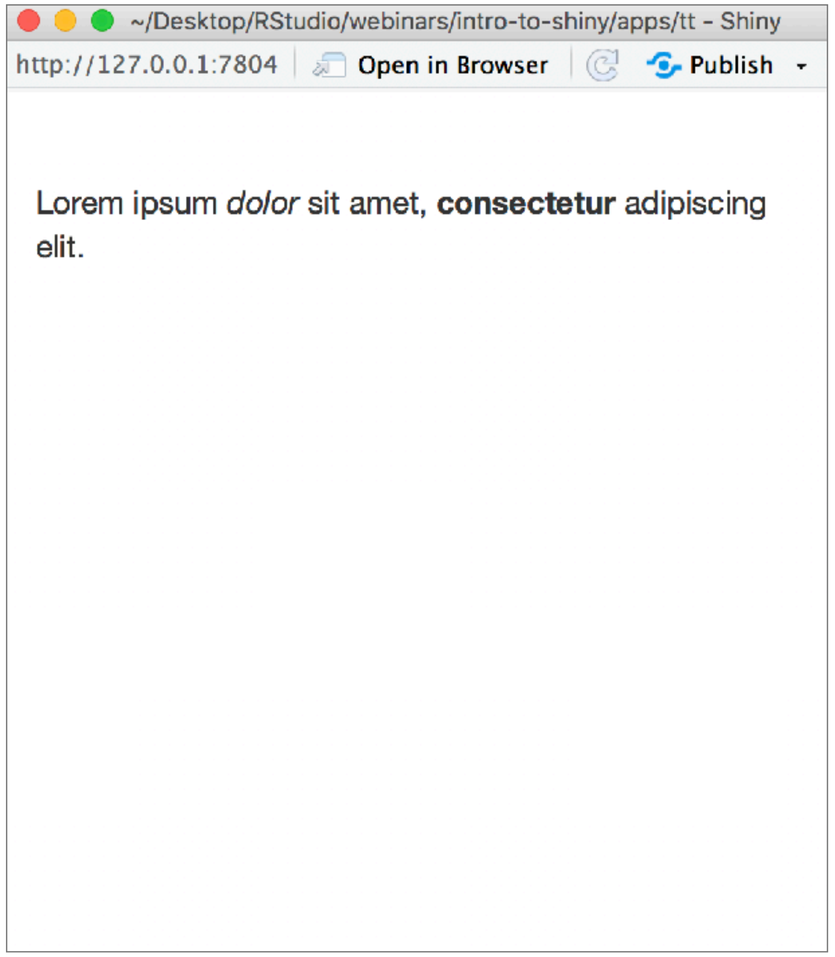
Linked text

```
library(shiny)

# Define UI with tags
ui <- fluidPage(
  tags$h1("Awesome title"),
  tags$br(), # line break
  tags$a("This app is built with
Shiny.", href = "http://
shiny.rstudio.com/")
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```



Nested tags

```
library(shiny)

# Define UI with tags
ui <- fluidPage(
    tags$p("Lorem ipsum",
           tags$em("dolor"), "sit amet,",
           tags$b("consectetur"),
           "adipiscing elit.")

)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```



Common tags

```
tags$p(...)          p(...)
tags$h1(...)         h1(...)
tags$h2(...)         h2(...)
tags$h3(...)         h3(...)
tags$h4(...)         h4(...)
tags$h5(...)         h5(...)
tags$h6(...)         h6(...)
tags$a(...)      →   a(...)
tags$br(...)         br(...)
tags$div(...)        div(...)
tags$span(...)       span(...)
tags$pre(...)        pre(...)
tags$code(...)       code(...)
tags$img(...)        img(...)
tags$strong(...)     strong(...)
tags$em(...)         em(...)
tags$hr(...)         hr(...)
```

Common tags

```
> tags$a("Anchor text")
<a>Anchor text</a>
> a("Anchor text")
<a>Anchor text</a>
```

```
> tags$br()
<br/>
> br()
<br/>
```

```
> tags$code("Monospace text")
<code>Monospace text</code>
> code("Monospace text")
<code>Monospace text</code>
```

```
> tags$h1("First level header")
<h1>First level header</h1>
> h1("First level header")
<h1>First level header</h1>
```

HTML

```
> HTML("Hello world, <br/> and then a line break.")
Hello world, <br/> and then a line break.
```

BUILDING WEB APPLICATIONS IN R WITH SHINY

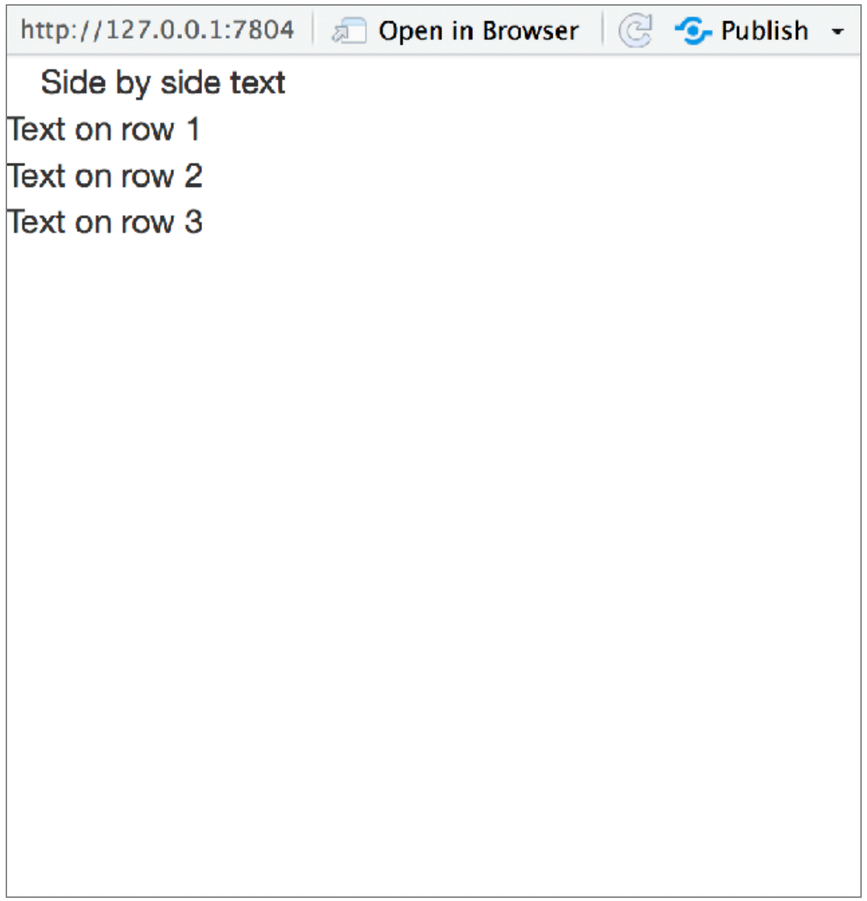
Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

Layout panels

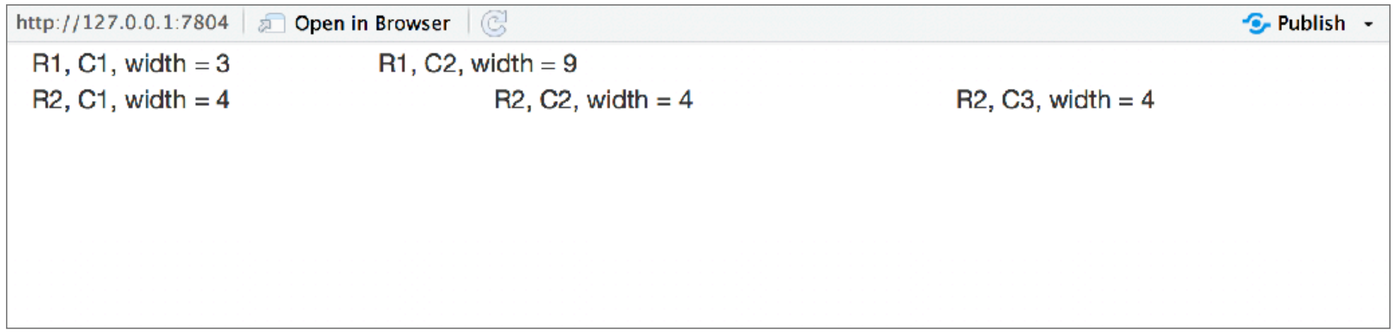
fluidrow()

```
library(shiny)

# Define UI with fluid rows
ui <- fluidPage(
  "Side", "by", "side", "text",
  fluidRow("Text on row 1"),
  fluidRow("Text on row 2"),
  fluidRow("Text on row 3")
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```

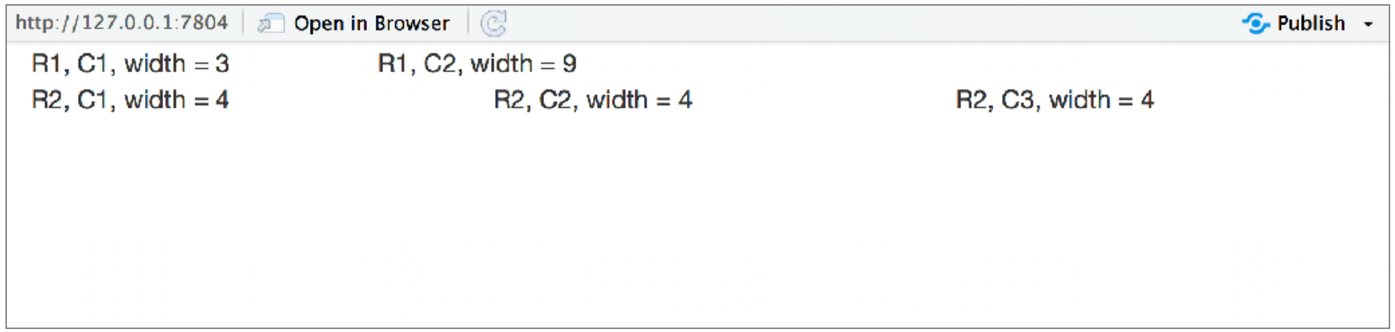


column()

```
library(shiny)

# Define UI with fluid rows and columns
ui <- fluidPage(
    fluidRow(
      column("R1, C1, width = 3", width = 3),
      column("R1, C2, width = 9", width = 9)
    ),
  fluidRow(
      column("R2, C1, width = 4", width = 4),
      column("R2, C2, width = 4", width = 4),
      column("R2, C3, width = 4", width = 4)
  )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```



column()



3 + 9 = 12

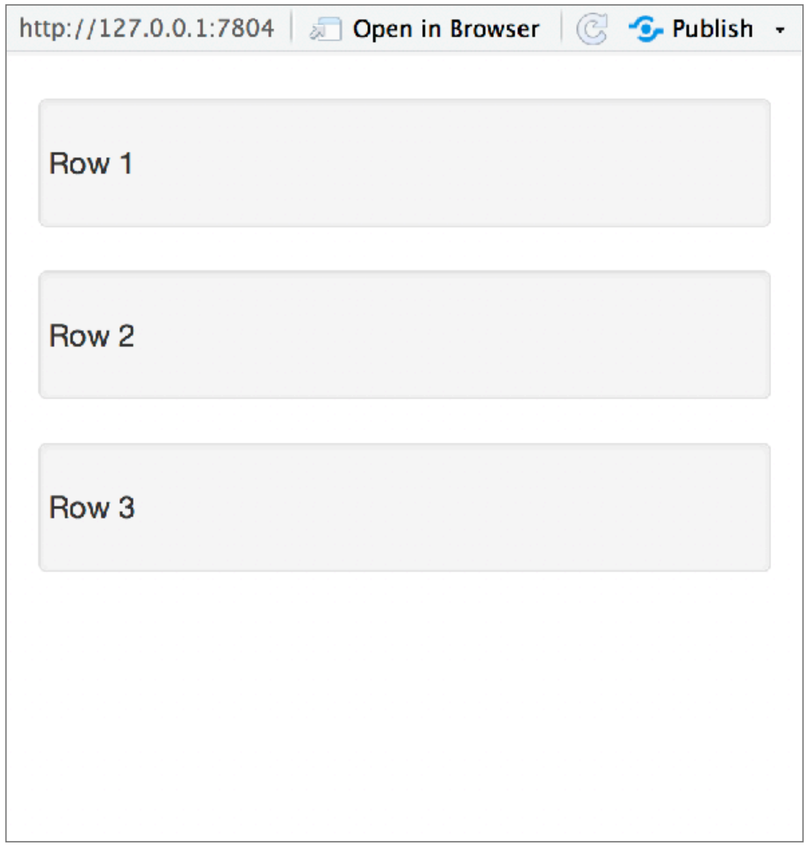
4 + 4 + 4 = 12

Panels

- Use panels to group multiple elements into a single element that has its own properties.
- Especially important and useful for complex apps with a large number of inputs and outputs such that it might not be clear to the user where to get started.

wellPanel()

```
library(shiny)

# Define UI with wellPanels
ui <- fluidPage(
  wellPanel( fluidRow("Row 1") ),
  wellPanel( fluidRow("Row 2") ),
  wellPanel( fluidRow("Row 3") )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```



Panels

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

sidebarPanel() and mainPanel()

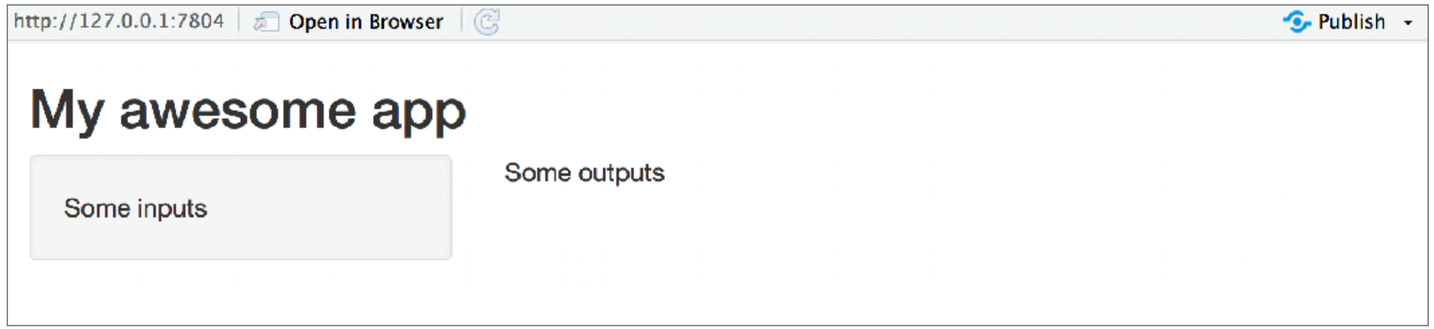
```
library(shiny)

# Define UI with default width sidebar
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel("Usually inputs go here"),
    mainPanel("Usually outputs go here")
  )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```

sidebarPanel() and mainPanel()

```
library(shiny)

# Define UI with custom width sidebar
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel("Usually inputs go here”,
                 width = 6),
    mainPanel("Usually outputs go here”,
              width = 6)
  )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```

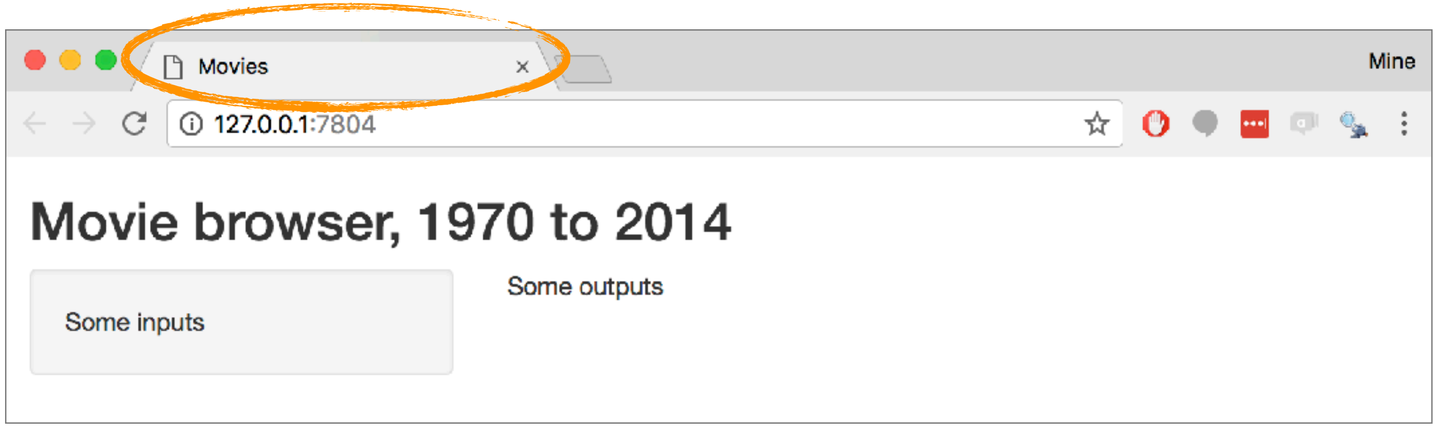
titlePanel()

```
library(shiny)

# Define UI with title panel
ui <- fluidPage(
  titlePanel("My awesome app"),
  sidebarLayout(
    sidebarPanel("Some inputs"),
    mainPanel("Some outputs")
  )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```

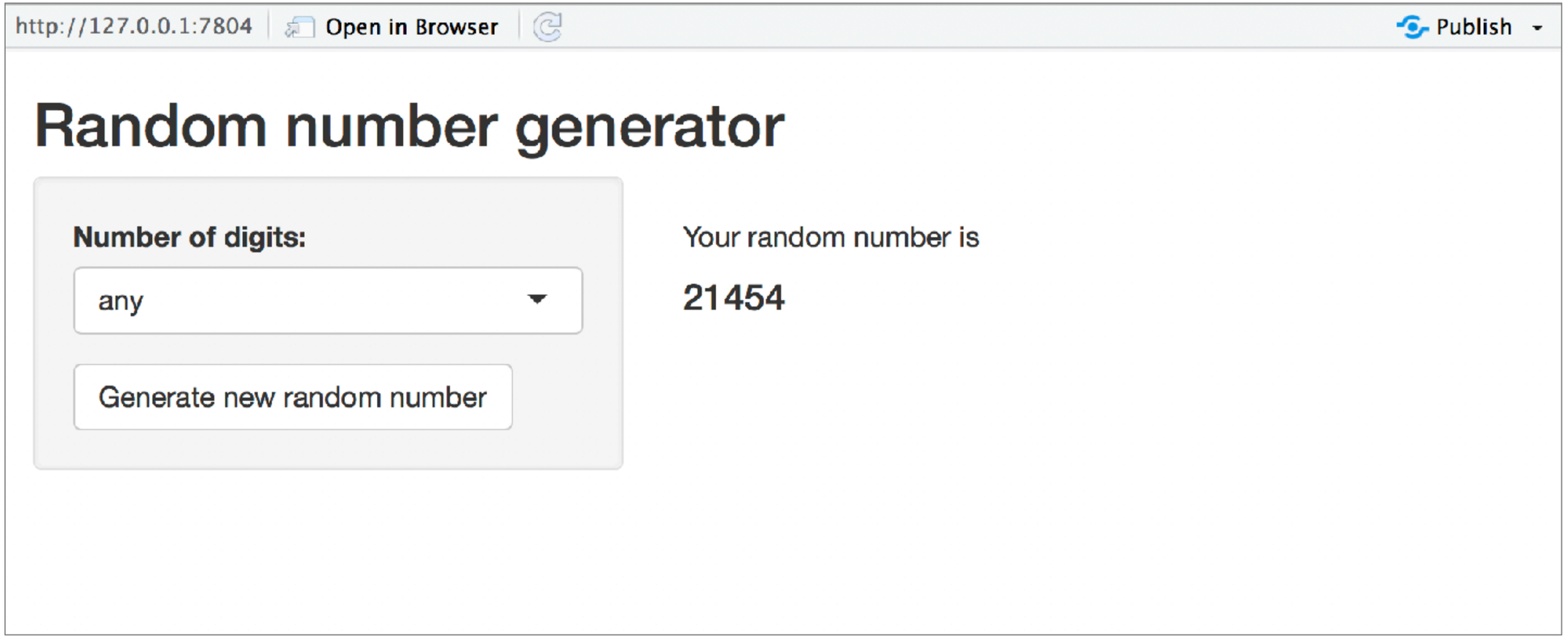


titlePanel() with windowTitle

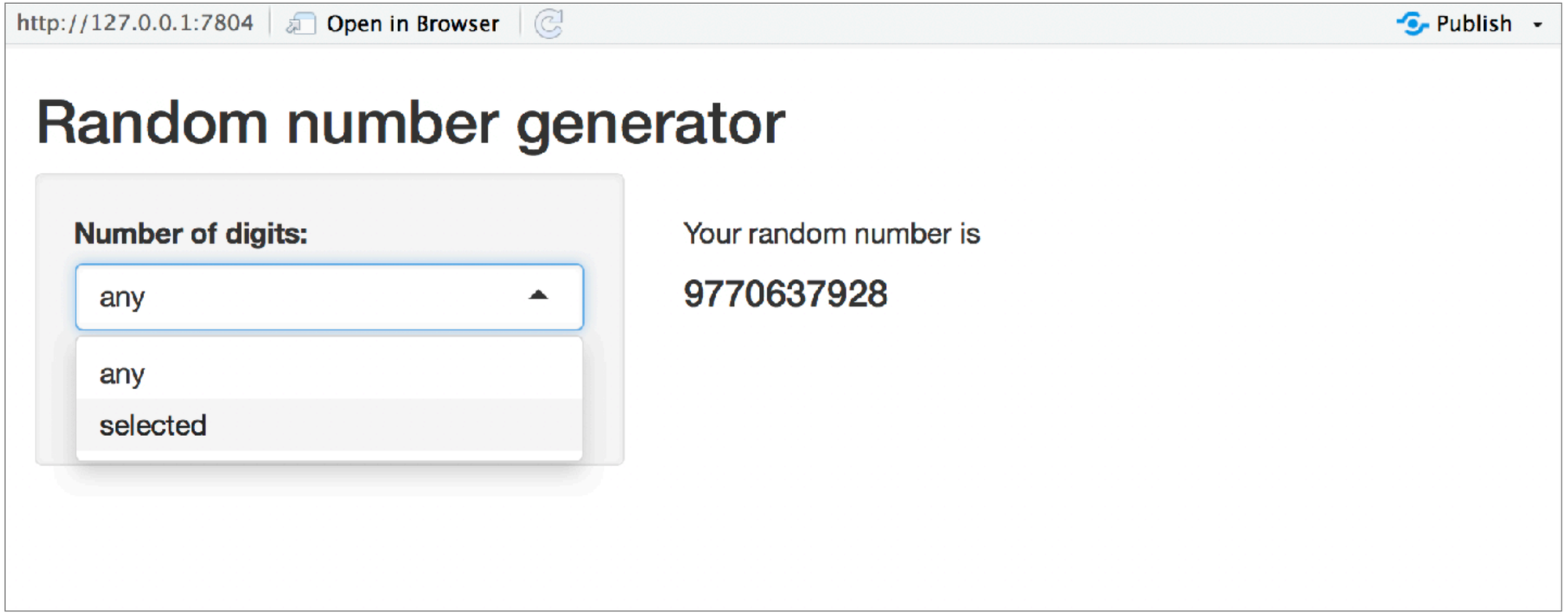
```
library(shiny)

# Define UI with title panel
ui <- fluidPage(
  titlePanel("Movie browser, 1970 to 2014",
             windowTitle = "Movies"),
  sidebarLayout(
    sidebarPanel("Some inputs"),
    mainPanel("Some outputs")
  )
)

# Define server fn that does nothing :)
server <- function(input, output) {}

# Create the app object
shinyApp(ui = ui, server = server)
```



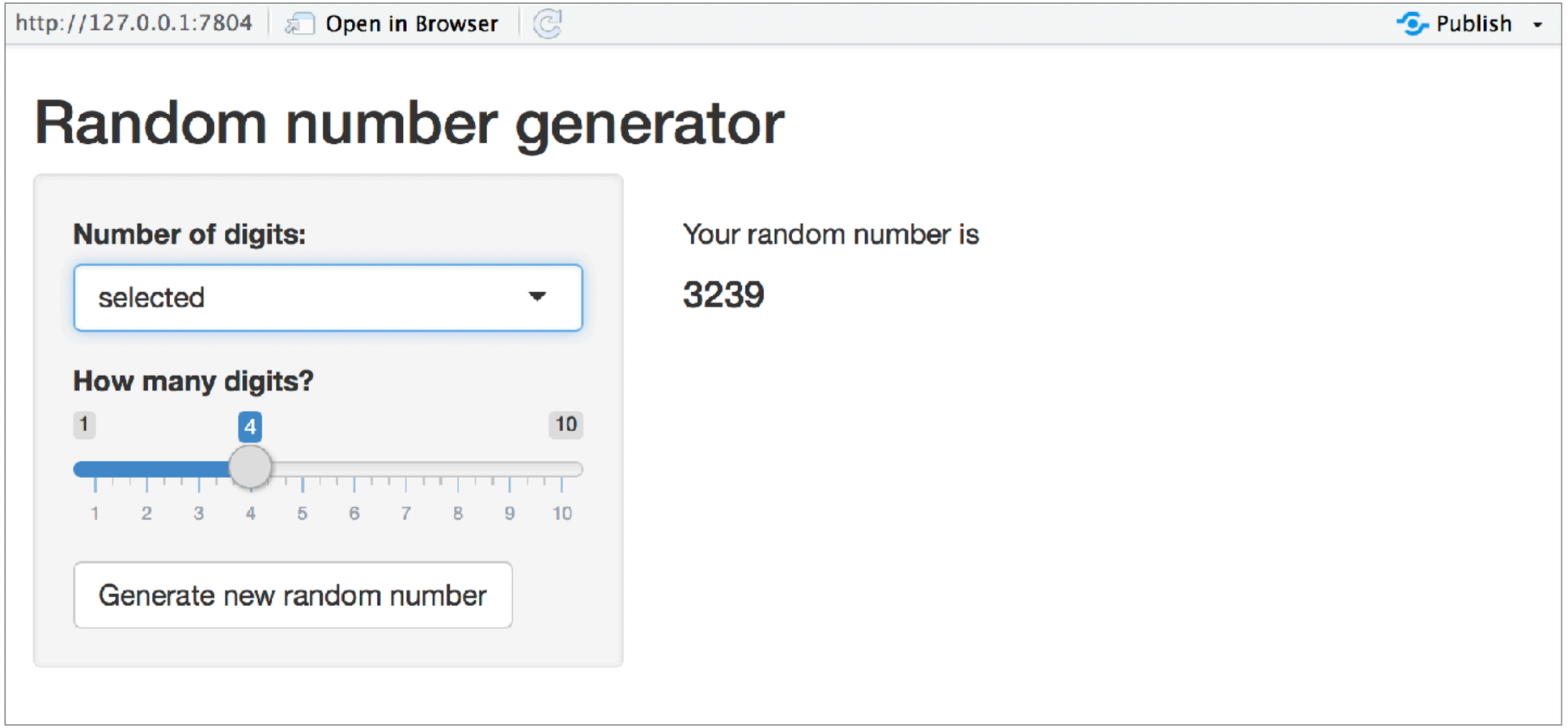
conditionalPanel()



conditionalPanel()



conditionalPanel()



BUILDING WEB APPLICATIONS IN R WITH SHINY

Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

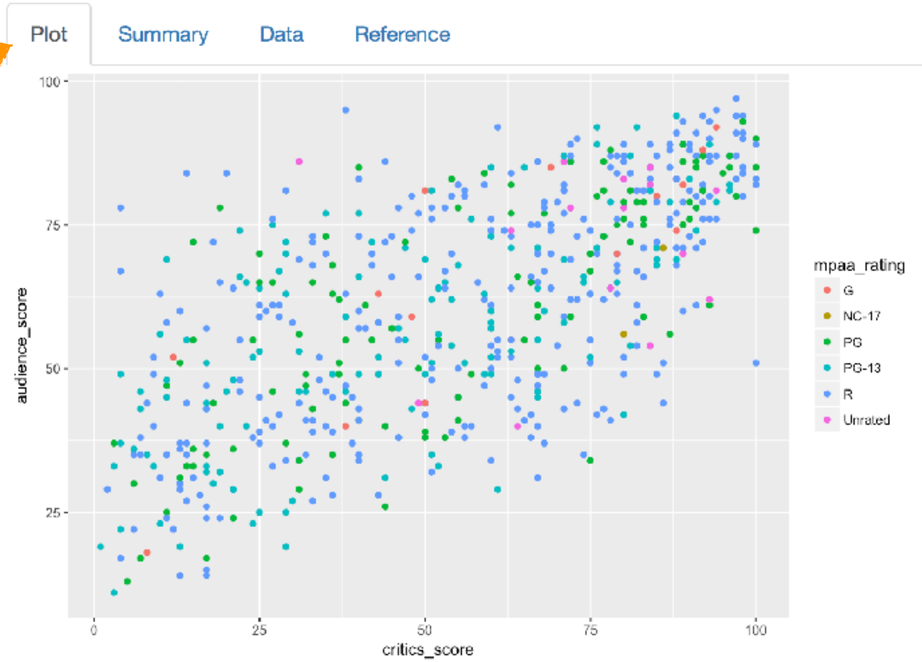
Tabs and tabsets

tabsetPanel() and tabPanel()

```
mainPanel(
  tabsetPanel(type = "tabs",
              tabPanel("Plot", plotOutput("plot")),
              tabPanel("Summary", tableOutput("summary")),
              tabPanel("Data", DT::dataTableOutput("data")),
              tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://www.rottentomatoes.com/"),
“."),
    tags$p("The data represent", nrow(movies), "randomly sampled movies
released between 1972 to 2014 in the United States.")
    )
  )
```

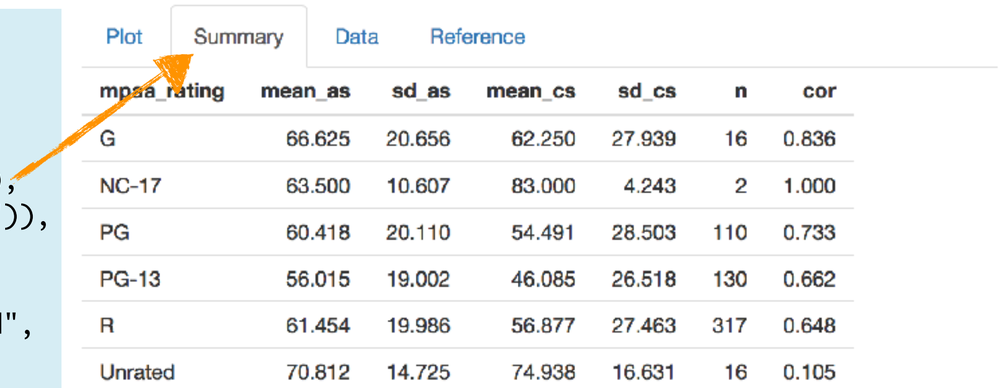
tabPanel()

```
mainPanel(
  tabsetPanel(type = "tabs",
              tabPanel("Plot", plotOutput("plot")),
              tabPanel("Summary", tableOutput("summary")),
              tabPanel("Data", DT::dataTableOutput("data")),
              tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://
www.rottentomatoes.com/"), “."),
    tags$p("The data represent", nrow(movies), "randomly
sampled movies released between 1972 to 2014 in the United
States.")
    )
  )
```



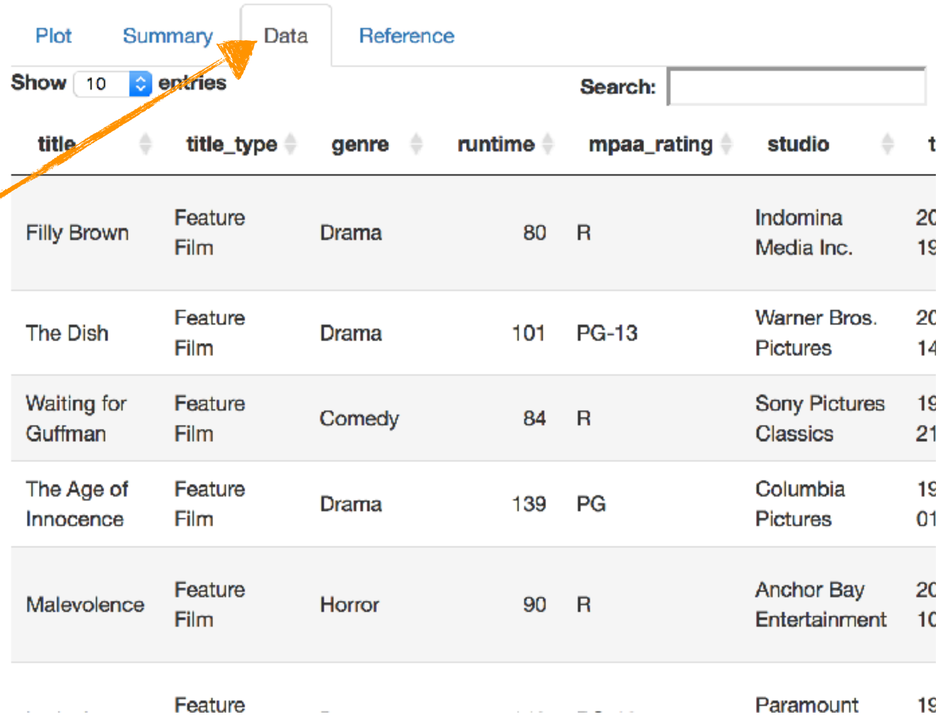
tabPanel()

```
mainPanel(
  tabsetPanel(type = "tabs",
              tabPanel("Plot", plotOutput("plot")),
              tabPanel("Summary", tableOutput("summary")),
              tabPanel("Data", DT::dataTableOutput("data")),
              tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://
www.rottentomatoes.com/"), “."),
    tags$p("The data represent", nrow(movies), "randomly
sampled movies released between 1972 to 2014 in the United
States.")
    )
  )
```



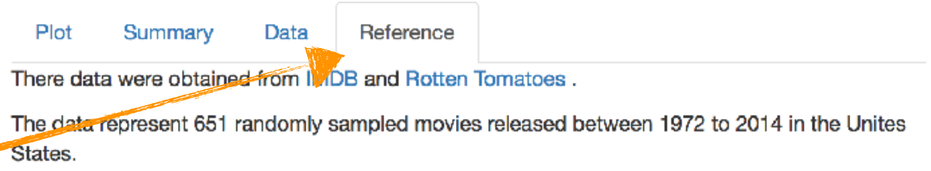
tabPanel()

```
mainPanel(
  tabsetPanel(type = "tabs",
              tabPanel("Plot", plotOutput("plot")),
              tabPanel("Summary", tableOutput("summary")),
              tabPanel("Data", DT::dataTableOutput("data")),
              tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://
www.rottentomatoes.com/"), “.”),
    tags$p("The data represent", nrow(movies), "randomly
sampled movies released between 1972 to 2014 in the United
States.")
    )
  )
```

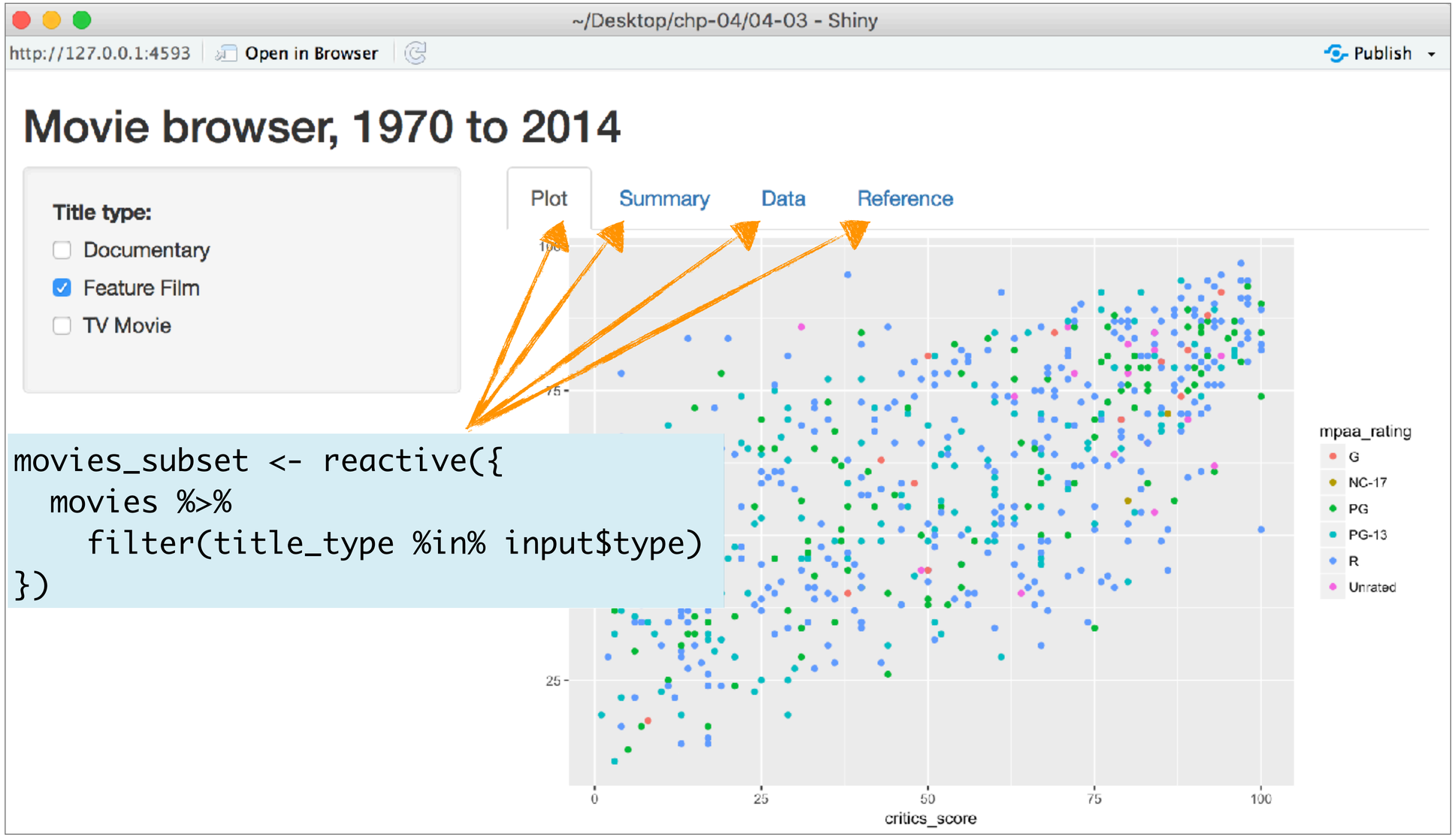


tabPanel()

```
mainPanel(
  tabsetPanel(type = "tabs",
              tabPanel("Plot", plotOutput("plot")),
              tabPanel("Summary", tableOutput("summary")),
              tabPanel("Data", DT::dataTableOutput("data")),
              tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://
www.rottentomatoes.com/"), “.”),
    tags$p("The data represent", nrow(movies), "randomly
sampled movies released between 1972 to 2014 in the United
States.")
    )
  )
```



Tabs and reactivity



navlistPanel()

```
mainPanel(
  navlistPanel(tabPanel("Plot", plotOutput("plot")),
               tabPanel("Summary", tableOutput("summary")),
               tabPanel("Data", DT::dataTableOutput("data")),
               tabPanel("Reference",
    tags$p("There data were obtained from",
      tags$a("IMDB", href = "http://www.imdb.com/"), "and",
      tags$a("Rotten Tomatoes", href = "https://
www.rottentomatoes.com/"), “.”),
    tags$p("The data represent", nrow(movies), "randomly
sampled movies released between 1972 to 2014 in the United
States.")
    )
  )
```

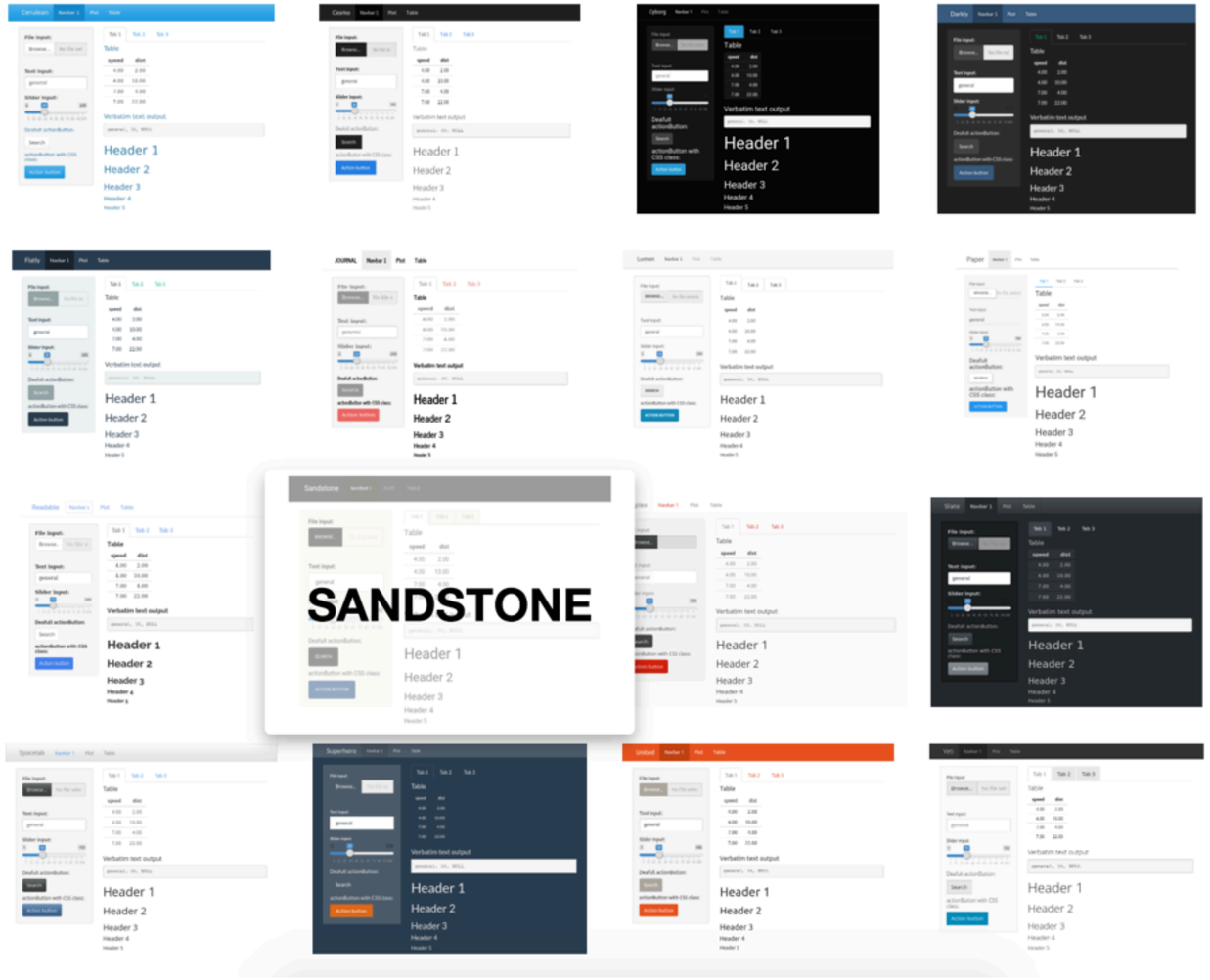
Tabs

Plot | Summary | Data | Reference

There data were obtained from IMDB and Rotten Tomatoes .

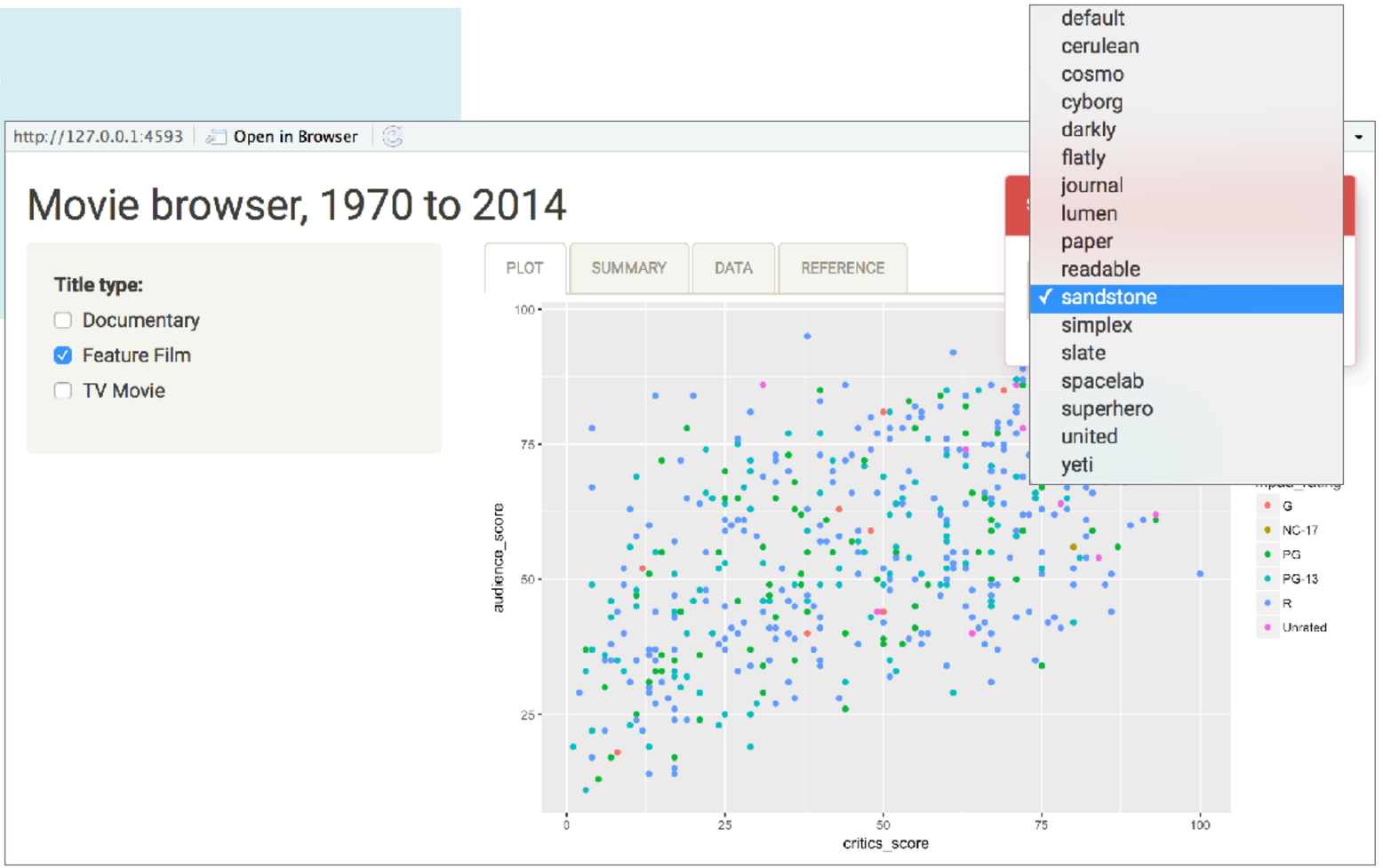
The data represent 651 randomly sampled movies released between 1972 to 2014 in the Unites States.

shinythemes



shinythemes

```
library(shiny)
library(shinythemes)

ui <- fluidPage(
  themeSelector(),
  …
)
```



BUILDING WEB APPLICATIONS IN R WITH SHINY

Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

Congratulations!

What did we learn?

- Design a Shiny app from scratch
- Essentials of reactive programming
- Customizing your app's UI
- Reactivity best practices

shinyapps.io

- Server maintained by RStudio
- Easy to use, secure, and scalable
- Comes with built in metrics
- Free tier available!

Shiny Server

- Free and open source
- Runs on premises, moving your computation closer to your data

Pro options

- RStudio Server Pro
- RStudio Connect
- Find out more at shiny.rstudio.com/deploy

BUILDING WEB APPLICATIONS IN R WITH SHINY

Congratulations!