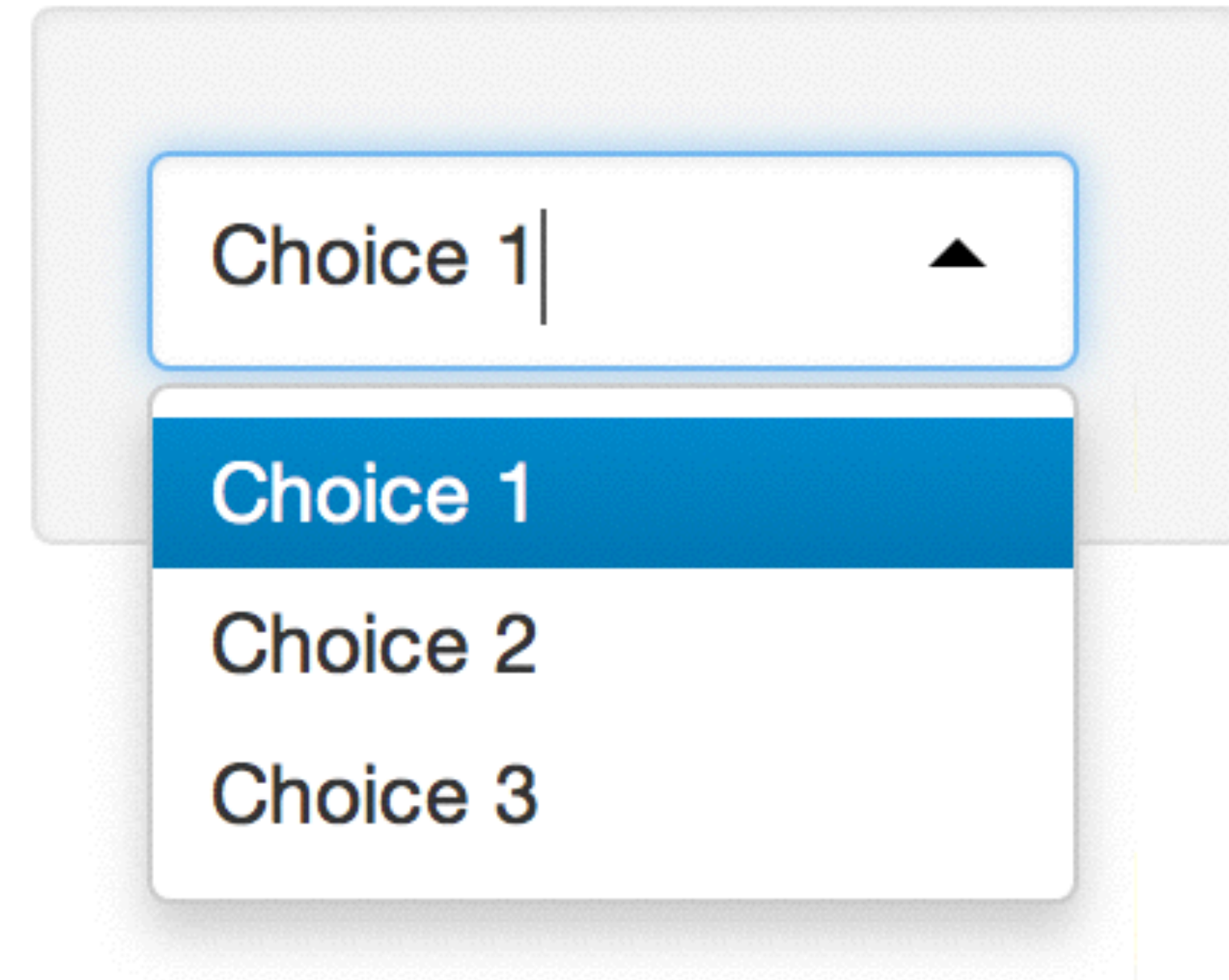


All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

The Training Materials are licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

How to start with Shiny, Part 1

How to build a Shiny App

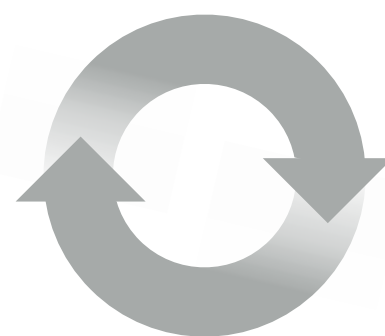


Garrett Golemund

Data Scientist and Master Instructor
May 2015
Email: garrett@rstudio.com

Code and slides at:
bit.ly/shiny-quickstart-1

How to start with Shiny



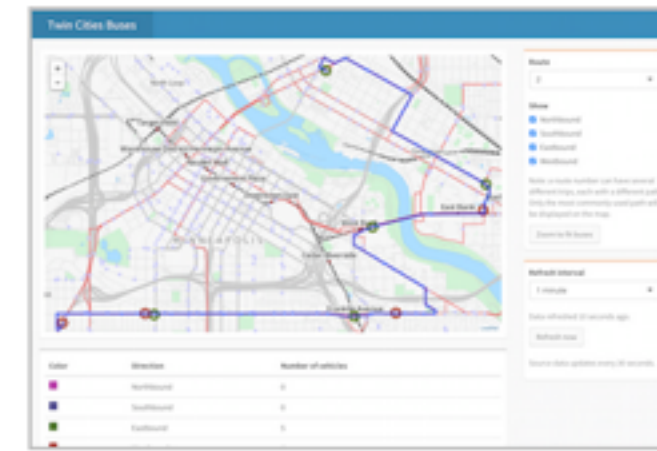
1. How to build a Shiny app (Today)
2. How to customize reactions (May 27)
3. How to customize appearance (June 3)

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 Showcase

www.rstudio.com/products/shiny/shiny-user-showcase/

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.

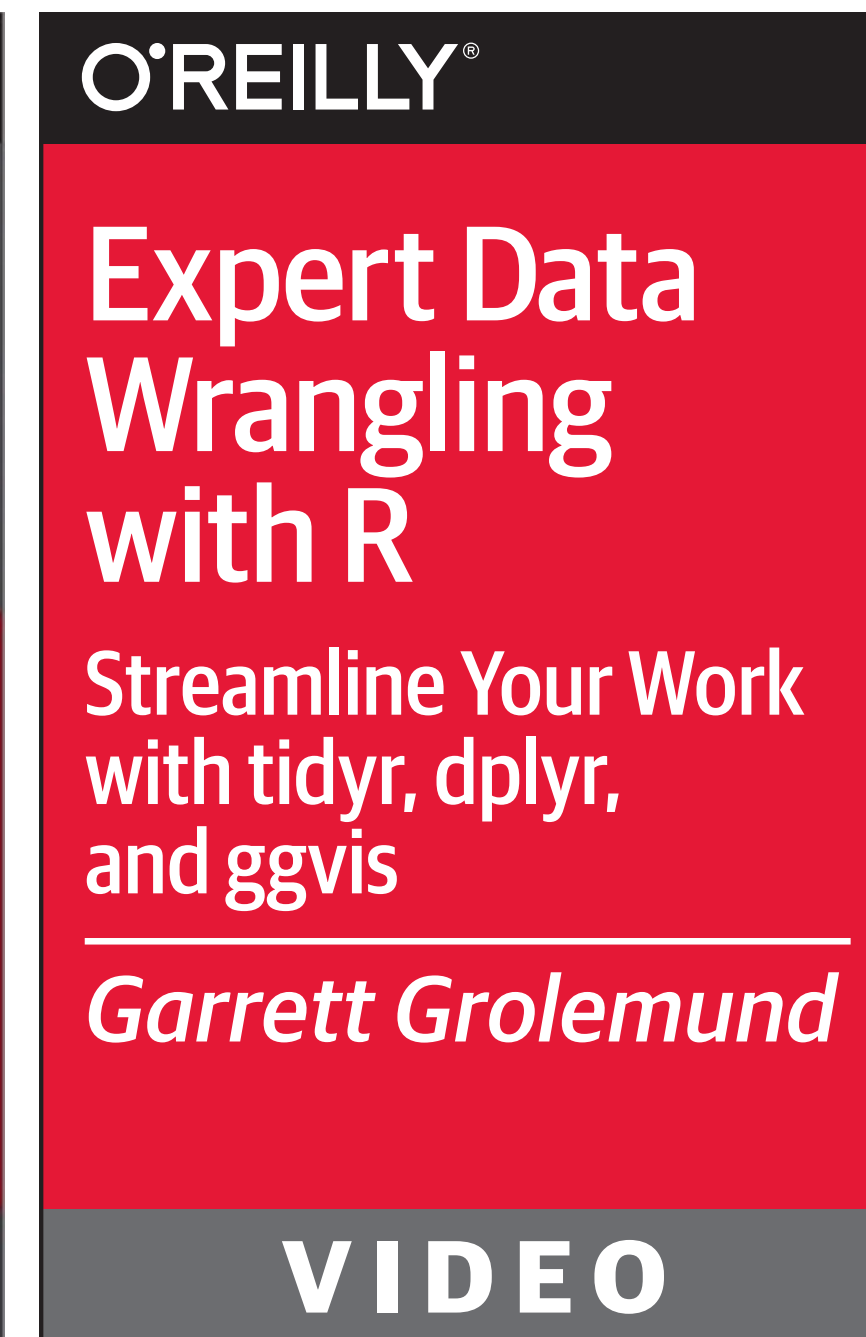
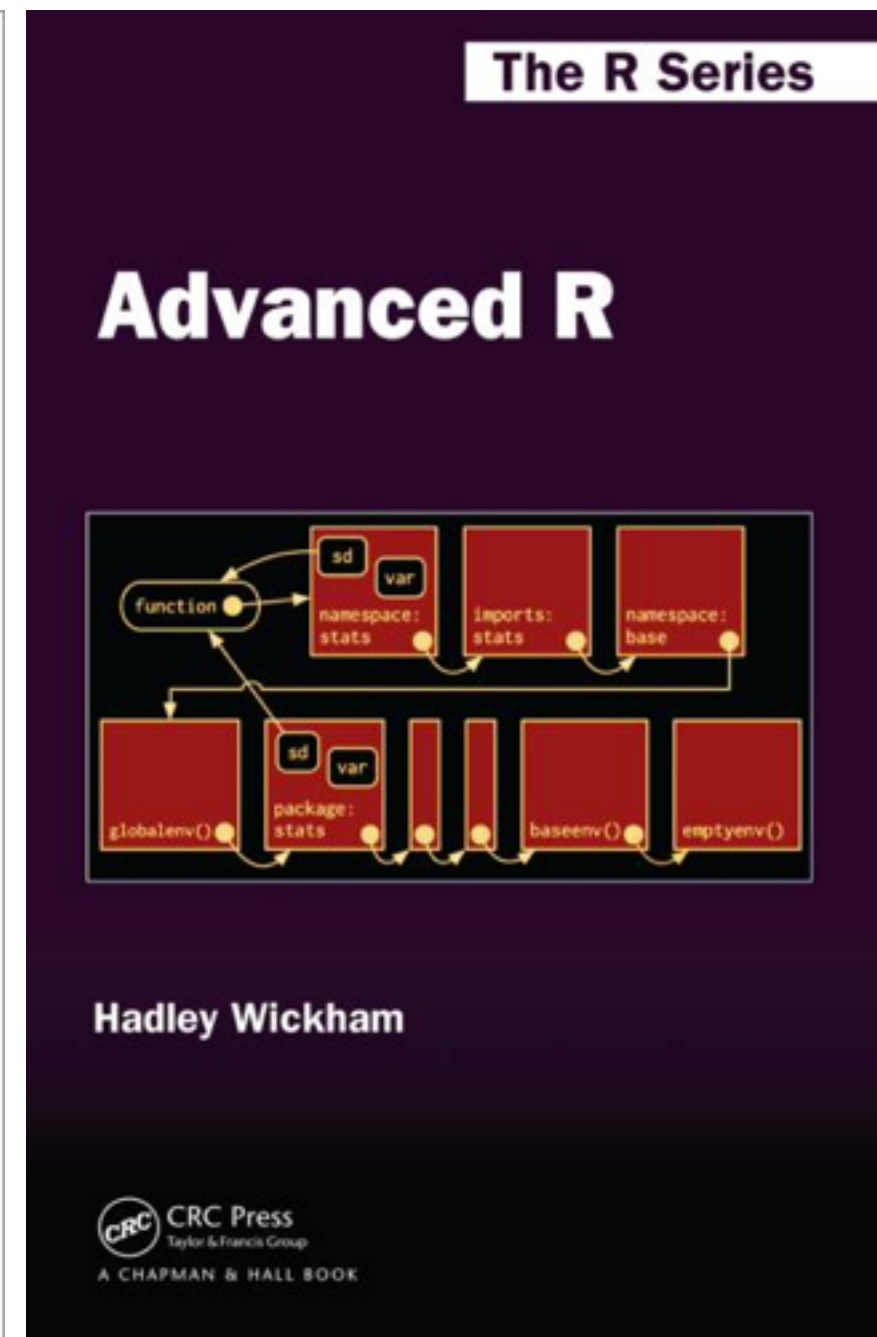
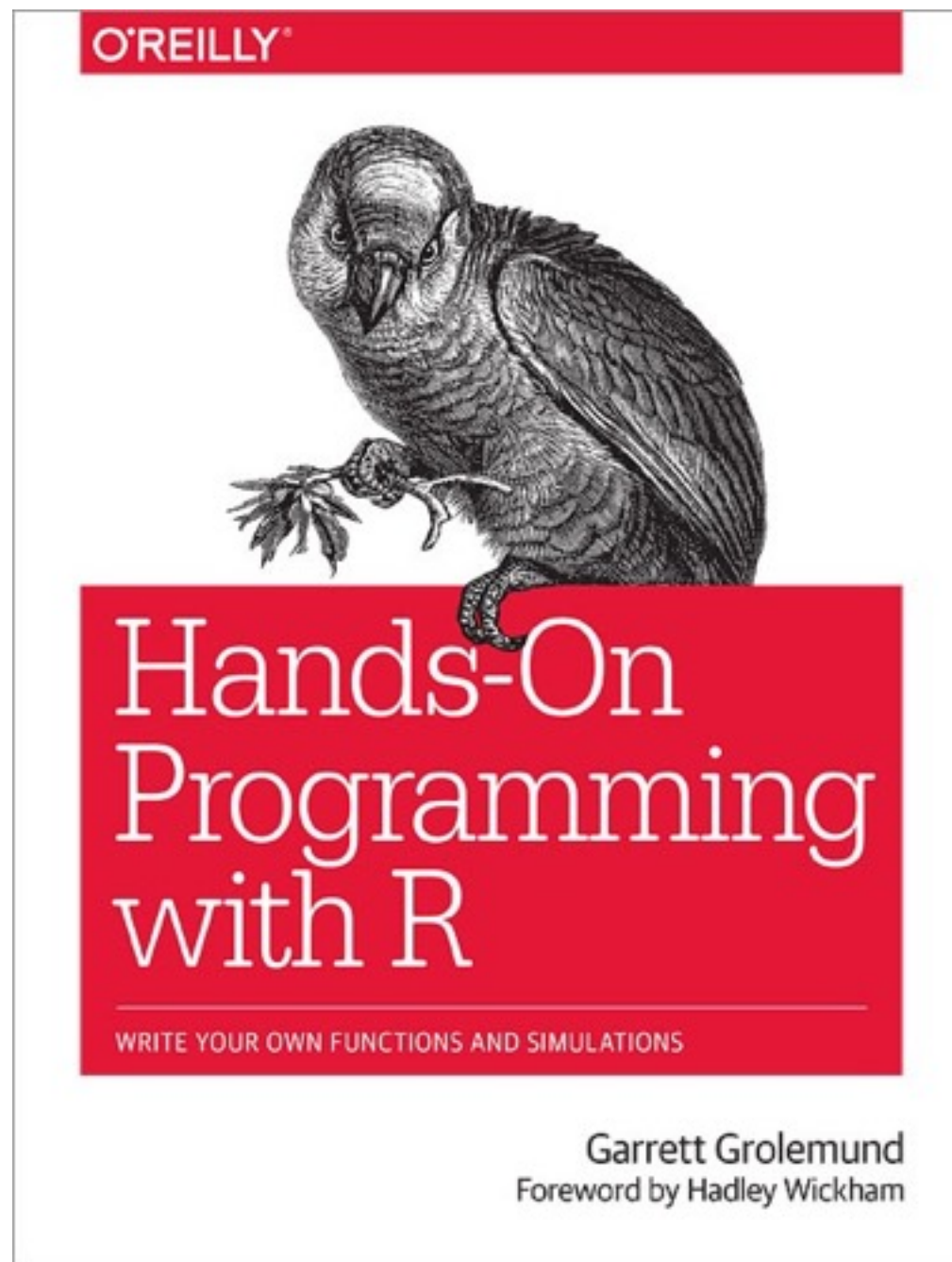


Learn R

Books

Videos

Interactive tutorials



DataCamp

www.datacamp.com

shop.oreilly.com/product/0636920028574.do

adv-r.had.co.nz/

shop.oreilly.com/product/0636920034834.do

shop.oreilly.com/product/0636920035992.do

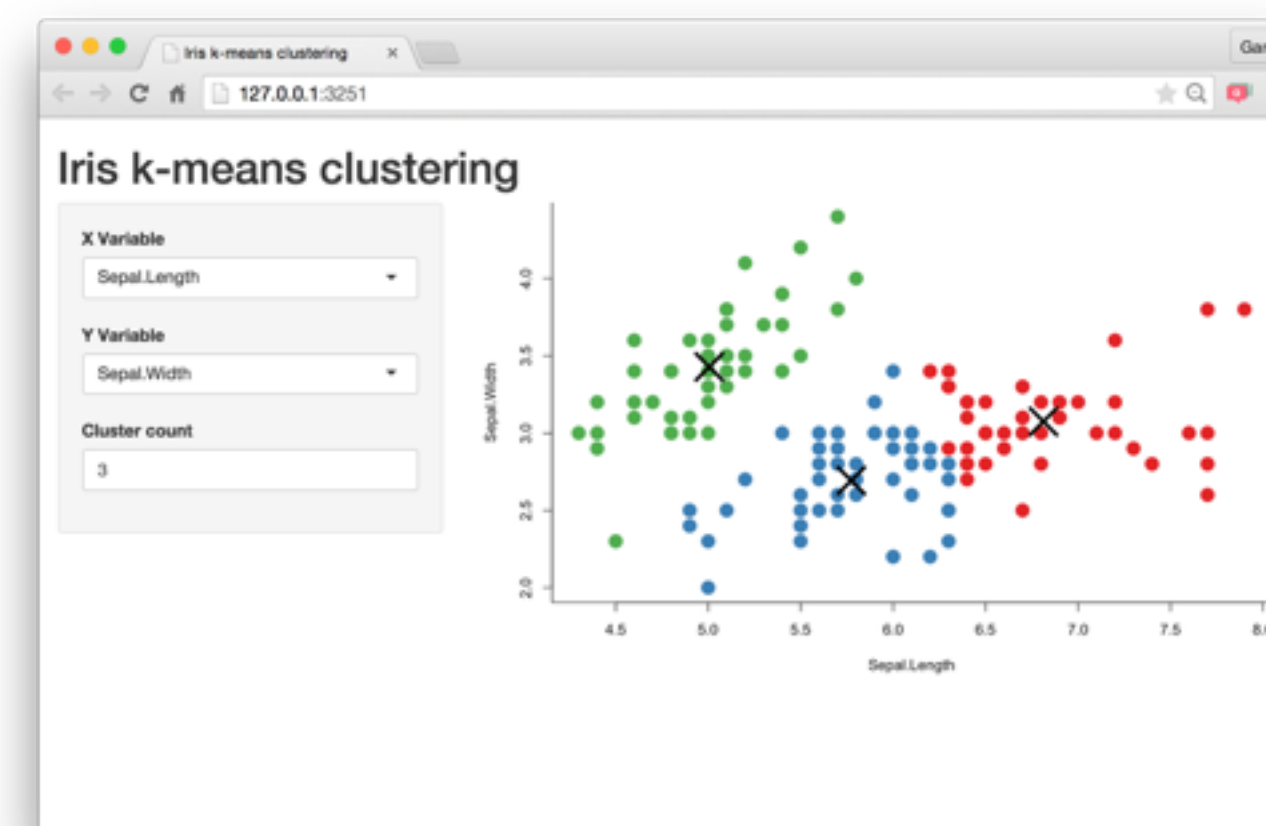
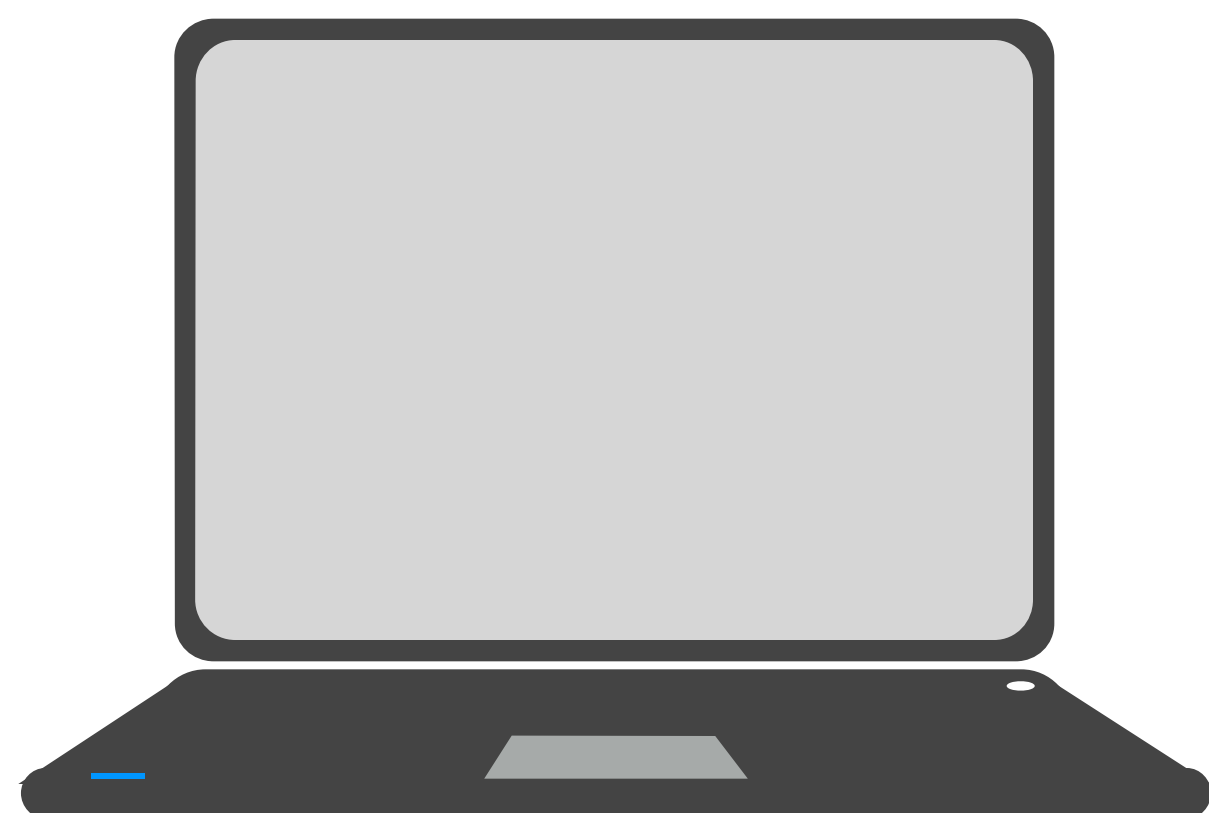


More at

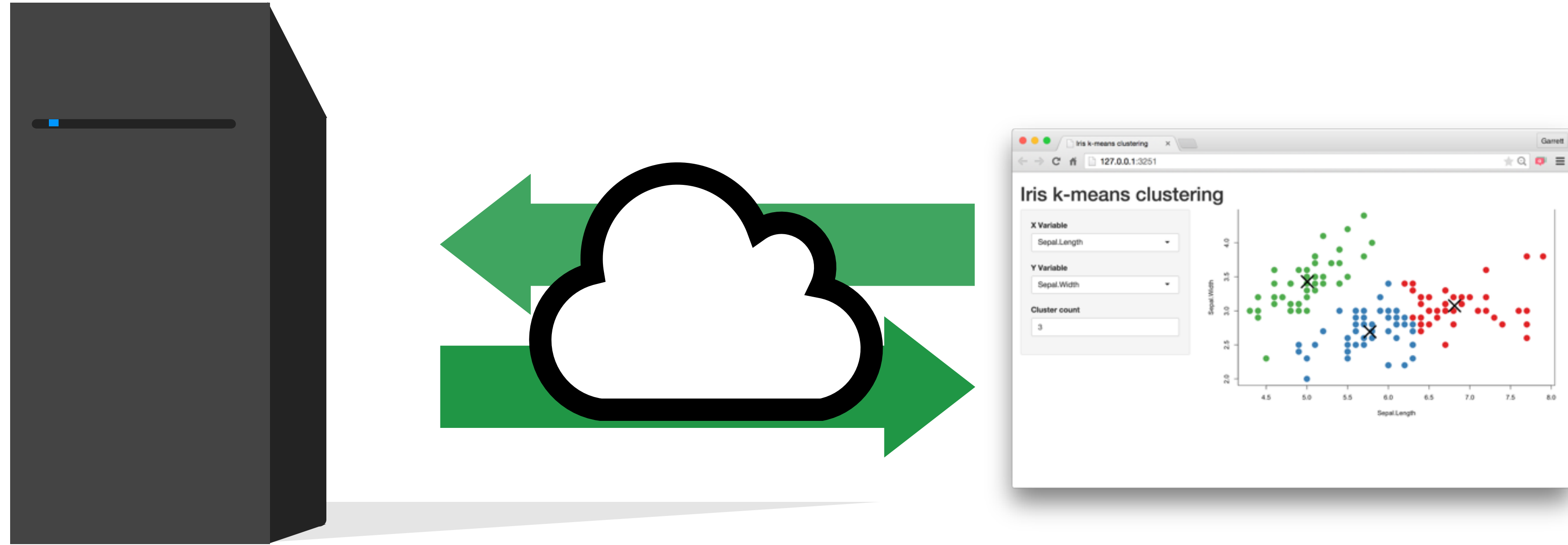
www.rstudio.com/resources/training/online-learning/

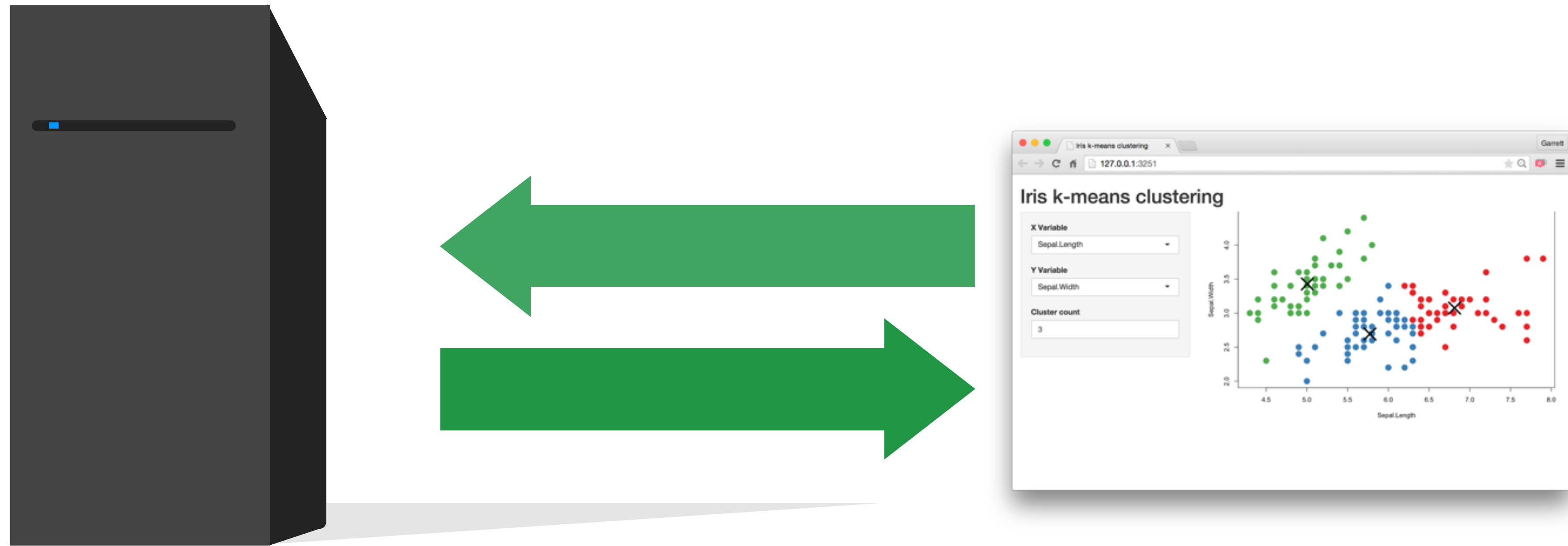
**Understand the
architecture**

Every Shiny app is maintained by a computer running R



Every Shiny app is maintained by a computer running R





Server Instructions



User Interface (UI)

**Use the
template**

App template

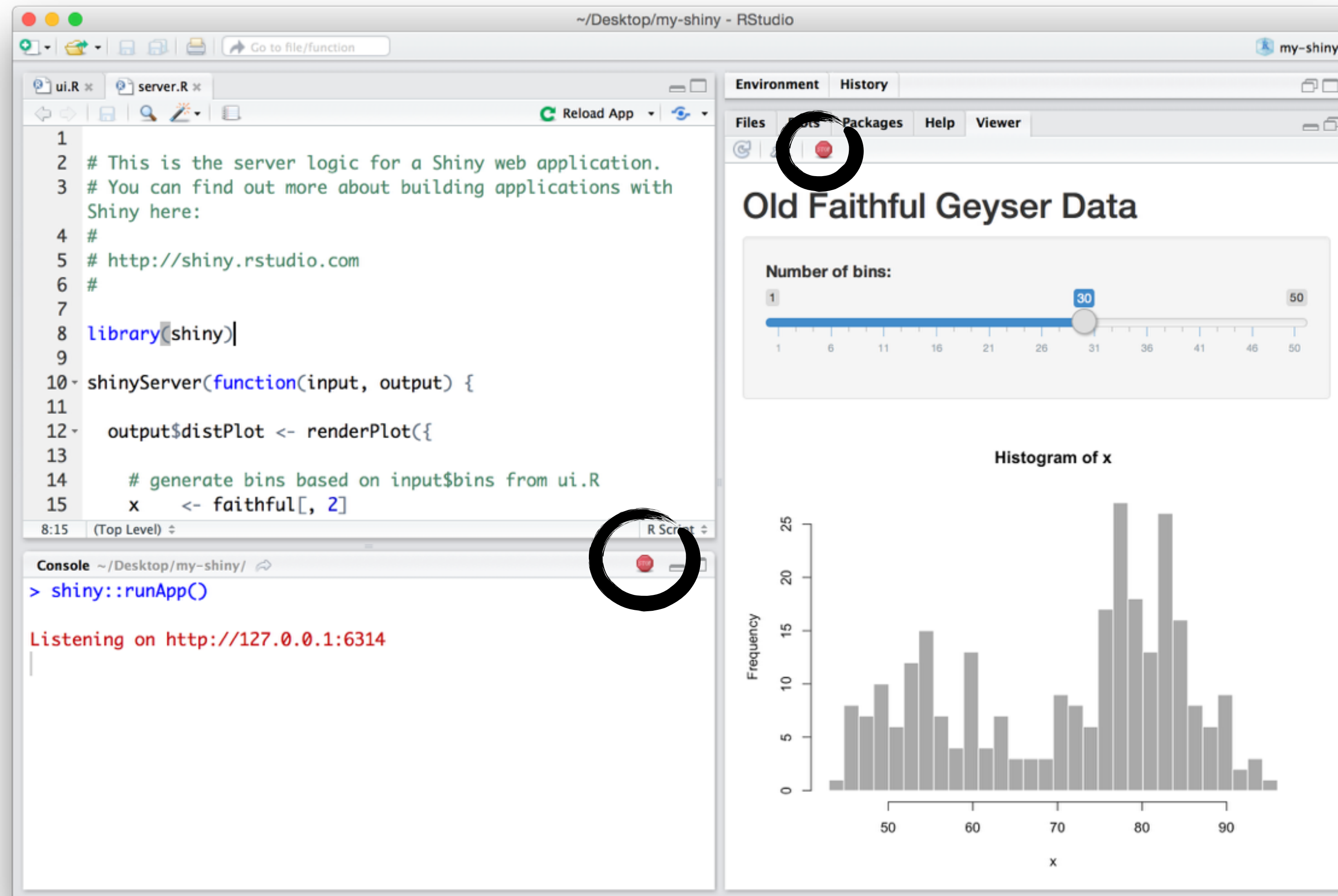
The shortest viable shiny app

```
library(shiny)
ui <- fluidPage()

server <- function(input, output) {}

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

Close an app



The screenshot shows the RStudio interface with a Shiny application running. The application displays "Old Faithful Geyser Data" with a histogram and a slider for "Number of bins". The console shows the command `shiny::runApp()` and the message "Listening on http://127.0.0.1:6314". Two red stop buttons are circled in black, indicating how to close the application.

```
1 # This is the server logic for a Shiny web application.
2 # You can find out more about building applications with
3 # Shiny here:
4 #
5 # http://shiny.rstudio.com
6 #
7
8 library(shiny)
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate bins based on input$bins from ui.R
15     x <- faithful[, 2]
```

Console: `> shiny::runApp()`
Listening on http://127.0.0.1:6314

Number of bins: 1 30 50

Histogram of x

Frequency

x

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

```
library(shiny)
ui <- fluidPage("Hello World")

server <- function(input, output) {}

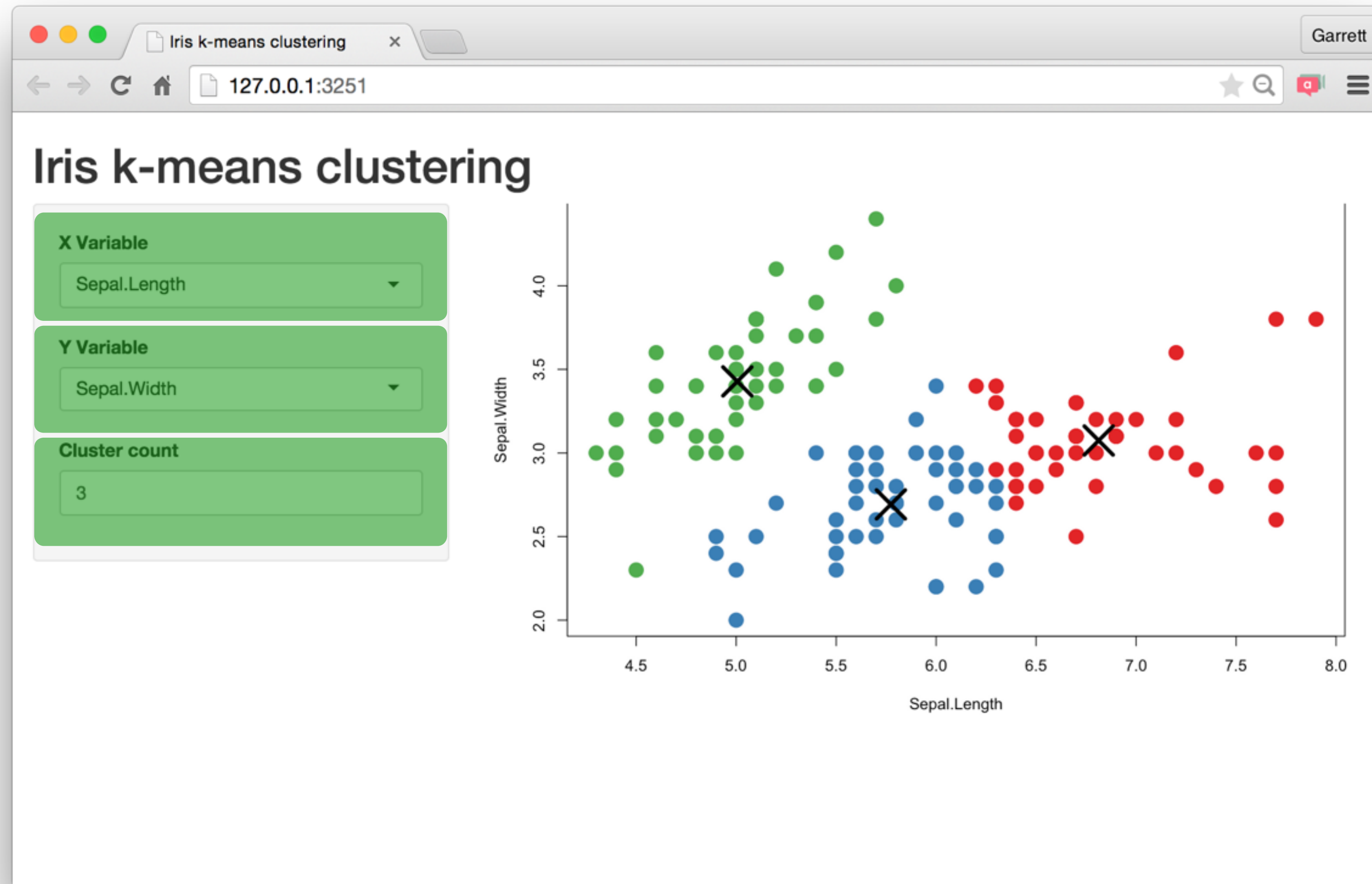
shinyApp(ui = ui, server = server)
```

Build your app around

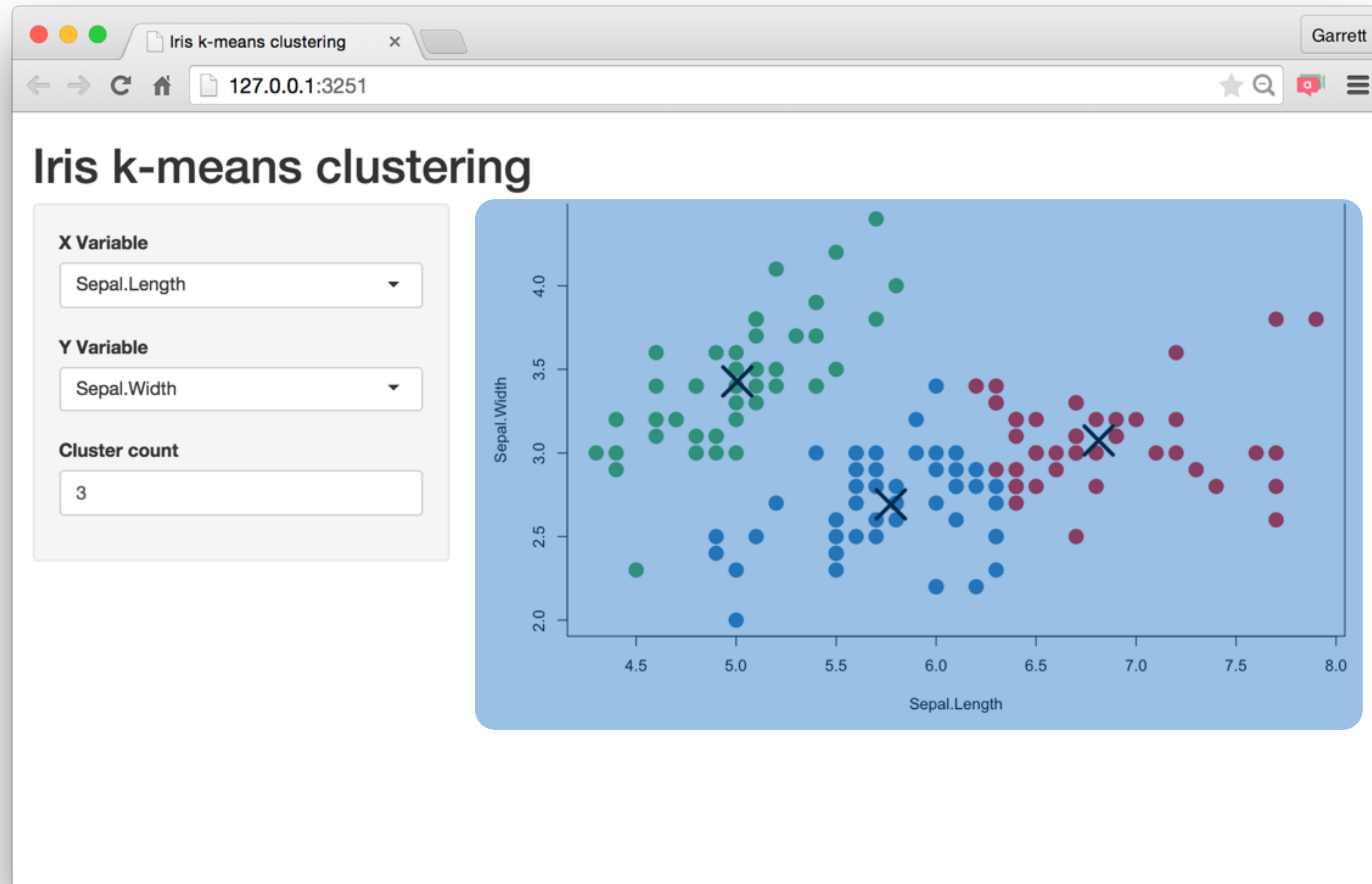
Inputs and

Outputs

Build your app around **inputs** and **outputs**



Build your app around **inputs** and **outputs**



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

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

Inputs

Create an input with an ***Input()** function.

```
sliderInput(inputId = "num",  
            label = "Choose a number",  
            value = 25, min = 1, max = 100)
```

```
<div class="form-group shiny-input-container">  
  <label class="control-label" for="num">Choose a number</label>  
  <input class="js-range-slider" id="num" data-min="1" data-max="100"  
    data-from="25" data-step="1" data-grid="true" data-grid-num="9.9"  
    data-grid-snap="false" data-pretty-separator="," data-keyboard="true"  
    data-keyboard-step="1.01010101010101"/>  
</div>
```

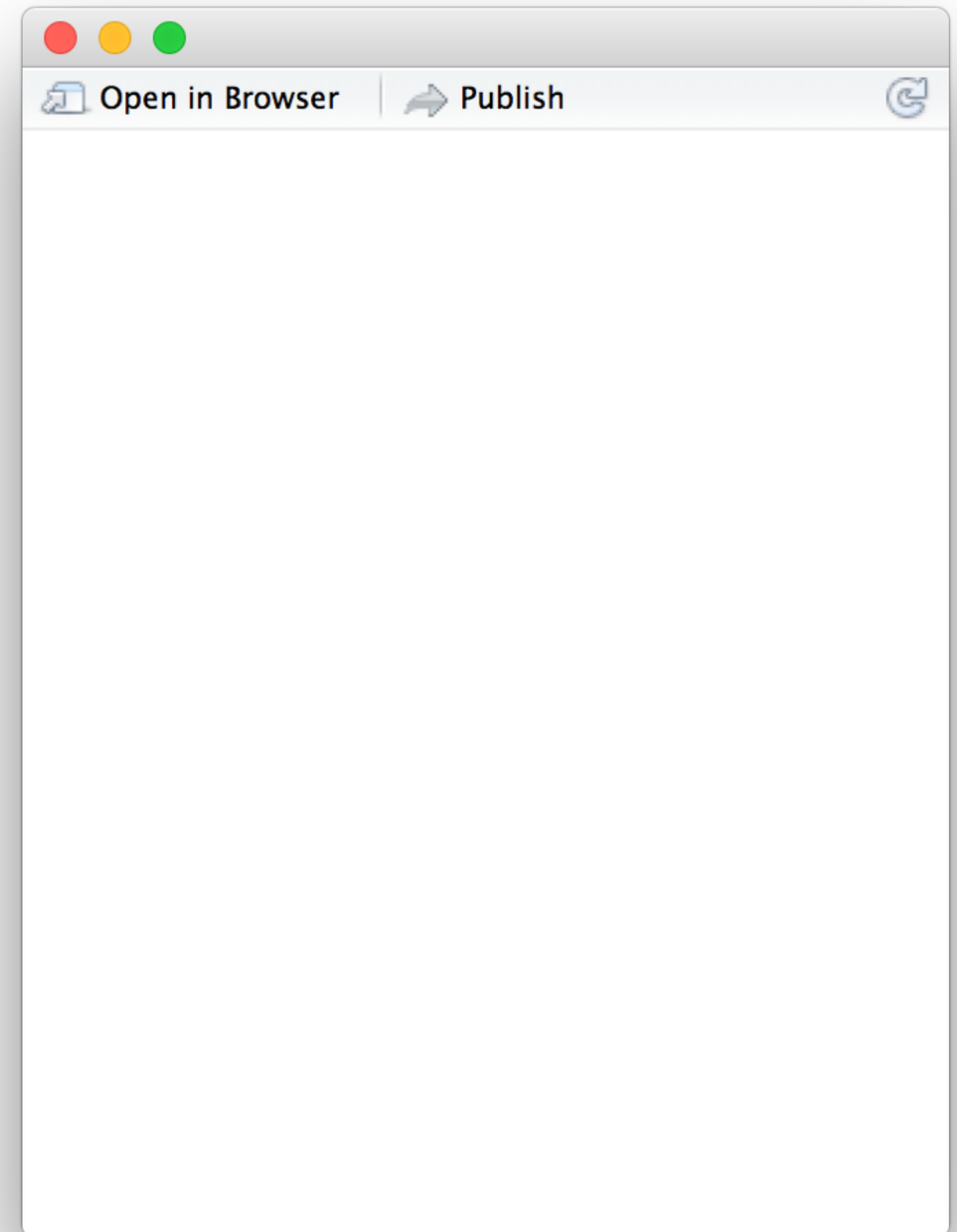
Create an input with an input function.

```
library(shiny)
ui <- fluidPage(

)

server <- function(input, output) {}

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

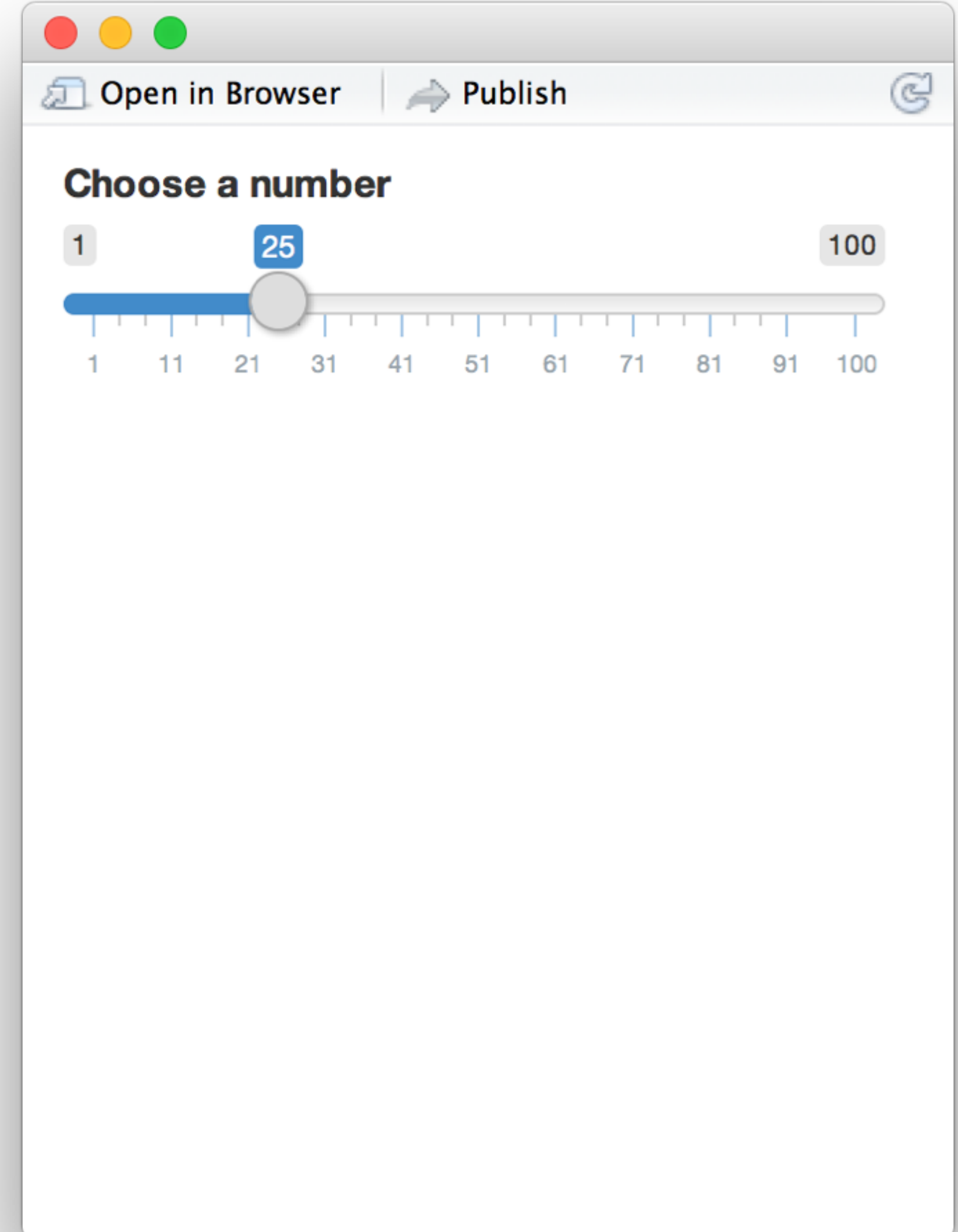


Create an input with an input function.

```
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)
```



Buttons

`actionButton()`
`submitButton()`

Date range

 to

`dateRangeInput()`

Radio buttons

- Choice 1
- Choice 2
- Choice 3

`radioButtons()`

Single checkbox

 Choice A

`checkboxInput()`

Checkbox group

- Choice 1
- Choice 2
- Choice 3

`checkboxGroupInput()`

Date input

`dateInput()`

File input

 No file chosen

`fileInput()`

Numeric input

`numericInput()`

Password Input

`passwordInput()`

Select box

`selectInput()`

Sliders

0 50 100
0 25 75 100

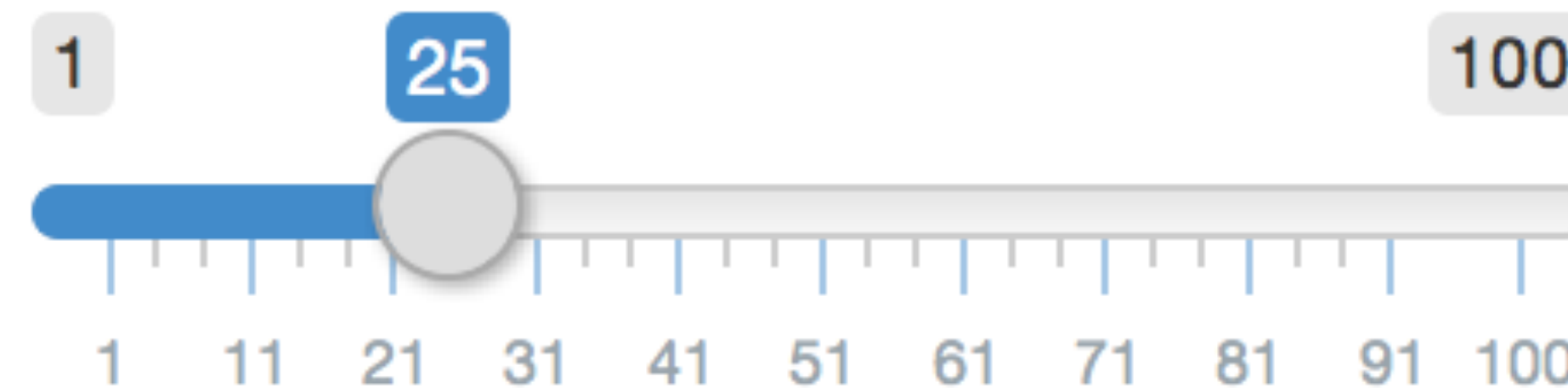
`sliderInput()`

Text input

`textInput()`

Syntax

Choose a number



```
sliderInput(inputId = "num", label = "Choose a number", ...)
```

input name
(for internal use)

Notice:
Id not ID

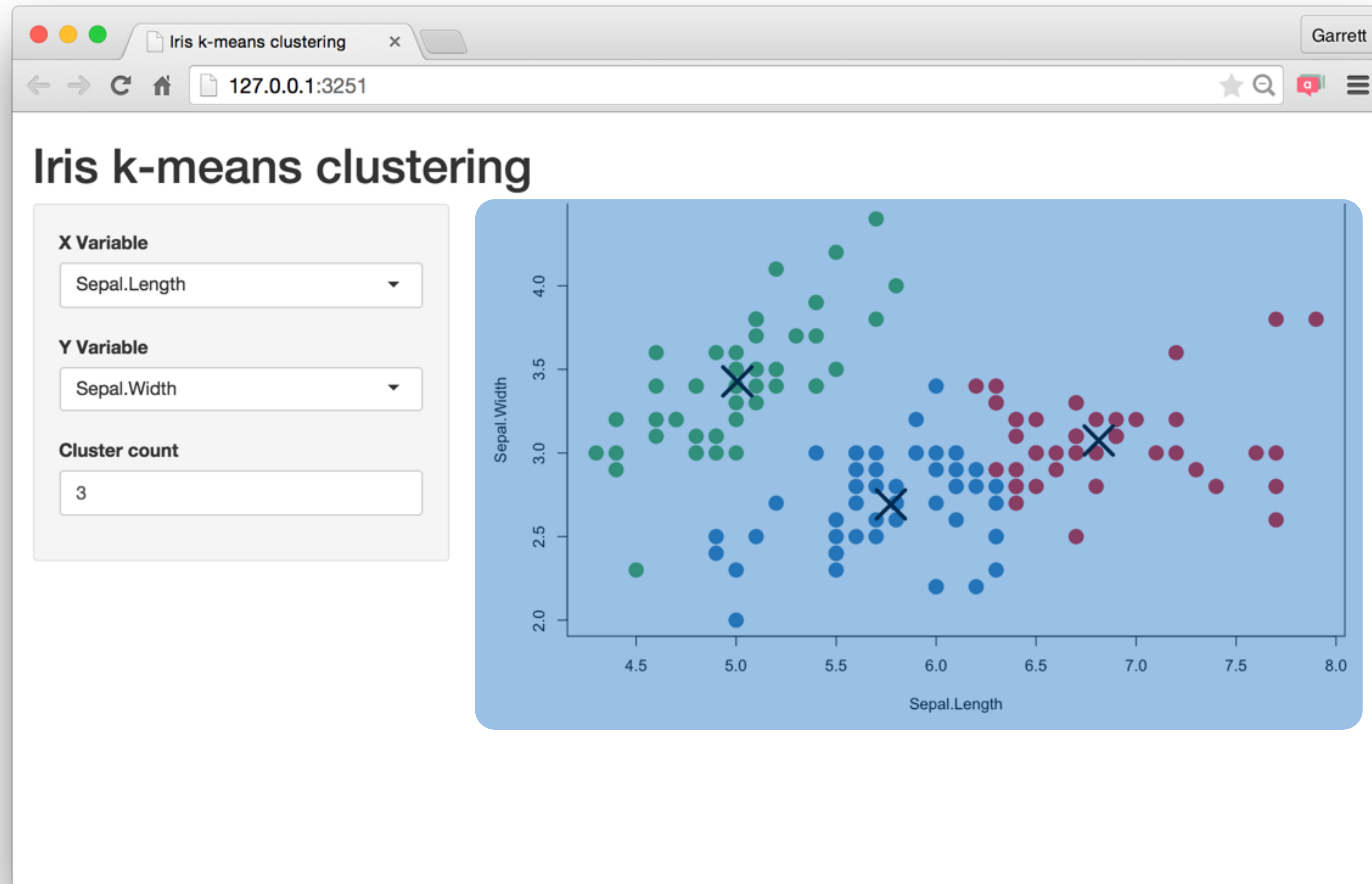
label to
display

input specific
arguments

?sliderInput

Outputs

Build your app around **inputs** and **outputs**



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

*Output()

To display output, add it to `fluidPage()` with an `*Output()` function

```
plotOutput("hist")
```

the type of output
to display

name to give to the
output object

```
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)
```

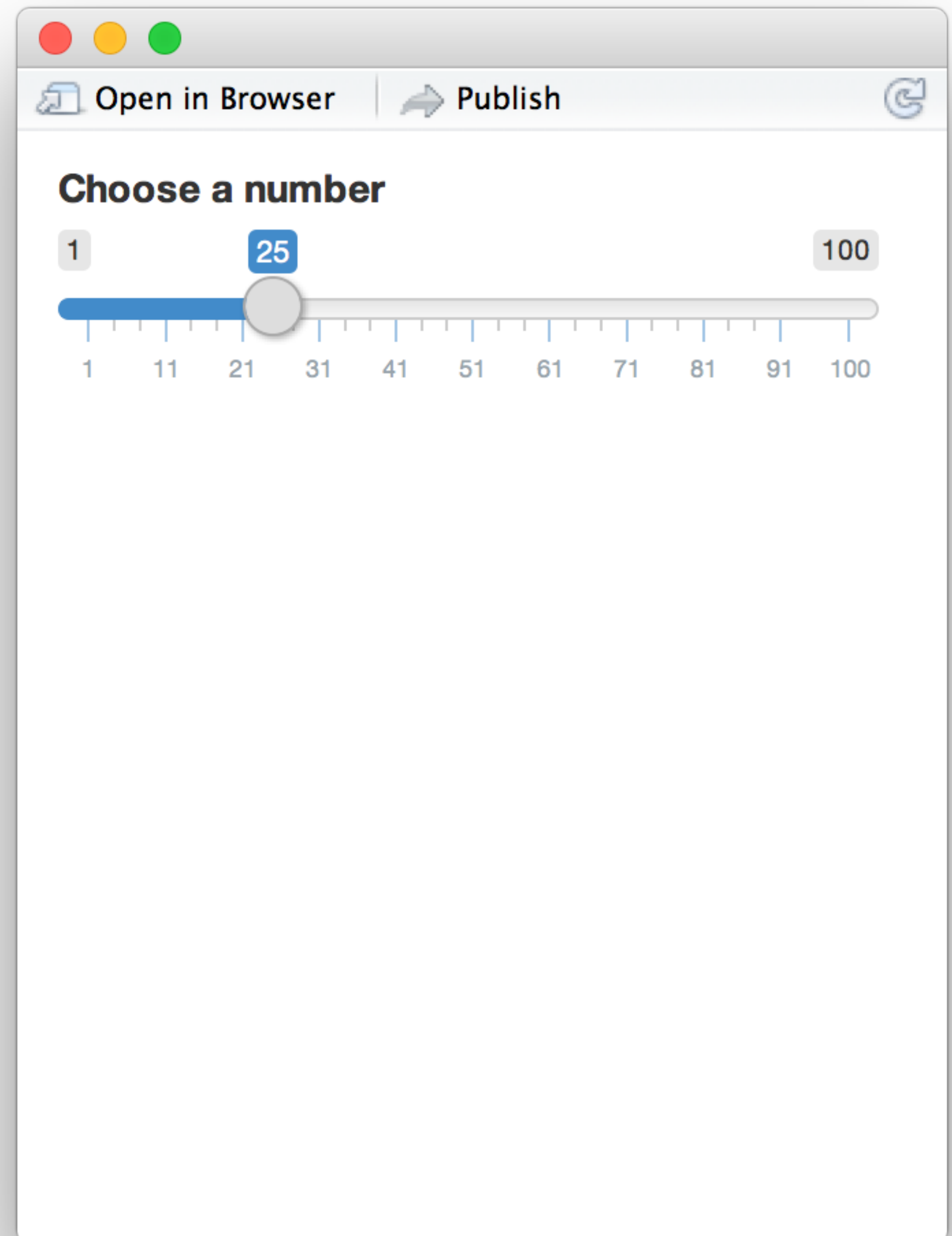
Comma between arguments

```
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)
```

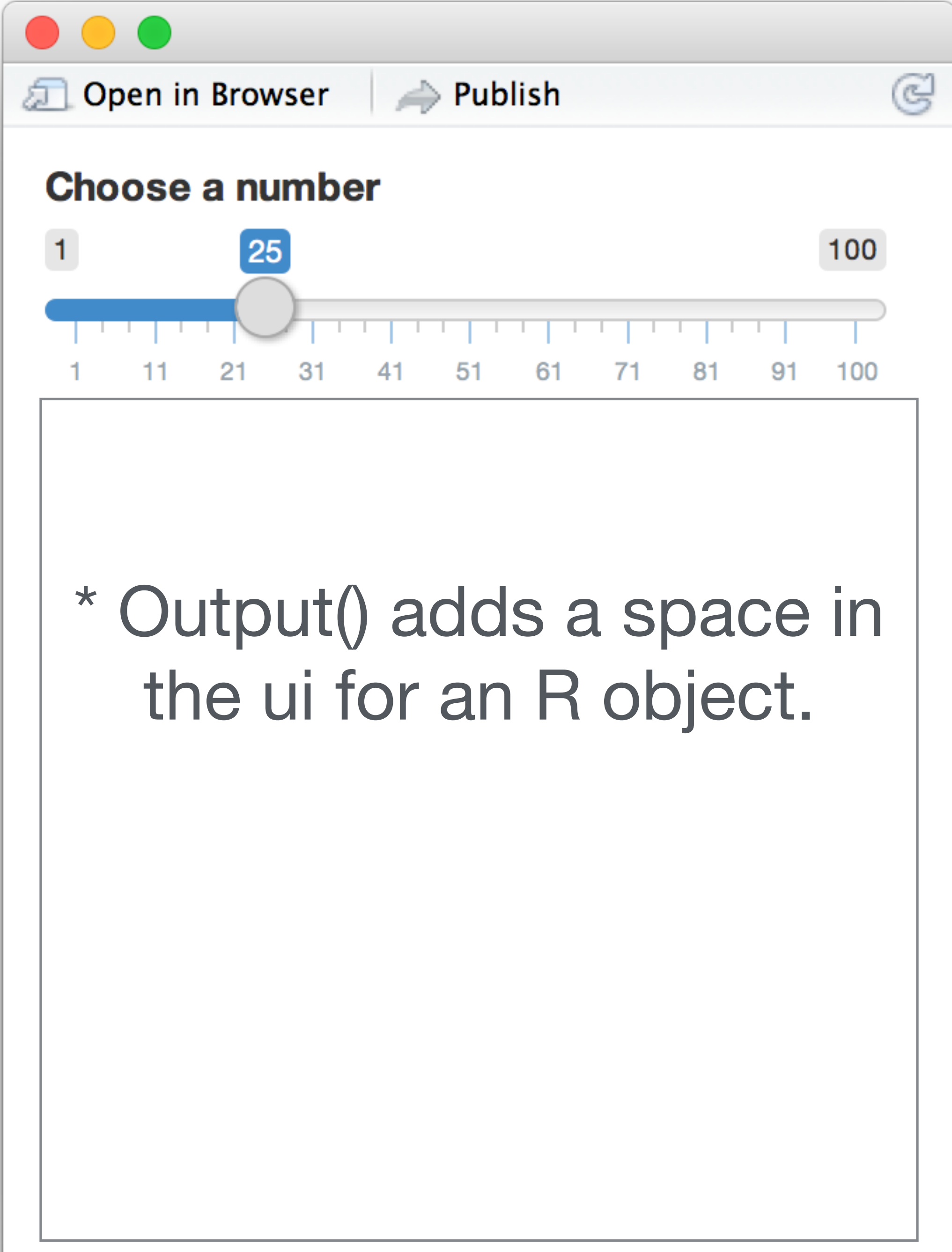


```
library(shiny)

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)
```



Choose a number

1 25 100

1 11 21 31 41 51 61 71 81 91 100

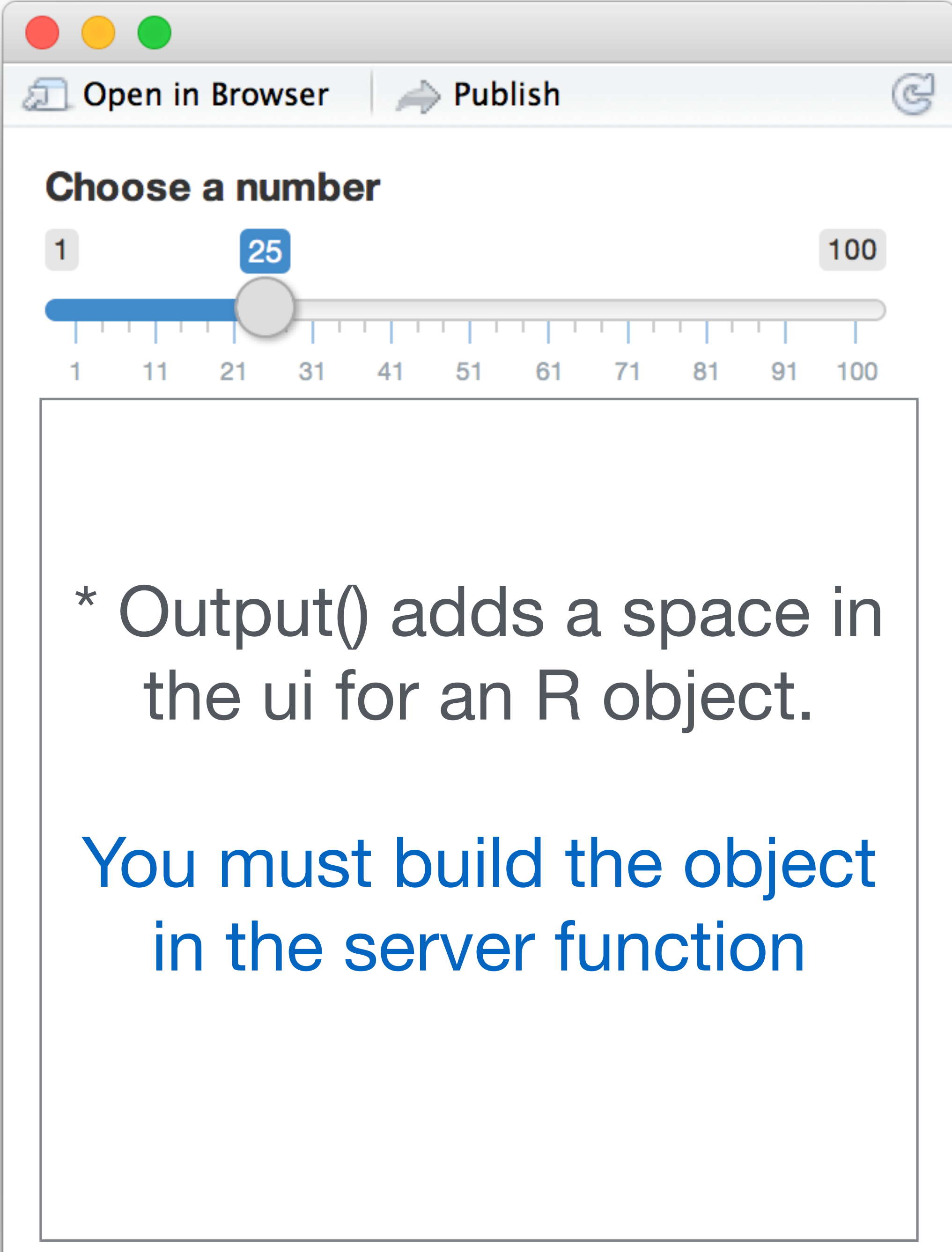
* Output() adds a space in the ui for an R object.


```
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)
```



Choose a number

1 25 100

1 11 21 31 41 51 61 71 81 91 100

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

You must build the object in the server function

Recap

Begin each app with the template

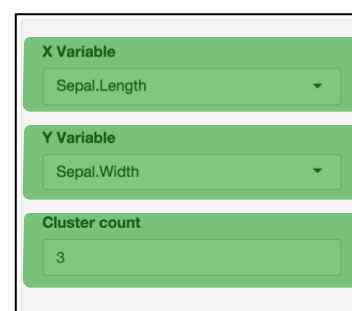
```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```



Hello World

Add elements as arguments to **fluidPage()**

Create reactive inputs with an ***Input()** function

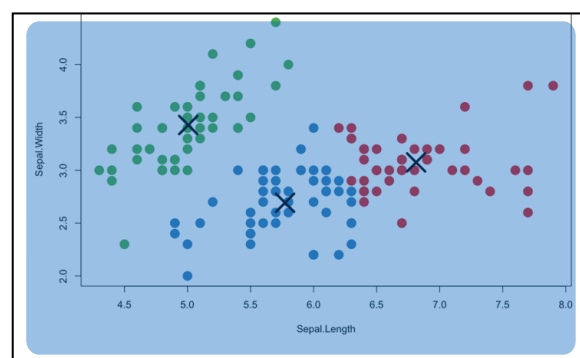


X Variable
Sepal.Length

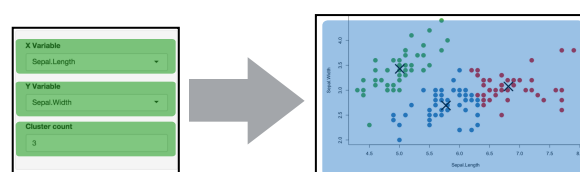
Y Variable
Sepal.Width

Cluster count
3

Display reactive results with an ***Output()** function



Assemble outputs from inputs in the server function



Tell the
server
how to assemble
inputs into outputs

1

Save objects to display to output\$

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

1

Save objects to display to output\$

```
output$hist
```



```
plotOutput("hist")
```

2

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

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

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

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

render*()

Builds reactive output to display in UI

```
renderPlot({ hist(rnorm(100)) })
```

type of object to
build

code block that builds
the object

2

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

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    hist(rnorm(100))  
  })  
}
```

2

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

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    title <- "100 random normal values"  
    hist(rnorm(100), main = title)  
  })  
}
```

3

Access **input** values with `input$`

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

3

Access **input** values with `input$`

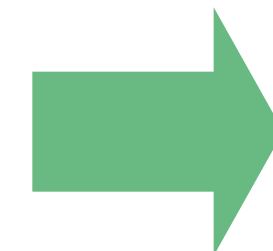
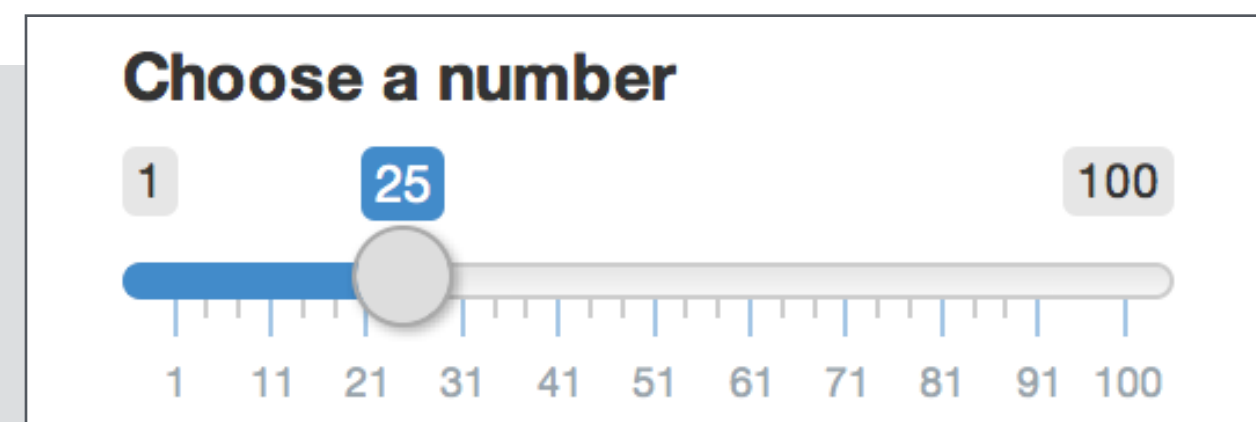
```
sliderInput(inputId = "num", ...)
```



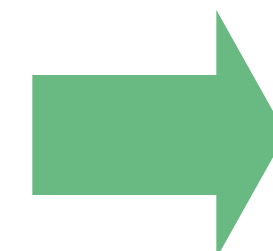
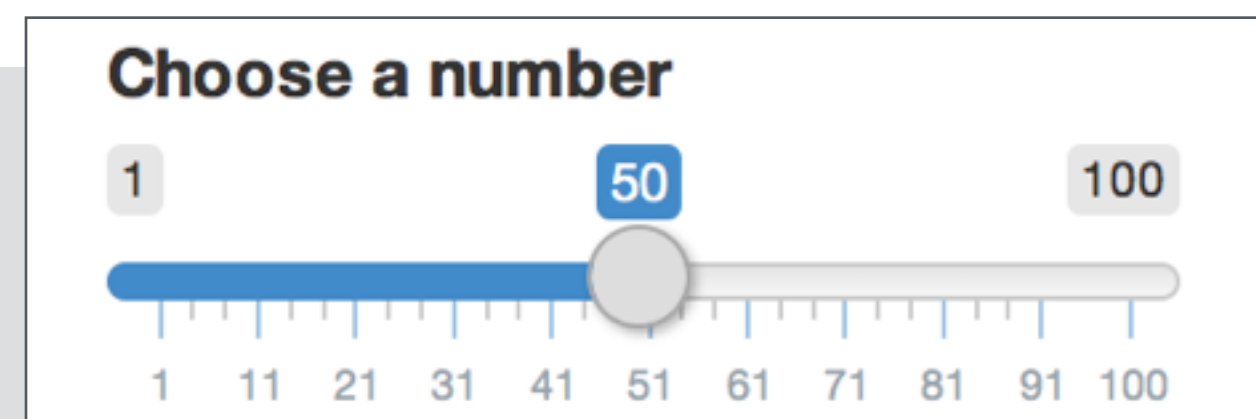
```
input$num
```

Input values

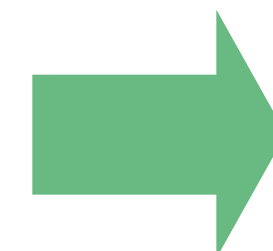
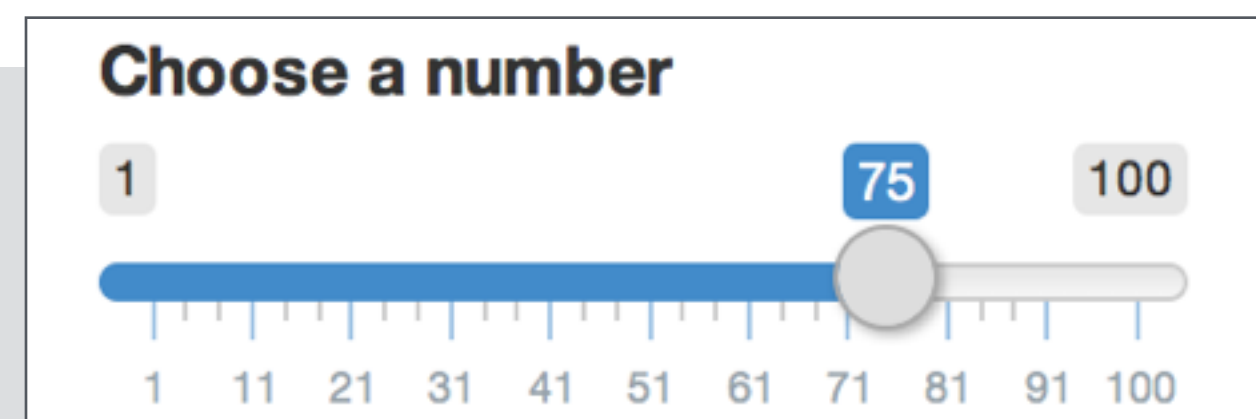
The input value changes whenever a user changes the input.



```
input$num = 25
```



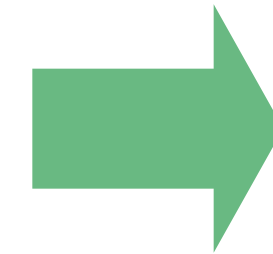
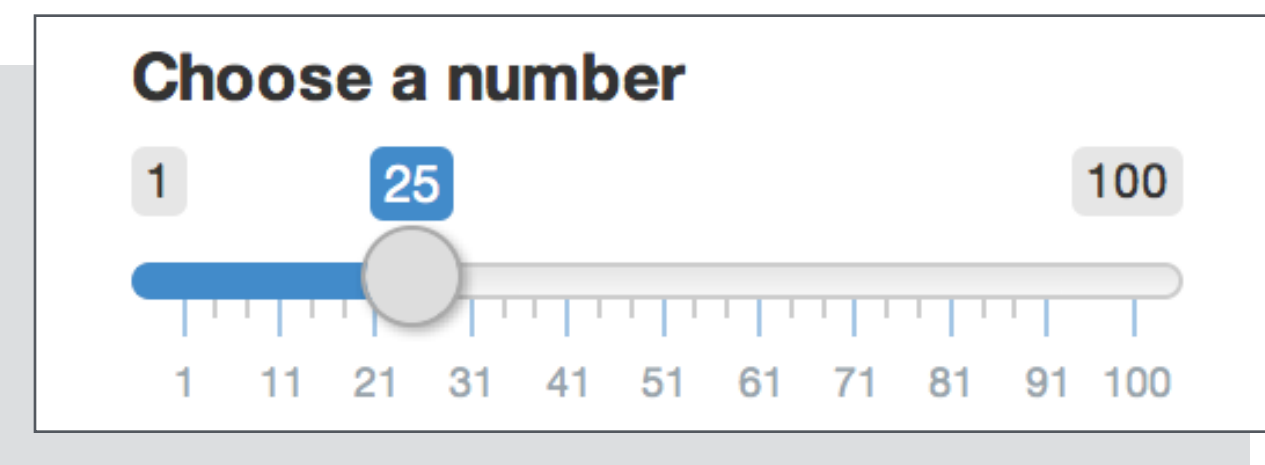
```
input$num = 50
```



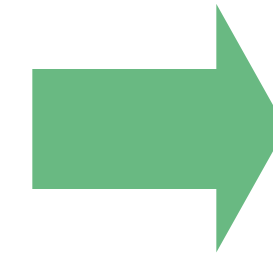
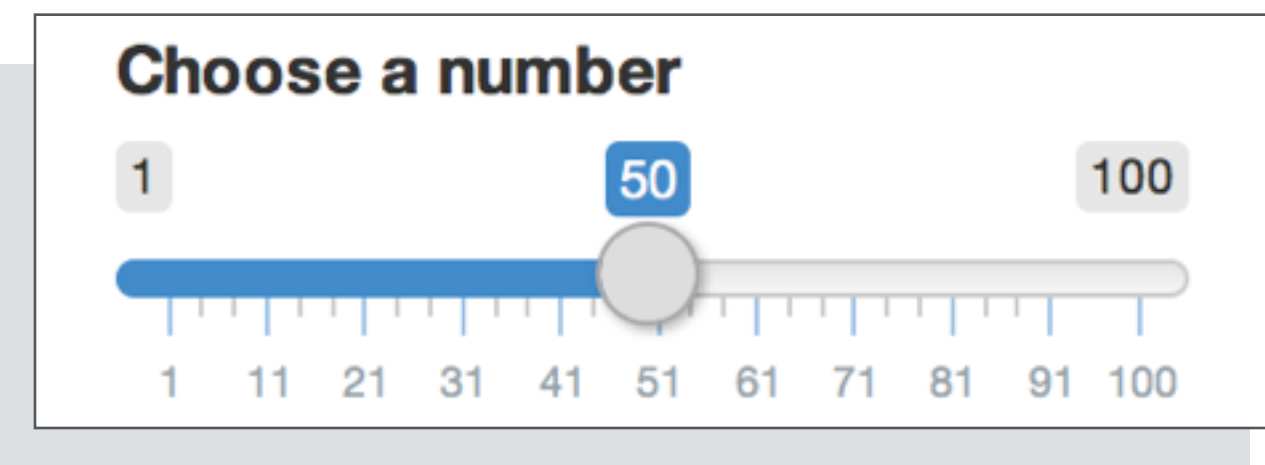
```
input$num = 75
```

Input values

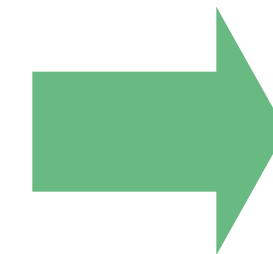
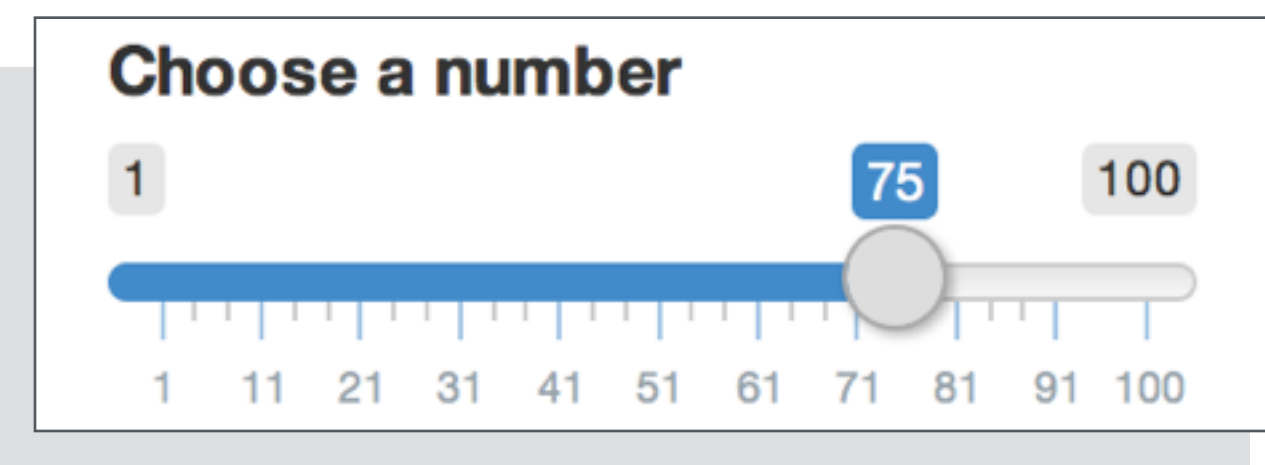
The input value changes whenever a user changes the input.



```
input$num = 25
```



```
input$num = 50
```



```
input$num =
```

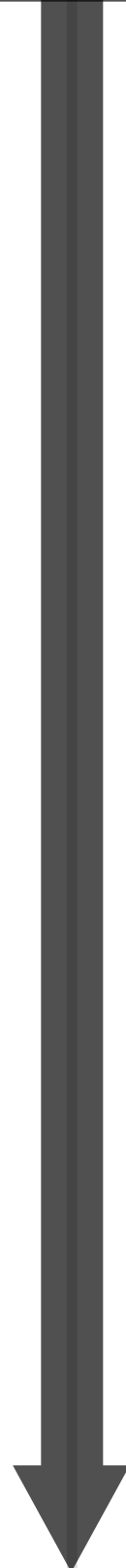
Output will automatically update
if you follow the 3 rules

Reactivity 101

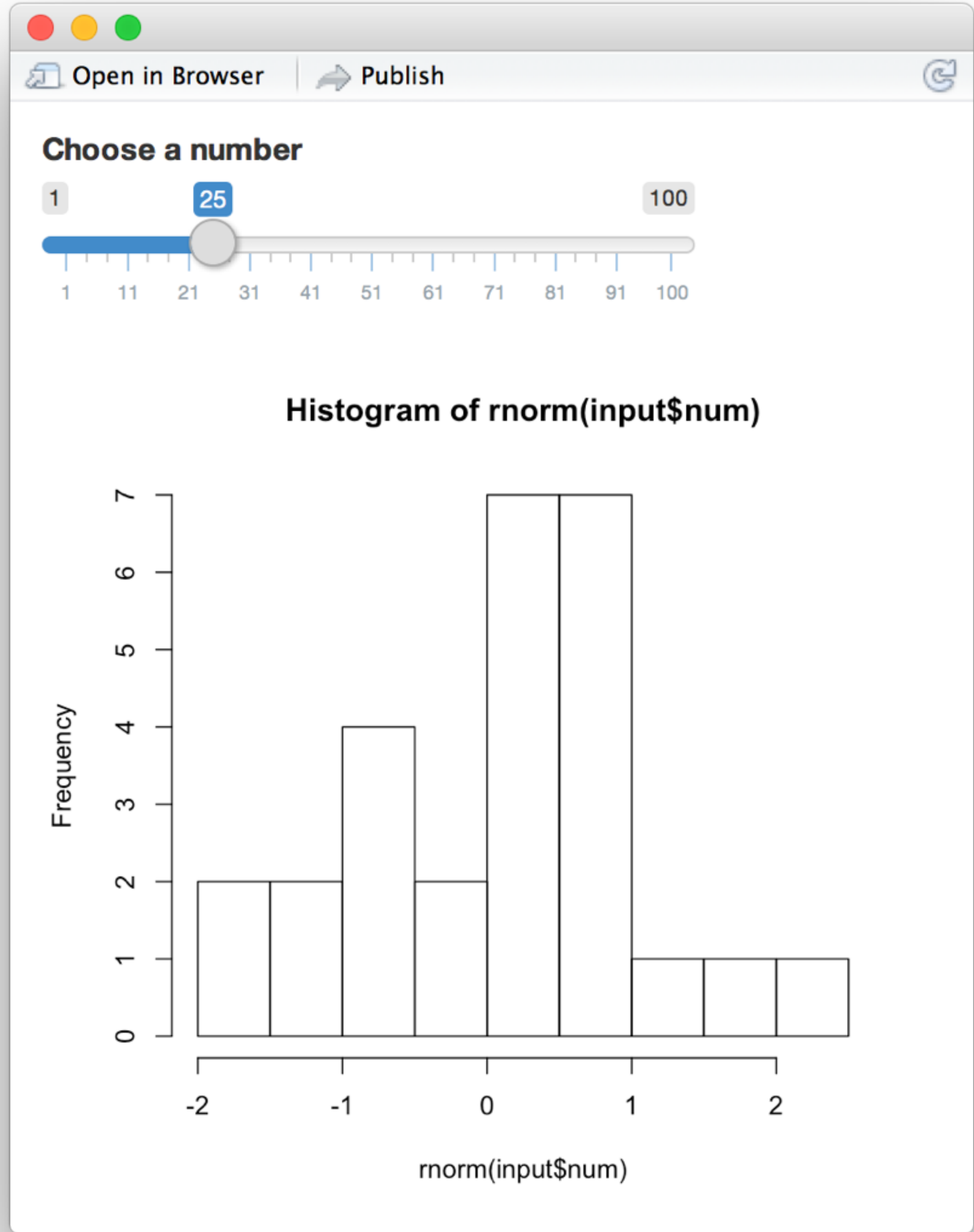
Reactivity automatically occurs whenever you use an input value to render an output object

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


input\$num

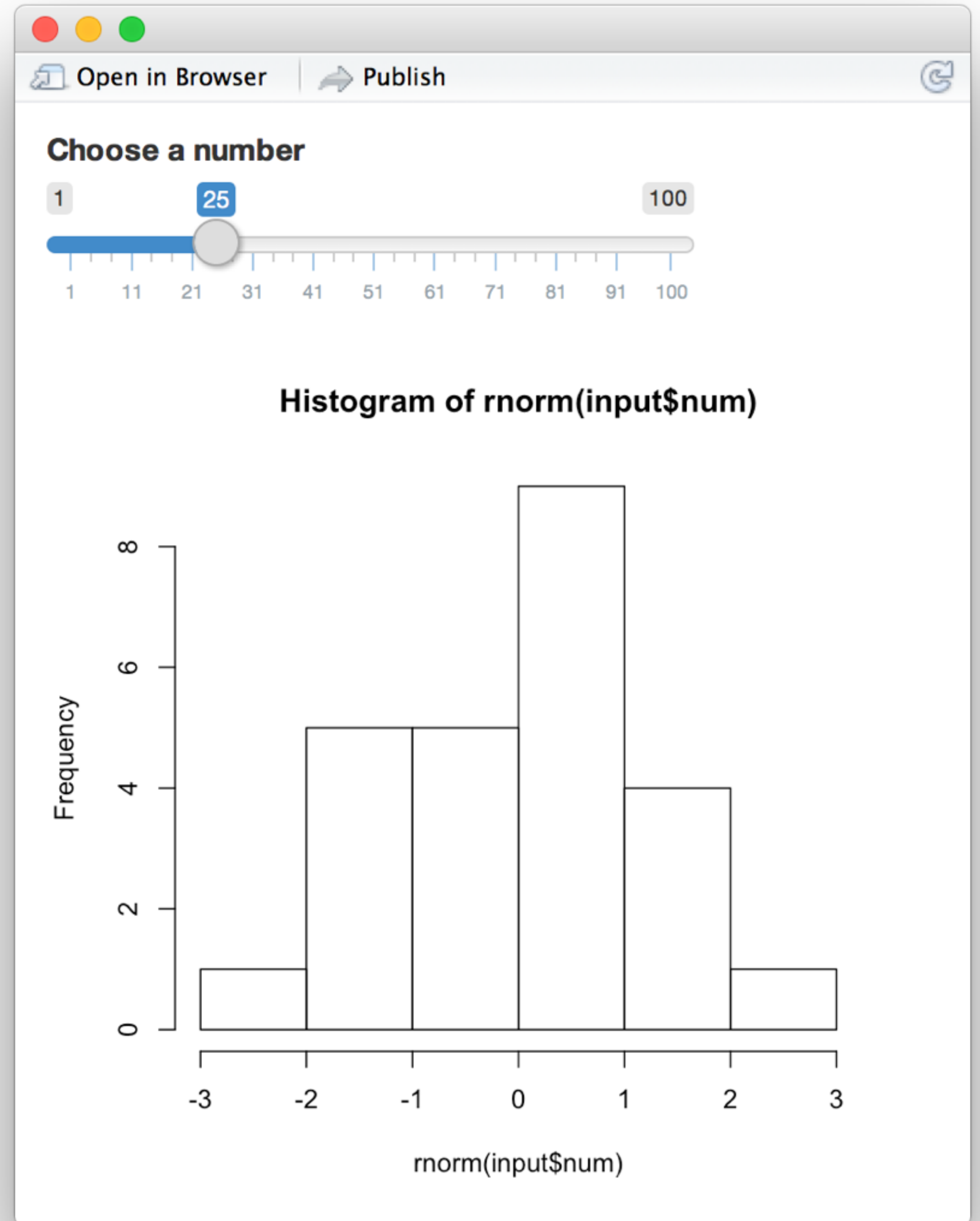


```
renderPlot({  
  hist(rnorm(input$num))  
})
```



input\$num

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



Recap: Server

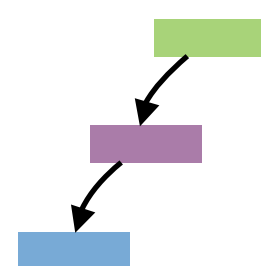


Use the server function to assemble inputs into outputs. Follow 3 rules:

output\$hist ←

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

input\$num



1. Save the output that you build to **output\$**

2. Build the output with a **render*()** function

3. Access input values with **input\$**

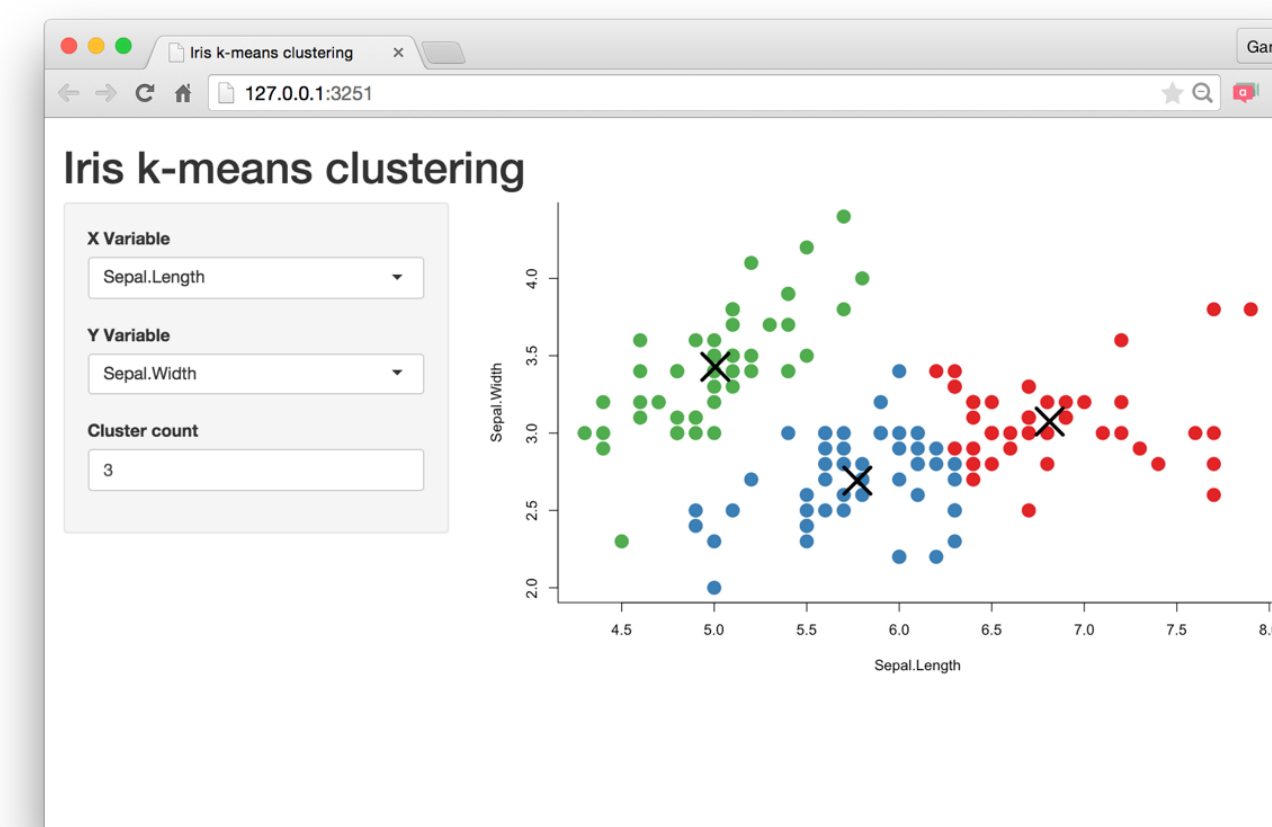
Create reactivity by using **Inputs** to build **rendered Outputs**

Share

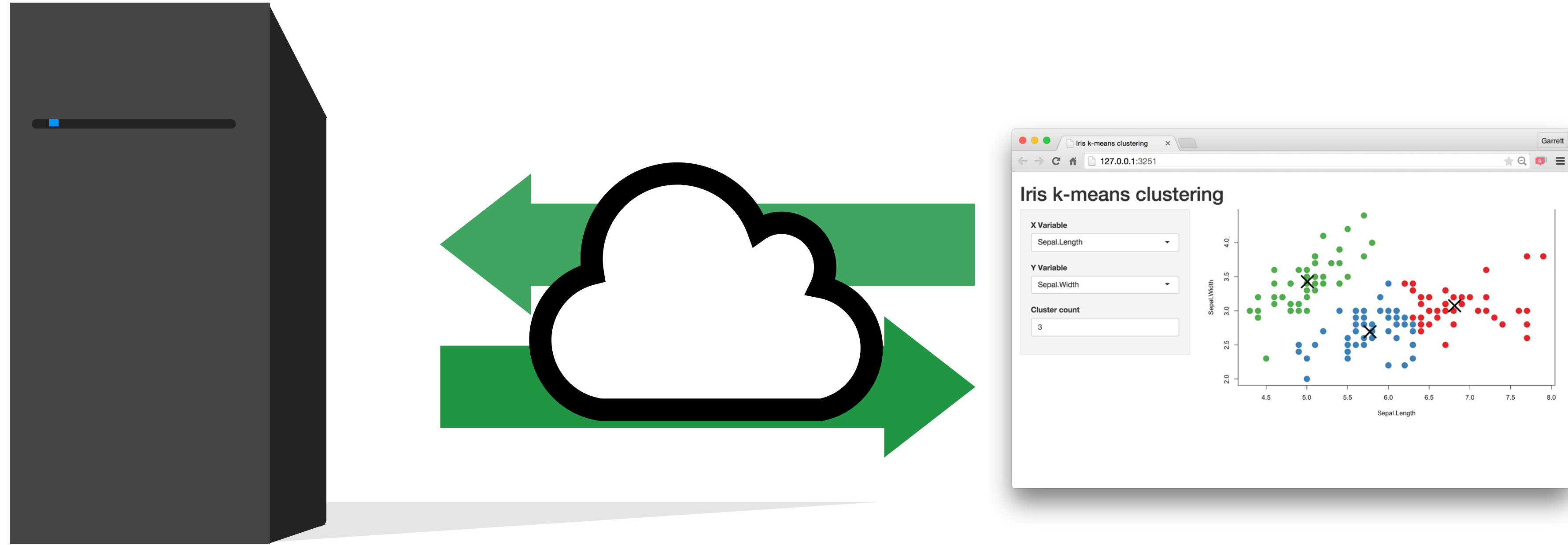
your app



Every Shiny app is maintained by a computer running R



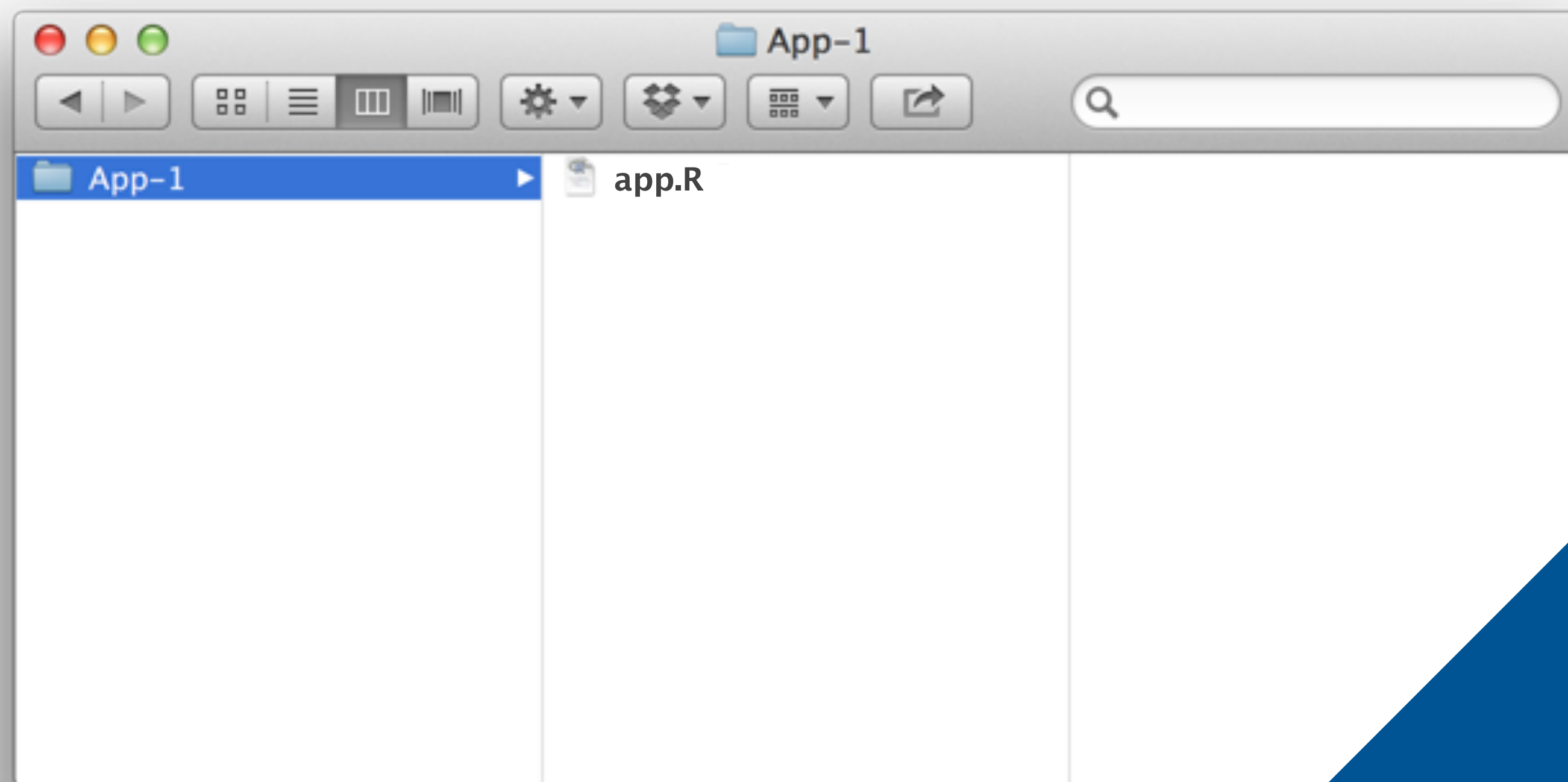
Every Shiny app is maintained by a computer running R



How to save your app

One directory with every file the app needs:

- **app.R** (*your script which ends with a call to `shinyApp()`*)
- **datasets, images, css, helper scripts, etc.**



You must use this exact name (**app.R**)

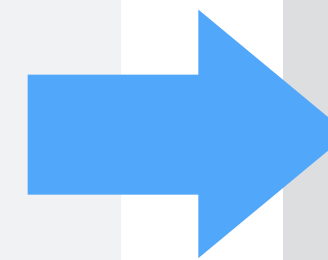
Two file apps

```
library(shiny)

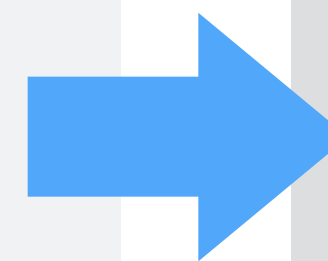
ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)

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

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



```
# ui.R
library(shiny)
fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)
```

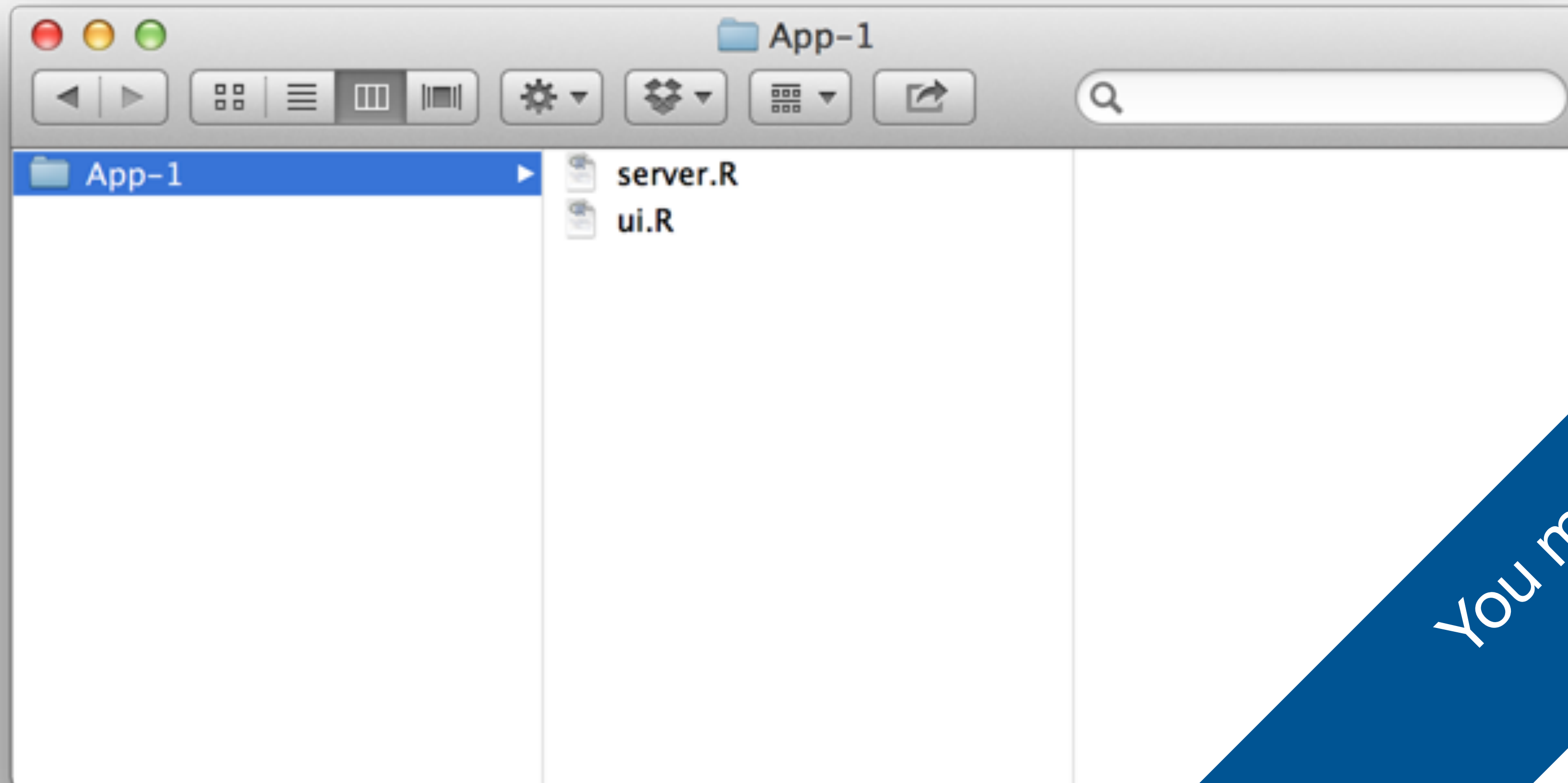


```
# server.R
library(shiny)
function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}
```


Two file apps

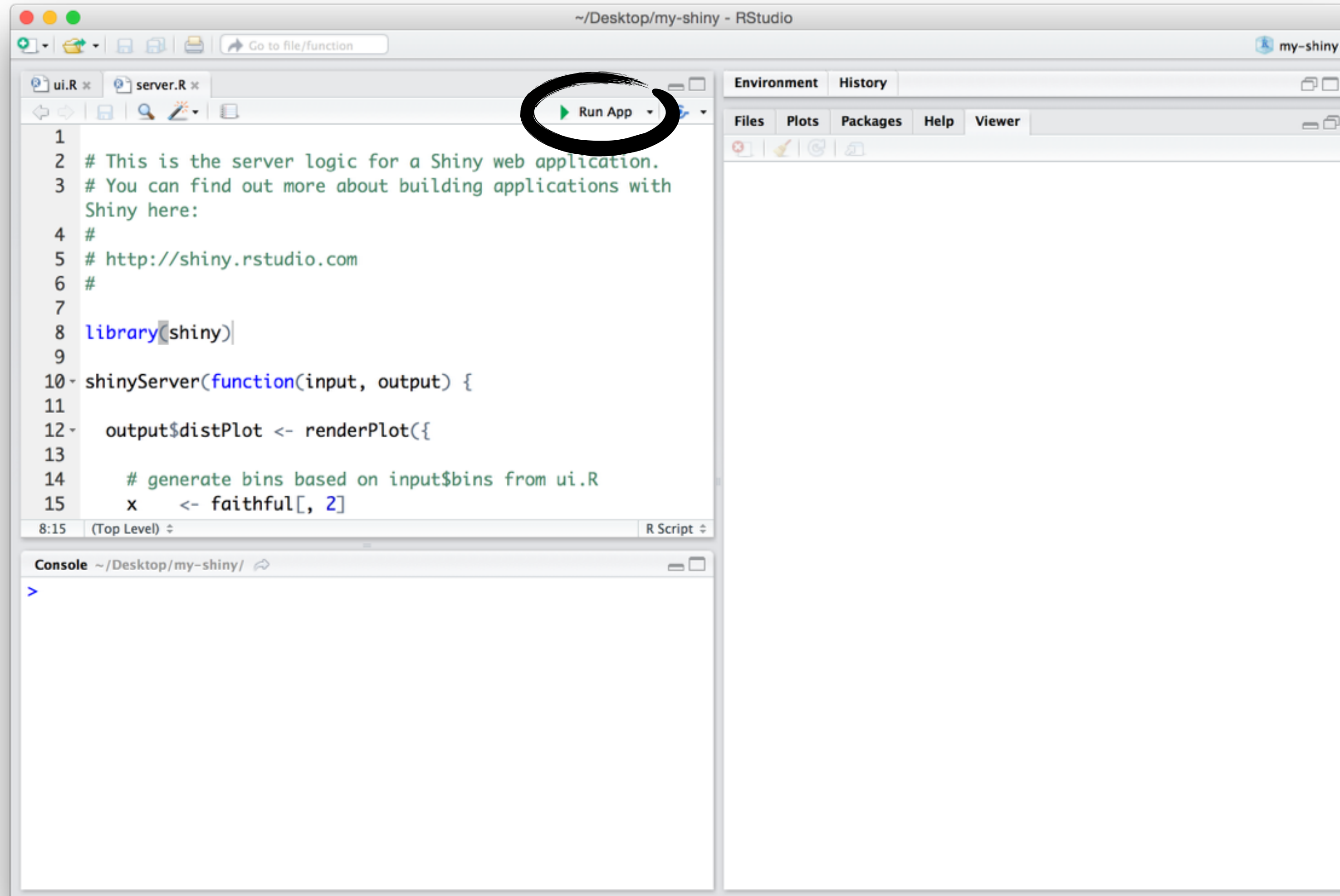
One directory with two files:

- `server.R`
- `ui.R`

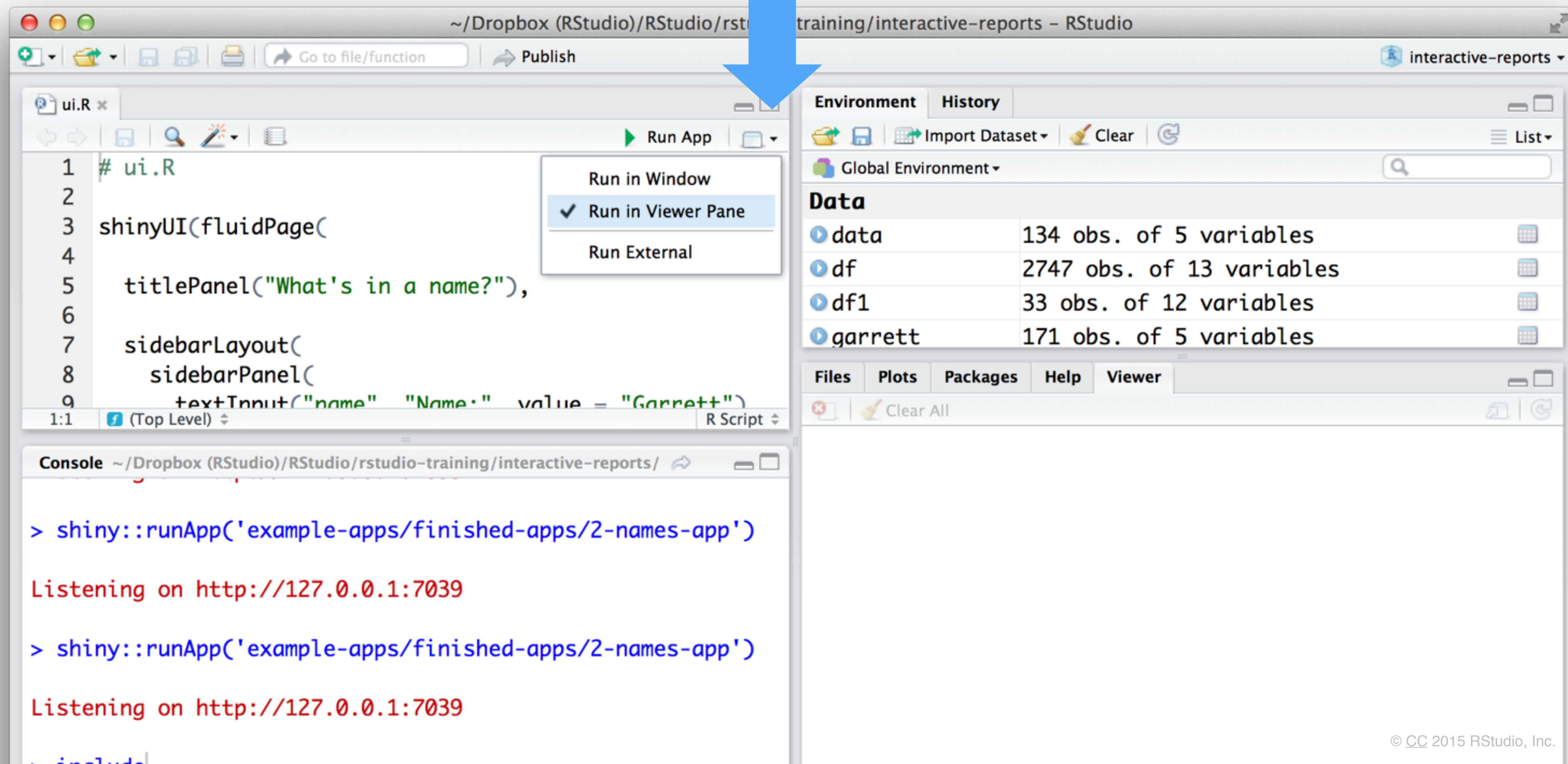


You must use these exact names

Launch an app



Display options



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for a Shiny application. The code is as follows:

```
1 # ui.R
2
3 shinyUI(fluidPage(
4
5   titlePanel("What's in a name?"),
6
7   sidebarLayout(
8     sidebarPanel(
9       textInput("name", "Name:" value = "Garrett")
10    )
11  )
12 )
```
- Run App Menu:** A dropdown menu is open, showing three options: "Run in Window", "Run in Viewer Pane" (which is selected with a checkmark), and "Run External". A large blue arrow points from the title "Display options" to this menu.
- Environment Panel:** Shows the Global Environment and a list of data objects:

Data	Obs.	Vars.
data	134	5
df	2747	13
df1	33	12
garrett	171	5
- Console:** Shows the command `> shiny::runApp('example-apps/finished-apps/2-names-app')` and the output `Listening on http://127.0.0.1:7039`.

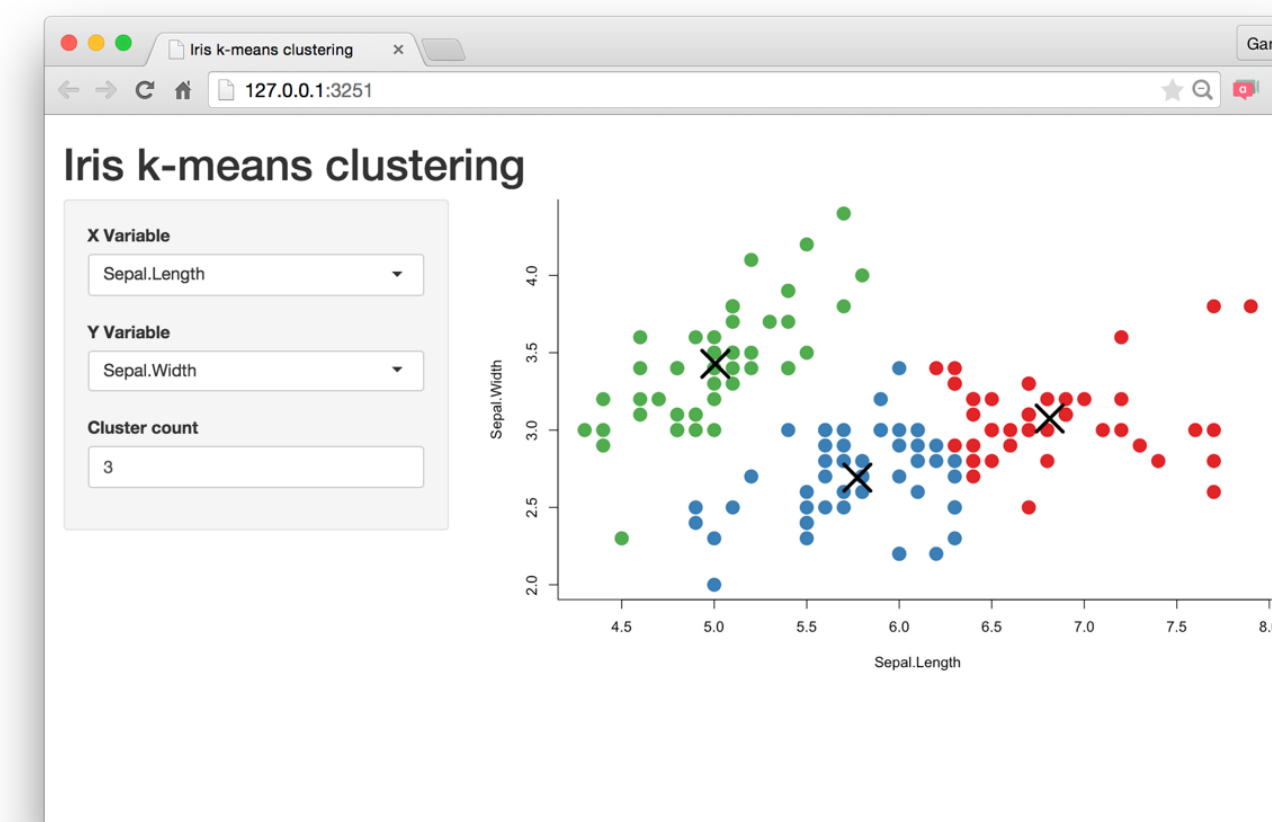
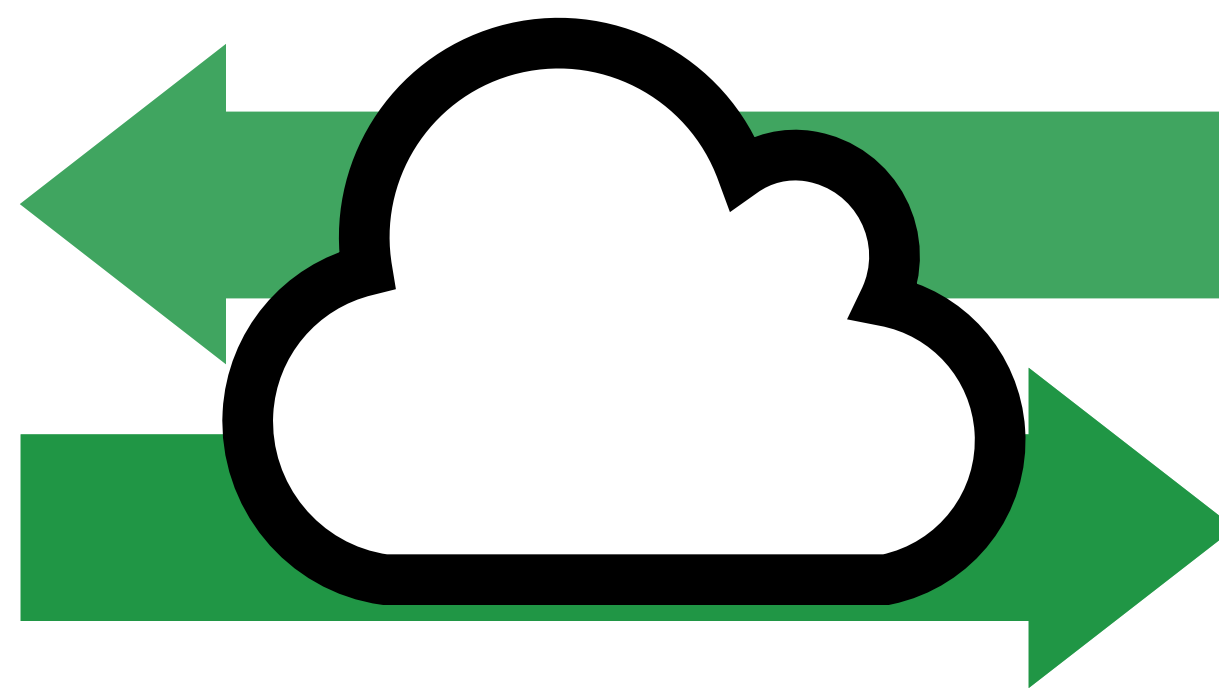
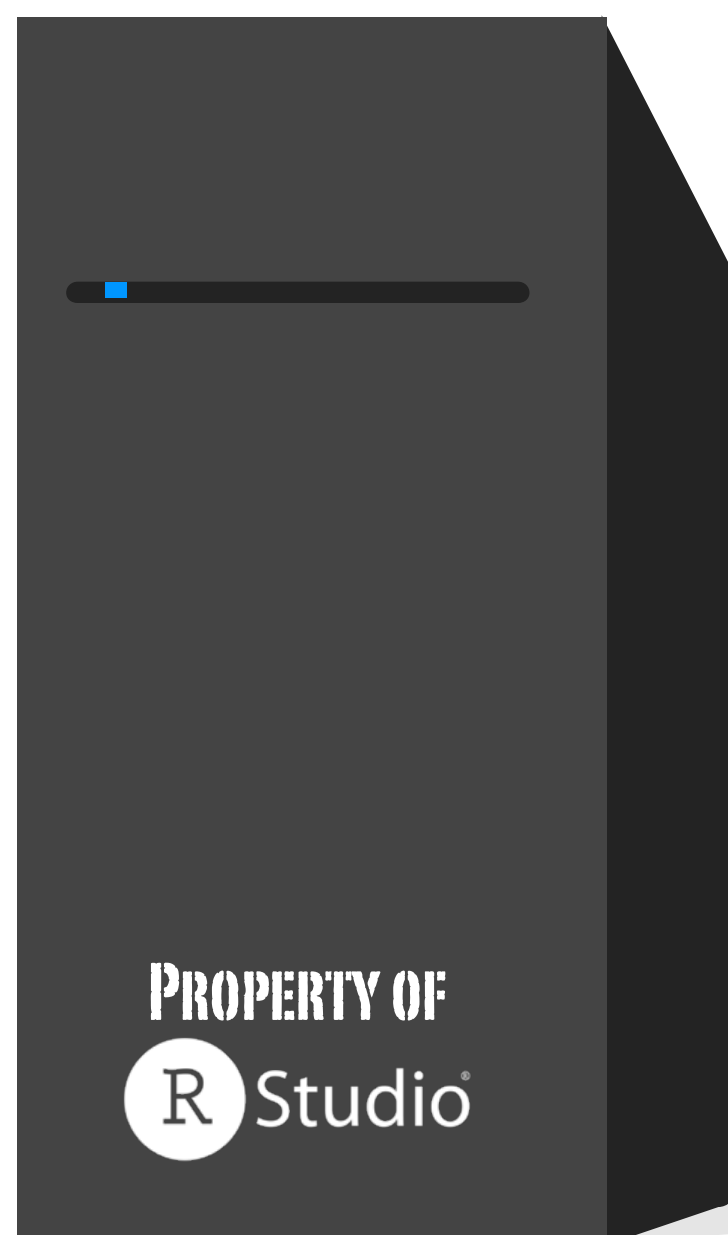
Use
shinyapps.io

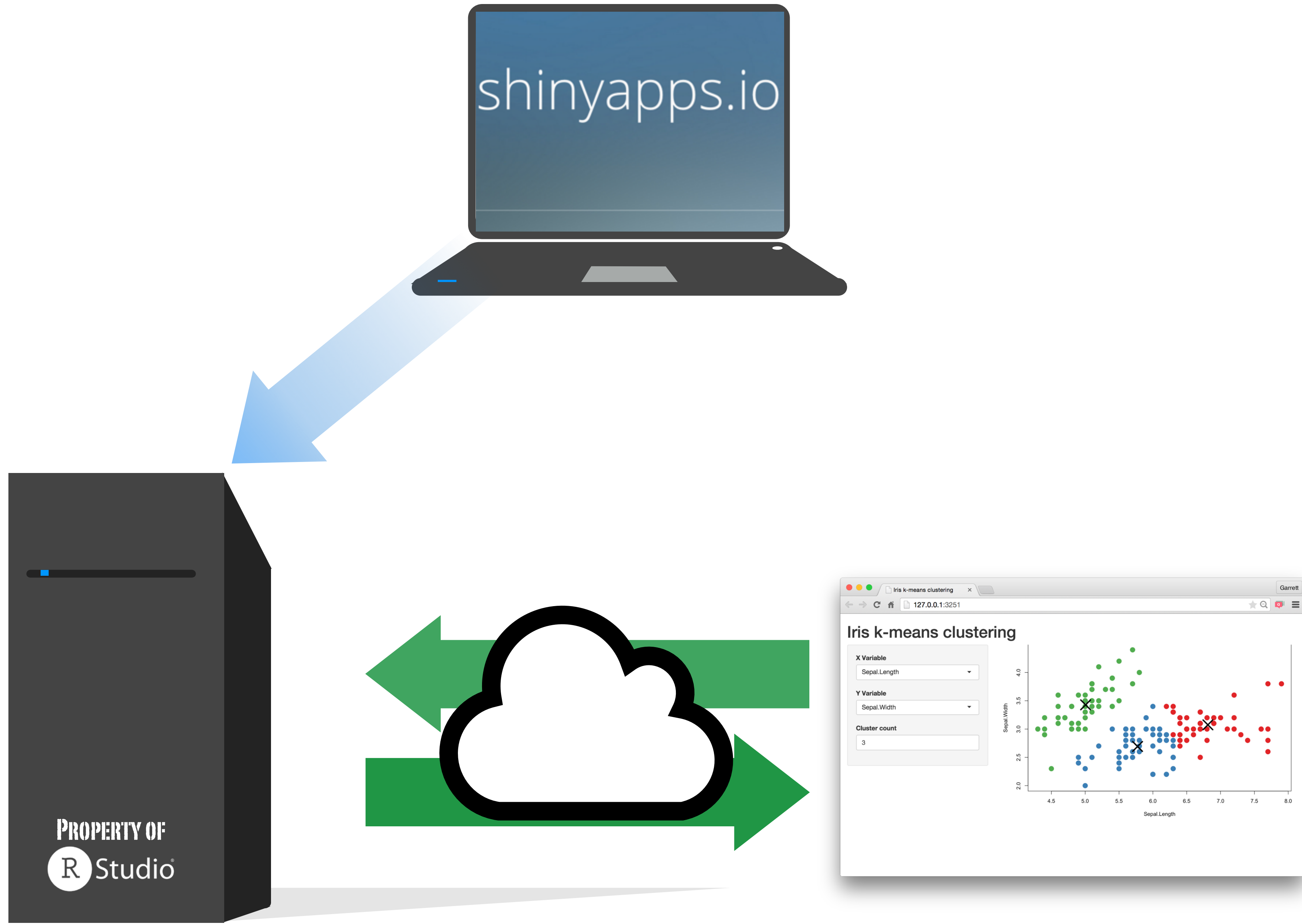


Shinyapps.io

A server maintained by RStudio

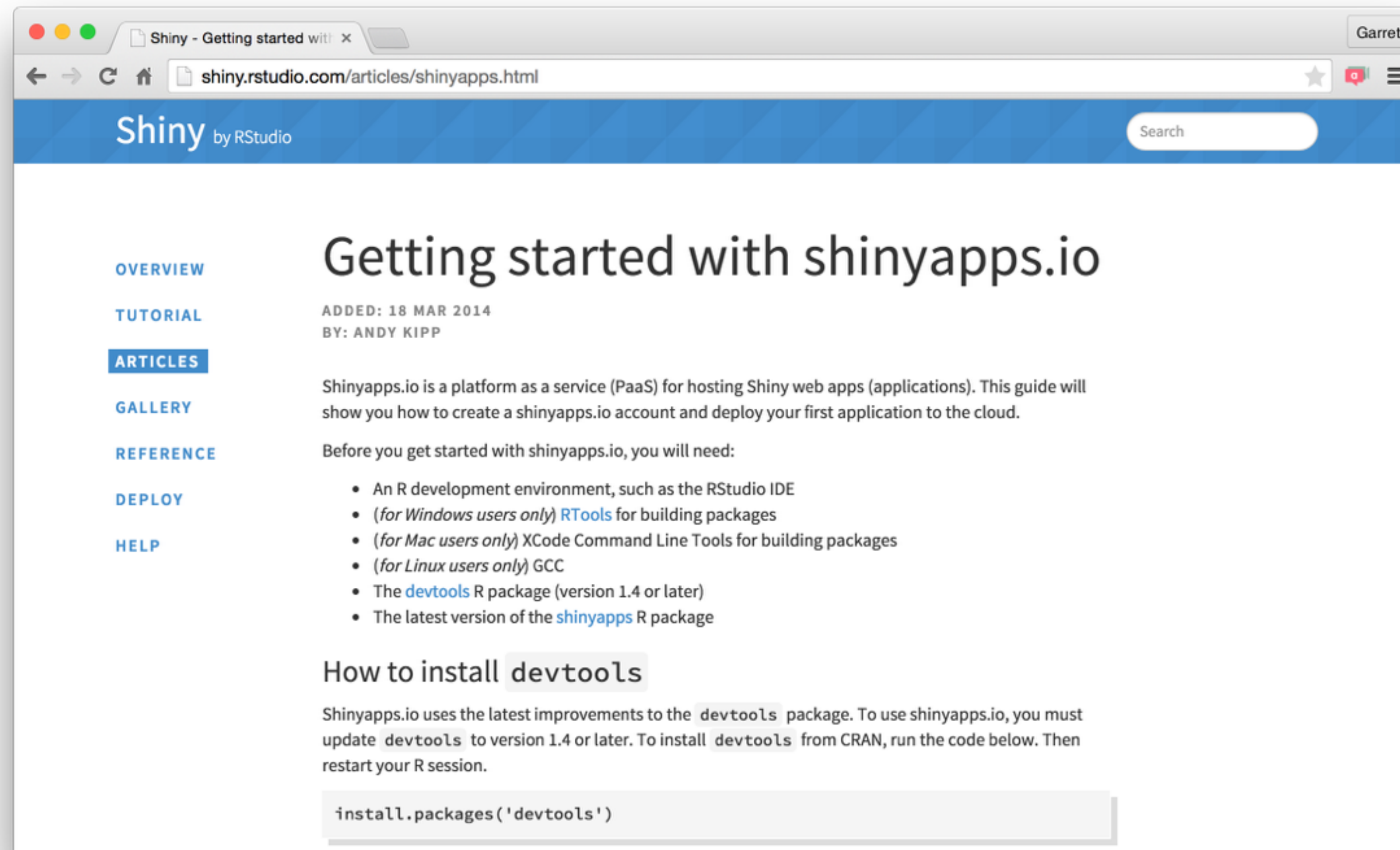
- free
- easy to use
- secure
- scalable





Getting started guide

shiny.rstudio.com/articles/shinyapps.html



The screenshot shows a web browser window displaying the article 'Getting started with shinyapps.io' on the Shiny website. The browser's address bar shows the URL 'shiny.rstudio.com/articles/shinyapps.html'. The page has a blue header with the 'Shiny by RStudio' logo and a search bar. On the left side, there is a navigation menu with links for OVERVIEW, TUTORIAL, ARTICLES (which is highlighted), GALLERY, REFERENCE, DEPLOY, and HELP. The main content area features the article title 'Getting started with shinyapps.io', the date 'ADDED: 18 MAR 2014', and the author 'BY: ANDY KIPP'. The article text explains that shinyapps.io is a PaaS for hosting Shiny web apps and provides a list of prerequisites: an R development environment (RStudio IDE, RTools for Windows, XCode for Mac, or GCC for Linux), the devtools R package (version 1.4 or later), and the latest version of the shinyapps R package. Below the prerequisites, there is a section titled 'How to install devtools' which states that shinyapps.io uses the latest improvements to the devtools package and provides instructions to update and install devtools from CRAN. At the bottom of the article, a code block shows the command 'install.packages('devtools')'.

FREE

\$0 /month

New to Shiny? Deploy your applications to the cloud for FREE. Perfect for teachers and students or those who want a place to learn and play. No credit card required.

5 Applications

25 Active Hours

✔ Community Support

❗ RStudio Branding

BASIC

\$39 /month
(or \$440/year)

Take your users' experience to the next level. shinyapps.io Basic lets you scale your application performance by adding R processes dynamically as usage increases.

Unlimited Applications

250 Active Hours

✔ Multiple Instances

✔ Email Support

STANDARD

\$99 /month
(or \$1,100/year)

Need password protection? shinyapps.io Standard lets you authenticate your application users.

Unlimited Applications

1000 Active Hours

✔ Authentication

✔ Multiple Instances

✔ Email Support

PROFESSIONAL

\$299 /month
(or \$3,300/year)

shinyapps.io Professional has it all. Share an account with others in your business or change your shinyapps.io domain into a URL of your own.

Unlimited Applications

5000 Active Hours

✔ Authentication

✔ Multiple Users

✔ Multiple Instances

✔ Custom Domains*

✔ Email Support

**Build
your own
server**



Shiny Server

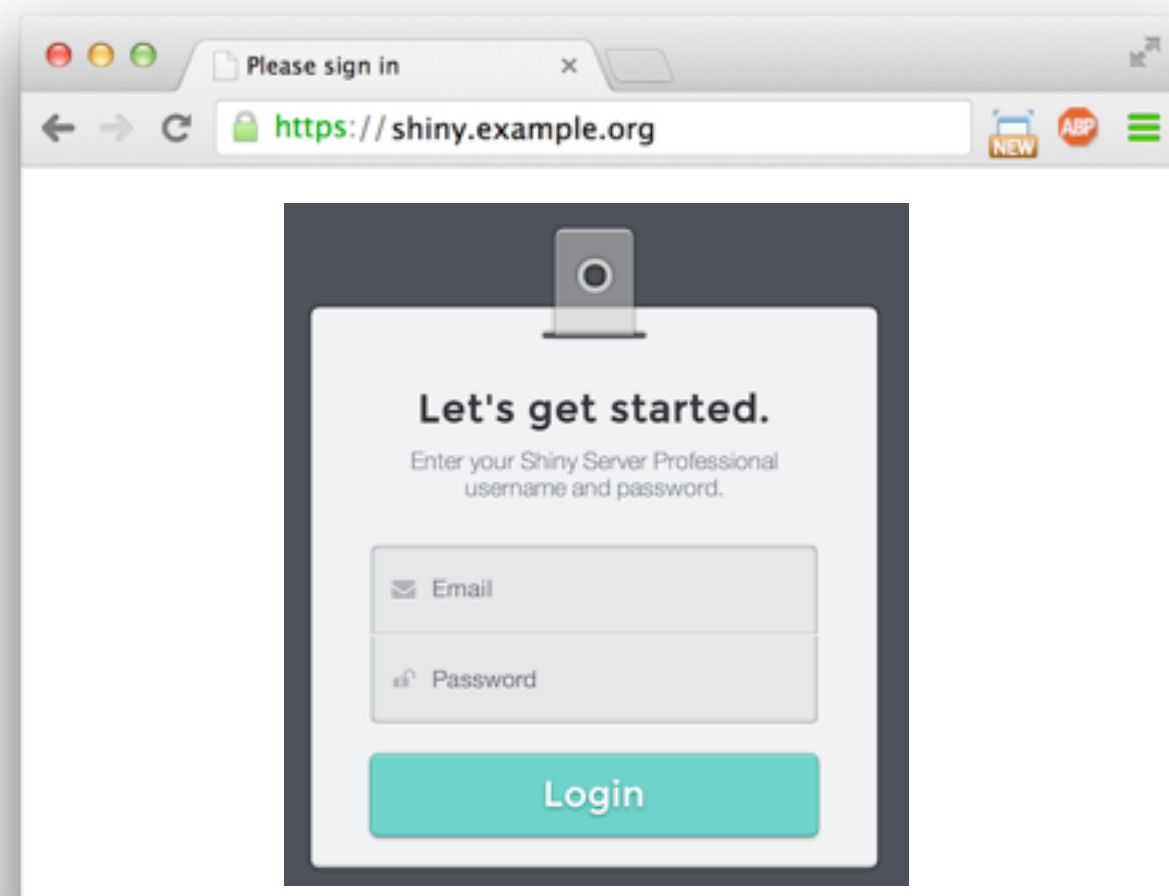
www.rstudio.com/products/shiny/shiny-server/

A back end program that builds a linux web server specifically designed to host Shiny apps.

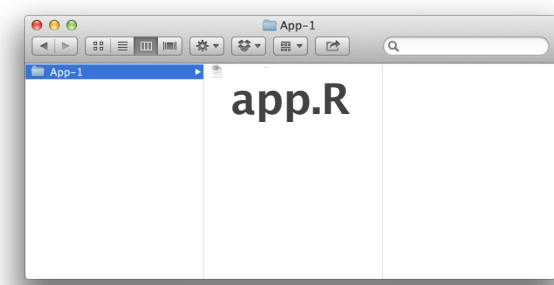
Shiny Server Pro

www.rstudio.com/products/shiny/shiny-server/

- ✓ **Secure access** - LDAP, GoogleAuth, SSL, and more
- ✓ **Performance** - fine tune at app and server level
- ✓ **Management** - monitor and control resource use
- ✓ **Support** - direct priority support



Recap: Sharing



Save your app in its own directory as **app.R**, or **ui.R** and **server.R**



Host apps at shinyapps.io by:



1. Sign up for a free shinyapps.io account

```
library(shinyapps)
```

2. Install the [shinyapps](https://github.com/rstudio/shinyapps) package



Build your own server with **Shiny Server** or **Shiny Server Pro**

Learn

more

You now how to

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

***Input()**

***Output()**

Build an app

`output$`
`render*()`
`input$`

`input$num`

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

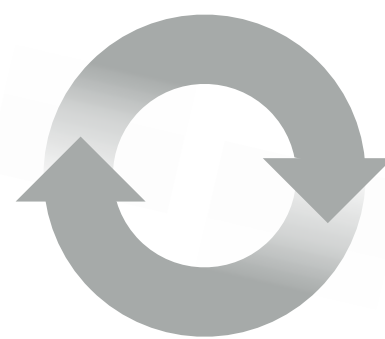
Create interactions

app.R

shinyapps.io

Share your apps

How to start with Shiny



1. How to build a Shiny app (Today)

2. How to customize reactions (May 27)

3. How to customize appearance (June 3)

The Shiny Development Center

shiny.rstudio.com

