RStudio

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212

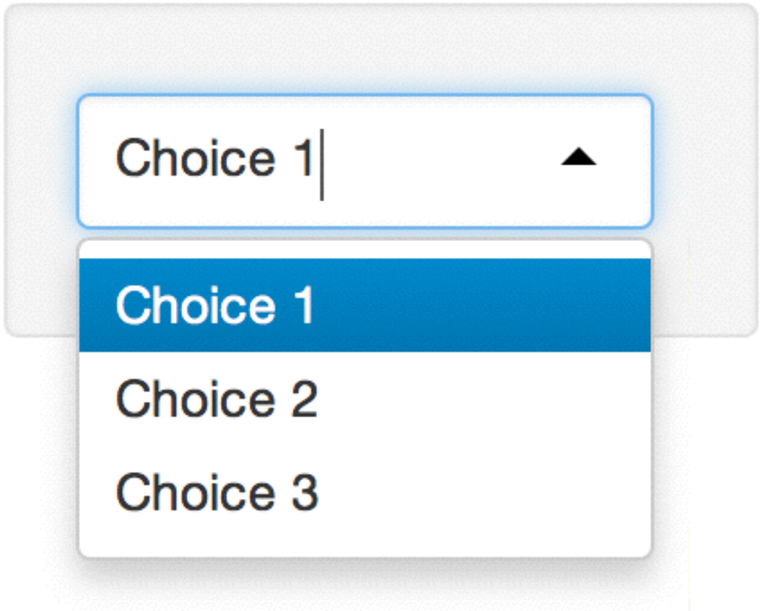
Email: info@rstudio.com
Web: http://www.rstudio.com

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.

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212
Email: info@rstudio.com
Web: http://www.rstudio.com

How to start with Shiny, Part 1

How to build a Shiny App



Garrett Grolemund

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

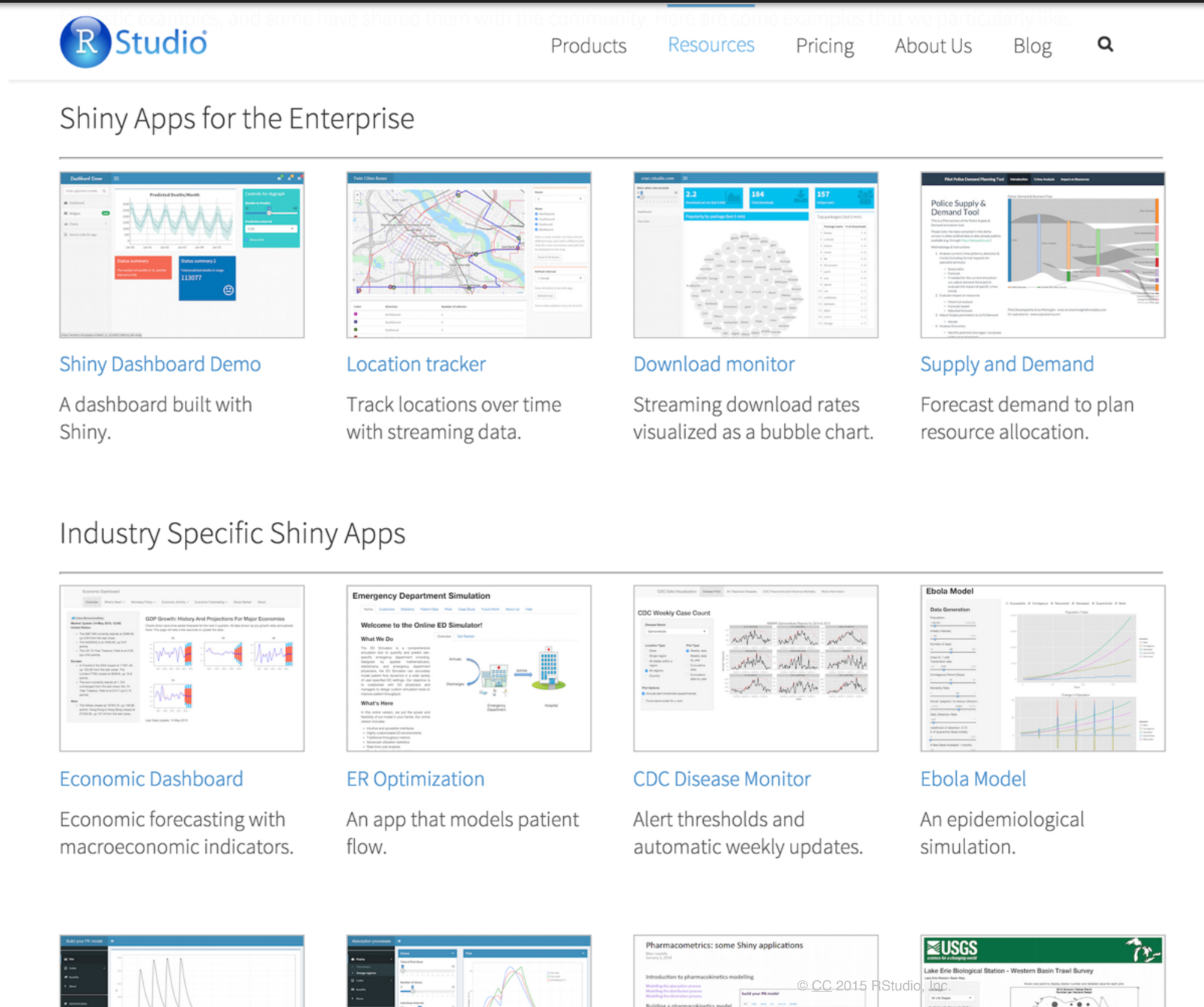
1. How to build a Shiny app (Today)

2. How to customize reactions (May 27)

3. How to customize appearance (June 3)

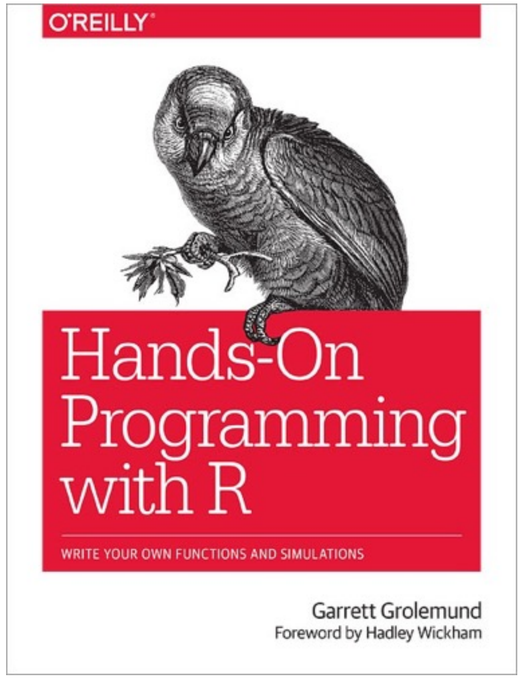
Shiny Showcase

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

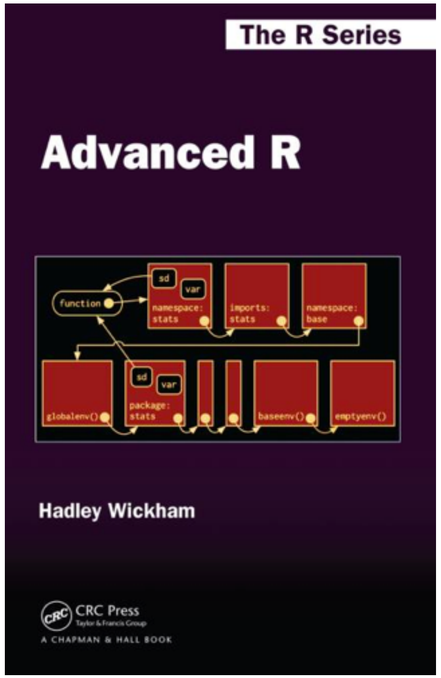


Learn R

Books

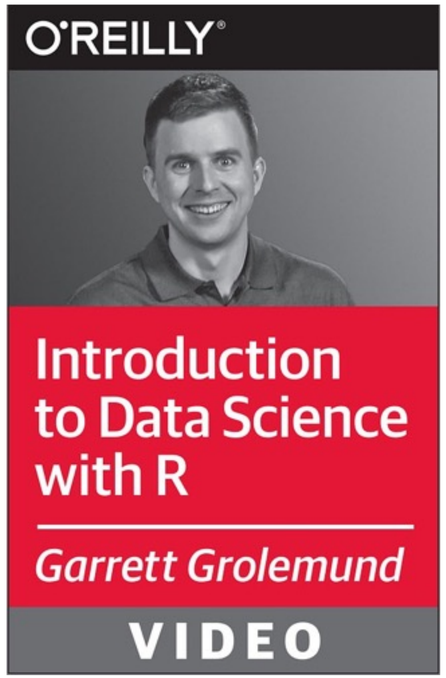


shop.oreilly.com/product/0636920028574.do

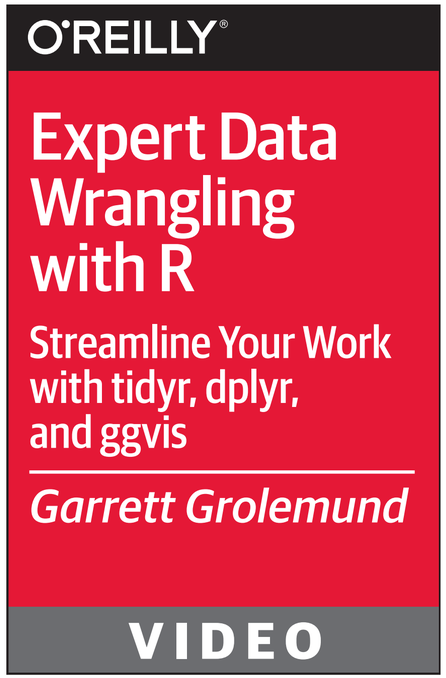


adv-r.had.co.nz/

Videos



shop.oreilly.com/product/0636920034834.do



shop.oreilly.com/product/0636920035992.do

Interactive tutorials

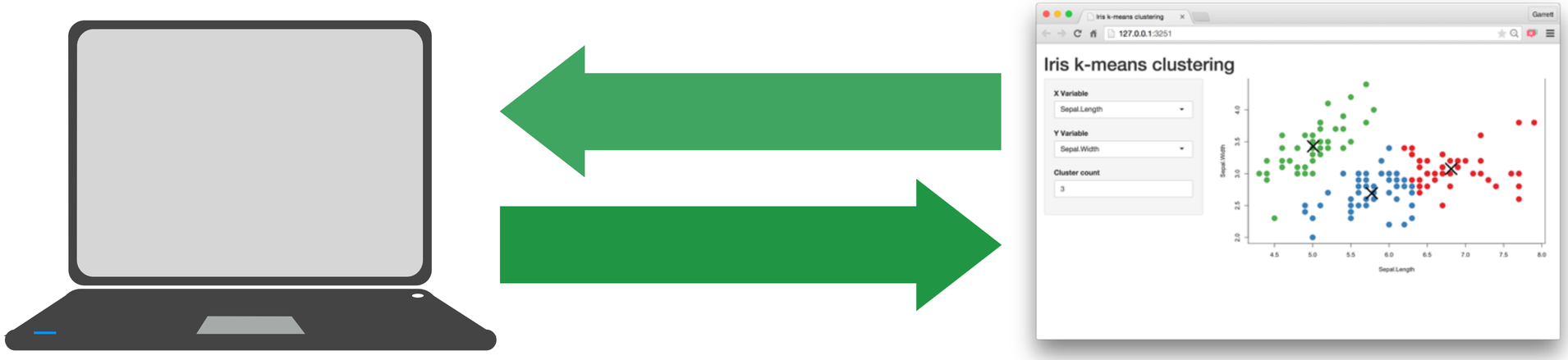
DataCamp

www.datacamp.com

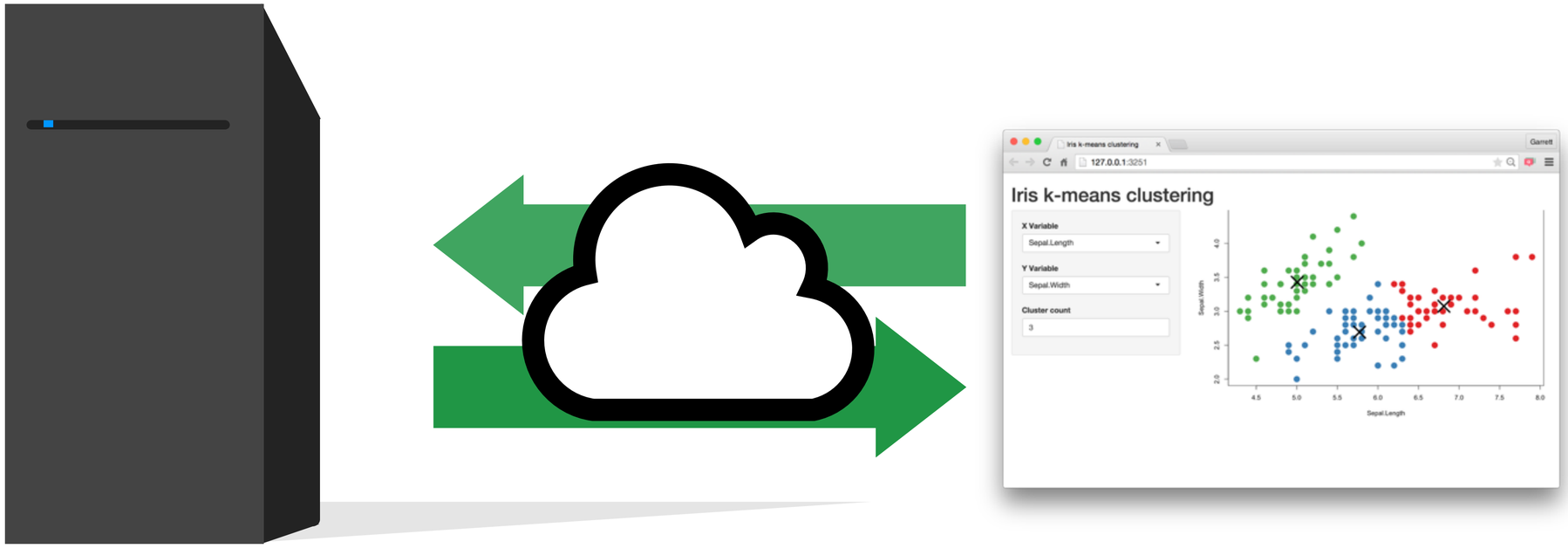
More at
www.rstudio.com/resources/training/online-learning/

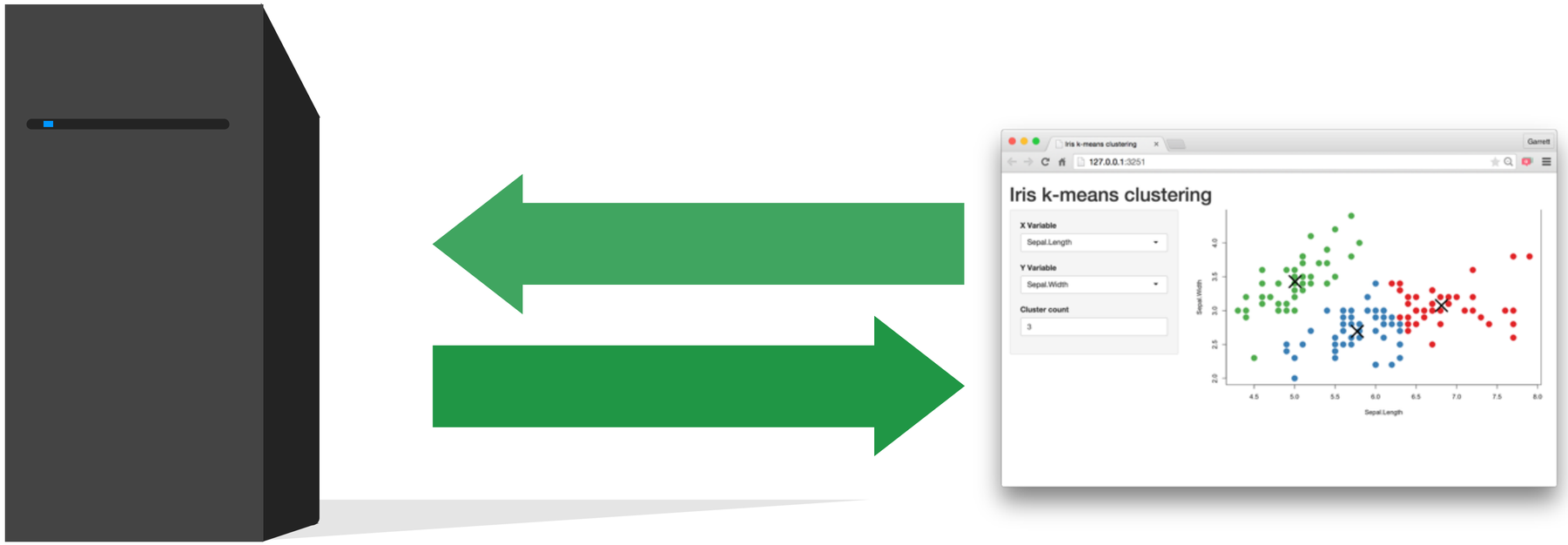
Understand the architecture

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)

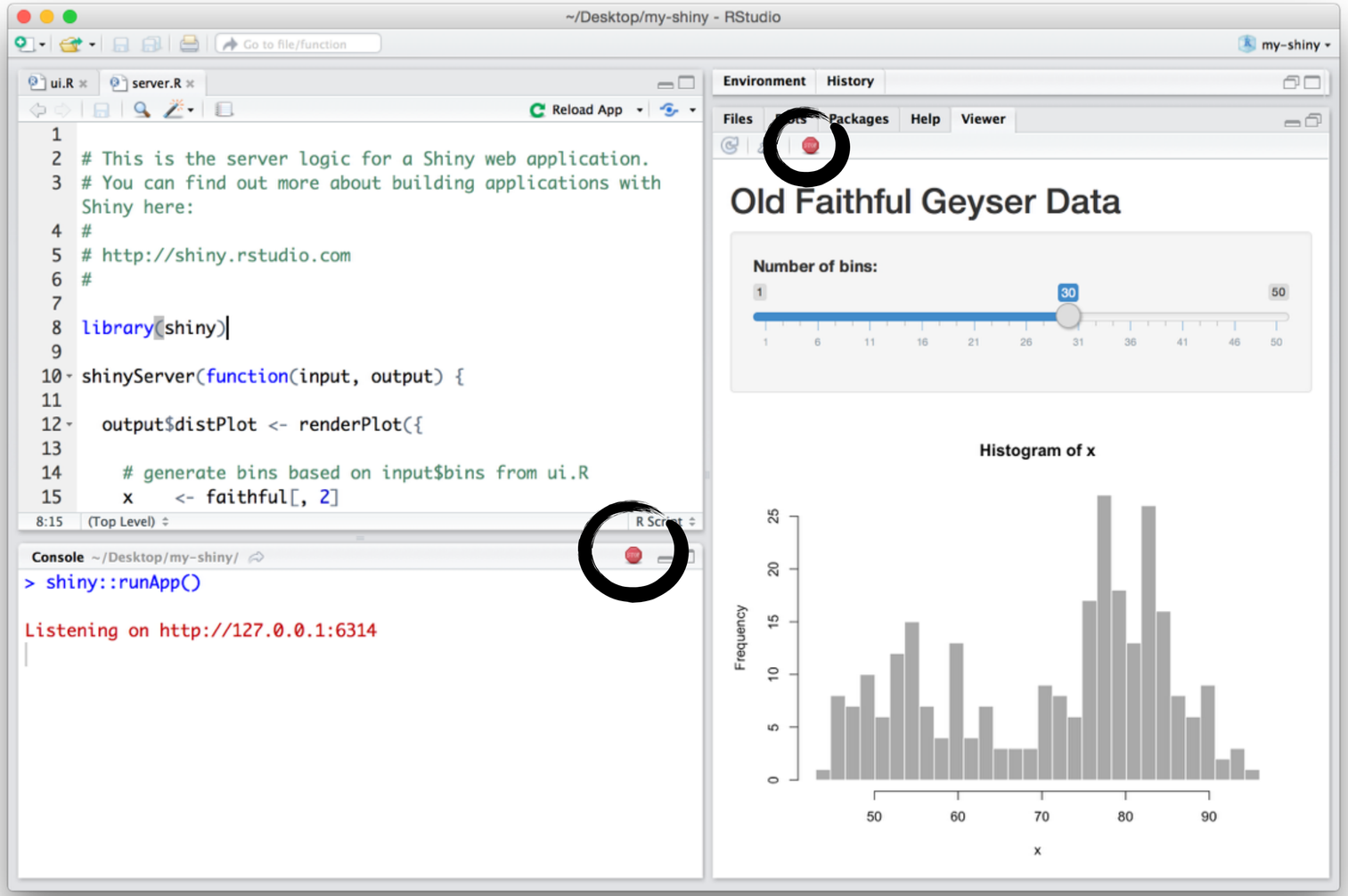
Use the template

App template

The shortest viable shiny app

```
library(shiny)

ui <- fluidPage()


server <- function(input, output) {}


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

Close an app

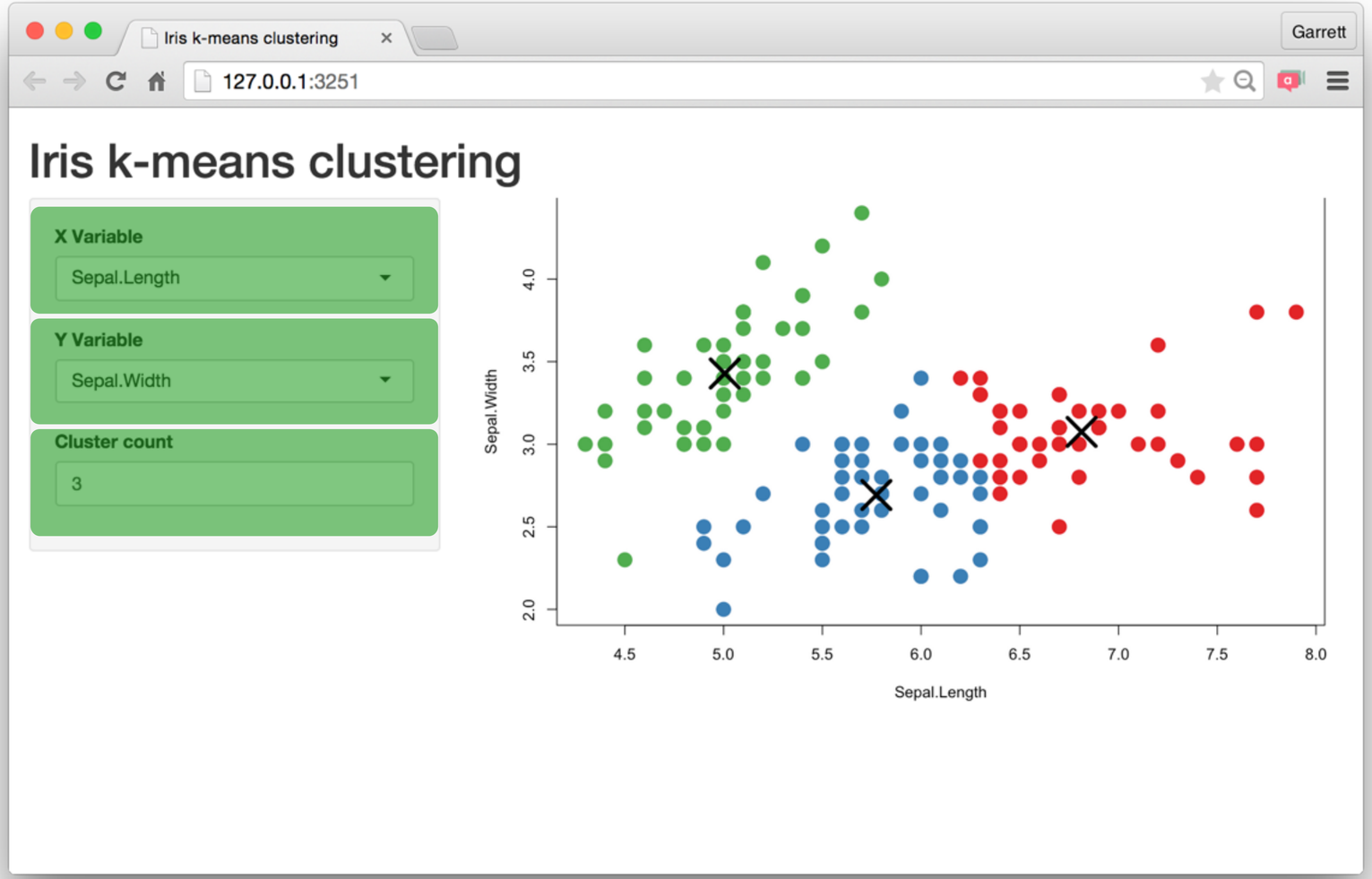


Add elements to your app as arguments to `fluidPage()`

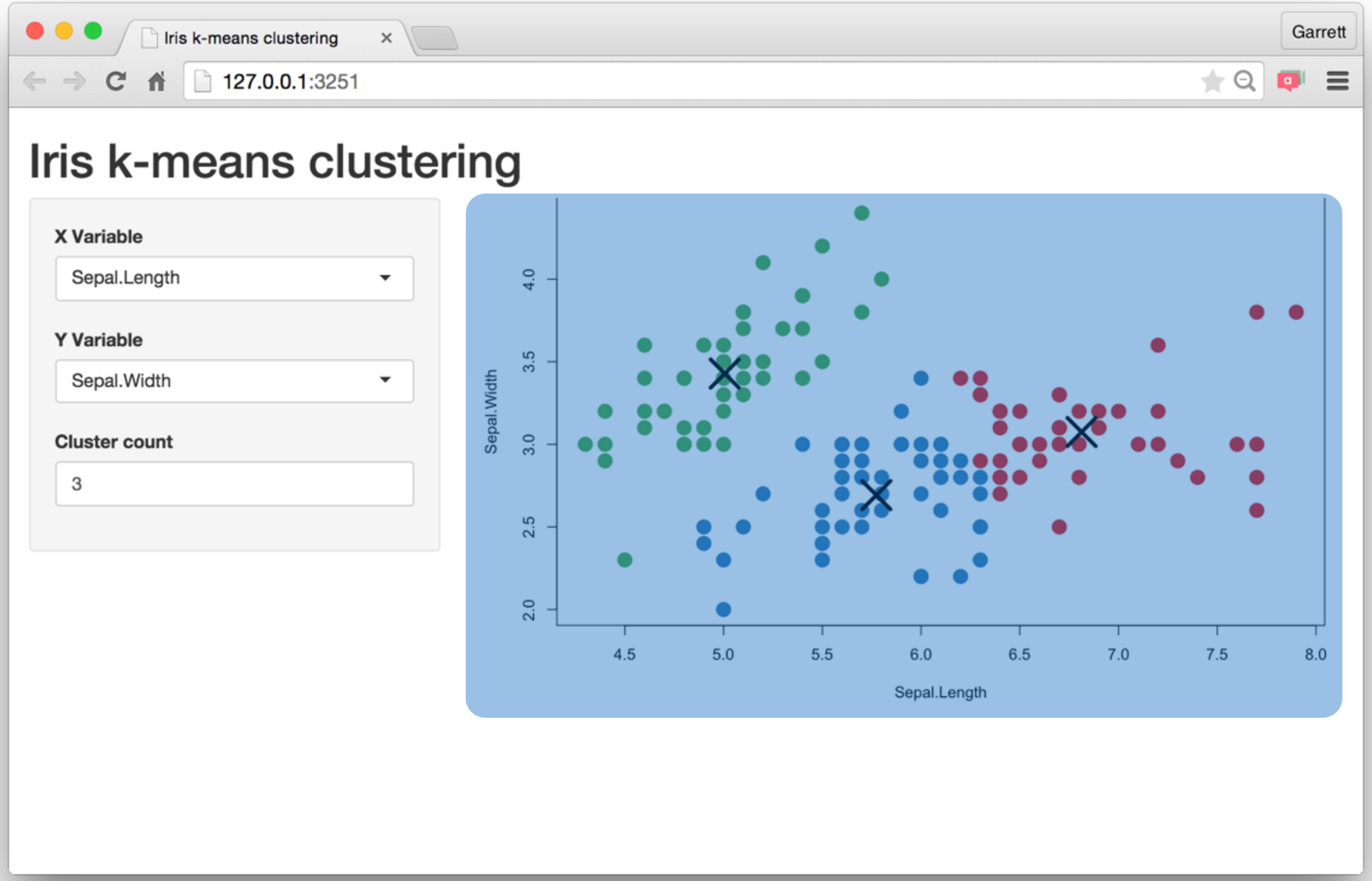
```
library(shiny)
ui <- fluidPage("Hello World")

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

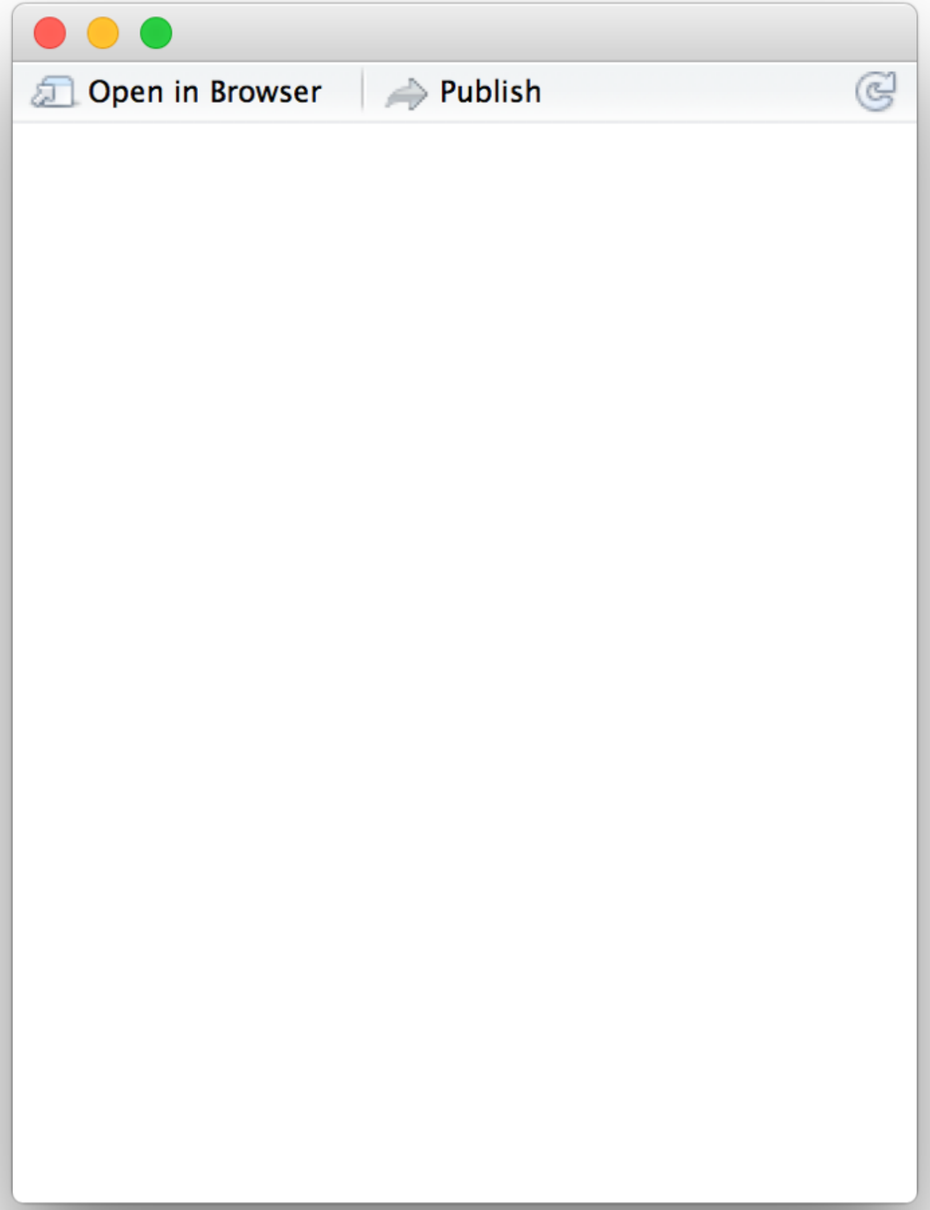
Create an input with an ***Input()** function.

```
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-prettify-separator="," data-keyboard="true"
   data-keyboard-step="1.01010101010101"/>
</div>
```

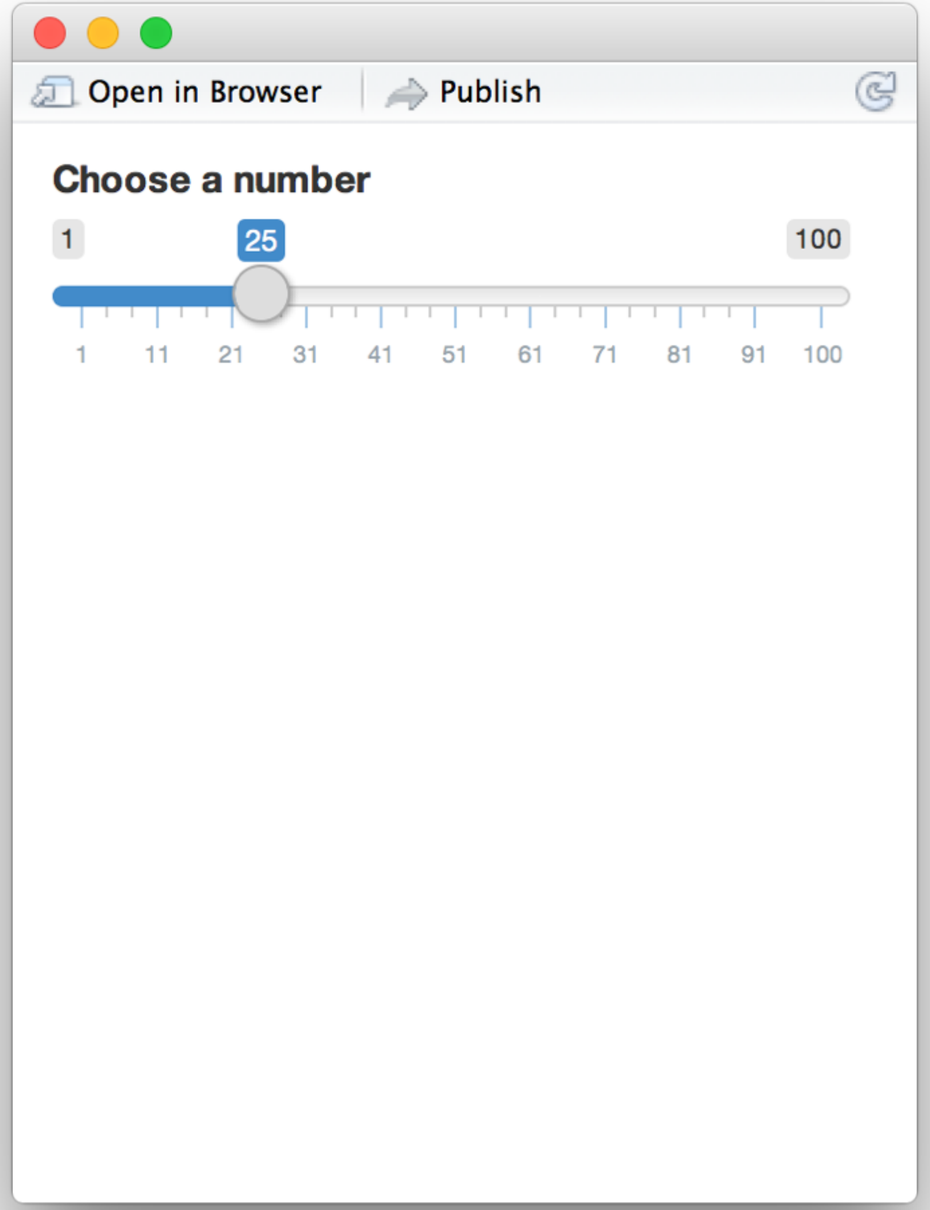
Create an input with an input function.

```
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

Action

Submit

actionButton()
submitButton()

Single checkbox

Choice A

checkboxInput()

Checkbox group

Choice 1
Choice 2
Choice 3

checkboxGroupInput()

Date input

2014-01-01

dateInput()

Date range

2014-01-24 to 2014-01-24

dateRangeInput()

File input

Choose File No file chosen

fileInput()

Numeric input

1

numericInput()

Password Input

passwordInput()

Radio buttons

Choice 1
Choice 2
Choice 3

radioButtons()

Select box

Choice 1

selectInput()

Sliders

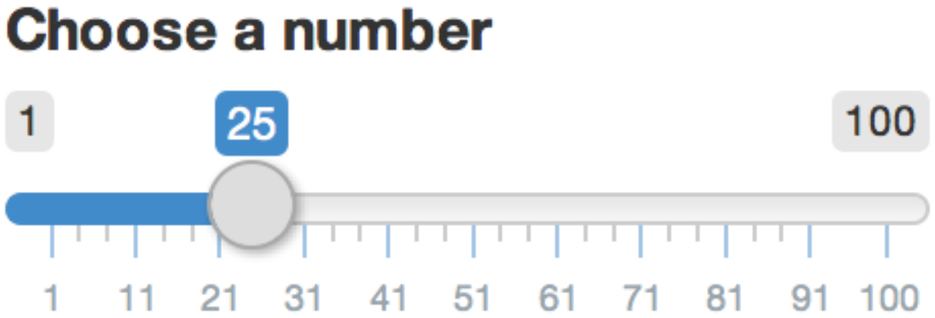
sliderInput()

Text input

Enter text...

textInput()

Syntax




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

input name
(for internal use)

Notice:
Id not ID

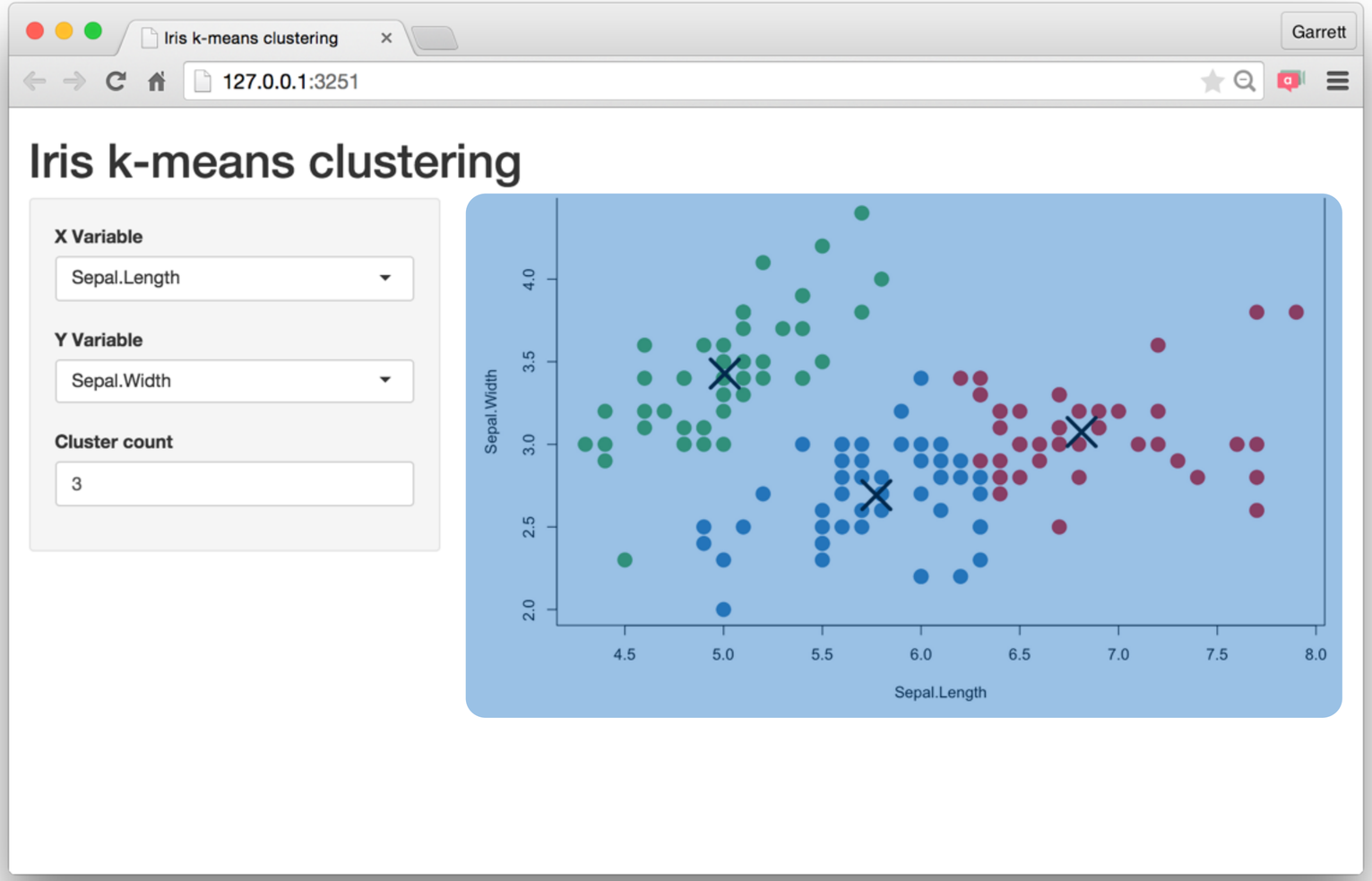
label to display

input specific arguments

```
?sliderInput
```

Outputs

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



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

*Output()

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

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

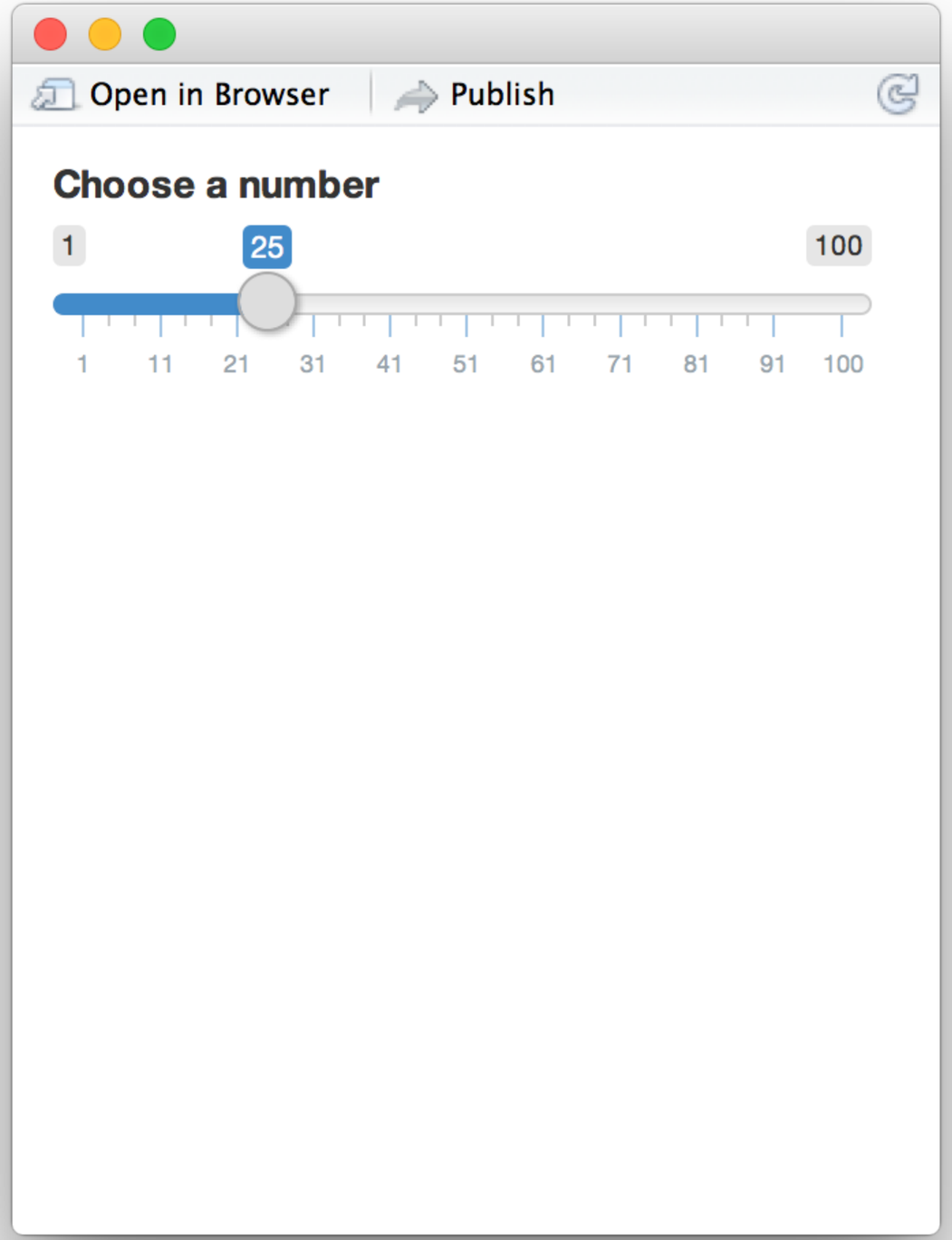
the type of output to display

name to give to the output object

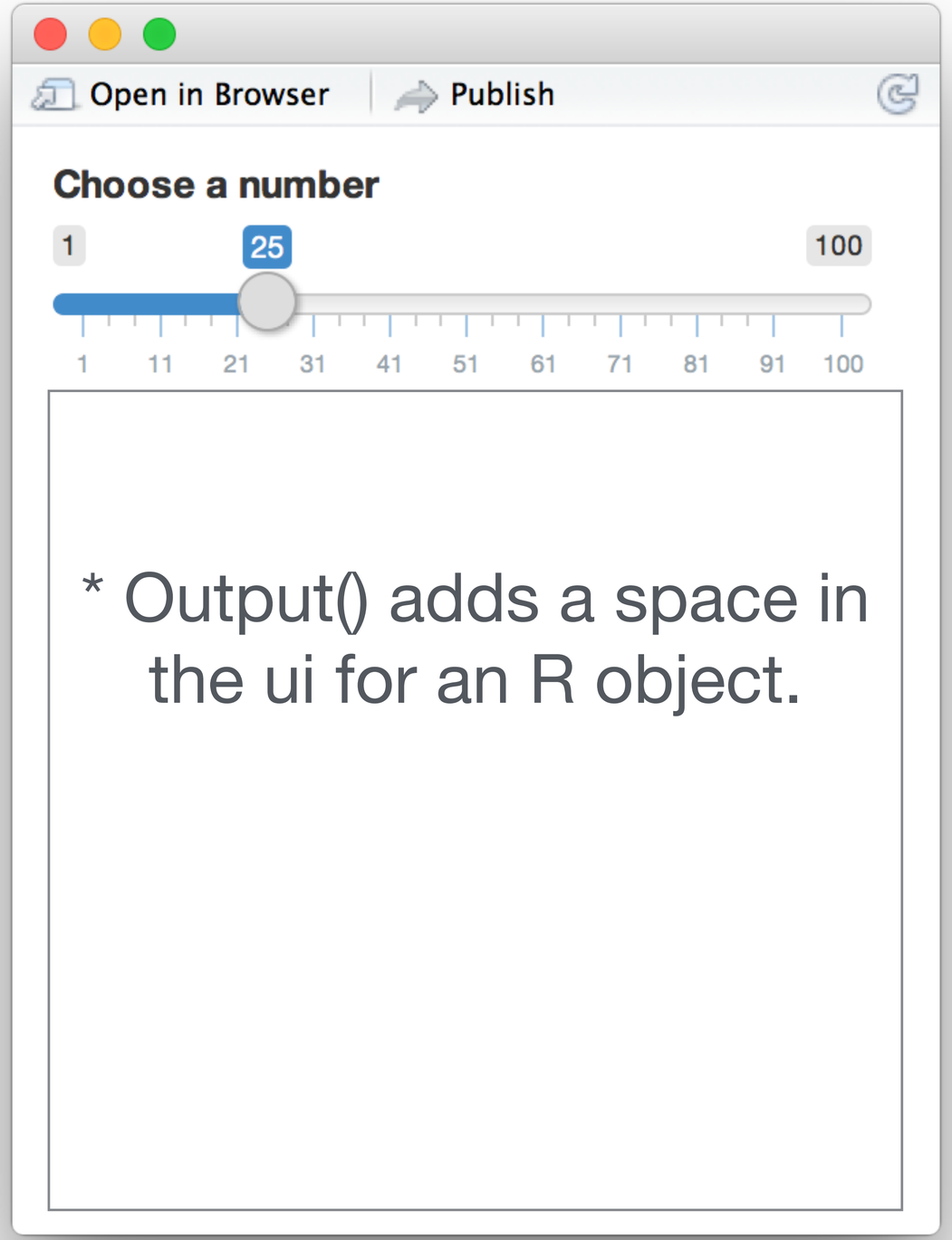
```
library(shiny)

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

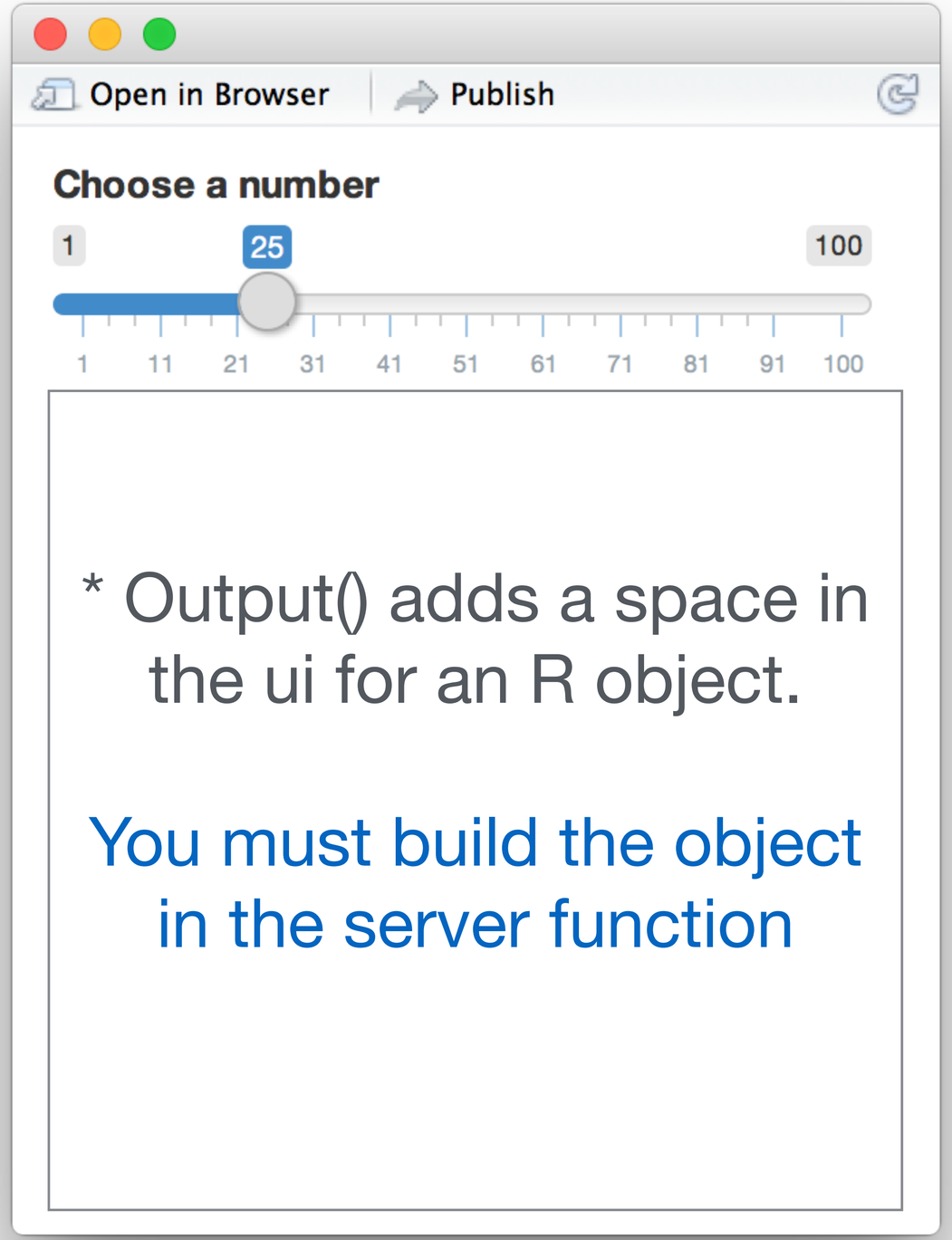
```
library(shiny)

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)
```



```
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)
```

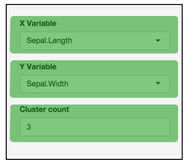


Recap

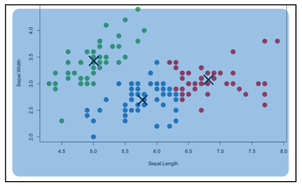
```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

Begin each app with the template

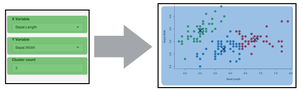
Add elements as arguments to **fluidPage()**



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



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



Assemble outputs from inputs in the server function

Tell the

server

how to assemble
inputs into outputs

Use 3 rules to write the server function

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




}
```

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
`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

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))
  })
}
```

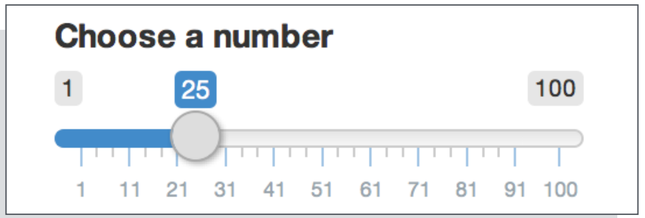
3 Access **input** values with input$

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

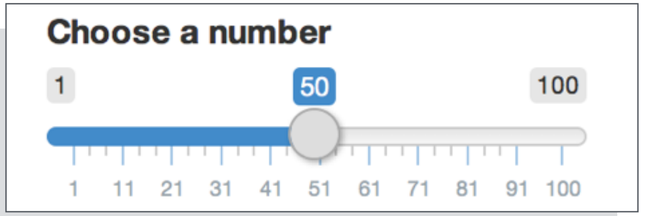
```
input$num
```

Input values

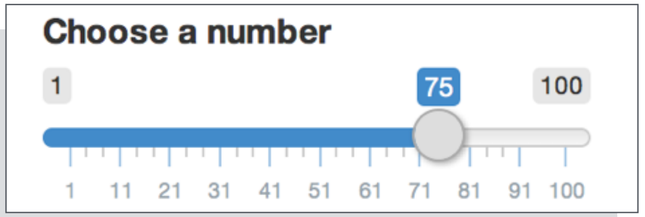
The input value changes whenever a user changes the input.




```
input$num = 25
```



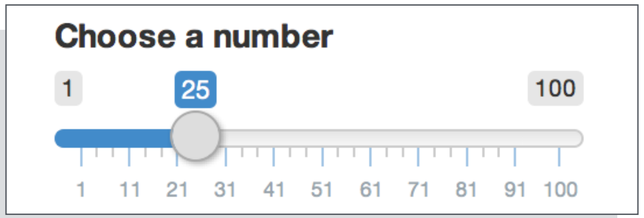

```
input$num = 50
```



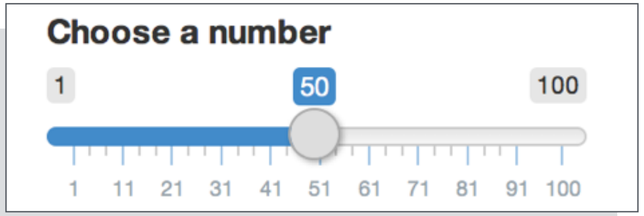

```
input$num = 75
```

Input values

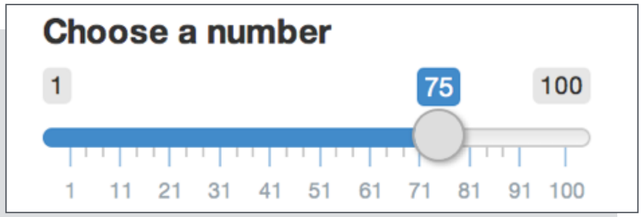
The input value changes whenever a user changes the input.




```
input$num = 25
```




```
input$num = 50
```




```
input$num =
```

Output will automatically update if you follow the 3 rules

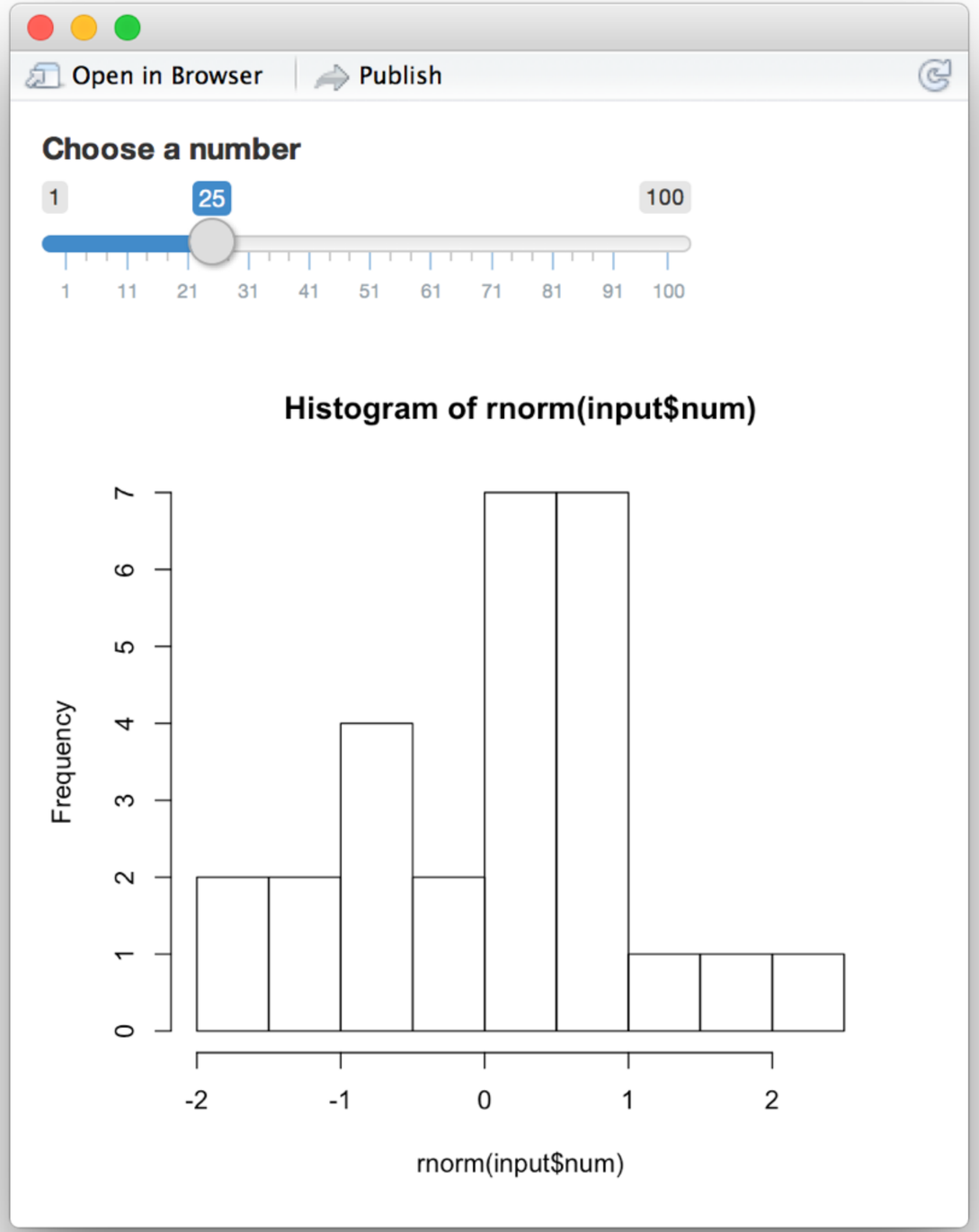
Reactivity 101

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

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

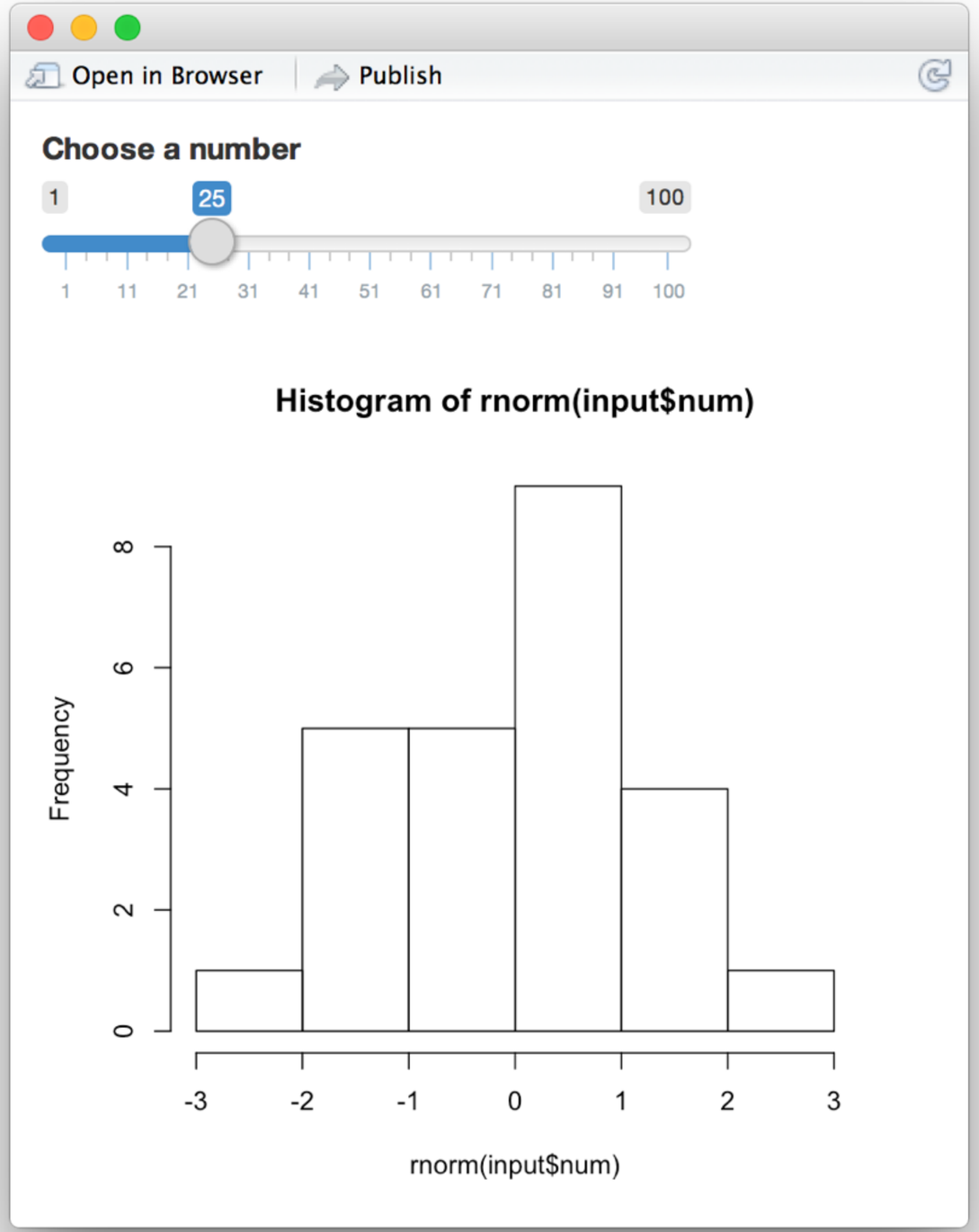
input$num

```
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 <-`

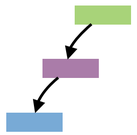
1. Save the output that you build to **output$**

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

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

`input$num`

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

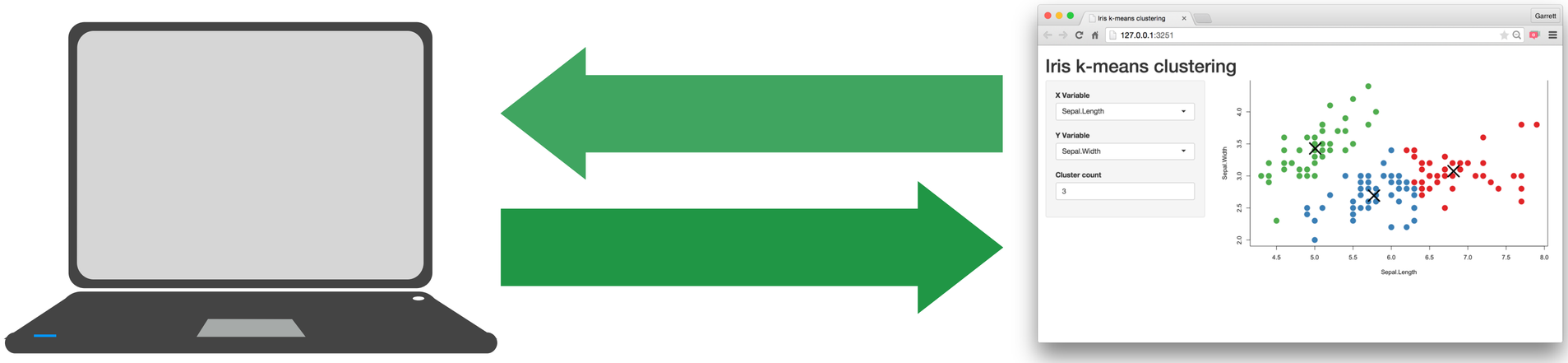


Create reactivity by using **Inputs** to build **rendered Outputs**

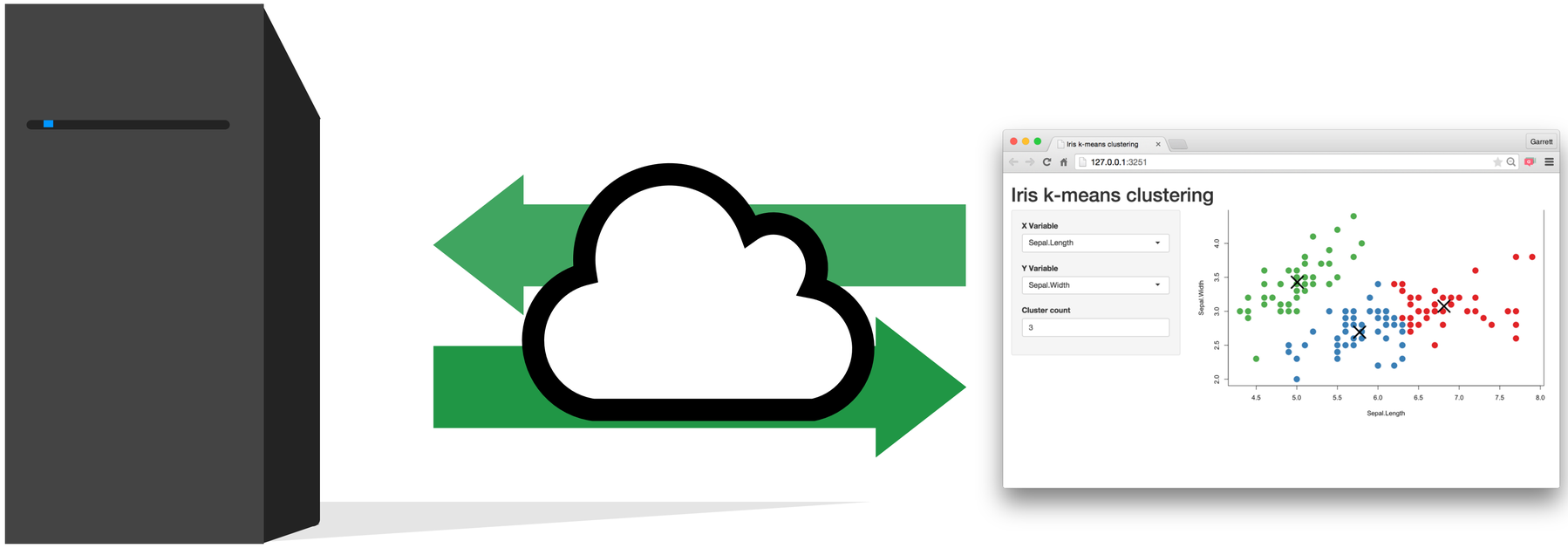
Share

your app

Every Shiny app is maintained by a computer running R



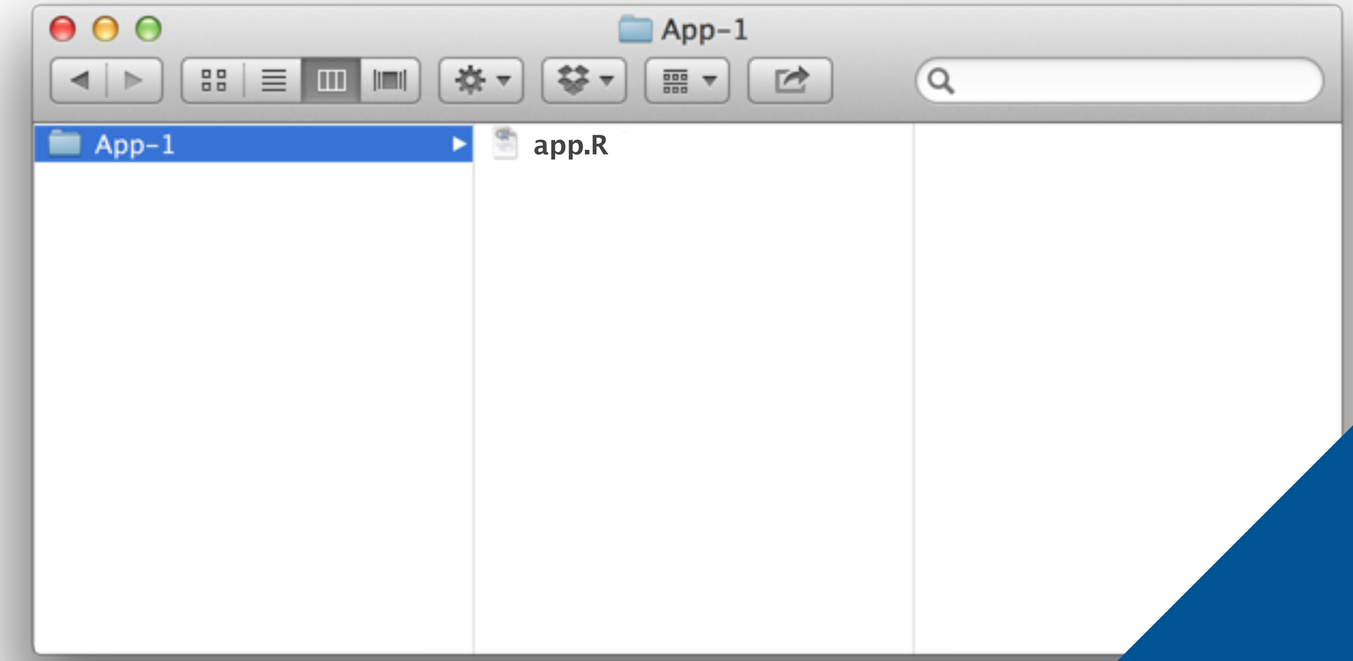
Every Shiny app is maintained by a computer running R



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)
```

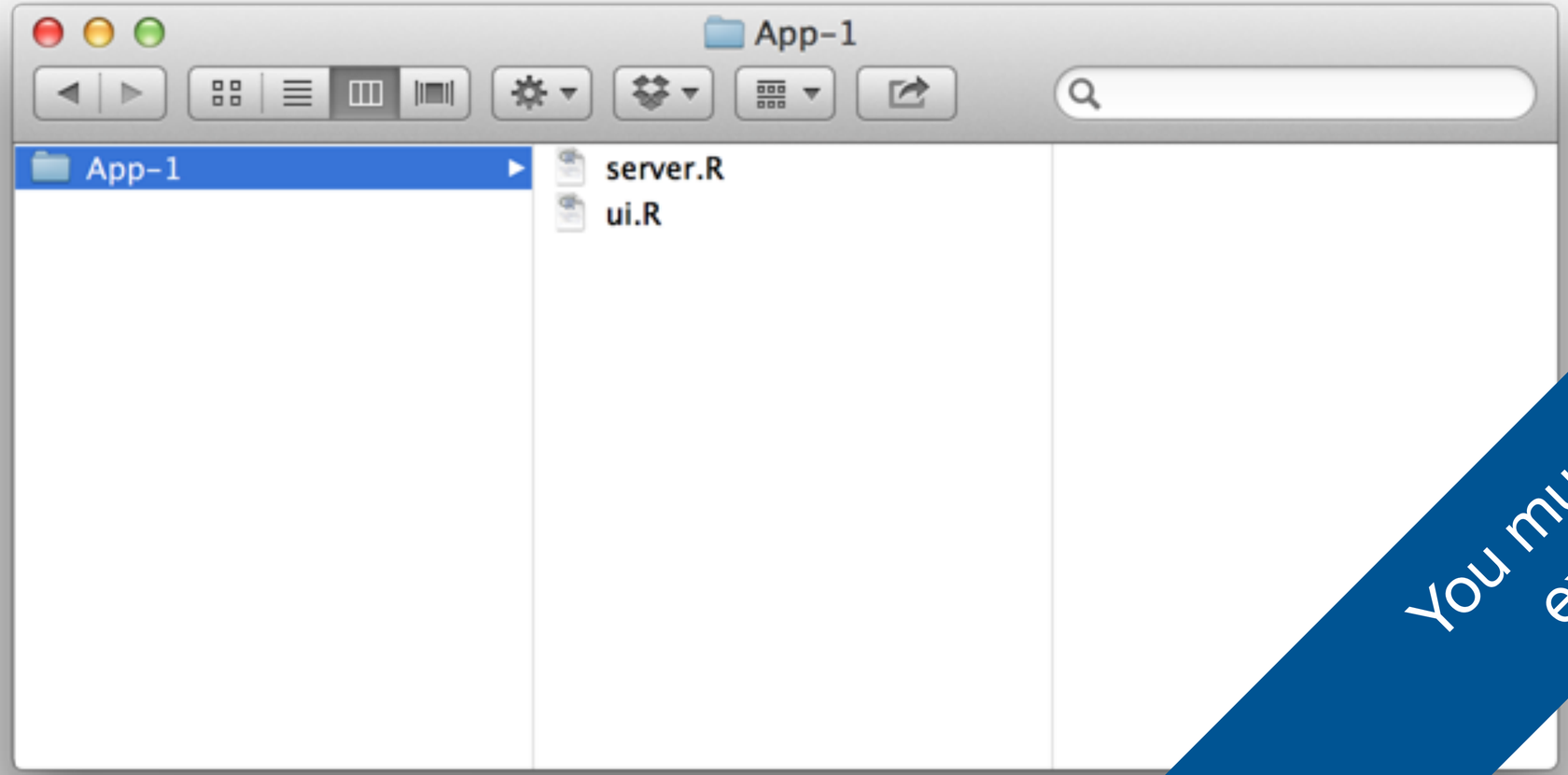
```
# ui.R
library(shiny)
fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)
```

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

Two file apps

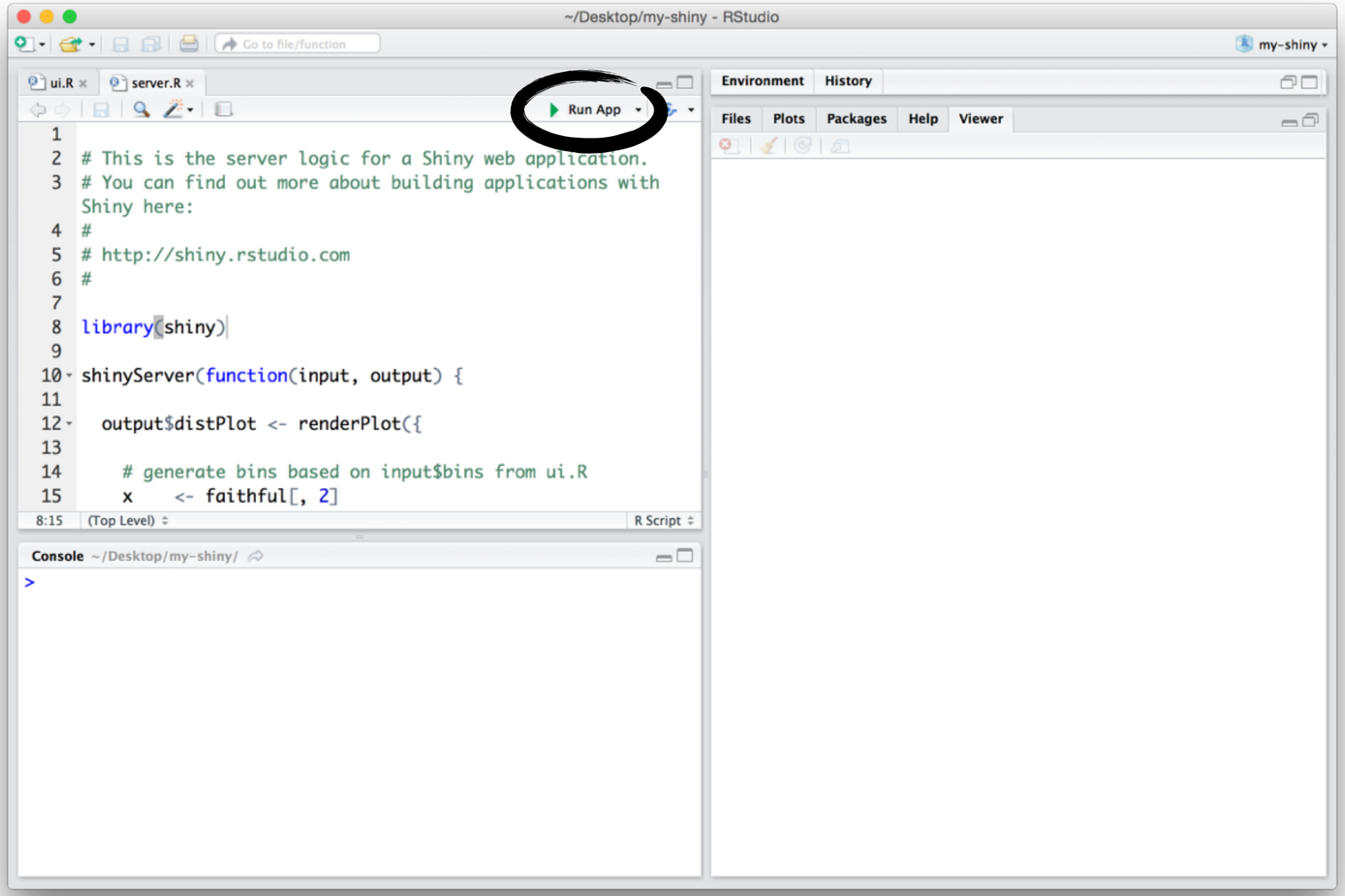
One directory with two files:

- server.R
- ui.R

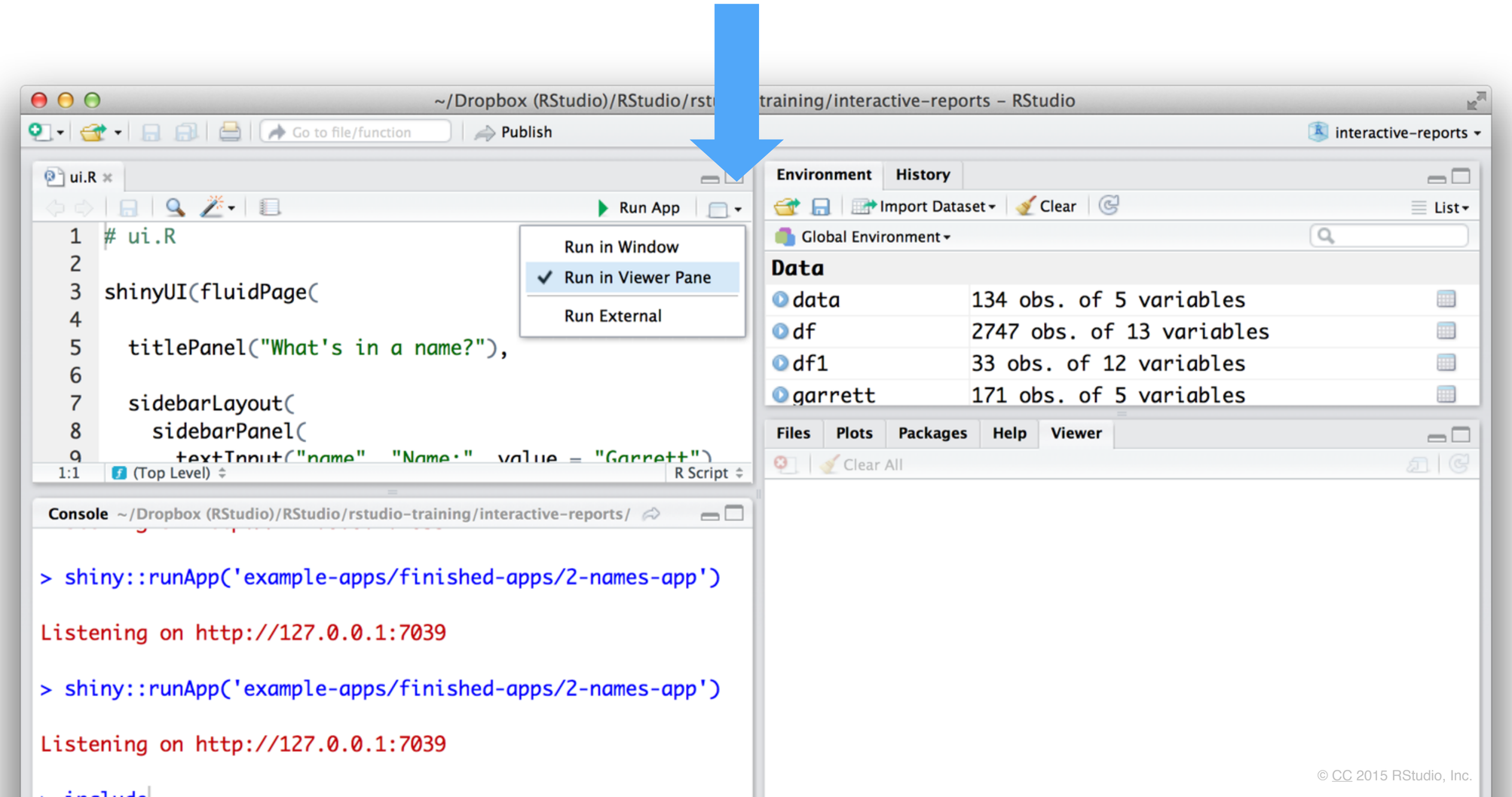


You must use these exact names

Launch an app



Display options

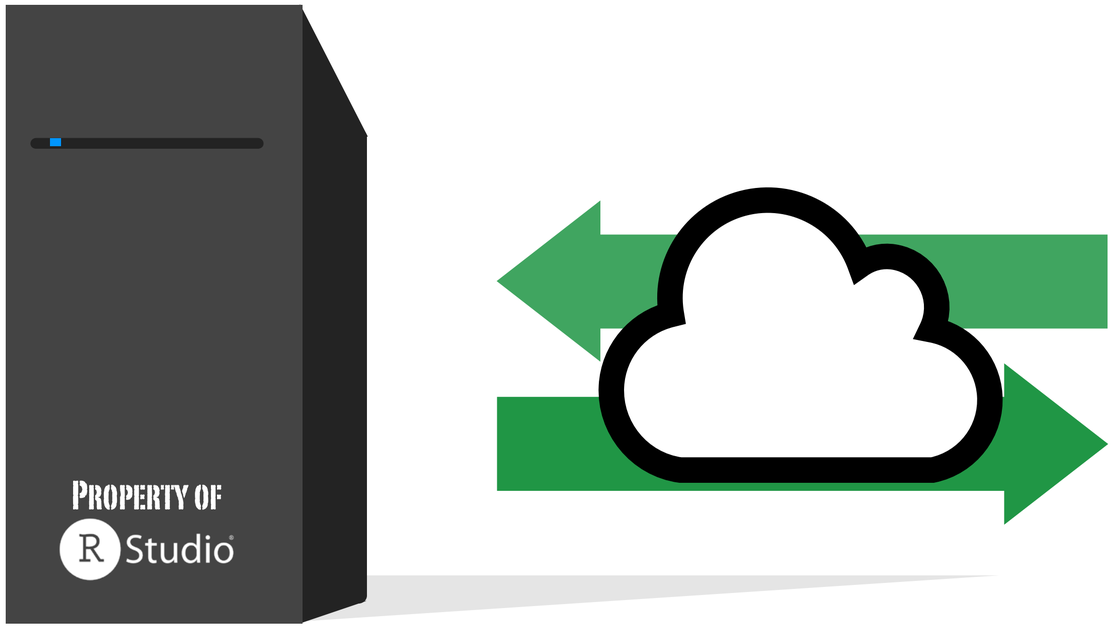


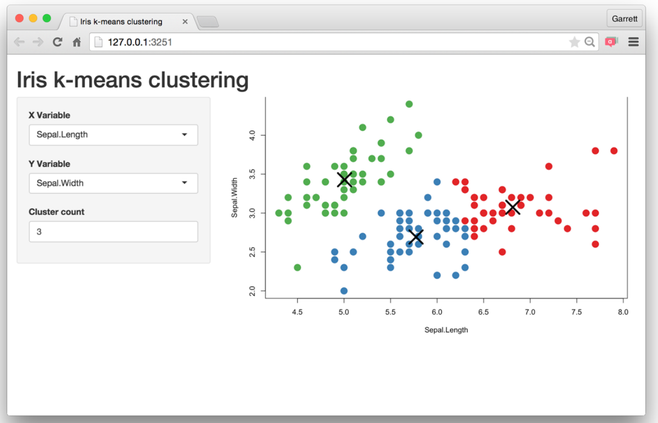
Use shinyapps.io

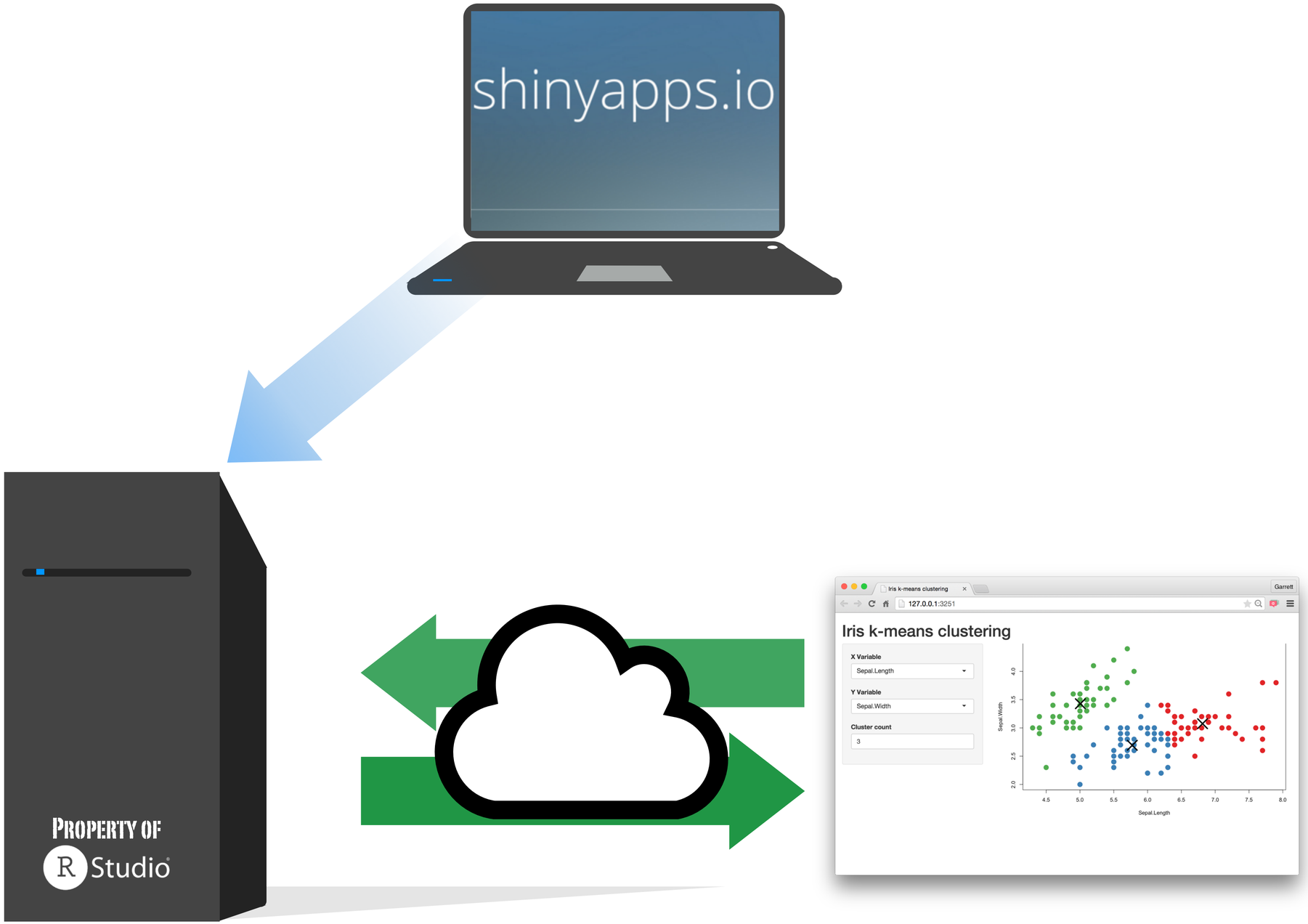
Shinyapps.io

A server maintained by RStudio

- free
- easy to use
- secure
- scalable

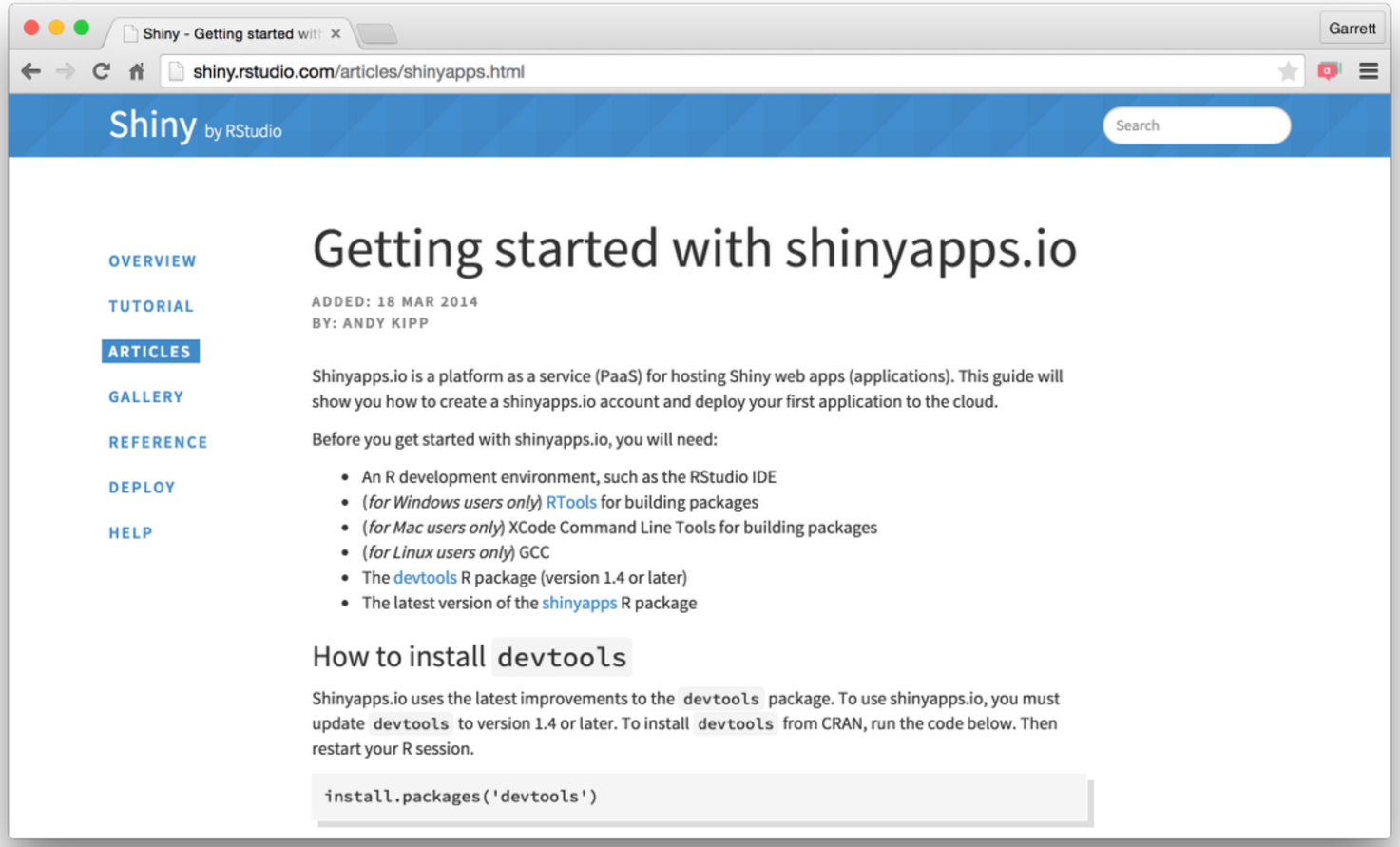






Getting started guide

shiny.rstudio.com/articles/shinyapps.html



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

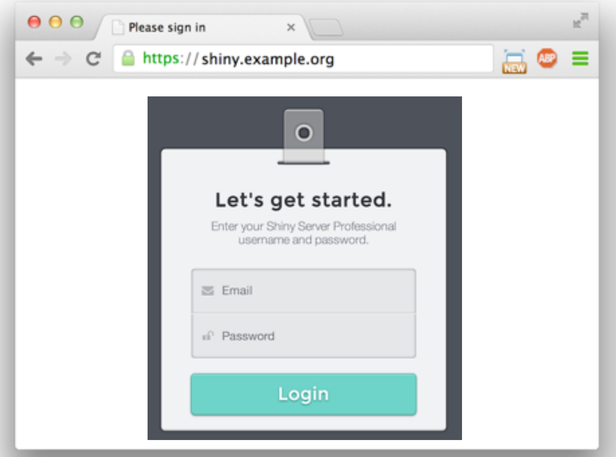
www.rstudio.com/products/shiny/shiny-server/

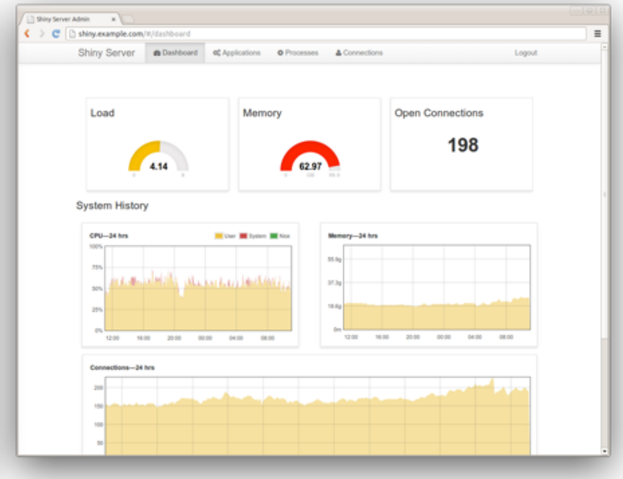
A back end program that builds a linux web server specifically designed to host Shiny apps.

Shiny Server **Pro**

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

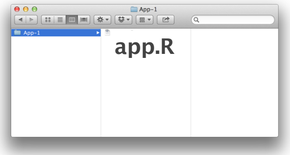
- **Secure access** - LDAP, GoogleAuth, SSL, and more
- **Performance** - fine tune at app and server level
- **Management** - monitor and control resource use
- **Support** - direct priority support





45 day evaluation free trial

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

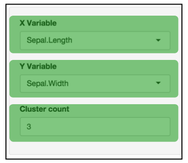
```
library(shinyapps)
```

2. Install the **`shinyapps`** package

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

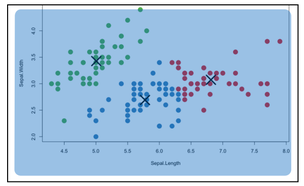
Learn more

You now how to



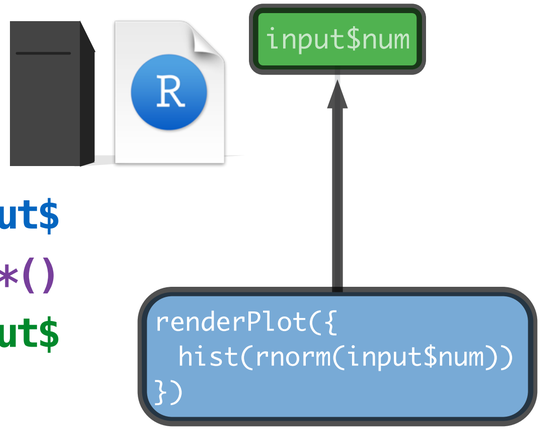
*Input()

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



*Output()

Build an app



Create interactions

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