

Why Shiny is just like sushi ...

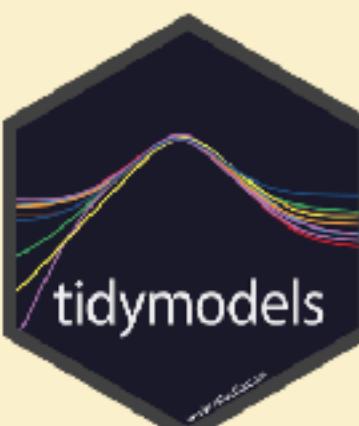


Taryn Morris
taryn@ixioanalytics.com
16 April 2020



Import

Tidy → Transform



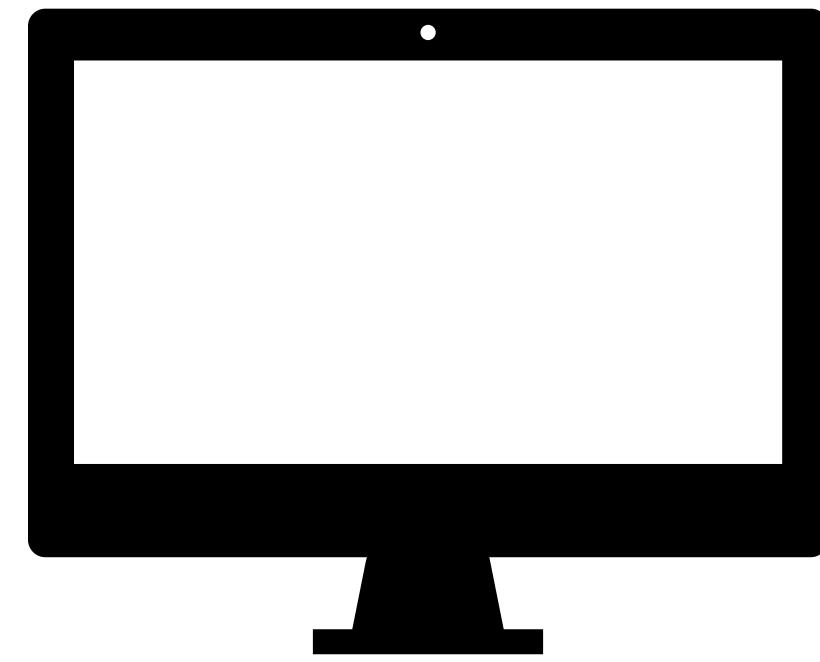
Visualize

Model



Share

https://vac-lshtm.shinyapps.io/ncov_tracker/

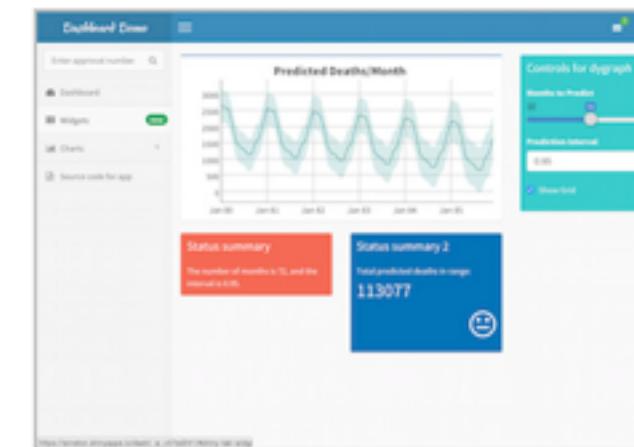


<https://alhill.shinyapps.io/COVID19seir/>

DEMO

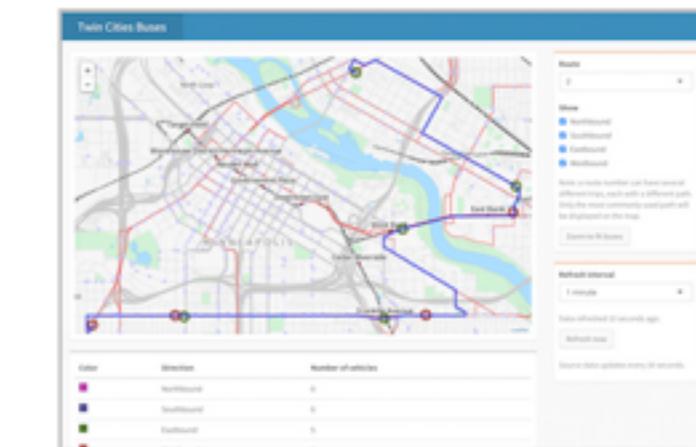


Shiny Apps for the Enterprise



Shiny Dashboard Demo

A dashboard built with Shiny.



Location tracker

Track locations over time with streaming data.



Download monitor

Streaming download rates visualized as a bubble chart.



Supply and Demand

Forecast demand to plan resource allocation.

shiny.rstudio.com/

Industry Specific Shiny Apps



Economic Dashboard

Economic forecasting with macroeconomic indicators.



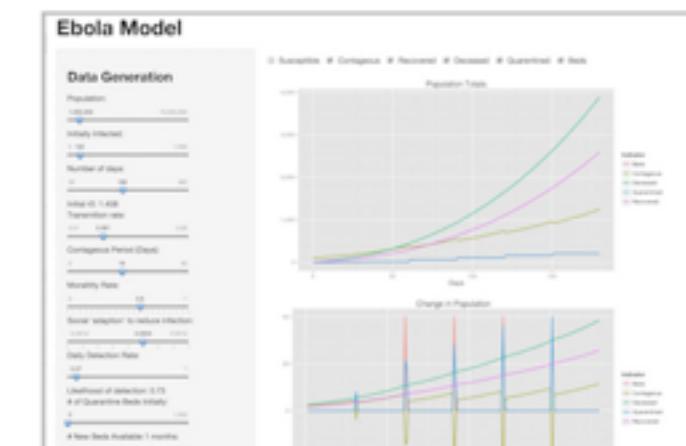
ER Optimization

An app that models patient flow.



CDC Disease Monitor

Alert thresholds and automatic weekly updates.



Ebola Model

An epidemiological simulation.

what is
Reactivity?

Think Excel.

A screenshot of the Microsoft Excel 2010 application window. The title bar reads "Workbook1". The ribbon is visible at the top, showing tabs for Home, Layout, Tables, Charts, SmartArt, Formulas, Data, and Review. The Home tab is selected. The ribbon tabs are "Edit", "Font", "Alignment", "Number", "Format", "Cells", and "Themes". The ribbon icons include Paste, Fill, Calibri (Body), 12pt, Bold (B), Italic (I), Underline (U), Alignment, Wrap Text, General, Conditional Formatting, Styles, Insert, Delete, Format, Themes, and Font Size. The main area shows a blank worksheet with a grid of cells from A1 to Q25. The status bar at the bottom shows "Normal View" and "Ready".

Workbook1

Tables Charts SmartArt Formulas Data Review

Font Alignment Number Format Cells

Body 12 A[▲] A[▼] abc Wrap Text General .00 .00

I *U* Merge Conditional Formatting Styles Insert Delete

C D E F G H I J K L M N O

100

101

Workbook1

Tables Charts SmartArt Formulas Data Review

Font Alignment Number Format Cells

Font: Arial (Body) Size: 12 Bold: A⁺ Italic: I Underline: U Font Color: Yellow Font Style: A₁

Alignment: Center Wrap Text: Wrap Text

Number: General Decimal Places: 0.00 Thousands Separator: , Currency: ₦.00

Format: Conditional Formatting Styles:

Cells: Insert Delete

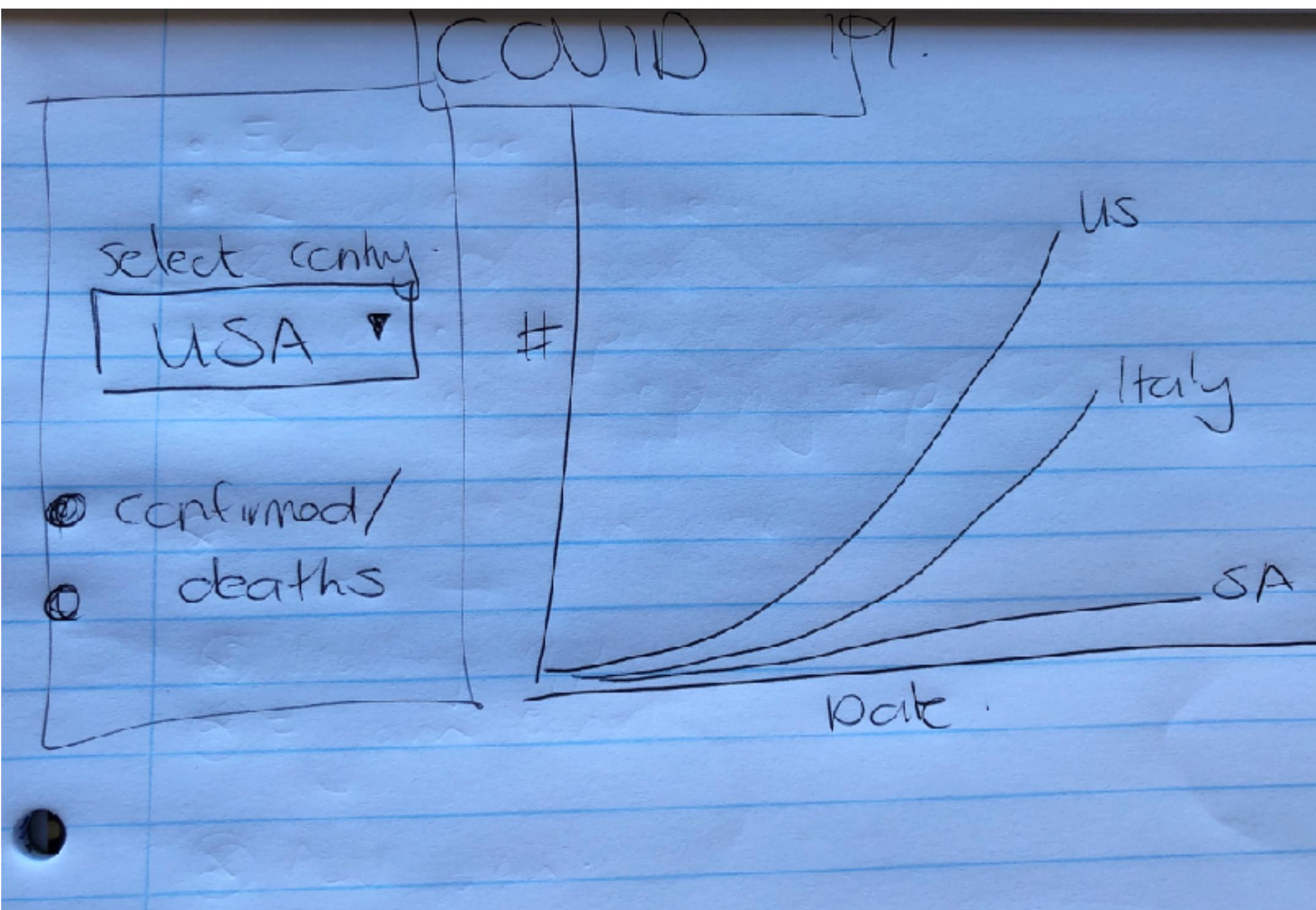
C D E F G H I J K L M N O

999

1000

Where to start:

1) inputs



2) layout

3) content

1) inputs

+

2) layout

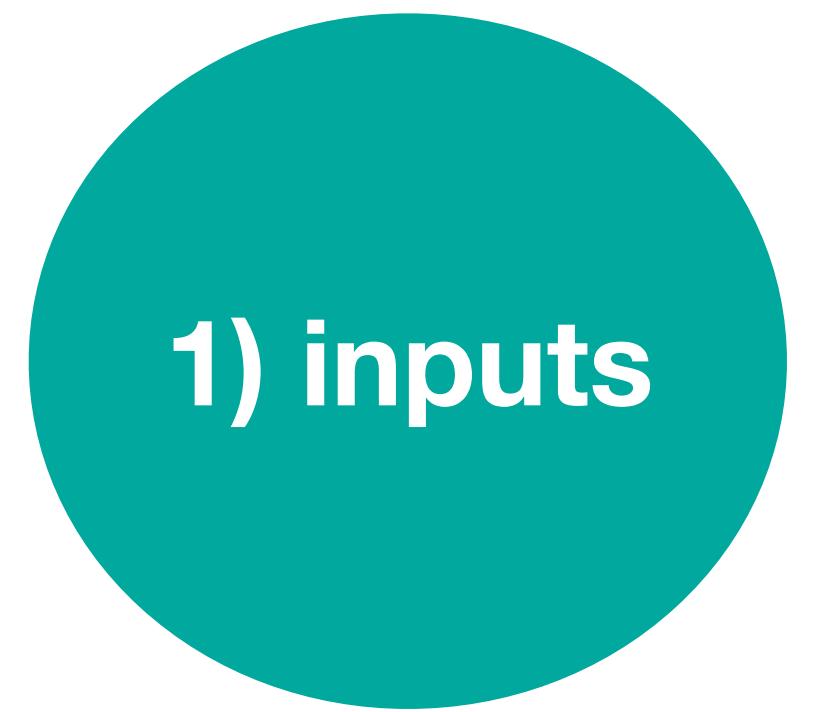
+

3) content

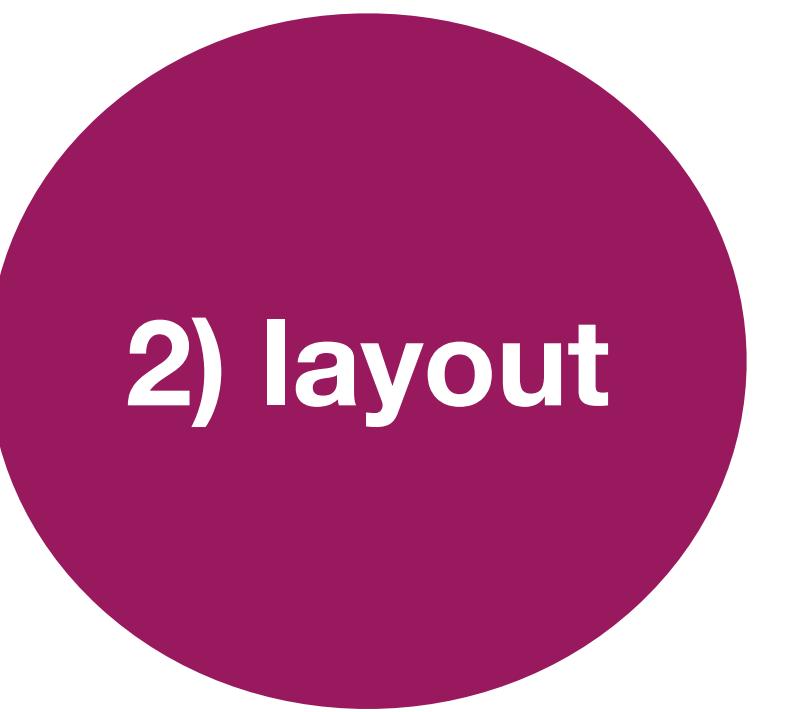
=



Why Shiny is just like sushi ...



+



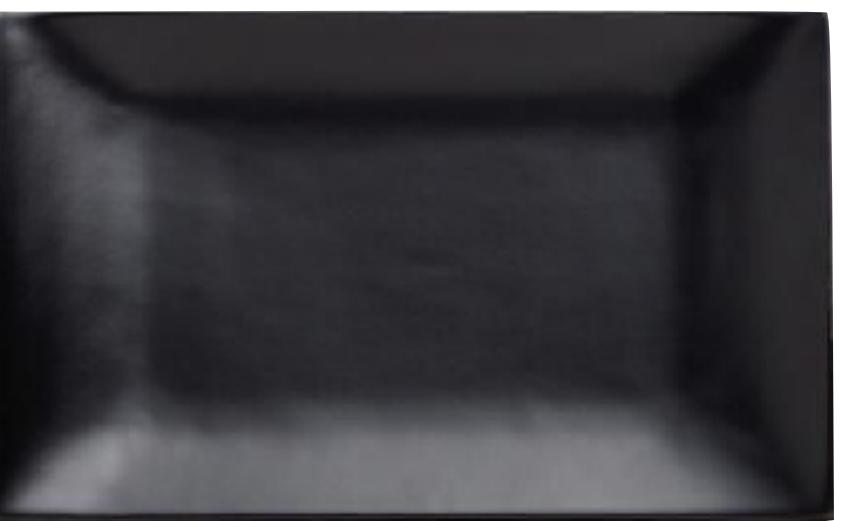
+



=



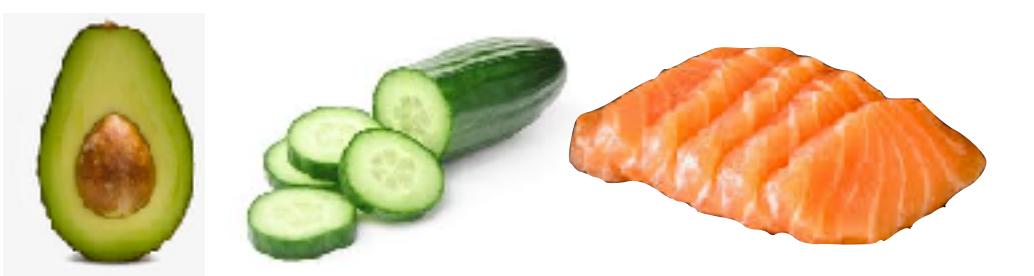
+



+



=



What's in an app?

```
library(shiny)
```

```
ui <- fluidPage()
```

Put a slider input 1:100

Have a heading followed by
a graph and then a table

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

Make an awesome
gg-plot scatterplot

```
shinyApp(ui = ui, server = server)
```

User interface
controls the
options and layout
of app

1) inputs

2) layout

Server function
contains recipes
needed to build visuals

3) content



What's in an app?

```
library(shiny)
```

```
ui <- fluidPage()
```



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



```
shinyApp(ui = ui, server = server)
```

1) inputs

2) layout

3) content



USER INTERFACE (UI)

```
ui <- fluidPage()  
Put a slider input 1:100  
Have a heading followed by  
a graph and then a table
```

User interface
controls the
options and layout
of app

1) inputs

2) layout

Add elements to your app as arguments to
`fluidPage()`

```
ui <- fluidPage(  
  # *Input() functions,  
  # *Output() functions  
)
```

USER INTERFACE (UI)

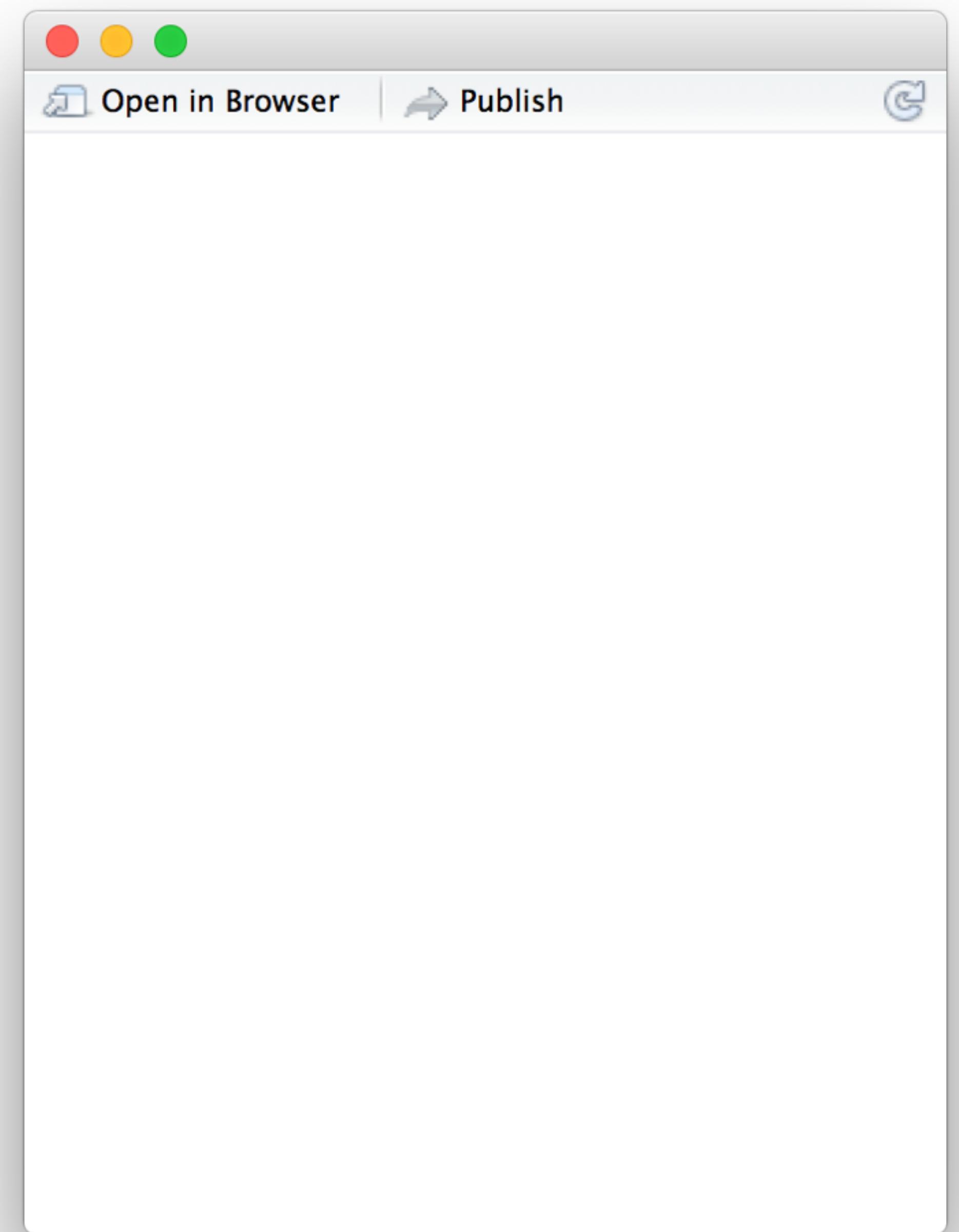
Inputs



Create an input with an input function.

UI

```
library(shiny)  
ui <- fluidPage(  
)  
  
server <- function(input, output) {}  
  
shinyApp(server = server, ui = ui)
```



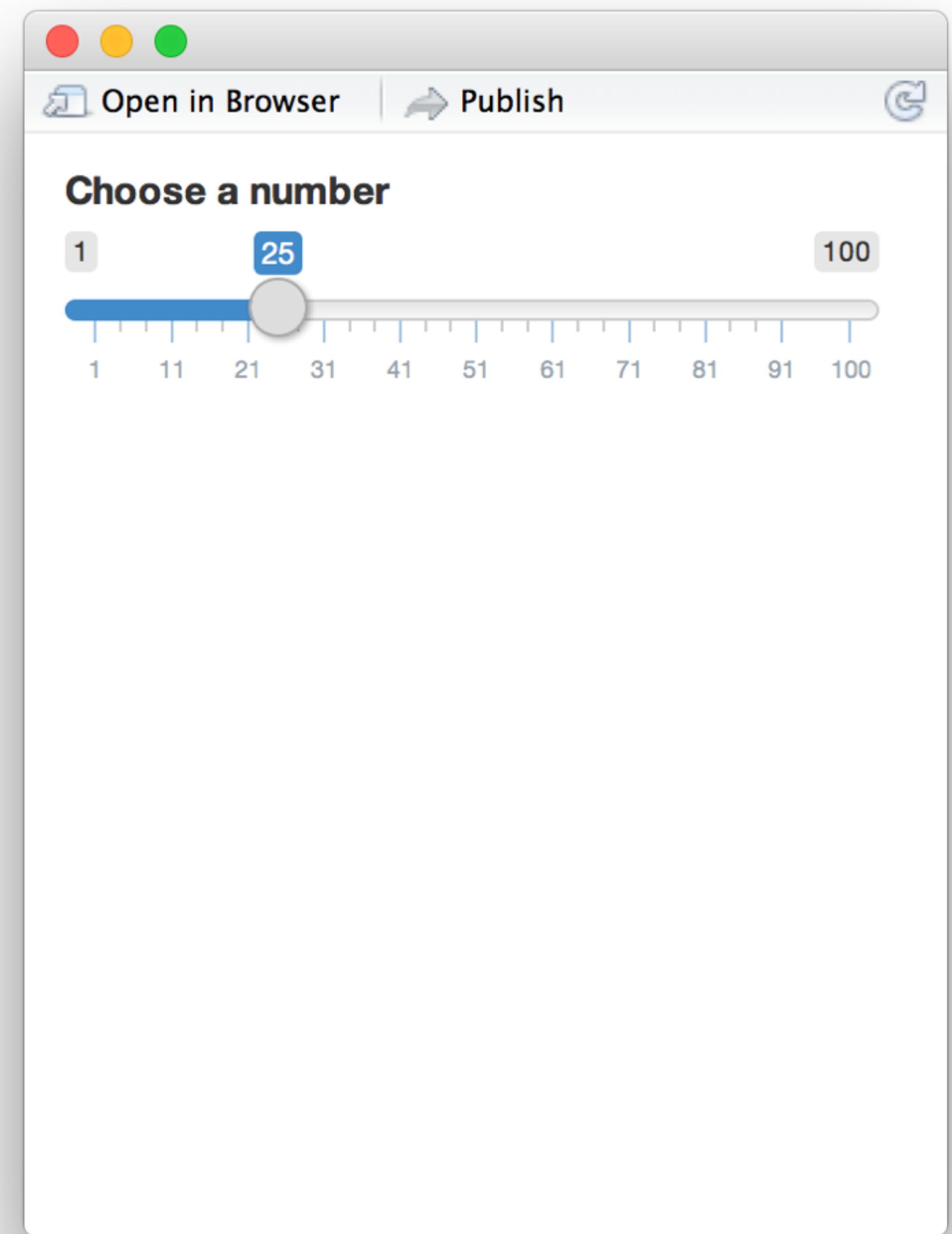
Create an input with an input function.

UI

```
library(shiny)
ui <- fluidPage(
  sliderInput(inputId = "num",
  label = "Choose a number",
  value = 25, min = 1, max = 100)
)
```

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

shinyApp(server = server, ui = ui)
```



Basic widgets

Buttons

Action

Submit

Date range

2017-06-21 to 2017-06-21

Radio buttons

Choice 1

Choice 2

Choice 3

Single checkbox

Choice A

File input

Browse... No file selected

Select box

Choice 1

Checkbox group

Choice 1

Choice 2

Choice 3

Date input

2014-01-01

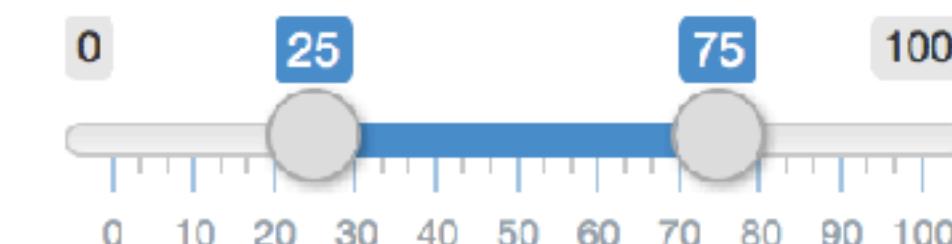
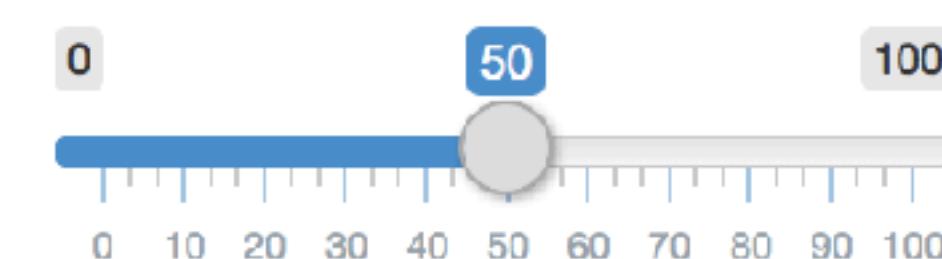
Help text

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

Numeric input

1

Sliders



Text input

Enter text...

USER INTERFACE (UI)

Outputs



Need to specify the type of output

Function	Inserts
dataTableOutput()	an interactive table
htmlOutput()	raw HTML
imageOutput()	image
plotOutput()	plot
tableOutput()	table
textOutput()	text
uiOutput()	a Shiny UI element
verbatimTextOutput()	text
plotlyOutput ()	plotlyOutput ()

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {}

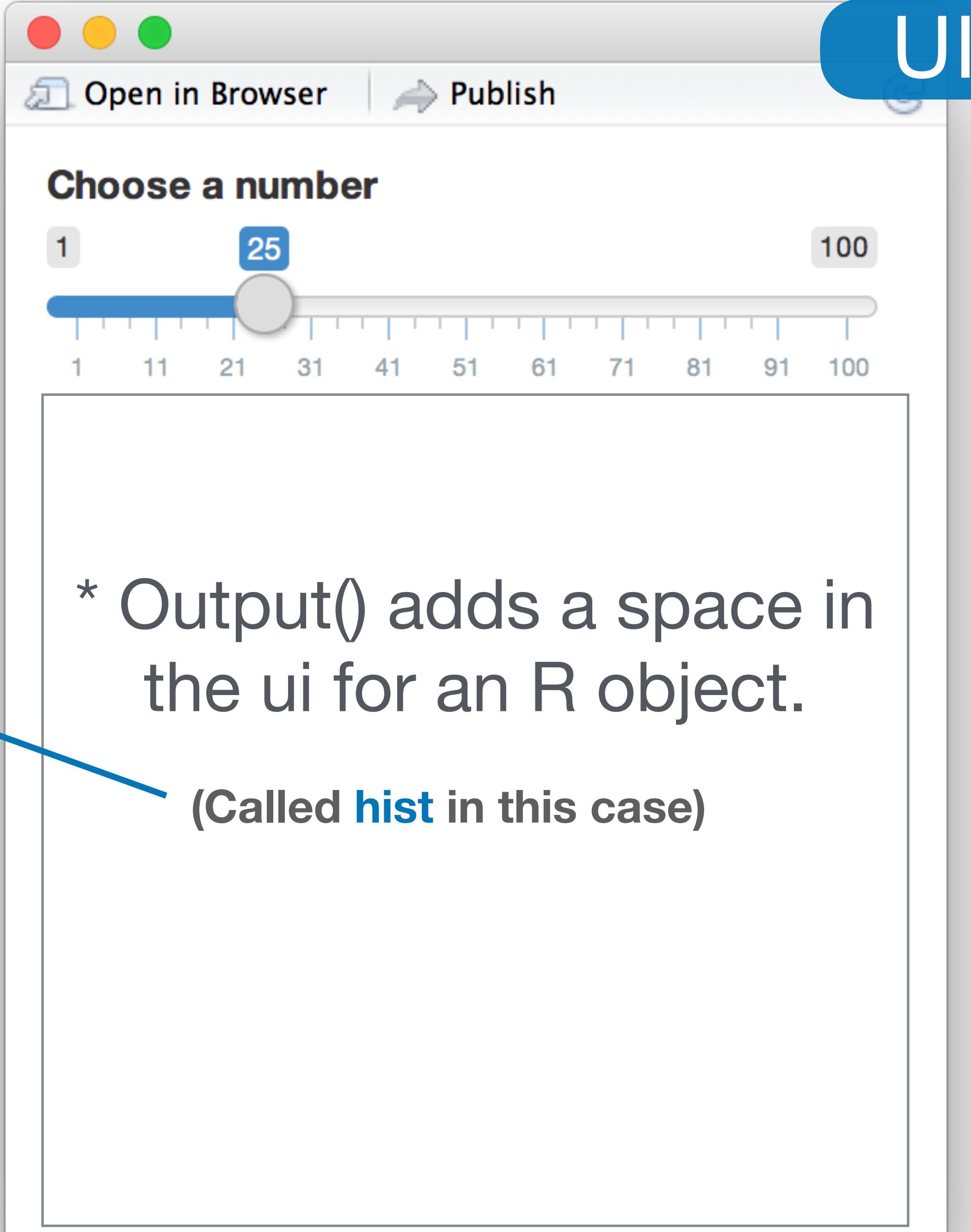
shinyApp(ui = ui, server = server)
```

Comma between
arguments

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

UI



The screenshot shows a Shiny application window. At the top, there are three buttons: a red circle, a yellow circle, and a green circle. Below them are two links: "Open in Browser" and "Publish". The main content area has a title "Choose a number". Below the title is a slider with a blue handle set to the value 25. The slider has numerical labels at 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, and 100. Below the slider is a large, empty rectangular box with a thin gray border. A blue arrow points from the text "plotOutput("hist")" in the R code to the top-left corner of this empty box. Inside the box, there is explanatory text: "* Output() adds a space in the ui for an R object. (Called **hist** in this case)".

Choose a number

1 11 21 31 41 51 61 71 81 91 100

* Output() adds a space in the ui for an R object.
(Called **hist** in this case)

Tell the **SERVER**

How to assemble
inputs into the outputs



In 3 steps ...

1

Save objects to display to output\$

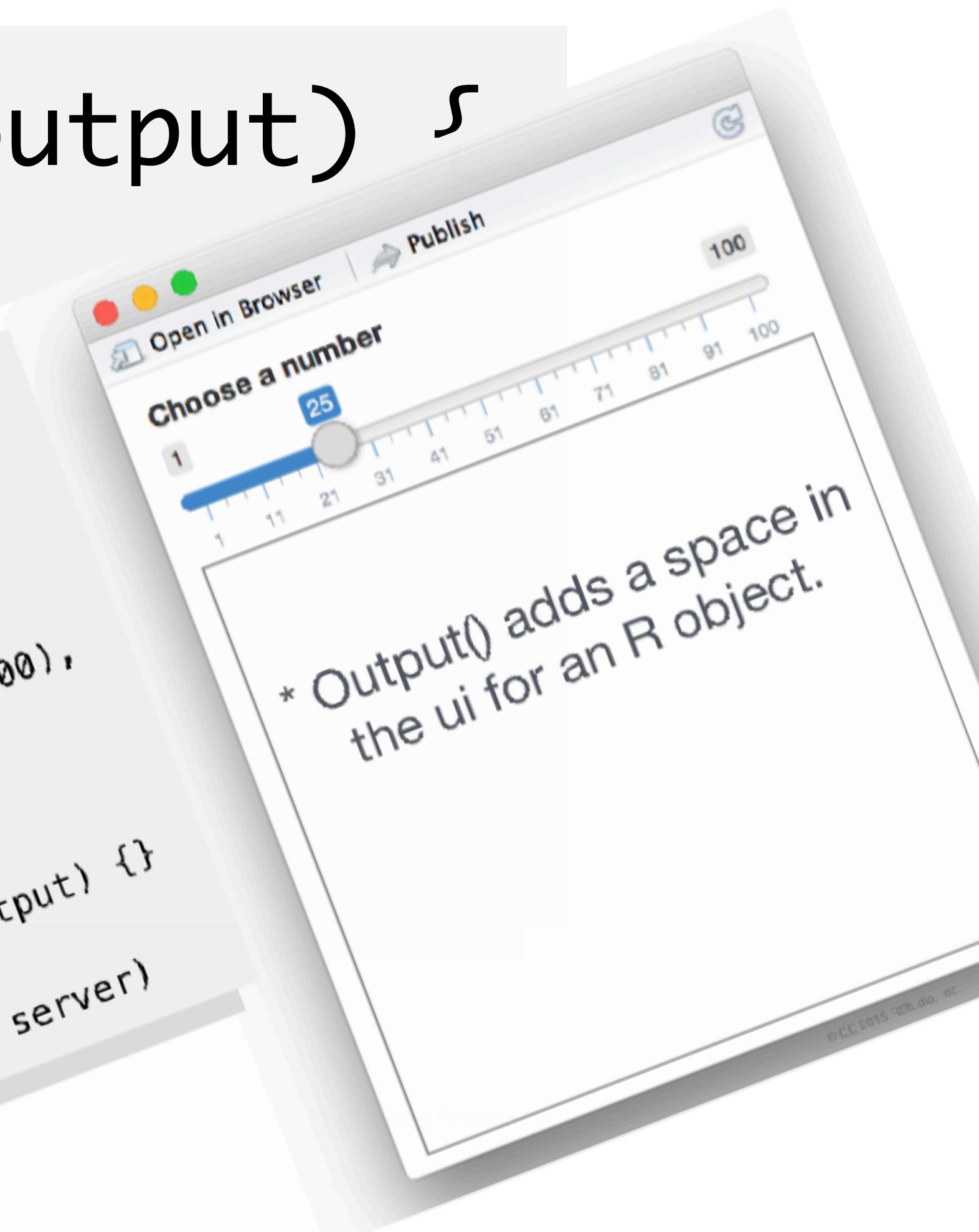
```
server <- function(input, output) {  
  output$hist <- # code  
}
```

1

Save objects to display to output\$

```
server <- function(input, output) {  
  output$hist <- # code  
}
```

```
library(shiny)  
ui <- fluidPage(  
  sliderInput(inputId = "num",  
    label = "Choose a number",  
    value = 25, min = 1, max = 100),  
  plotOutput("hist"))  
server <- function(input, output) {}  
shinyApp(ui = ui, server = server)
```



2 Build objects to display with `render*`()

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    })  
}
```

2 Build objects to display with `render*`()

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
  })  
}
```

```
library(shiny)  
ui <- fluidPage(  
  sliderInput(inputId = "num",  
    label = "Choose a number",  
    value = 25, min = 1, max = 100,  
    plotOutput("hist"))  
)  
server <- function(input, output) {}  
shinyApp(ui = ui, server = server)
```

* `Output()` adds a space in
the `ui` for an R object.

2

Use the **render***() function that creates the type of output you wish to make.

function	creates
renderDataTable()	An interactive table (from a data frame, matrix, or other table-like structure)
renderImage()	An image (saved as a link to a source file)
renderPlot()	A plot
renderPrint()	A code block of printed output
renderTable()	A table (from a data frame, matrix, or other table-like structure)
renderText()	A character string
renderUI()	a Shiny UI element

3

Build reactive output with linked inputs - “input\$...”

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    hist(rnorm(input$num))  
  })  
}
```

```
library(shiny)  
ui <- fluidPage(  
  sliderInput(inputId = "num",  
              label = "Choose a number",  
              value = 25, min = 1, max = 100)  
)  
server <- function(input, output) {}  
shinyApp(server = server, ui = ui)
```

3

Build reactive output with linked inputs - “input\$...”

```
renderPlot({ hist(rnorm(input$num))})
```



type of object to build



code block that builds the object

Reactivity 101

Reactivity automatically occurs whenever you use an input value to render an output object

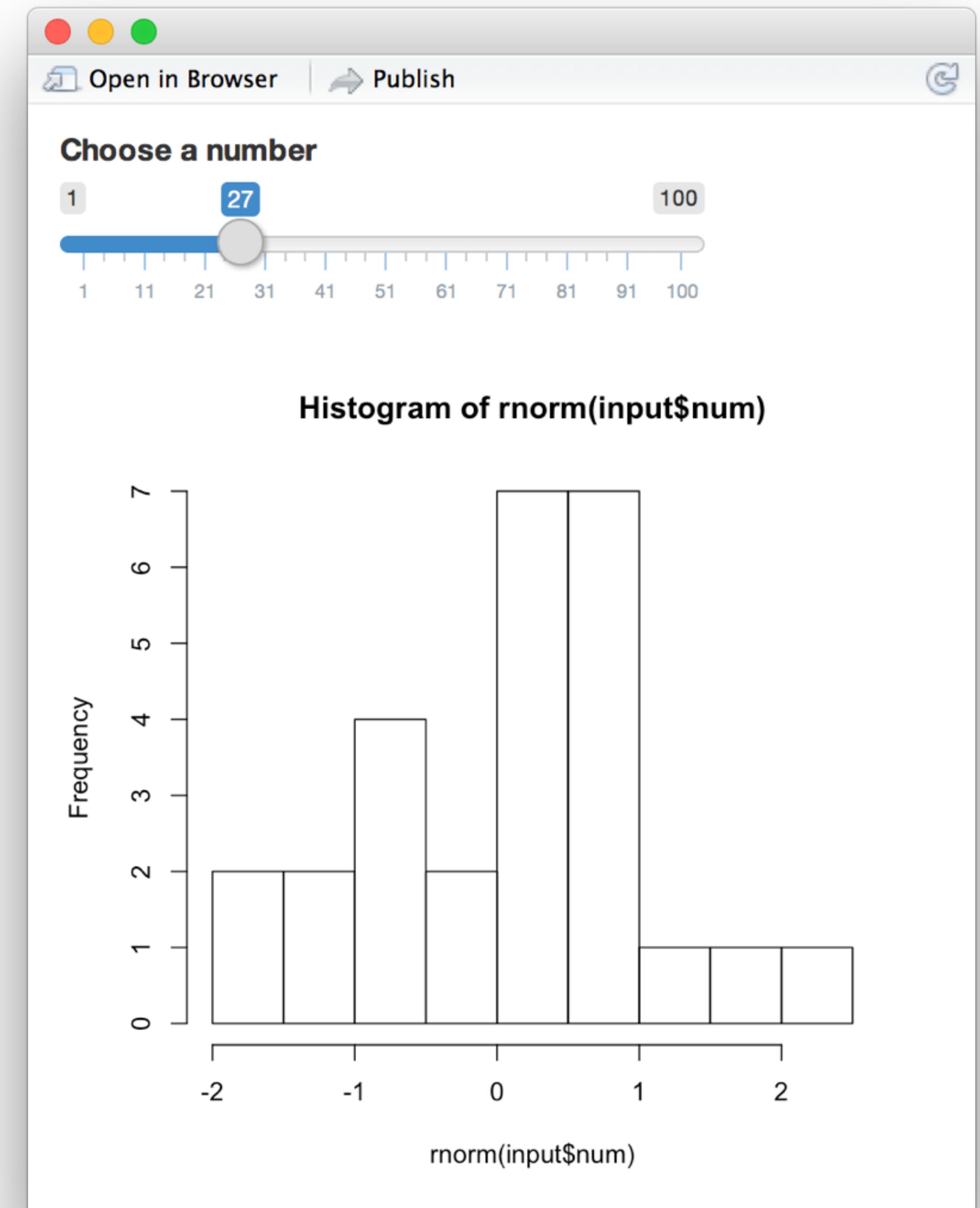
```
function(input, output) {  
  output$hist <- renderPlot({  
    hist(rnorm(input$num))  
  })  
}
```

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}

shinyApp(ui = ui, server = server)
```



How to learn

- If you build it ... you will learn
- RStudio Shiny website has guided learning modules (slow but thorough)
- Learn as you need ... LMGTFY
- Take an example off the web, play around with it, change input types, elements etc ...

Credit to RStudio for slide material



Ways to make your
app

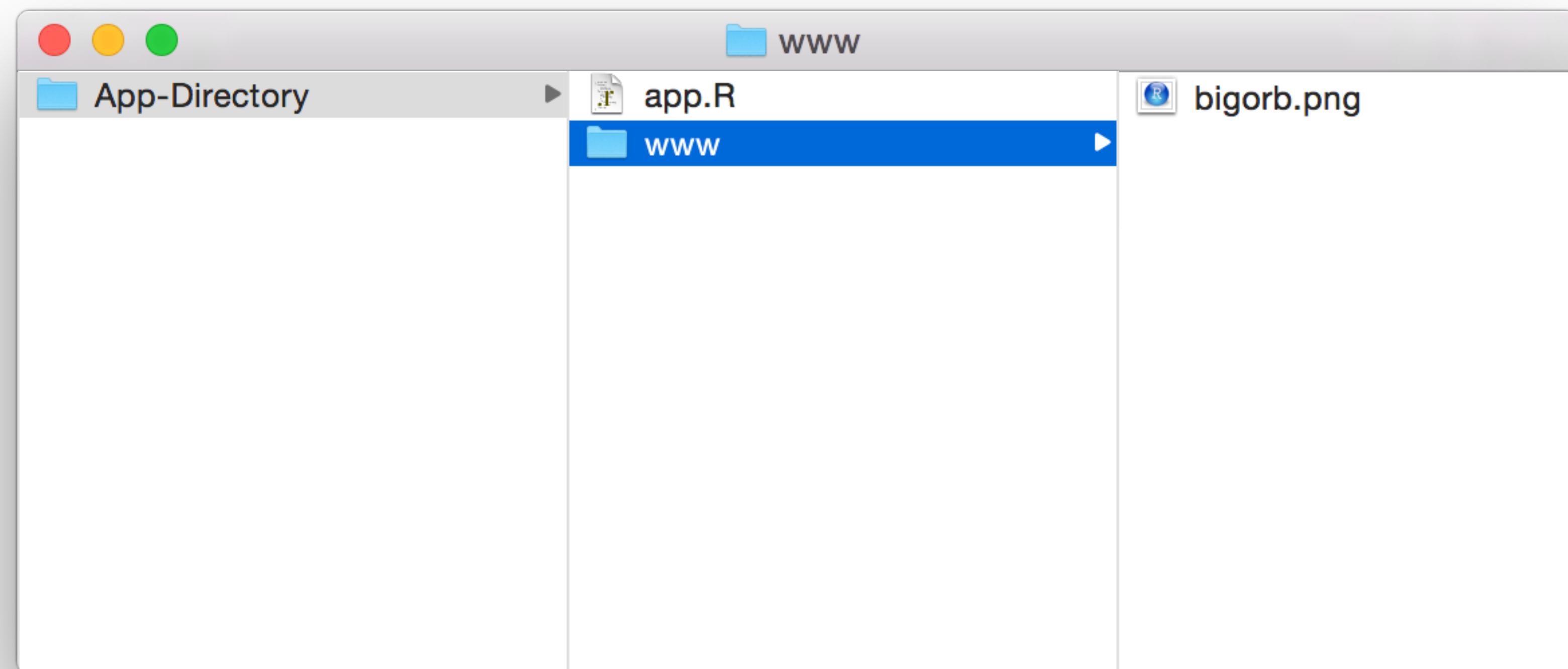
More ... pretty

Add some text - or images?

Function	Creates
a()	A Hyperlink
br()	A line break
code()	Text formatted like computer code
em()	Italicized (emphasized) text
h1(),h2(),h3(),h4(),h5(),h6()	Headers (First level to sixth)
hr()	A horizontal rule (line)
img()	An image
p()	A new paragraph
strong()	Bold (strong) text

Adding Images

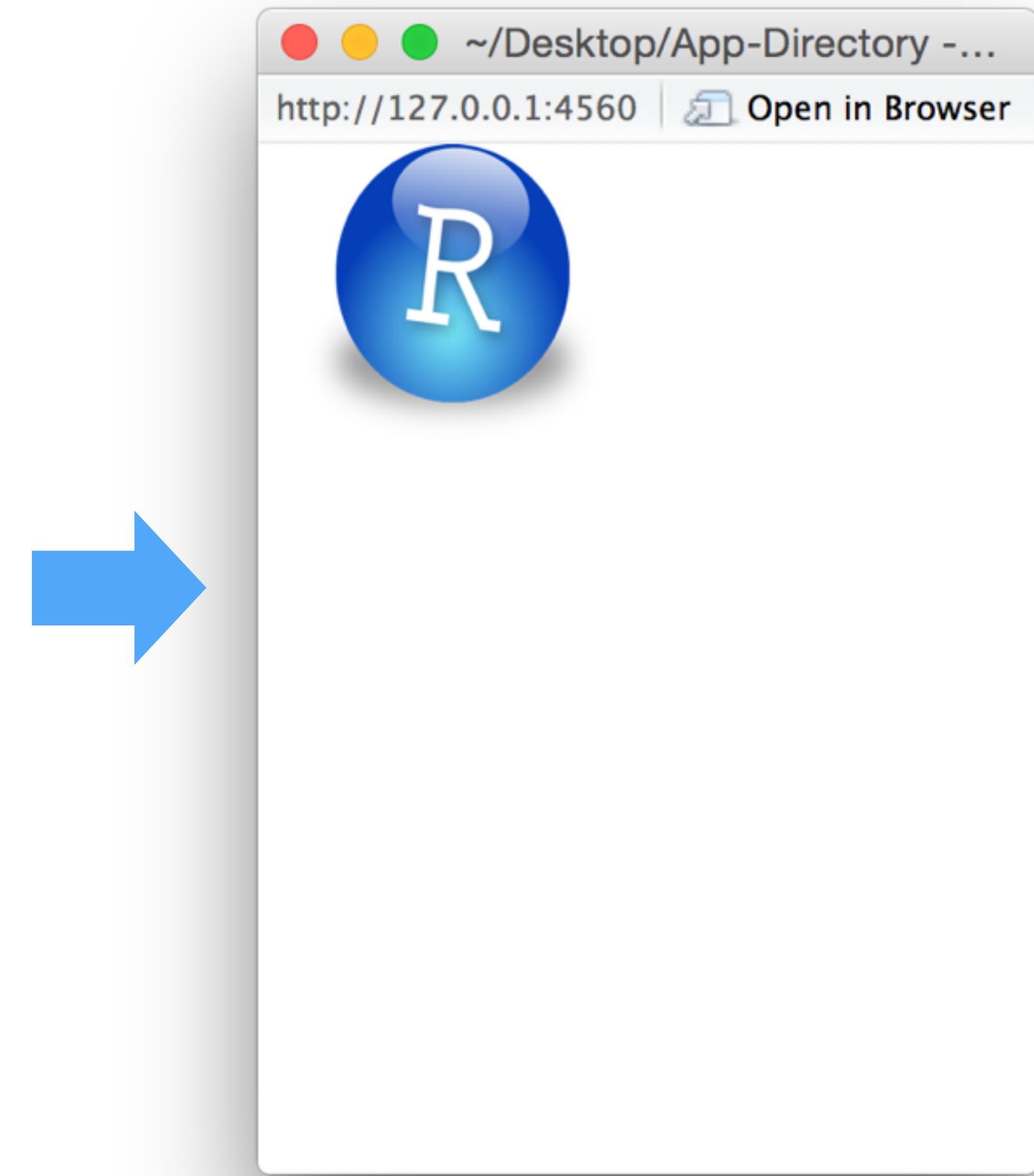
To add an image from a file, save the file in a subdirectory named **www**



img()

Add an img from a file in www

```
fluidPage(  
  img(  
    height = 100,  
    width = 100,  
    src = "bigorb.png")  
)
```

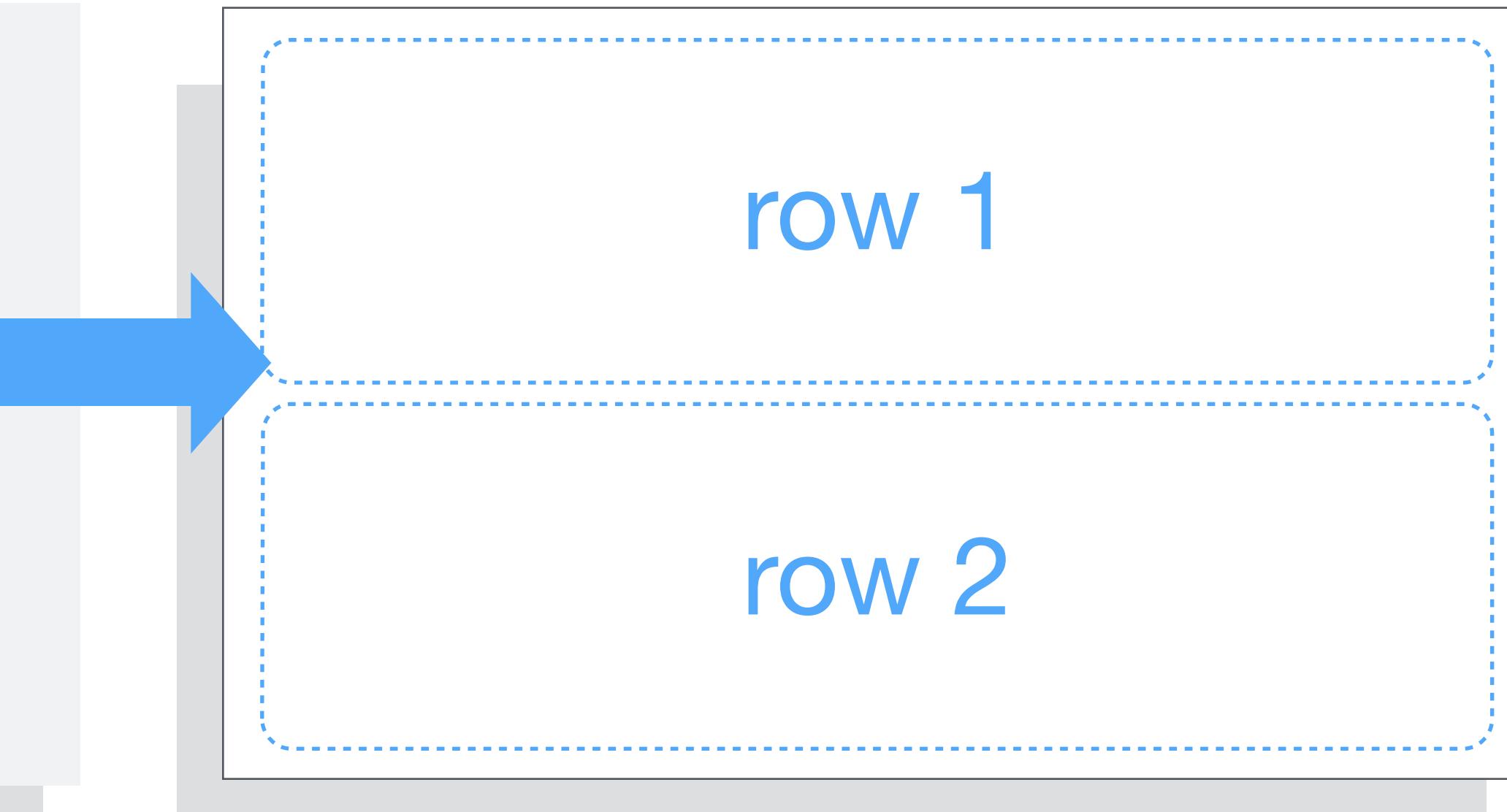


Change the layout

fluidRow()

`fluidRow()` adds rows to the grid. Each new row goes below the previous rows.

```
ui <- fluidPage(  
  fluidRow(),  
  fluidRow()  
)
```

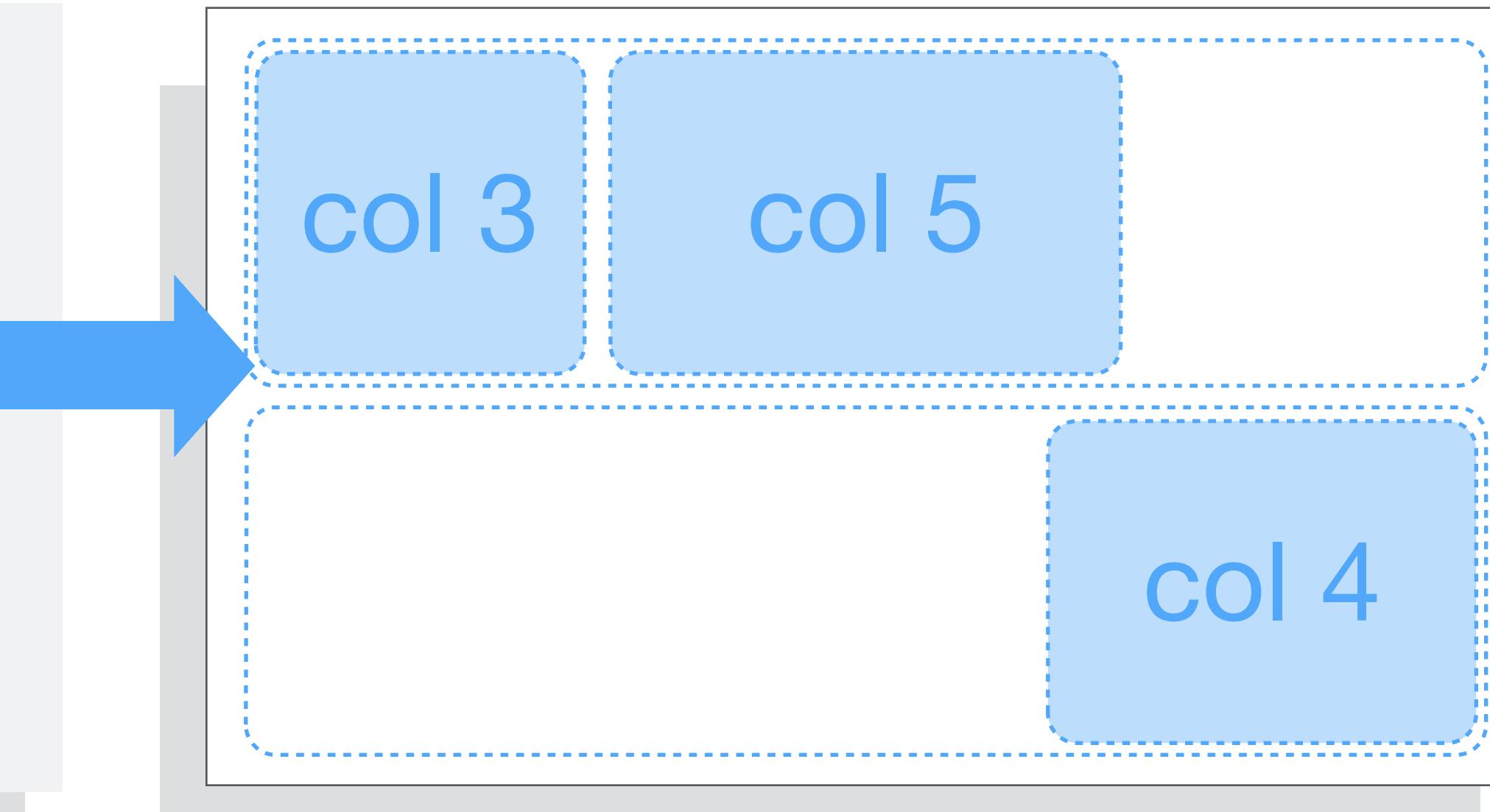


column()

column() adds columns within a row. Each new column goes to the left of the previous column.

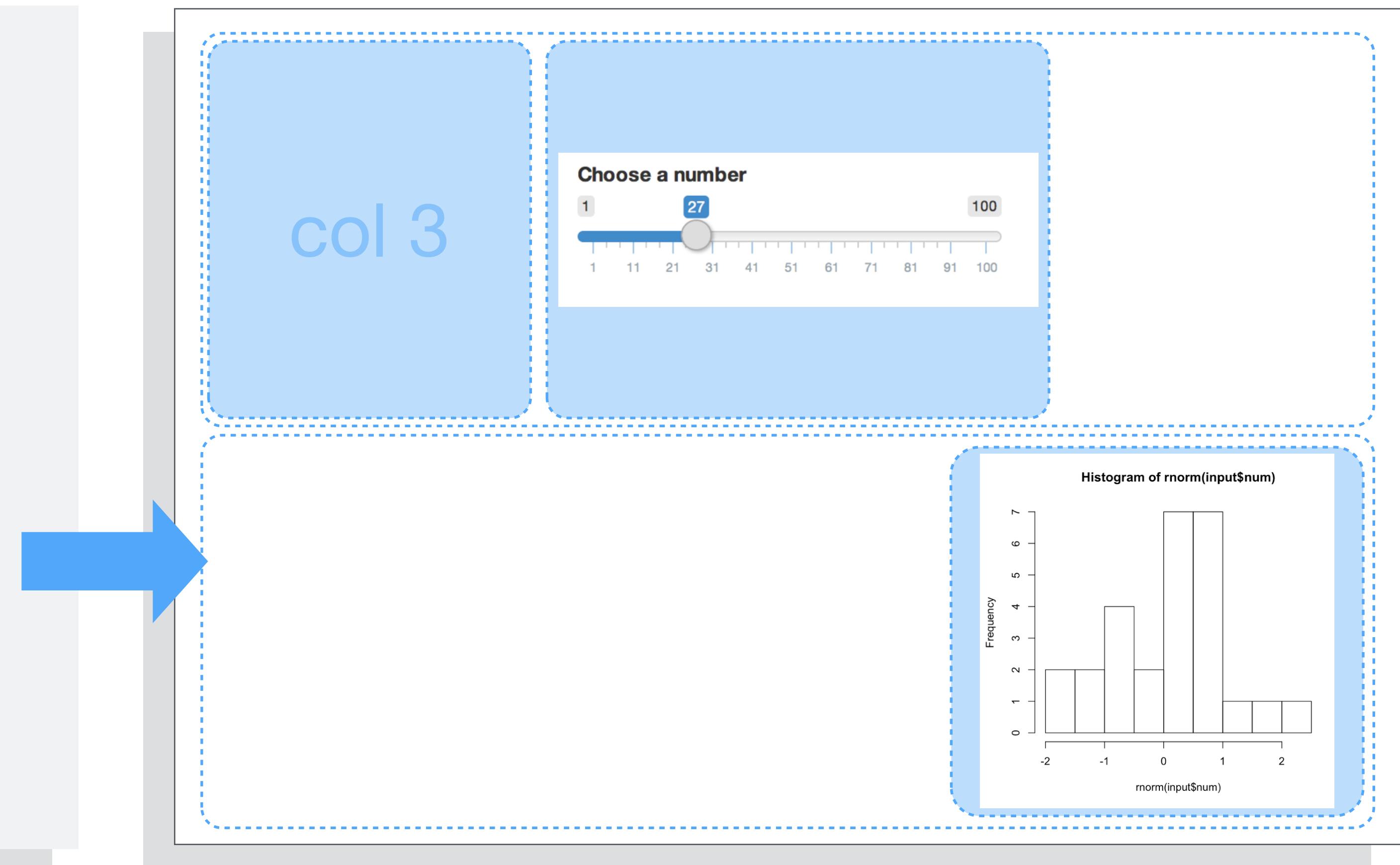
Specify the **width** and **offset** of each column out of 12

```
ui <- fluidPage(  
  fluidRow(  
    column(3),  
    column(5)),  
  fluidRow(  
    column(4, offset = 8))
```



To place an element in the grid, call it as an argument of a layout function

```
fluidPage(  
  fluidRow(  
    column(3),  
    column(5, sliderInput(...))  
  ),  
  fluidRow(  
    column(4, offset = 8,  
      plotOutput("hist")  
    )  
  )  
)
```

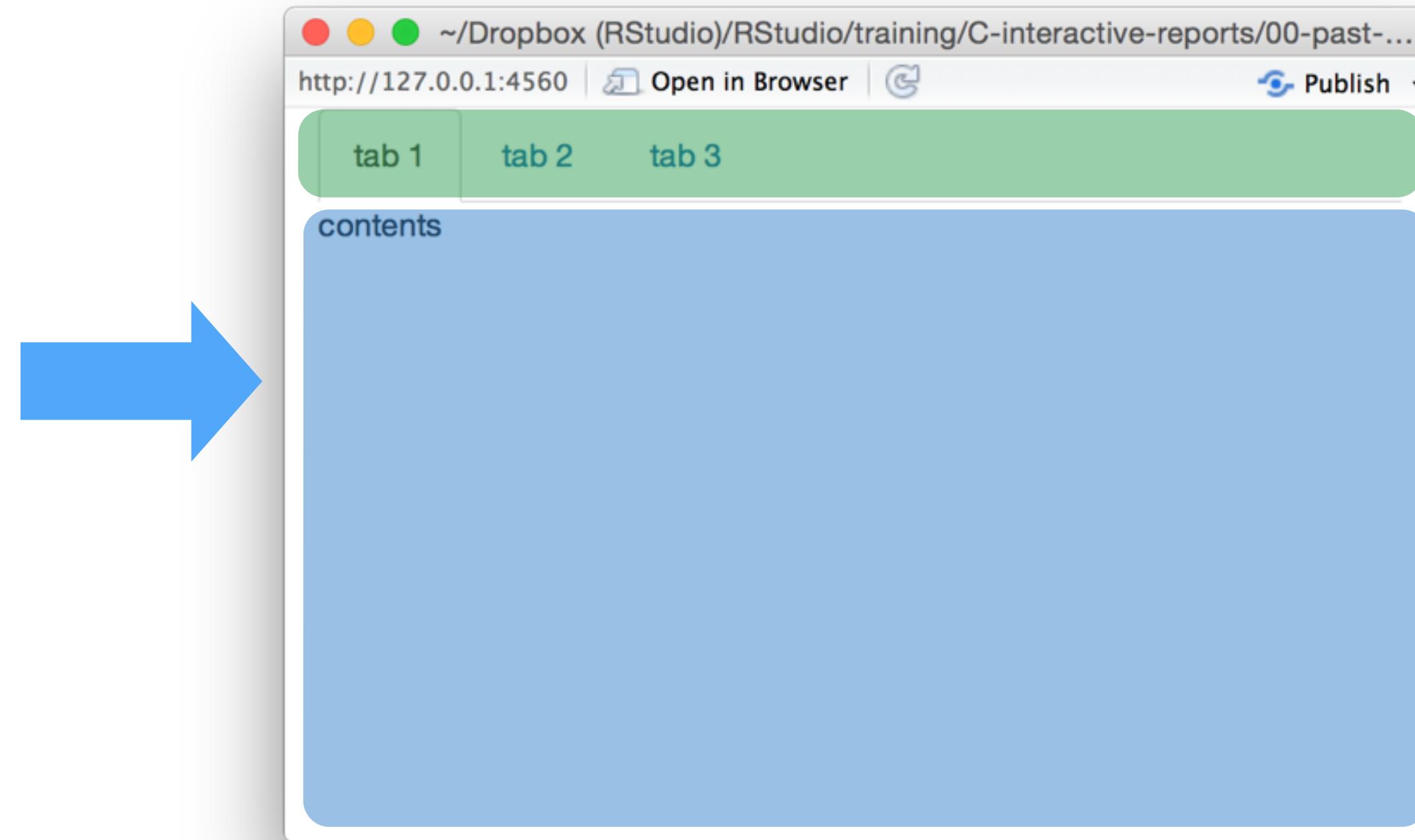


You could also add tabs if you have a whole lot of visuals

tabsetPanel()

`tabsetPanel()` combines tabs into a single *panel*.
Use *tabs* to navigate between tabs.

```
fluidPage(  
  tabsetPanel(  
    tabPanel("tab 1", "contents"),  
    tabPanel("tab 2", "contents"),  
    tabPanel("tab 3", "contents")  
  )  
)
```



absolutePanel()

Panel position set rigidly (absolutely), not fluidly

conditionalPanel()

A JavaScript expression determines whether panel is visible or not.

fixedPanel()

Panel is fixed to browser window and does not scroll with the page

headerPanel()

Panel for the app's title, used with pageWithSidebar()

inputPanel()

Panel with grey background, suitable for grouping inputs

mainPanel()

Panel for displaying output, used with pageWithSidebar()

navlistPanel()

Panel for displaying multiple stacked tabPanels(). Uses sidebar navigation

sidebarPanel()

Panel for displaying a sidebar of inputs, used with pageWithSidebar()

tabPanel()

Stackable panel. Used with navlistPanel() and tabsetPanel()

tabsetPanel()

Panel for displaying multiple stacked tabPanels(). Uses tab navigation

titlePanel()

Panel for the app's title, used with pageWithSidebar()

wellPanel()

Panel with grey background.

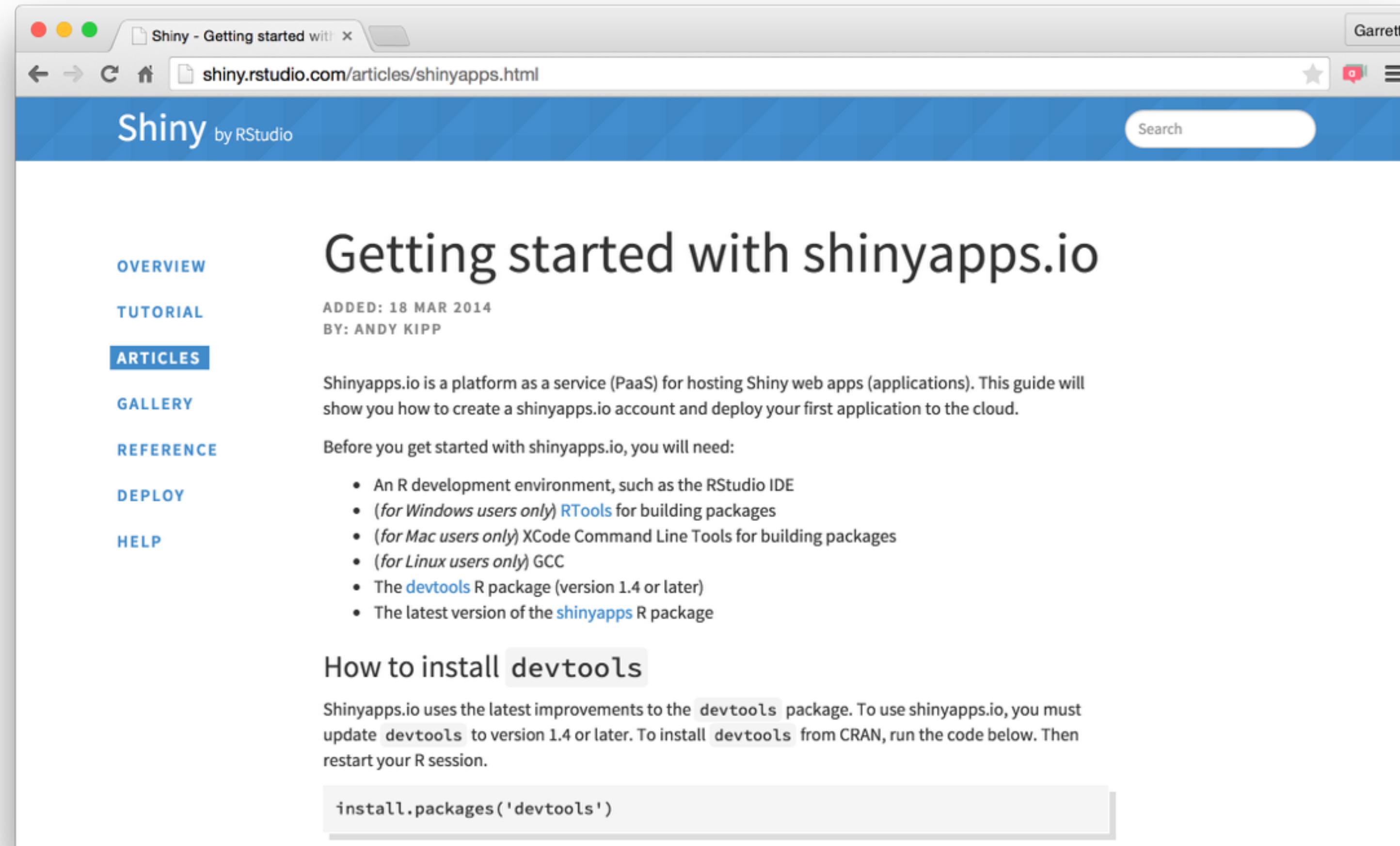
Share your app

Use
shinyapps.io



Getting started guide

shiny.rstudio.com/articles/shinyapps.html



The screenshot shows a web browser window with the title 'Shiny - Getting started with shinyapps.io'. The address bar contains the URL 'shiny.rstudio.com/articles/shinyapps.html'. The page itself is titled 'Getting started with shinyapps.io' and includes a sidebar with links for 'OVERVIEW', 'TUTORIAL', 'ARTICLES' (which is highlighted in blue), 'GALLERY', 'REFERENCE', 'DEPLOY', and 'HELP'. The main content area starts with a brief description of Shinyapps.io as a PaaS for hosting Shiny web apps, followed by a list of requirements for getting started, and a section on how to install the `devtools` package.

Shinyapps.io is a platform as a service (PaaS) for hosting Shiny web apps (applications). This guide will show you how to create a shinyapps.io account and deploy your first application to the cloud.

Before you get started with shinyapps.io, you will need:

- An R development environment, such as the RStudio IDE
- *(for Windows users only)* [RTools](#) for building packages
- *(for Mac users only)* XCode Command Line Tools for building packages
- *(for Linux users only)* GCC
- The [devtools](#) R package (version 1.4 or later)
- The latest version of the [shinyapps](#) R package

How to install `devtools`

Shinyapps.io uses the latest improvements to the `devtools` package. To use shinyapps.io, you must update `devtools` to version 1.4 or later. To install `devtools` from CRAN, run the code below. Then restart your R session.

```
install.packages('devtools')
```

Credit to RStudio for slide material