



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.

Analytic Web Applications with Shiny

Introduction and possibilities

1. Please download slides, and code from

bit.ly/rday-strata14

2. Please ensure that you have up to date versions of R and RStudio

3. Please install the following R packages:

`install.packages(c("ggplot2", "shiny", "dplyr"))`

Strata+ Hadoop WORLD

PRESENTED BY

O'REILLY®

cloudera



250 Northern Ave, Boston, MA 02210

Phone: 844-448-1212

Email: info@rstudio.com

Web: <http://www.rstudio.com>

Copyright 2014 RStudio | All Rights Reserved

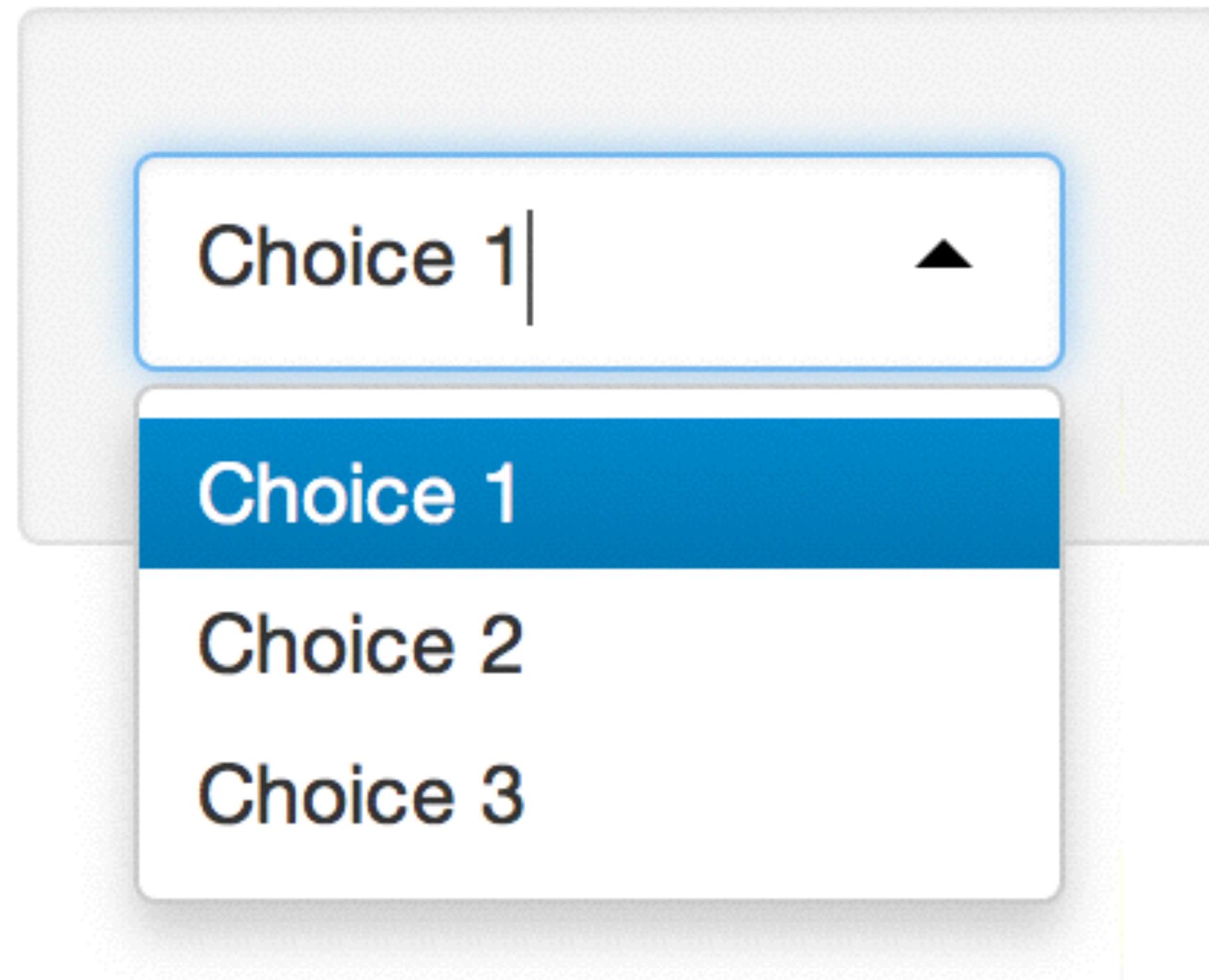
strataconf.com

#strataconf

#hadoopworld

Analytic Web Applications with Shiny

Introduction and possibilities



Garrett Grolemund

Data Scientist and Master Instructor

Email: garrett@rstudio.com

Follow [@StatGarrett](#)

HELLO
my name is

Garrett

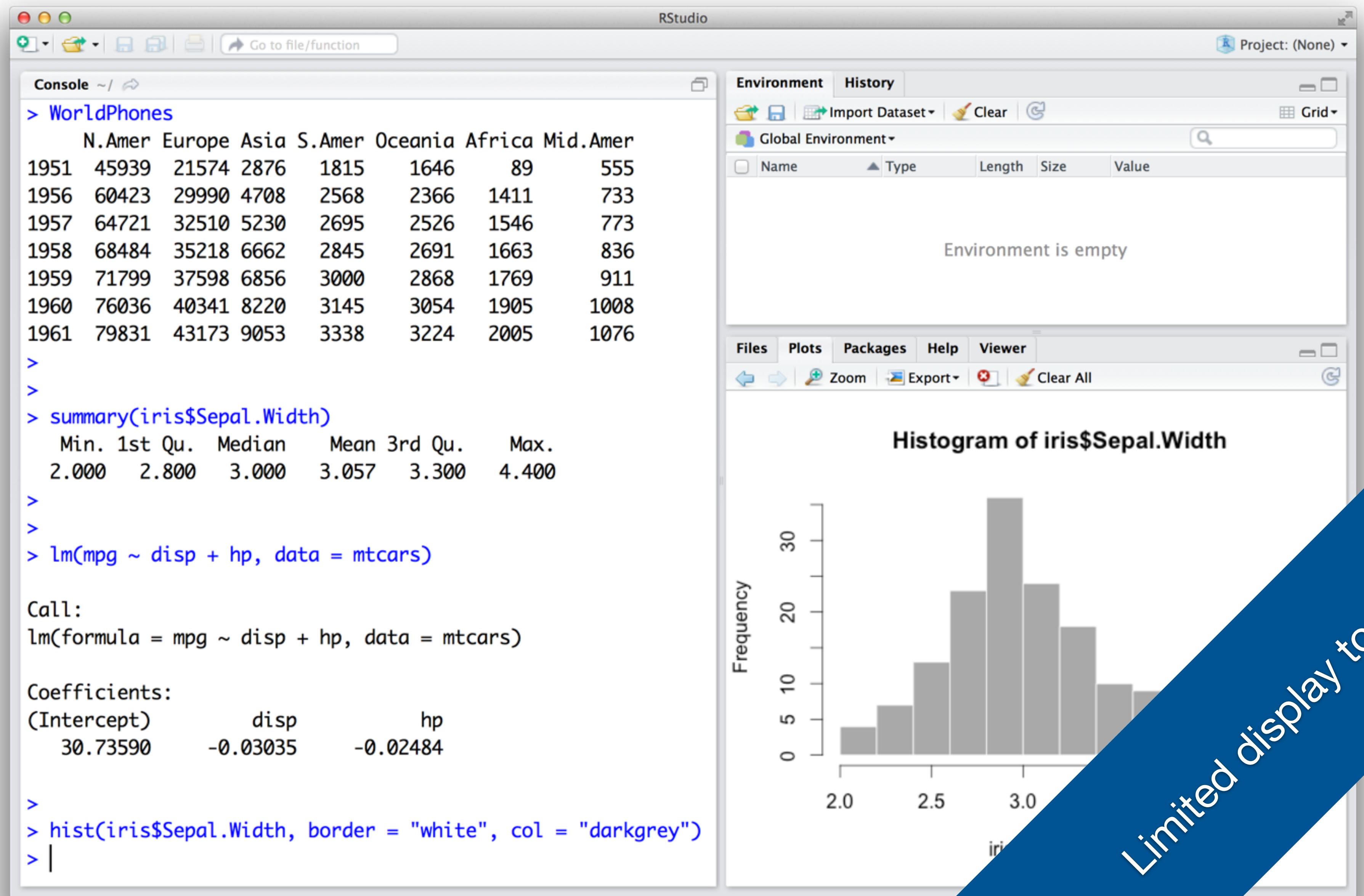


garrett@rstudio.com



@StatGarrett

**what is
Shiny?**

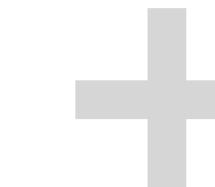


Limited display tools

Shiny



Analytic
Power



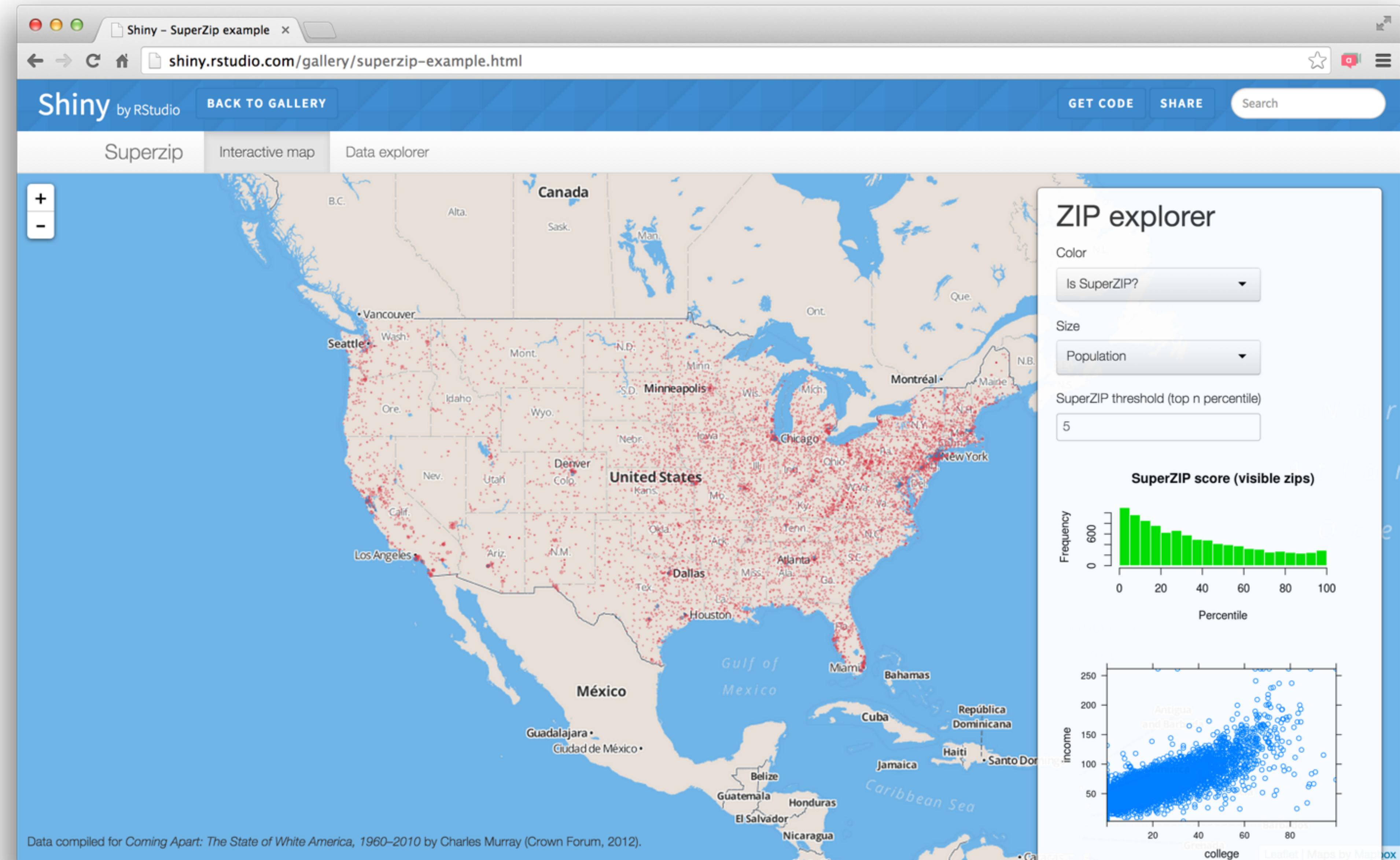
HTML



CSS



Display abilities
and interactivity



<http://shiny.rstudio.com/gallery/superzip-example.html>

Data Products

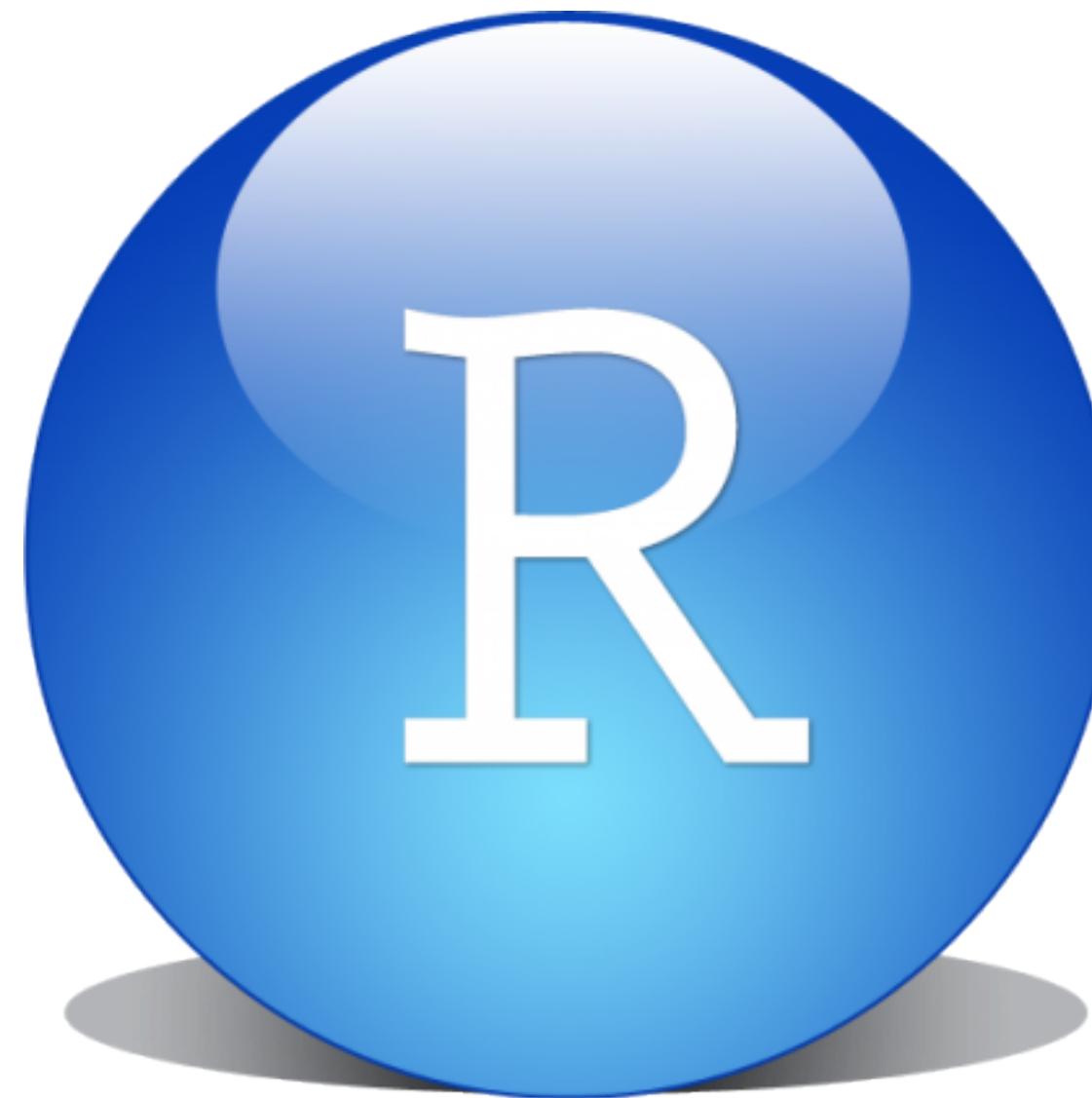
Extend ggvis for EDA

Extend R functions for non-programmers

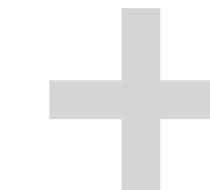
1. Reports in R
2. Interactivity
3. Layout Web Apps
4. Sharing

Quick Reports

R Markdown



Analytic
Power



Microsoft Word



Reveal.js
ioslides, Beamer



Report generation

```
--  
title: "Garrett Grolemund"  
output: html_document  
---
```

Data Scientist and Master Instructor
garrett@rstudio.com
[rstudio.com](www.rstudio.com)
Orlando, FL

```
## Expertise
```

I wrote the popular [lubridate](http://www.r-statistics.com/2012/03/do-more-with-dates-and-times-in-r-with-lubridate-1-1-0/) package and am the author of _Hands On Programming with R_ as well as _Data Science with R_, an upcoming book from O'Reilly Media. I have a Ph.D. in Statistics and specialize in the theory of data science. I enjoy:

- * __Teaching__ - I design and teach workshops for RStudio. I'm also the Editor-in-Chief of the [Shiny Development Center](http://shiny.rstudio.com), a great place to learn Shiny.

- * __Statistics__ - I have an M.A. in Statistics from Harvard University and a Ph.D. in Statistics from Rice University.

- * __Teaching statistics__ - I have travelled as far as New Zealand, the birthplace of R, to research better ways to teach statistics.

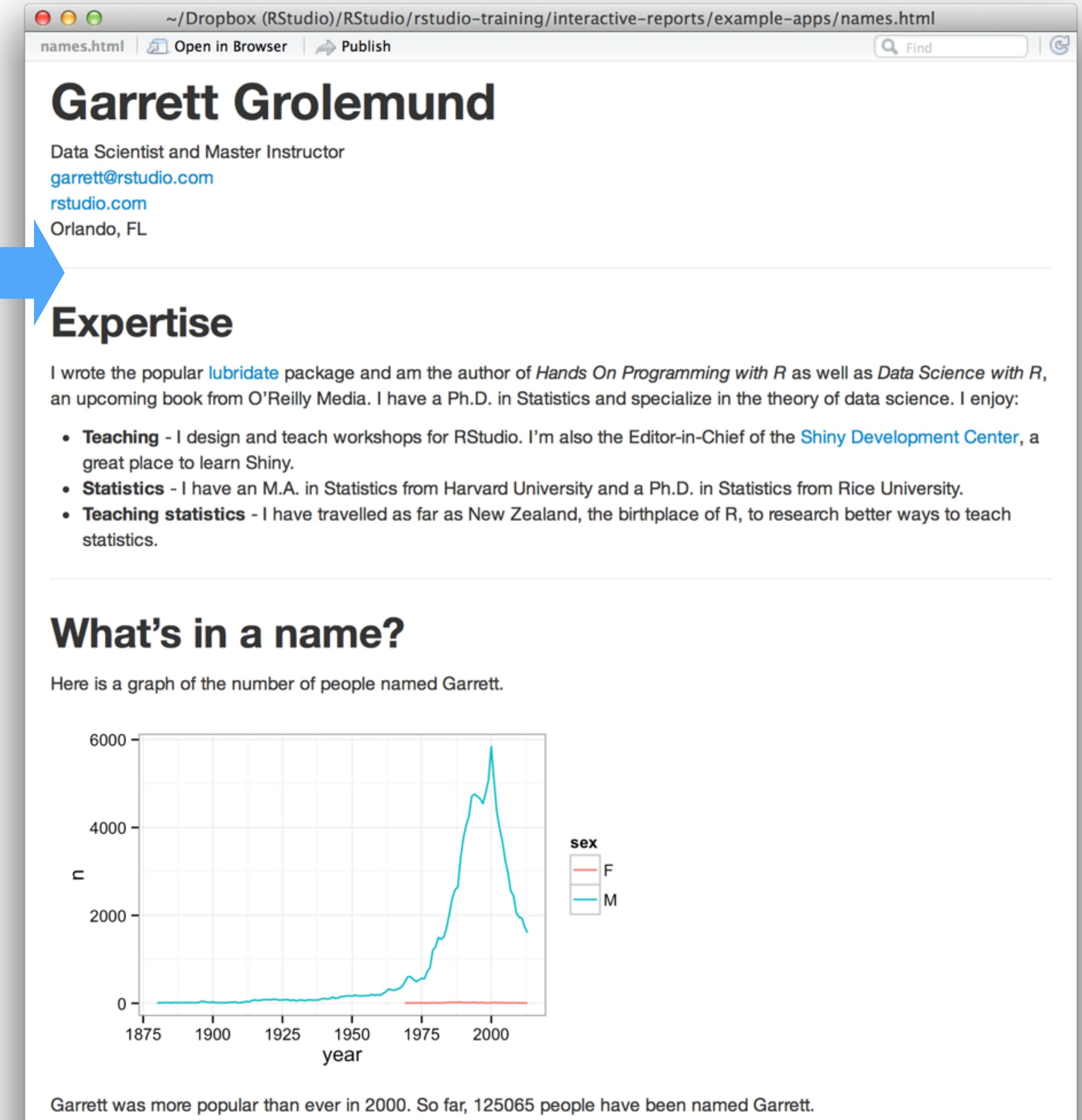
```
## What's in a name?
```

Here is a graph of the number of people named Garrett.

```
```{r echo=FALSE, fig.height=3, fig.width=5}  
library(babynames)
library(dplyr, warn.conflicts = FALSE)

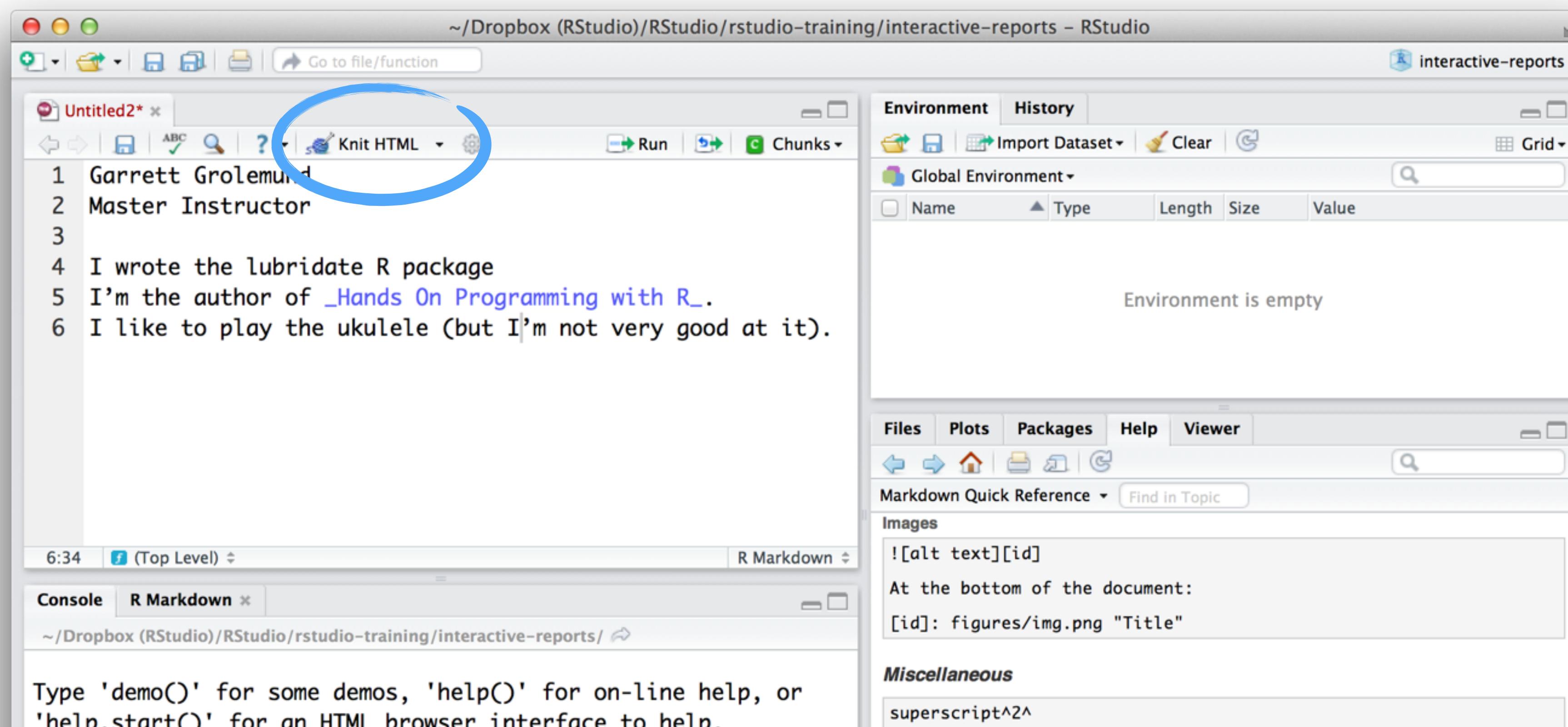
garrett <- filter(babynames, name == "Garrett", sex == "M")
plot(garrett$year, garrett$n, type = "l", col = "blue", xlab = "year", ylab = "n")
```
```

My name was more popular than ever in `r garrett\$year[which.max(garrett\$n)]`. So far, `r sum(garrett\$n)` people have been named Garrett.



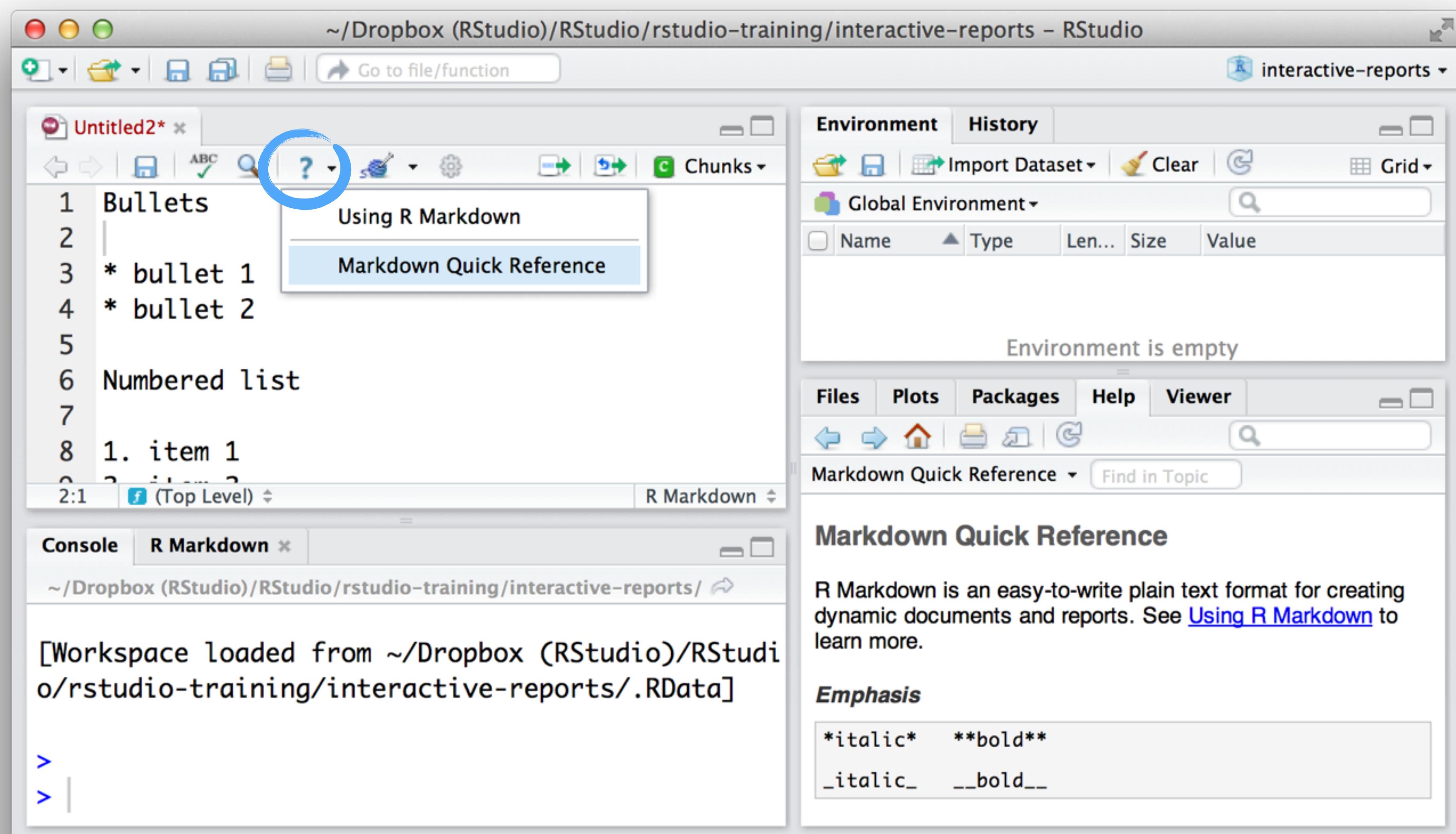
To compile markdown into HTML, click the "Knit HTML" button.

Note your file must have the extension `.md` or `.Rmd`



Markdown Quick Reference

Note your file must have the extension `.md` or `.Rmd`



Your Turn

Open names.Rmd and click "knit HTML."

With a partner:

1. identify which parts of the file are markdown and which are R code
2. inspect how the R code appears in the HTML output
3. change the .Rmd file to refer to your name.
Then click "knit HTML" again.



Insert a chunk of R code with

```
```{r}  
some code

```
```

When you compile, R markdown will run the code and include its results. R markdown will also remove the ```{r} and ```.

Insert R code *into a line of text* with

```
`r # some code`
```

When you compile, R markdown will run the code and include its results. R markdown will also remove the `r and `.

```
## What's in a name?
```

```
```{r echo=FALSE, message=FALSE}
library(babynames)
library(dplyr, warn.conflicts = FALSE)
library(ggplot2, warn.conflicts = FALSE)
```

```
garrett <- filter(babynames, name == "Garrett")
```

```

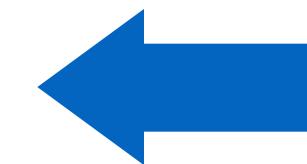
Here is a graph of the number of people named Garrett.

```
```{r echo=FALSE, fig.height=3, fig.width=5}
qplot(year, n, data = garrett, geom = "line", color = sex) + theme_bw()
```

```

Garrett was more popular than ever in `r garrett\$year[which.max(garrett\$n)]`. So far, `r sum(garrett\$n)` people have been named Garrett.

**Code chunk
embeds plot**



What's in a name?

```
```{r echo=FALSE, message=FALSE}
library(babynames)
library(dplyr, warn.conflicts = FALSE)
library(ggplot2, warn.conflicts = FALSE)

garrett <- filter(babynames, name == "Garrett")
```

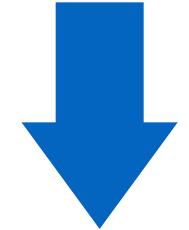
```

Here is a graph of the number of people named Garrett.

```
```{r echo=FALSE, fig.height=3, fig.width=5}
qplot(year, n, data = garrett, geom ="line", color = sex) + theme_bw()
```

```

inline code



Garrett was more popular than ever in `r garrett\$year[which.max(garrett\$n)]`. So far, `r sum(garrett\$n)` people have been named Garrett.

~/Dropbox (RStudio)/RStudio/rstudio-training/interactive-reports/example-apps/names.html
names.html | Open in Browser | Publish | Find | G

Garrett Grolemund

Data Scientist and Master Instructor
garrett@rstudio.com
rstudio.com
Orlando, FL

Expertise

I wrote the popular [lubridate](#) package and am the author of *Hands On Programming with R* as well as *Data Science with R*, an upcoming book from O'Reilly Media. I have a Ph.D. in Statistics and specialize in the theory of data science. I enjoy:

- **Teaching** - I design and teach workshops for RStudio. I'm also the Editor-in-Chief of the [Shiny Development Center](#), a great place to learn Shiny.
- **Statistics** - I have an M.A. in Statistics from Harvard University and a Ph.D. in Statistics from Rice University.
- **Teaching statistics** - I have travelled as far as New Zealand, the birthplace of R, to research better ways to teach statistics.

What's in a name?

Here is a graph of the number of people named Garrett.

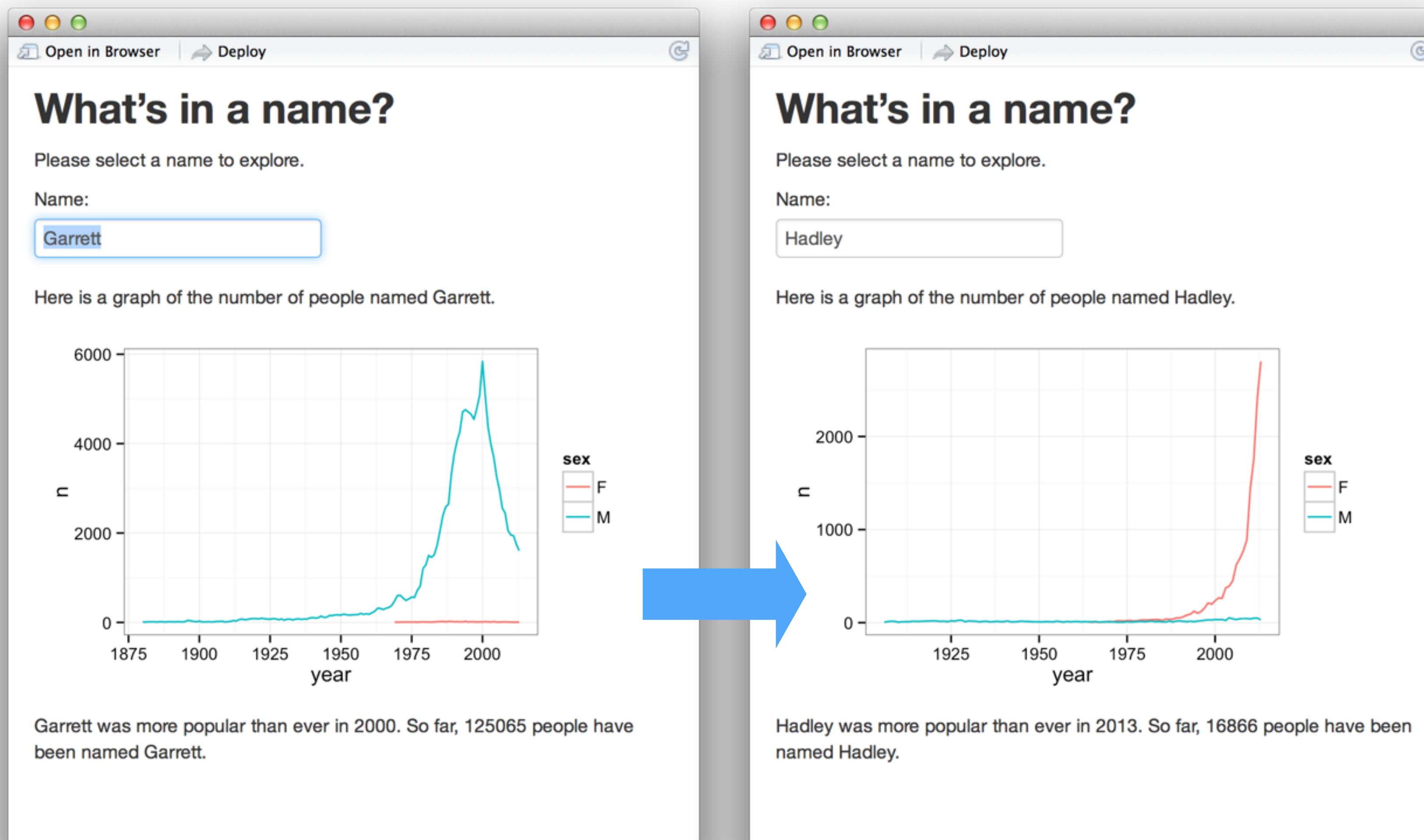
The graph displays the popularity of the name "Garrett" over time. The y-axis represents the count of people, ranging from 0 to 6,000. The x-axis represents the year, from 1875 to 2010. Two lines are shown: a red line for females (F) and a blue line for males (M). The male line shows a significant peak around 2000, reaching approximately 5,800. The female line remains near zero throughout the period.

| Year | Males (F) | Females (M) |
|------|-----------|-------------|
| 1875 | ~10 | ~10 |
| 1900 | ~10 | ~10 |
| 1925 | ~10 | ~10 |
| 1950 | ~100 | ~100 |
| 1975 | ~1,000 | ~100 |
| 1990 | ~4,500 | ~100 |
| 2000 | ~5,800 | ~100 |
| 2010 | ~2,000 | ~100 |

Garrett was more popular than ever in 2000. So far, 125065 people have been named Garrett.

Interactive reports

Goal: make graphs and text *interactive*



Open in Browser Deploy G

What's in a name?

Please select a name to explore.

Name:

Here is a graph of the number of people named Garrett.

The graph shows the number of people (n) on the y-axis (0 to 6000) versus year on the x-axis (1875 to 2013). A legend indicates 'sex' with 'F' (red line) and 'M' (blue line). The male population (M) shows a significant peak around 2000, reaching nearly 6000. The female population (F) remains very low, near zero throughout the period.

Garrett was more popular than ever in 2000. So far, 125065 people have been named Garrett.

Open in Browser Deploy G

What's in a name?

Please select a name to explore.

Name:

Here is a graph of the number of people named Hadley.

The graph shows the number of people (n) on the y-axis (0 to 2000) versus year on the x-axis (1875 to 2013). A legend indicates 'sex' with 'F' (red line) and 'M' (blue line). The female population (F) shows a sharp increase starting around 2000, peaking at approximately 2500 in 2013. The male population (M) remains very low, near zero throughout the period.

Hadley was more popular than ever in 2013. So far, 16866 people have been named Hadley.

runtime: shiny

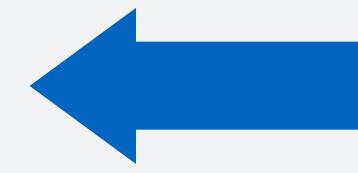
```
---
```

```
title: "Garrett Grolemund"
```

```
output: html_document
```

```
runtime: shiny
```

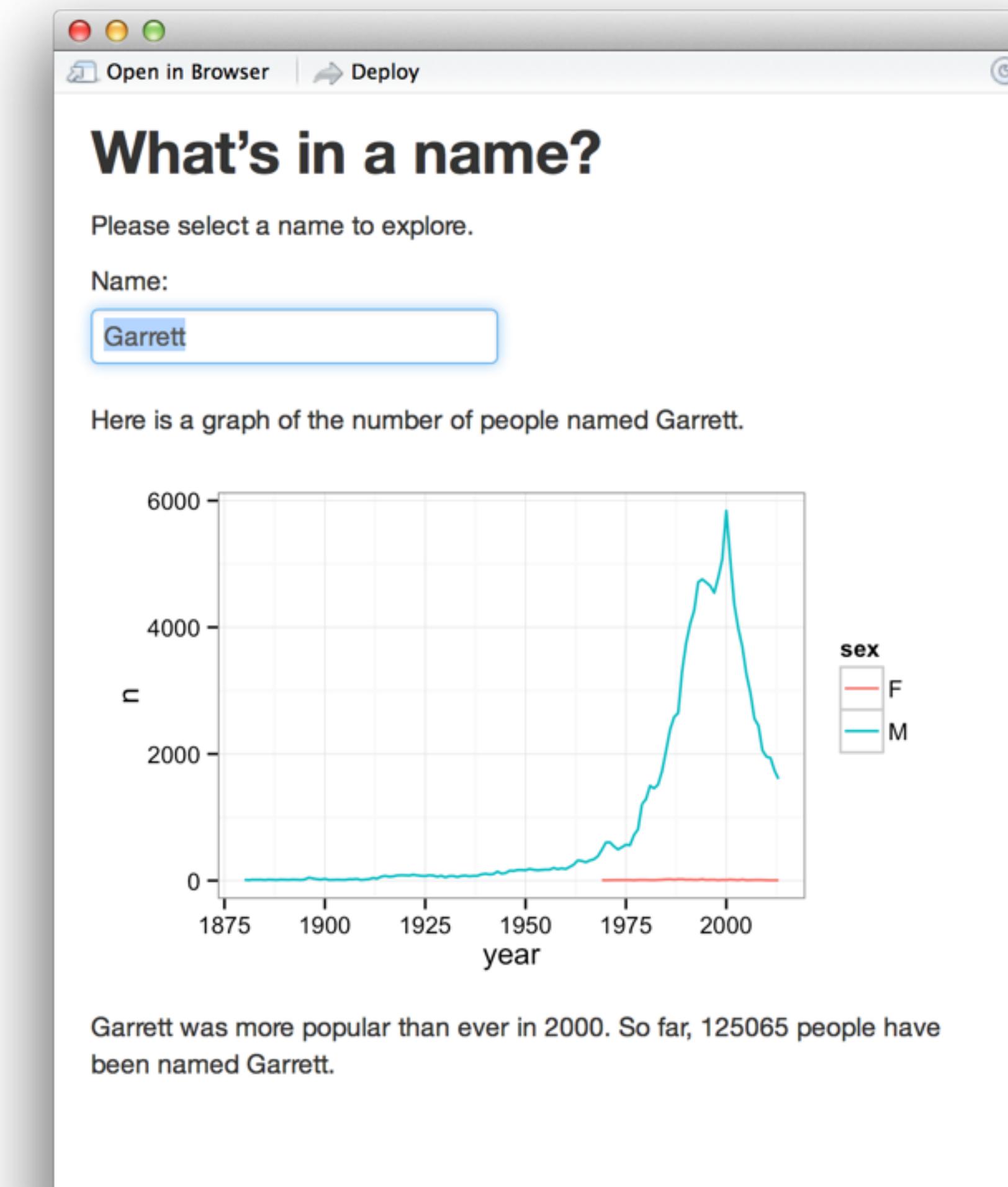
```
---
```



**Tell R Markdown to run
the doc as a shiny app**

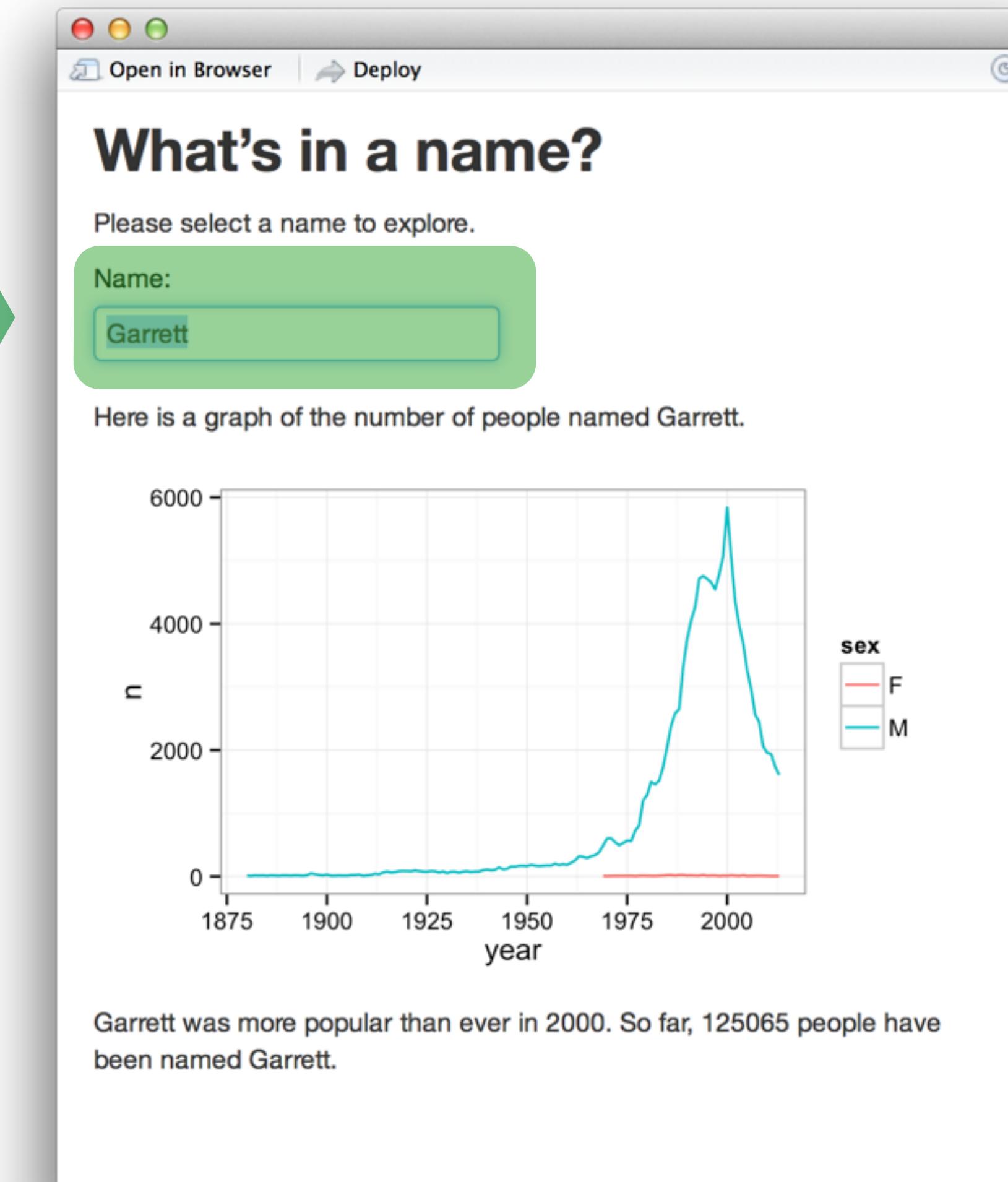
Data Scientist and Master Instructor
`garrett@rstudio.com`
`[rstudio.com](www.rstudio.com)`
Orlando, FL

Types of reactive R objects



Types of reactive R objects

Inputs - let users set a value by typing, clicking, etc.

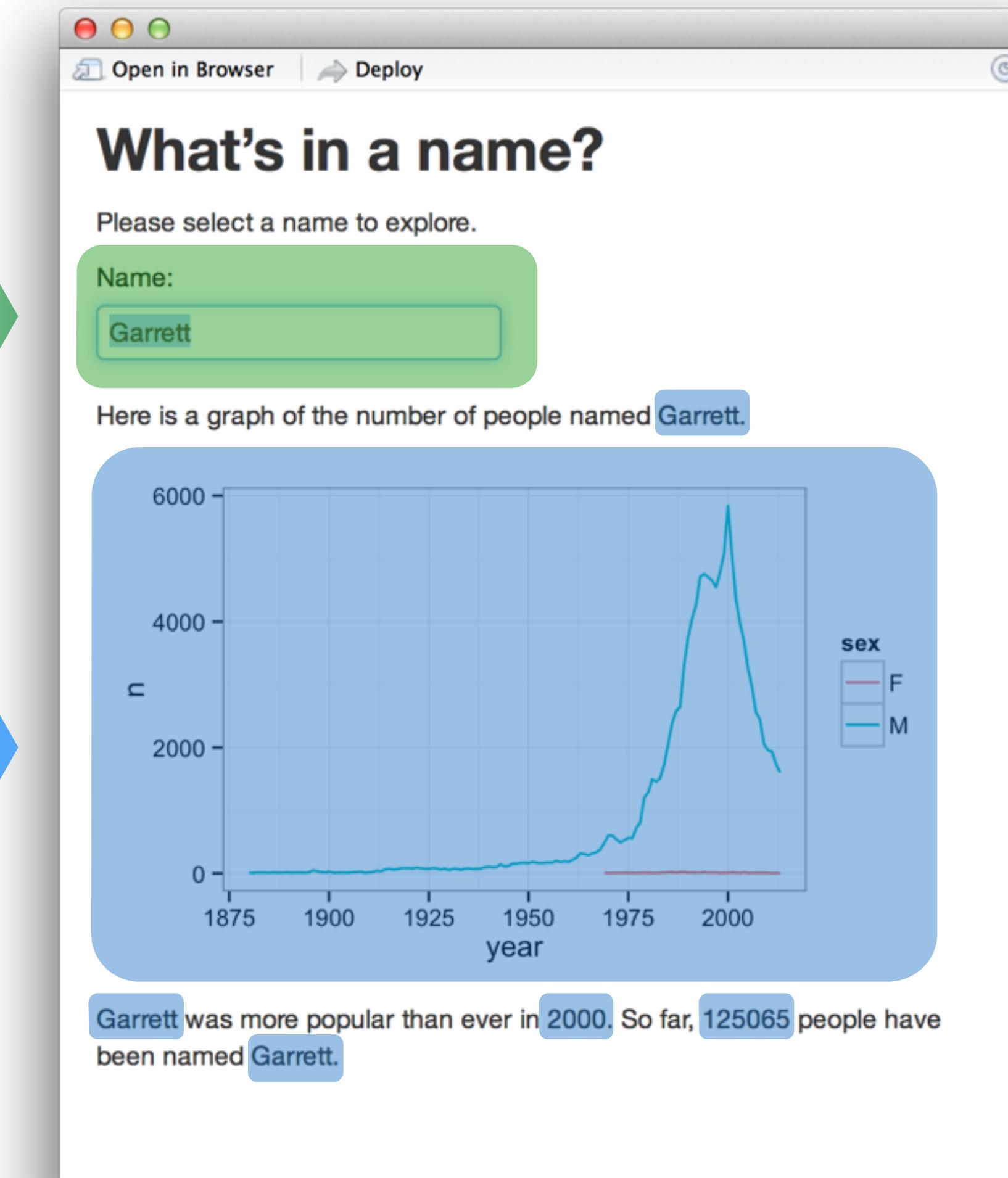
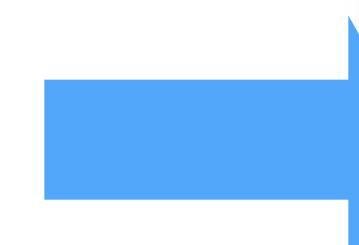


Types of reactive R objects

Inputs - let users set a value by typing, clicking, etc.



Rendered (outputs) - respond whenever a widget value changes



| function | input object |
|--------------------|--|
| actionButton | Action button |
| checkboxGroupInput | Group of checkboxes |
| checkboxInput | Single checkbox |
| dateInput | Calendar to aid date selection |
| dateRangeInput | Pair of calendars for selecting date range |
| fileInput | File upload control wizard |
| helpText | Help text to accompany other widgets |
| numericInput | Field to enter numbers |
| radioButtons | Set of radio buttons |
| selectInput | Box with choices to select from |
| sliderInput | Slider bar |
| submitButton | Submit button |
| textInput | Field to enter text |

Buttons

Action

Submit

actionButton
submitButton

Date range

2014-01-24 to 2014-01-24

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

2014-01-01

dateInput

File input

No file chosen

fileInput

Help text

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

helpText

Numeric input

1

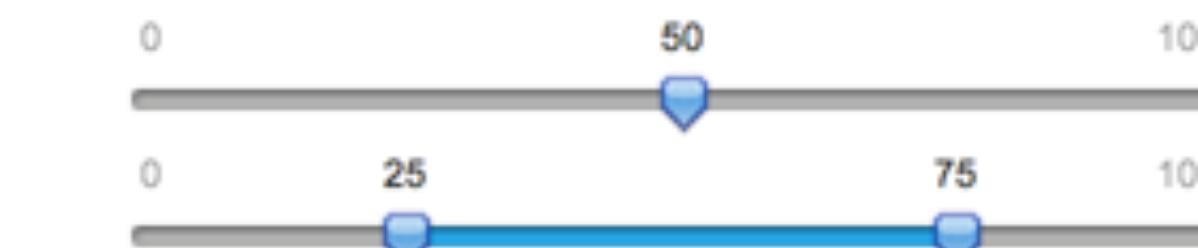
numericInput

Select box

Choice 1

selectInput

Sliders



sliderInput

Text input

Enter text...

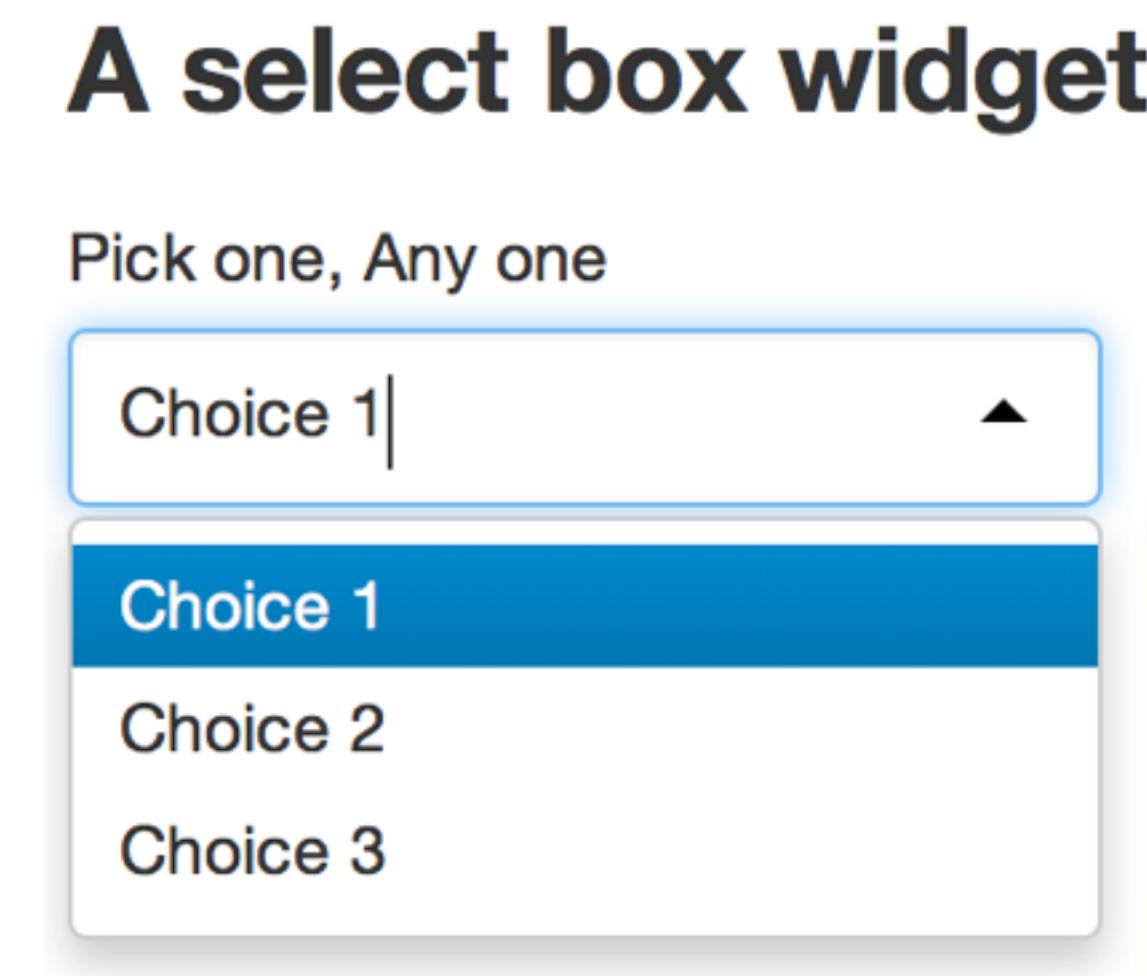
textInput

Inputs

Create a input with an input function. Since the function is R code, you'll need to put it in a code chunk.

```
### A select box widget
```{r echo=FALSE}
selectInput(name = "choice",
 label = "Pick one, Any one",
 choices = c("Choice 1",
 "Choice 2",
 "Choice 3"))
```

```



Syntax

A select box widget

Pick one, Any one

Choice 1

Choice 2

Choice 3

input
function

input name
(for internal use)

label to
display

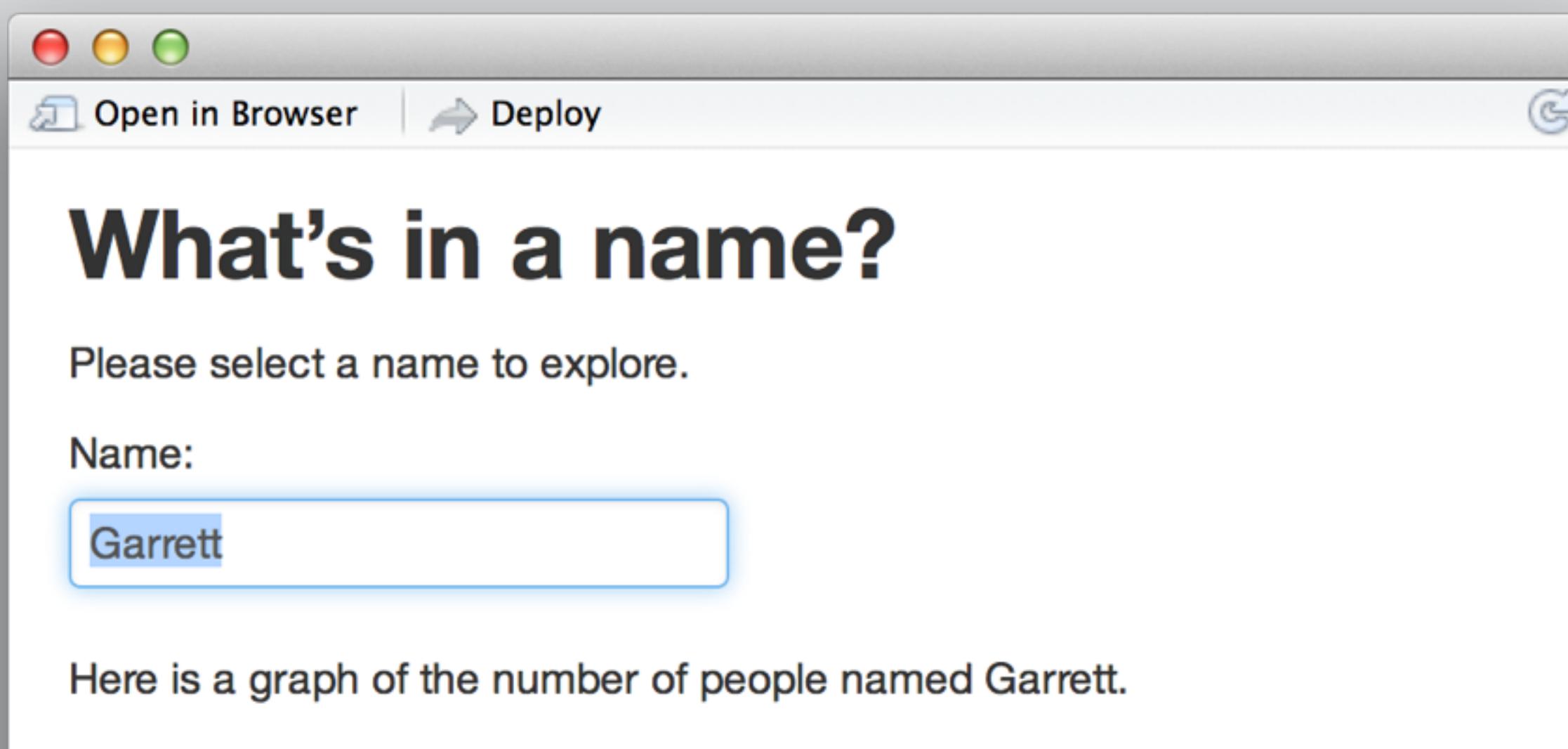
input specific
arguments

```
selectInput(name = "choice", label = "Pick one, Any one", ...)
```

Your Turn

Replace Sections 1 and 2 with a Shiny input that lets readers type in their name as **text**.

*Tips: Don't forget to add **runtime: shiny** to the top of your doc. Load the shiny package and look at the help page of **text input** function*



The screenshot shows a Shiny application window. At the top, there are three colored window control buttons (red, yellow, green) and two menu items: "Open in Browser" and "Deploy". Below the menu is a title "What's in a name?". A sub-instruction "Please select a name to explore." is displayed. A "Name:" label is followed by a text input field containing the name "Garrett". At the bottom, a message states "Here is a graph of the number of people named Garrett.".



Input values

A select box widget

Pick one, Any one

Choice 1

Choice 1
Choice 2
Choice 3

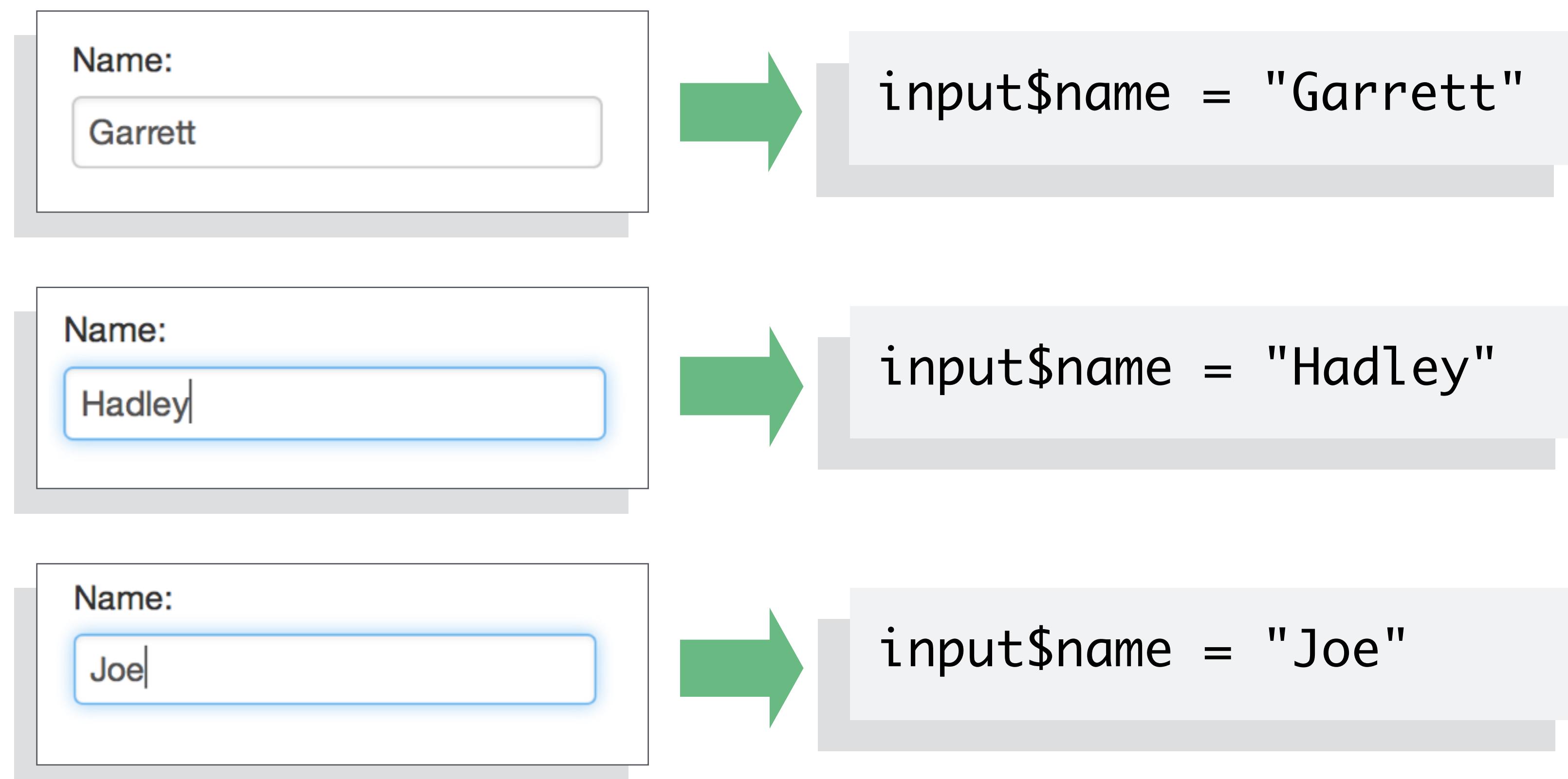
Each input returns a value that changes whenever the input changes.

```
selectInput(name = "choice", label = "Pick one, Any one", ...)
```

this input will provide a value saved as `input$choice`

Reactivity 101

The input value changes whenever a user changes the input.
*(Any Shiny output that uses the *input* value will also change).*



Reactivity 101

You **CANNOT** call an input value with a normal R function because it is *reactive*. (Shiny won't let you).

```
names <- filter(babyname, name == input$name)
```

Standard values

input\$name

ERROR!

R expression

Reactive values

Reactive expressions

To use a reactive value in an R expression, **wrap the expression in reactive**.

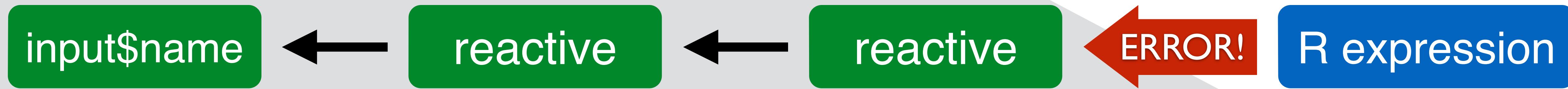
```
names <- reactive({  
  filter(babynames, name == input$name)  
})
```

Call an object created by reactive **as if it were a function**.

```
names()
```

Reactive uses reactive values (like inputs) to make new **reactive** values.

Standard values



Reactive values

Reactivity 101

You **CAN** call a reactive value when you wrap an R expression with one of *reactive*, *render**, *isolate*, or *observe*.

```
reactive({nchar(input$name)})
```



```
renderText({nchar(input$name)})
```



```
isolate({nchar(input$name)})
```



```
observe({nchar(input$name)})
```



Reactivity 101

You **CAN** call a widget value when you wrap an R expression with one of *reactive*, *render**, *isolate*, or *observe*.

```
reactive({nchar(input$name)})
```



```
renderText({nchar(input$name)})
```



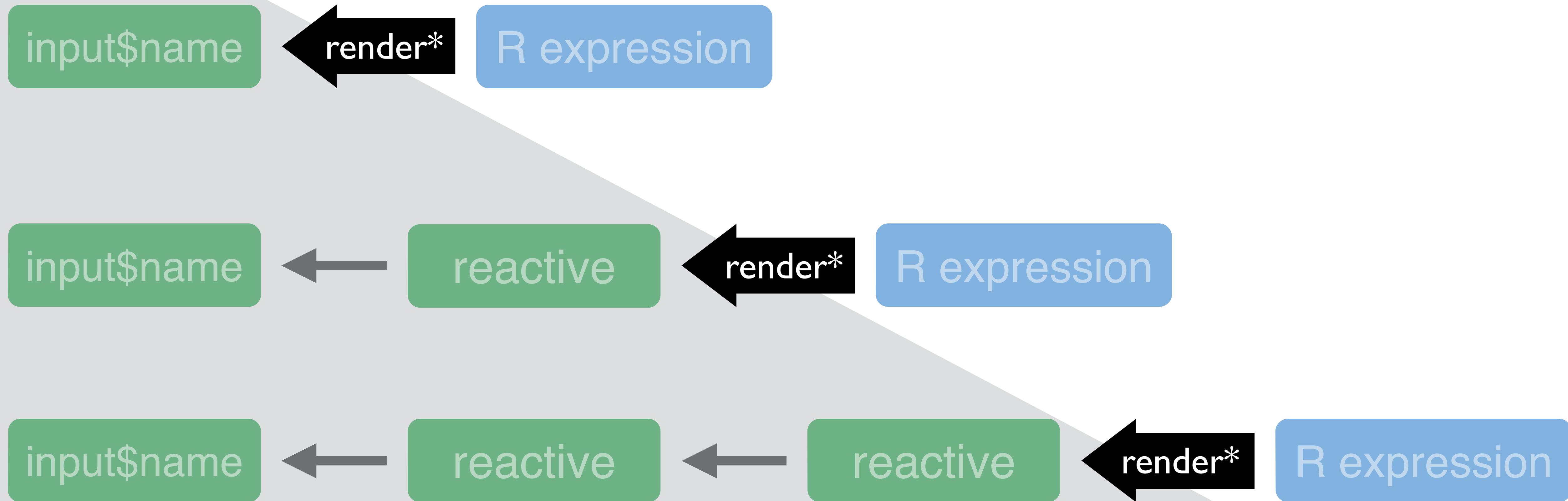
```
isolate({nchar(input$name)})
```



```
observe({nchar(input$name)})
```



Standard values



Reactive values

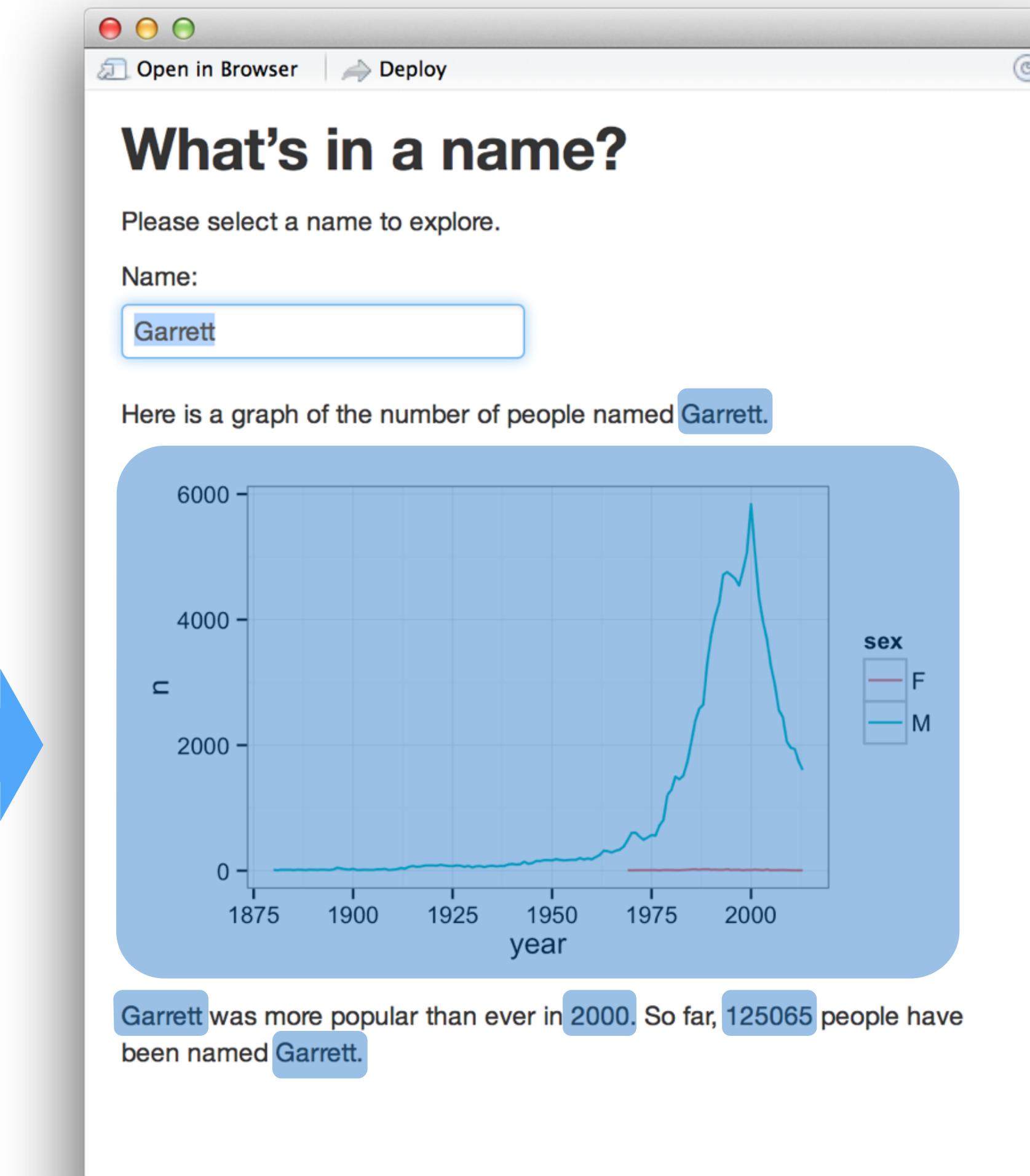
render*

A family of functions that make output that uses reactive values and **update** whenever a reactive value changes.

```
renderText({  
  names <- subset(babynames, name == input$name)  
  sum(names$n)  
})
```

render*

Rendered (outputs) -
respond whenever a
reactive value changes



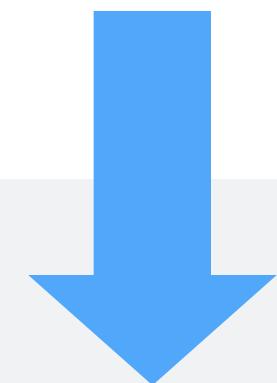
Use the function that creates the type of object you wish to make.

| 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*

Each render function takes one argument: the code needed to build the final output. (*Use as many lines as you like*)

```
renderText({  
  names <- subset(babynames, name == input$name)  
  sum(names$n)  
})
```



Putting it together

Once you make an input, you can refer to it in later code blocks.

```
```{r echo=FALSE}
textInput("name", "Name:",
 value = "Garrett")
```
There were this many people with that name:
```{r echo=FALSE}
names <- reactive({
 filter(babynames,
 name == input$name)
})
renderText({sum(names()$n)})
```
```

Name:

Garrett

There were this many people with that name:

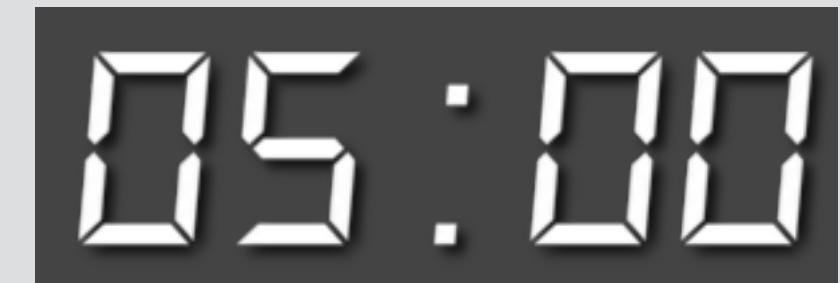
125065



Your Turn

Rewrite the R components of names.Rmd so they **re-render** whenever an input changes.

Hint: each call uses the same data set, which you can save as a reactive expression.



```
--  
title: "What's in a name?"  
runtime: shiny  
output: html_document  
--  
  
Please select a name to explore.  
```{r echo=FALSE, message=FALSE}  
library(babynames)
library(dplyr, warn.conflicts = FALSE)
library(ggplot2, warn.conflicts = FALSE)

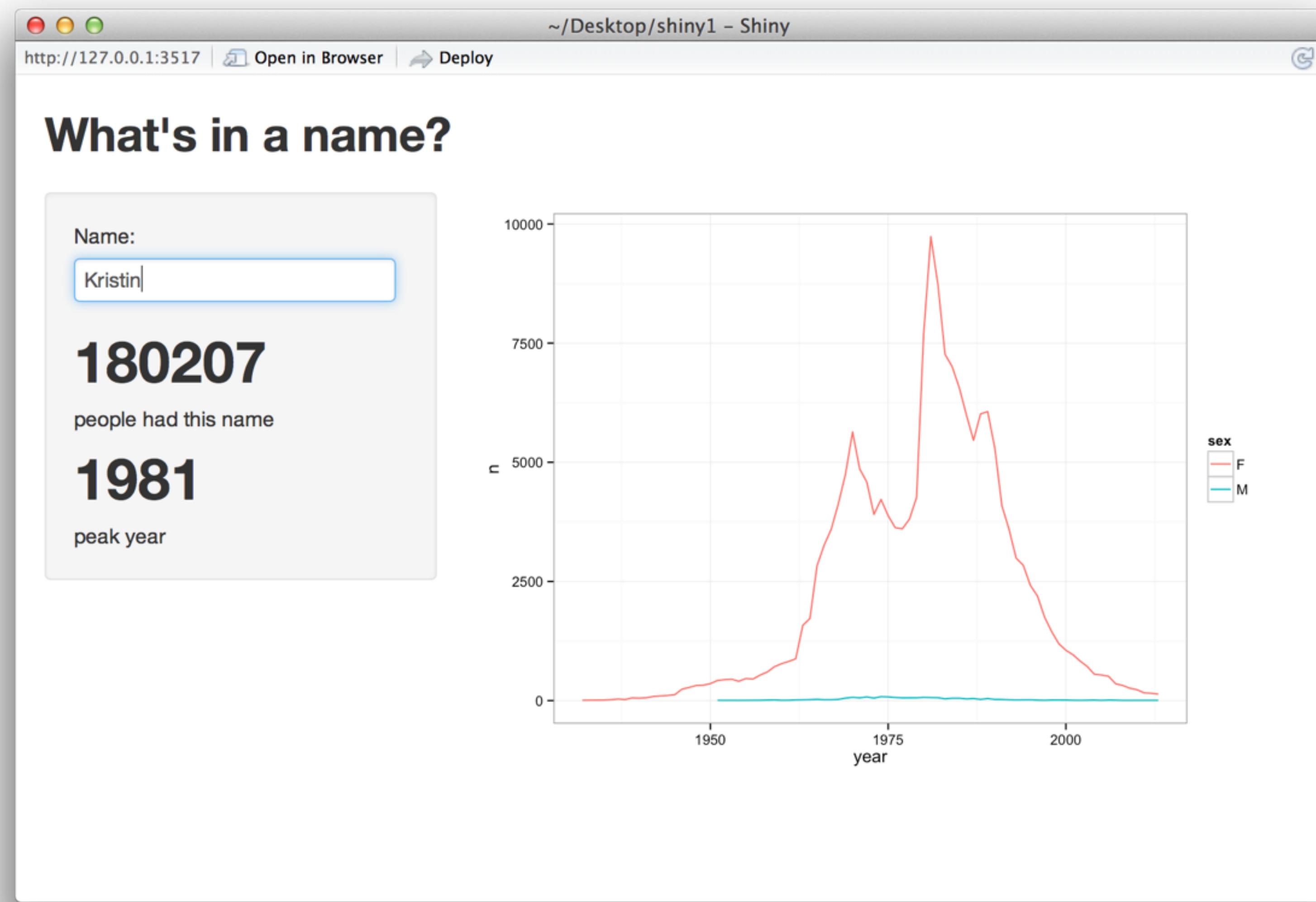
textInput("name", "Name:", value = "Garrett")

names <- reactive({
 filter(babynames, name == input$name)
})
...

Here is a graph of the number of people named `r renderText(input$name)`.
```{r echo=FALSE, fig.height=3, fig.width=5}  
renderPlot({  
  qplot(year, n, data = names(), geom ="line", color = sex) + theme_bw()  
})  
...  
`r renderText(input$name)` was more popular than ever in `r renderText(names()$year[which.max(names()$n)])`.  
So far, `r renderText(sum(names()$n))` people have been named `r renderText(input$name)`.
```

Layout Web Apps

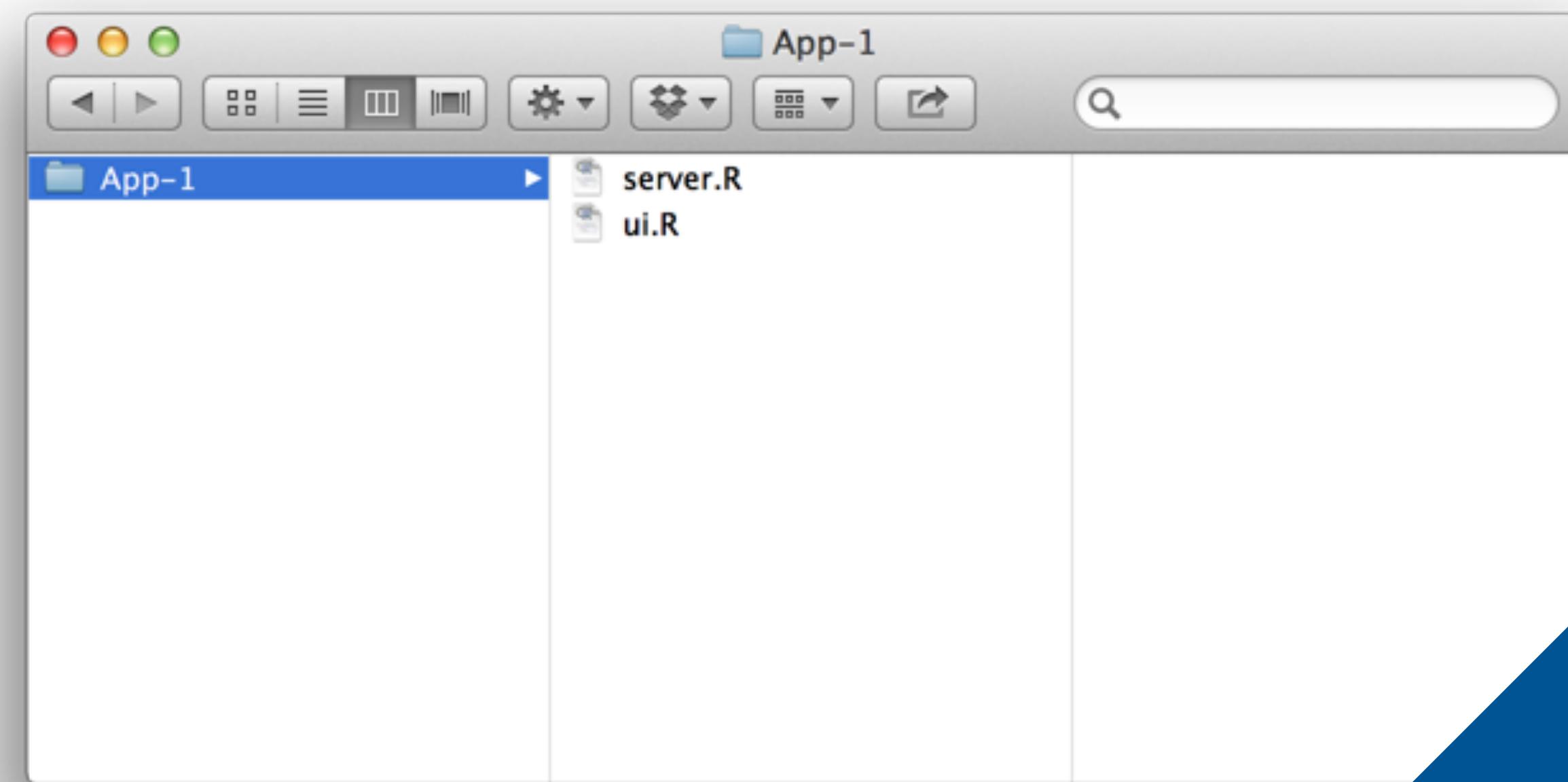
Web apps



Structure of a Shiny web app

One directory with two files:

- server.R
- ui.R



You must use these
exact names

runApp

You launch the app with `runApp` (your computer will build a local web site that hosts the app).

```
runApp("~/Documents/App-1")
```

File path to app directory.

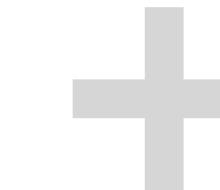
R will append the file path to the working directory, if path does not begin at the home directory

Typical Shiny App



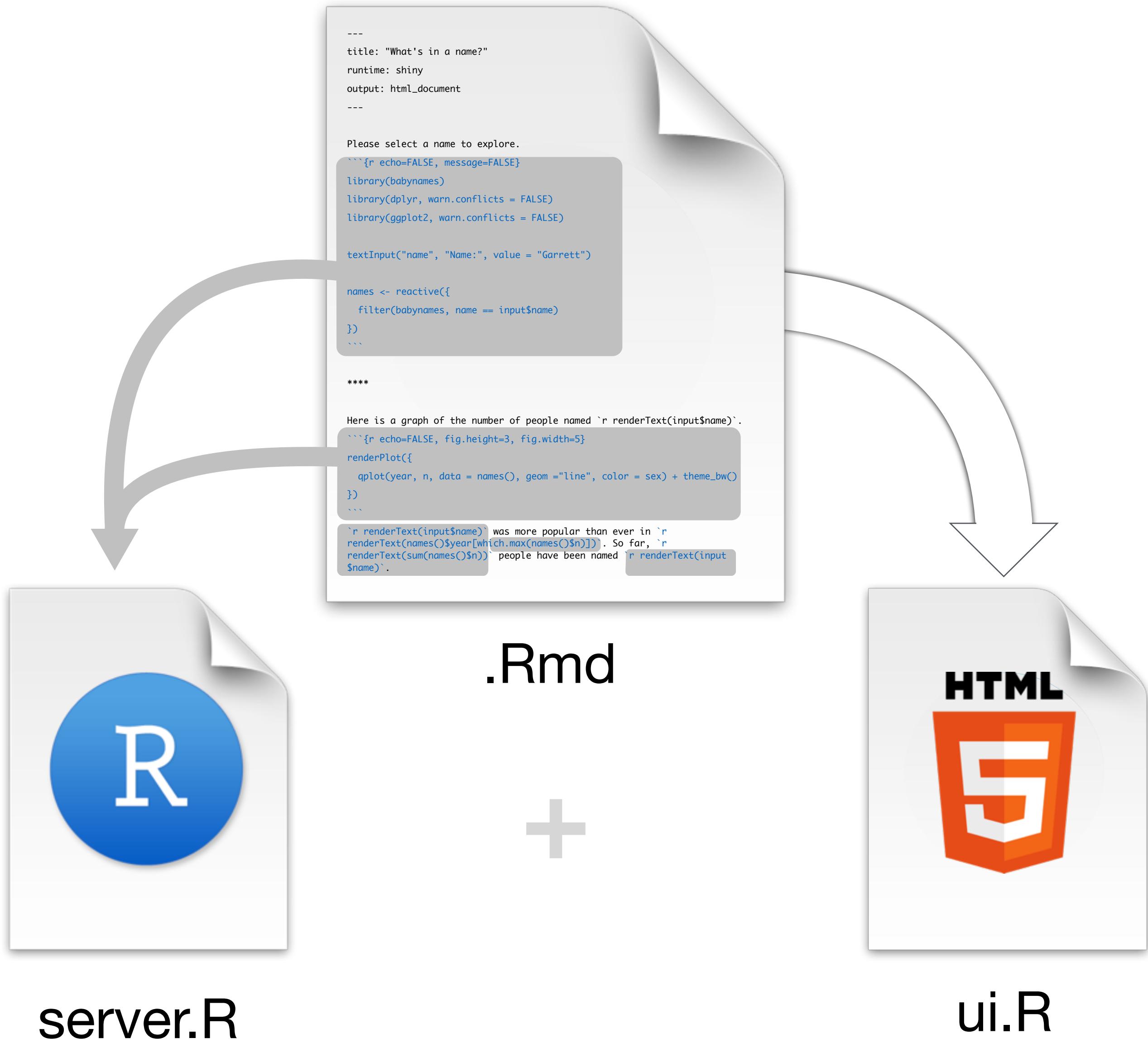
server.R

builds output
with R



ui.R

builds HTML page
to hold output



server.R

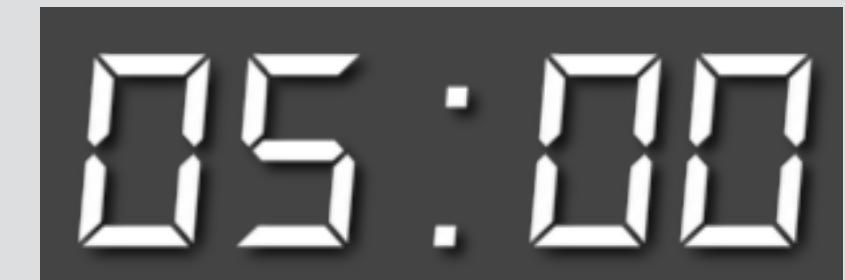
Your Turn

Open server.R in RStudio.

Find where the file builds:

1. the names reactive expression
2. the names plot
3. the total number of children with the name
4. the most popular year for the name

What else is in the file?



1

server.R contains all of the instructions (R code) your app needs to build its rendered outputs.

server.R **always** includes the following code.

```
shinyServer(function(input, output) {  
})
```

2

Server.R saves **every** rendered object to an element of **output**

```
library(babynames)
library(dplyr)
library(ggplot2)

shinyServer(function(input, output) {
  names <- reactive({filter(babynames, name == input$name)})

  output$trend <- renderPlot({
    qplot(year, n, data = names(), geom ="line", color = sex) + theme_bw()
  })

  output$total <- renderText({sum(names()$n)})

  output$peak <- renderText({names()$year[which.max(names()$n)]})
})
```

2

Server.R saves **every** rendered object to an element of output

```
library(babynames)
library(dplyr)
library(ggplot2)

shinyServer(function(input, output) {
  names <- reactive({filter(babynames, name == input$name)})

  output$trend <- renderPlot({
    qplot(year, n, data = names(), geom ="line", color = sex) + theme_bw()
  })

  output$total <- renderText({sum(names()$n)})

  output$peak <- renderText({names()$year[which.max(names()$n)]})
})
```

The name you will use to refer
to the element in the ui.R file



output\$trend

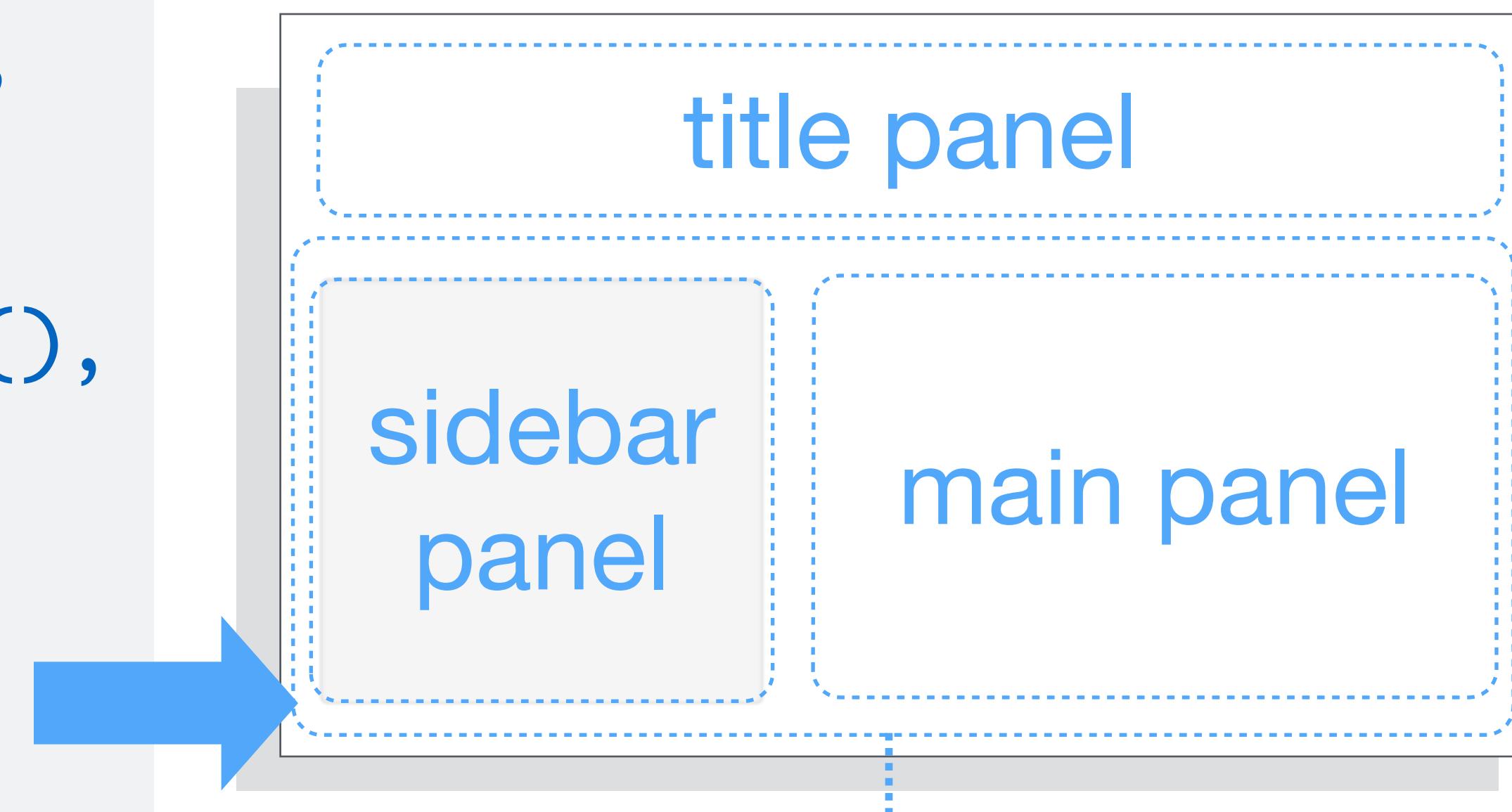
ui.R
(layouts)

The shortest viable ui.R file:

```
shinyUI(fluidPage())
```

Place layout functions inside `fluidPage` to layout the app

```
shinyUI(fluidPage(  
  titlePanel(""),  
  sidebarLayout(  
    sidebarPanel(),  
    mainPanel()  
)  
)
```



sidebar layout

text

Place raw text into any panel you like.

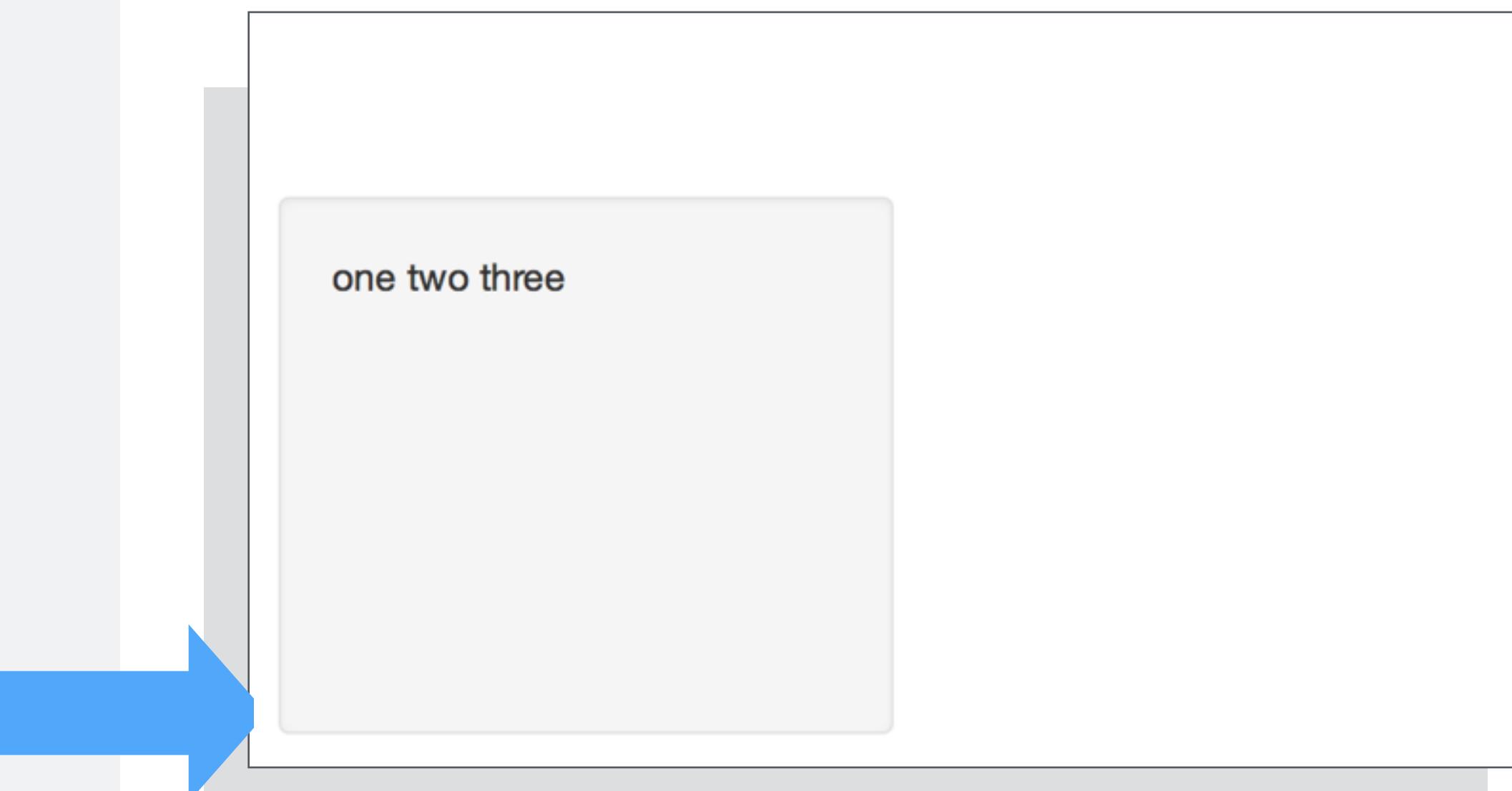
```
shinyUI(fluidPage(  
  titlePanel("one"),  
  sidebarLayout(  
    sidebarPanel("two"),  
    mainPanel("three"))  
)  
)
```



text

Place raw text into any panel you like.

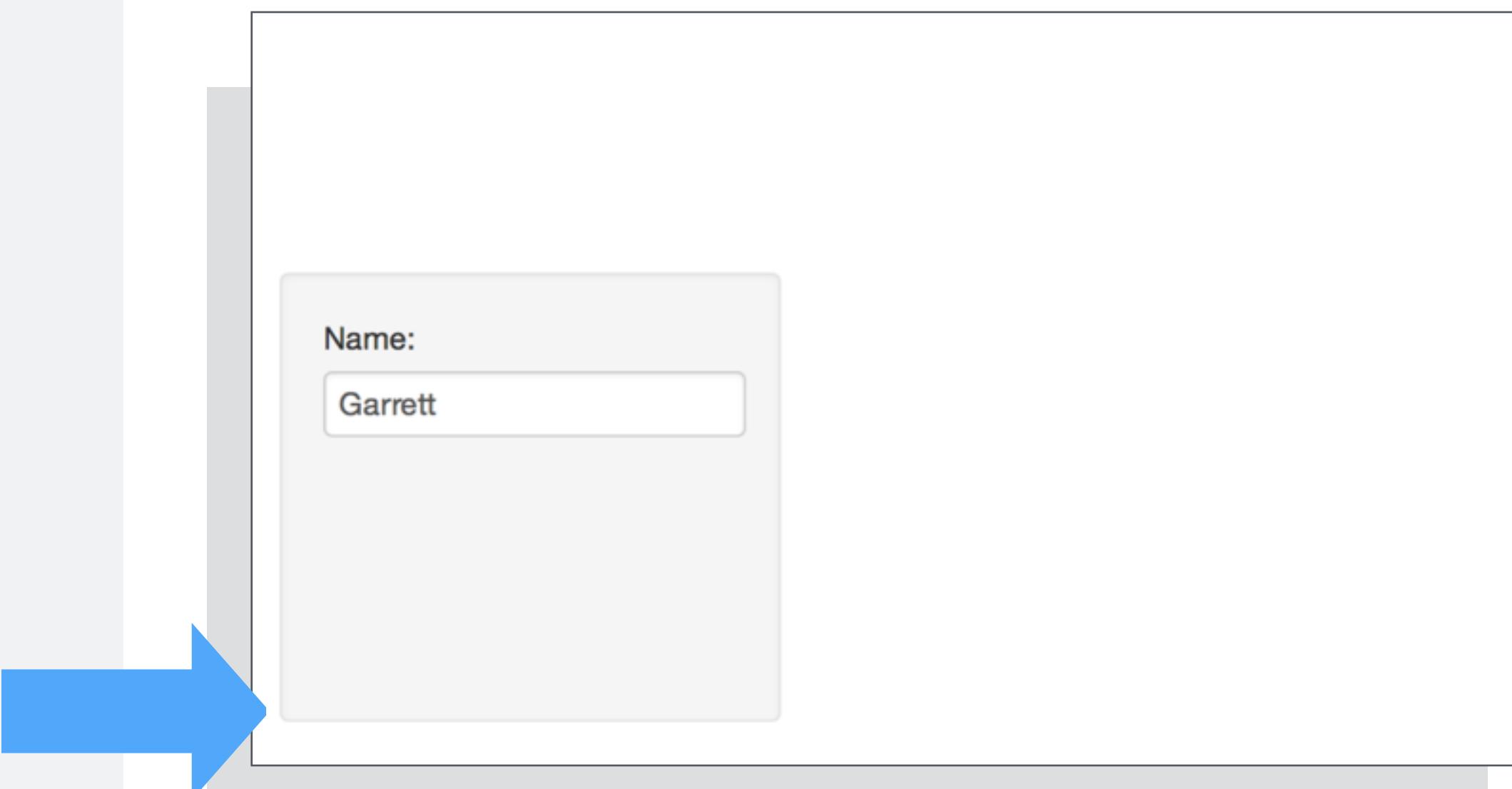
```
shinyUI(fluidPage(  
  titlePanel(""),  
  sidebarLayout(  
    sidebarPanel(  
      "one",  
      "two",  
      "three"),  
    mainPanel()  
  )  
)
```



inputs

Place your input functions where you want the inputs to appear. (*titlePanel* is not an option)

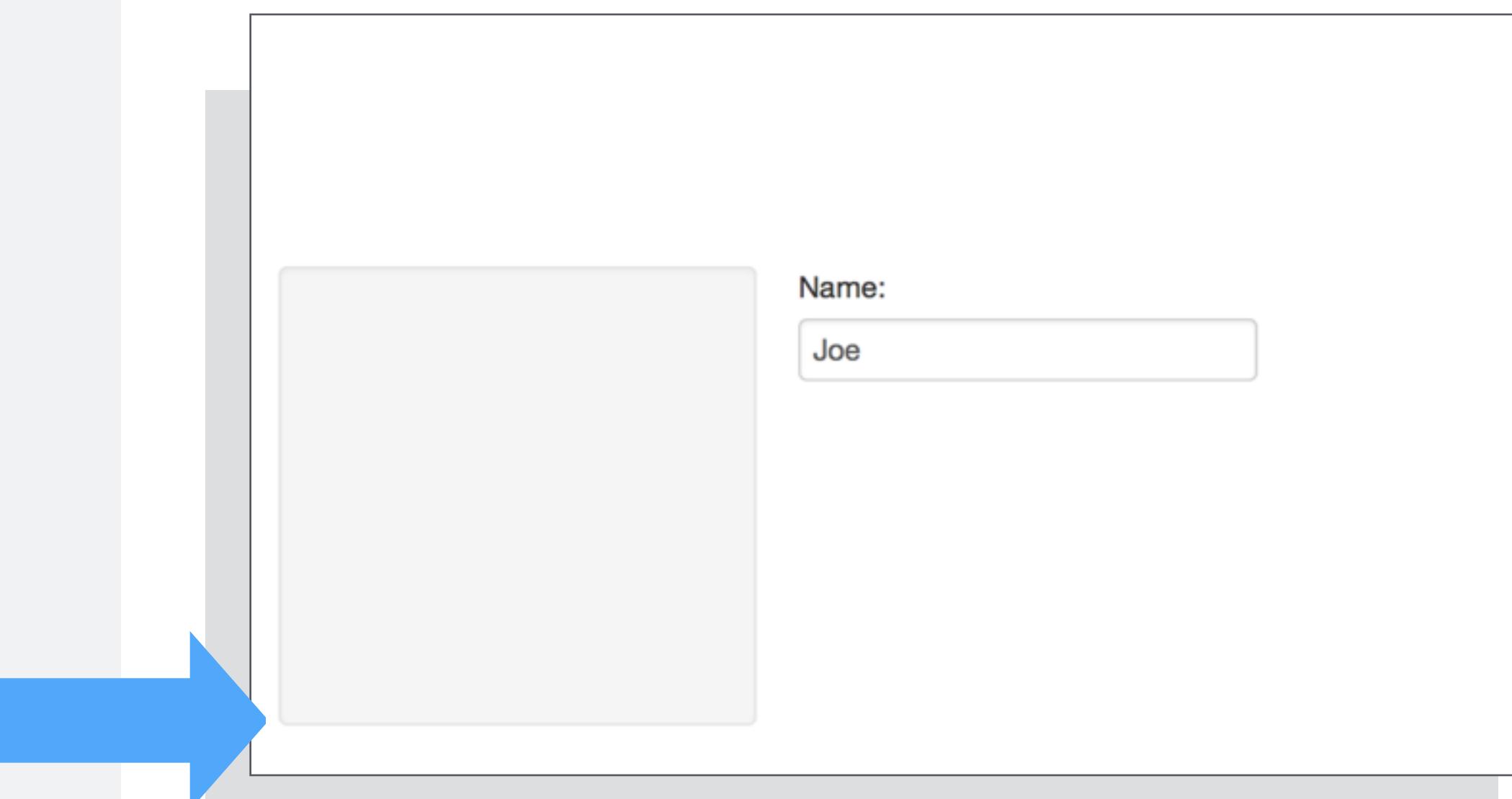
```
shinyUI(fluidPage(  
  titlePanel(""),  
  sidebarLayout(  
    sidebarPanel(  
  
     textInput("name", "Name:",  
        value = "Joe")  
    ),  
    mainPanel()  
  )  
)
```



inputs

Place your input functions where you want the inputs to appear. (*titlePanel* is not an option)

```
shinyUI(fluidPage(  
  titlePanel(""),  
  sidebarLayout(  
    sidebarPanel(),  
    mainPanel(  
  
     textInput("name", "Name:",  
               value = "Joe")  
    )  
  )  
)
```



ui.R

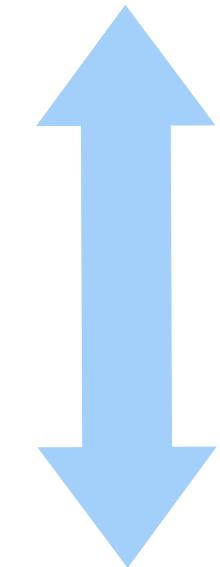
Use an *Output function to place each rendered element in a panel.

`plotOutput("trend")`

a *Output function that
matches the type of
object

name of rendered
element (set in server.R)

output\$trend

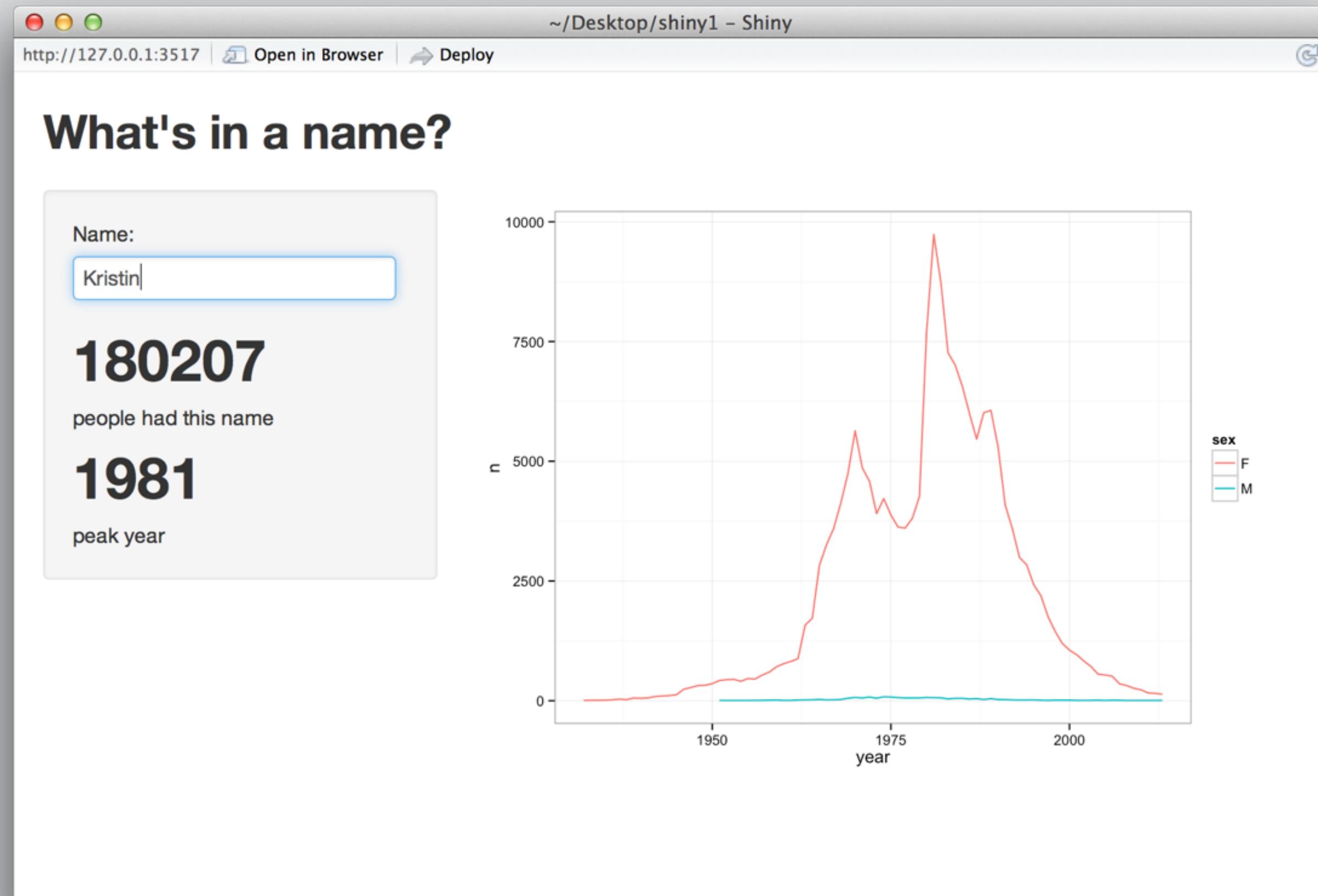


plotOutput("trend")

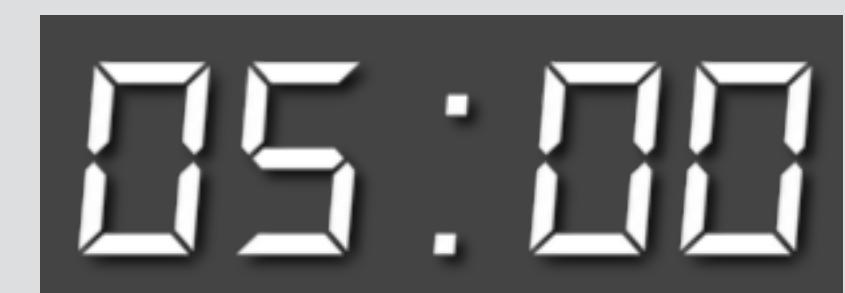
Function	Inserts
dataTableOutput	an interactive table
htmlOutput	rawHTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	a Shiny UI element
verbatimText	text

Your turn

Modify your ui.R file to recreate the web app.
Run the app.



Tip: You can increase the size of text (or `textOutput(...)`) by surrounding it with `h1()`, e.g. `h1("Header One size")`



```
shinyUI(fluidPage(  
  
  titlePanel("What's in a name?"),  
  
  sidebarLayout(  
    sidebarPanel(  
     textInput("name", "Name:", value = "Garrett"),  
      h1(textOutput("total")),  
      "people had this name",  
      h1(textOutput("peak")),  
      "peak year"  
    ),  
    mainPanel(  
      plotOutput("trend")  
    )  
  )  
)
```

```
shinyUI(fluidPage(  
  
  titlePanel("What's in a name?"),  
  
  sidebarLayout(  
    sidebarPanel(  
     textInput("name", "Name:", value = "Garrett"),  
      h1(textOutput("total")),  
      "people had this name",  
      h1(textOutput("peak")),  
      "peak year"  
    ),  
    mainPanel(  
      plotOutput("trend")  
    )  
  )  
)
```

Customize your UI with HTML

<http://shiny.rstudio.com/articles/html-tags.html>

You can add any HTML element to your app with one of Shiny's `tags` function.

```
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"                "eventsouce"            "fieldset"
```

Run App

A screenshot of the RStudio interface demonstrating how to run a Shiny application. A large blue arrow points from the title 'Run App' down to the 'Run App' button in the toolbar.

The left pane shows the `ui.R` script:

```
# ui.R
shinyUI(fluidPage(
  titlePanel("What's in a name?"),
  sidebarLayout(
    sidebarPanel(
      textInput("name", "Name:", value = "Garrett")
```

The right pane shows the Environment pane with the following data objects:

Object	Description
data	134 obs. of 5 variables
df	2747 obs. of 13 variables
df1	33 obs. of 12 variables
garrett	171 obs. of 5 variables

The bottom pane shows the Console output:

```
> shiny::runApp('example-apps/finished-apps/2-names-app')
Listening on http://127.0.0.1:7039
> shiny::runApp('example-apps/finished-apps/2-names-app')
```

Display options

A screenshot of the RStudio interface demonstrating how to run a Shiny application. The left pane shows the code for a Shiny UI script named ui.R:

```
# ui.R
shinyUI(fluidPage(
  titlePanel("What's in a name?"),
  sidebarLayout(
    sidebarPanel(
      textInput("name", "Name:", value = "Garrett")
    ),
    mainPanel(
      textOutput("nameOutput")
    )
  )
))
```

The right pane shows the Environment and Data panes. A blue arrow points from the 'Run App' button in the toolbar to a context menu that is open over the Run App button. The context menu has three options: 'Run in Window', 'Run in Viewer Pane' (which is selected), and 'Run External'. The Data pane lists several objects:

Object	Description
data	134 obs. of 5 variables
df	2747 obs. of 13 variables
df1	33 obs. of 12 variables
garrett	171 obs. of 5 variables

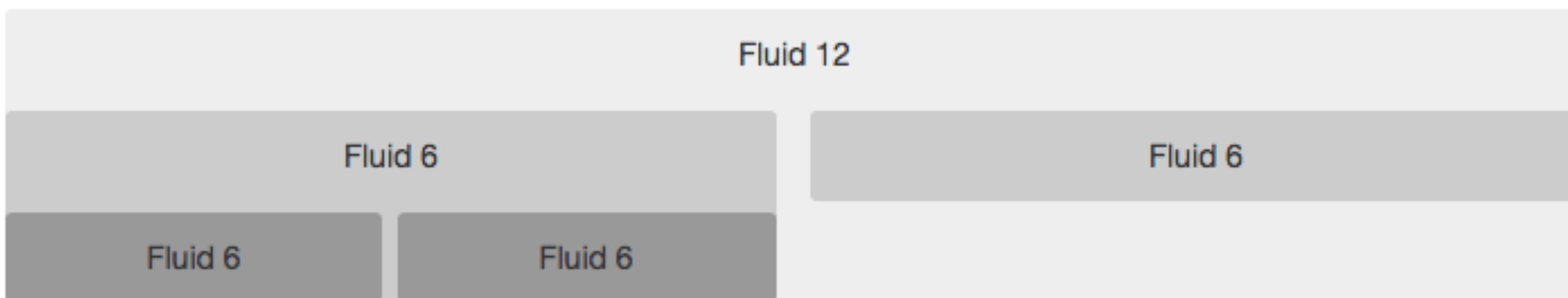
The bottom pane shows the Console output:

```
> shiny::runApp('example-apps/finished-apps/2-names-app')
Listening on http://127.0.0.1:7039
> shiny::runApp('example-apps/finished-apps/2-names-app')
```

© 2014 RStudio, Inc. All rights reserved.

Grid layouts

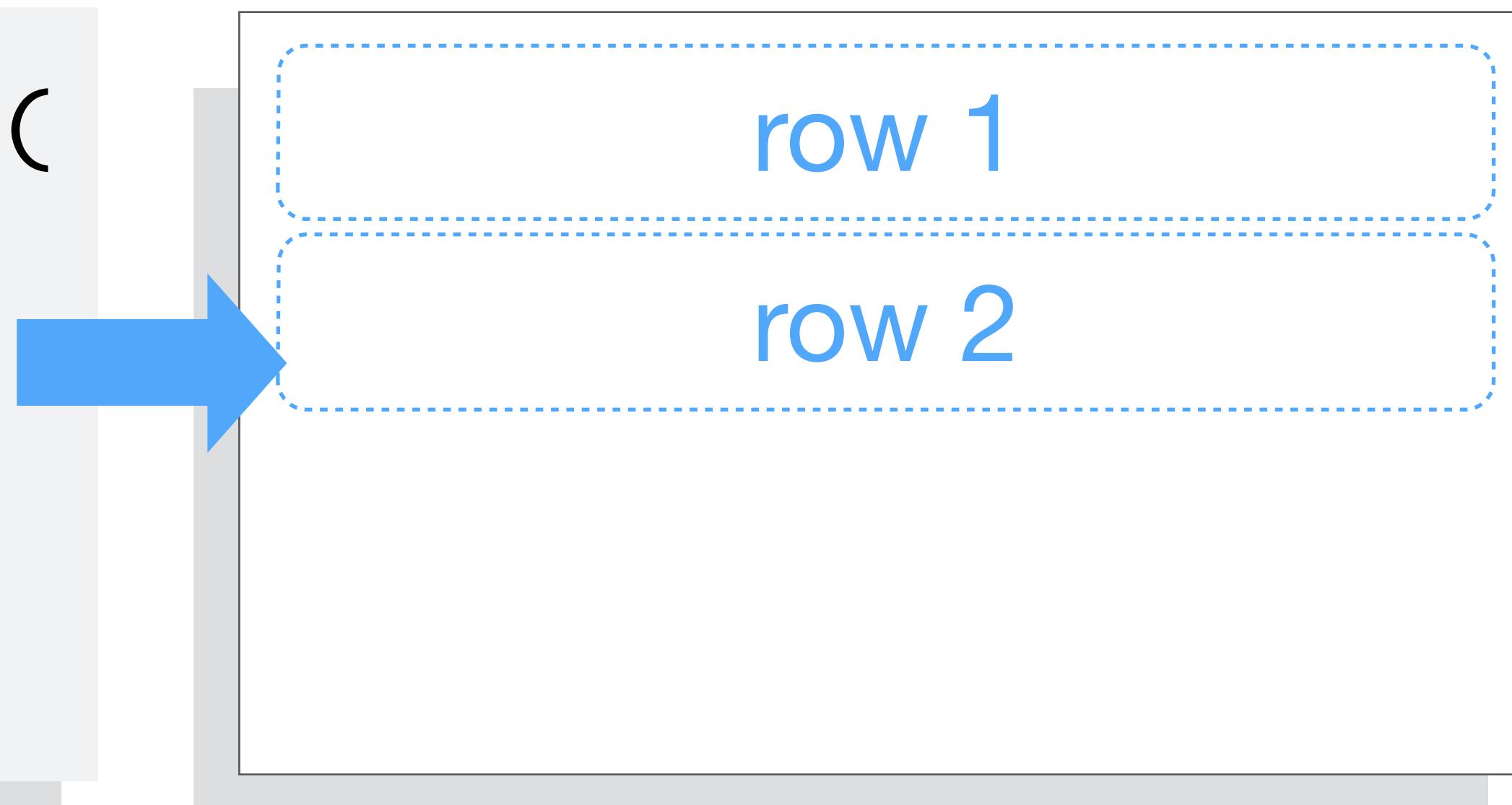
Alternatively, you can use `fluidRow` and `column` to layout your app with a grid system.



fluidRows

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

```
shinyUI(fluidPage(  
  fluidRow(),  
  fluidRow()  
)
```



column

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

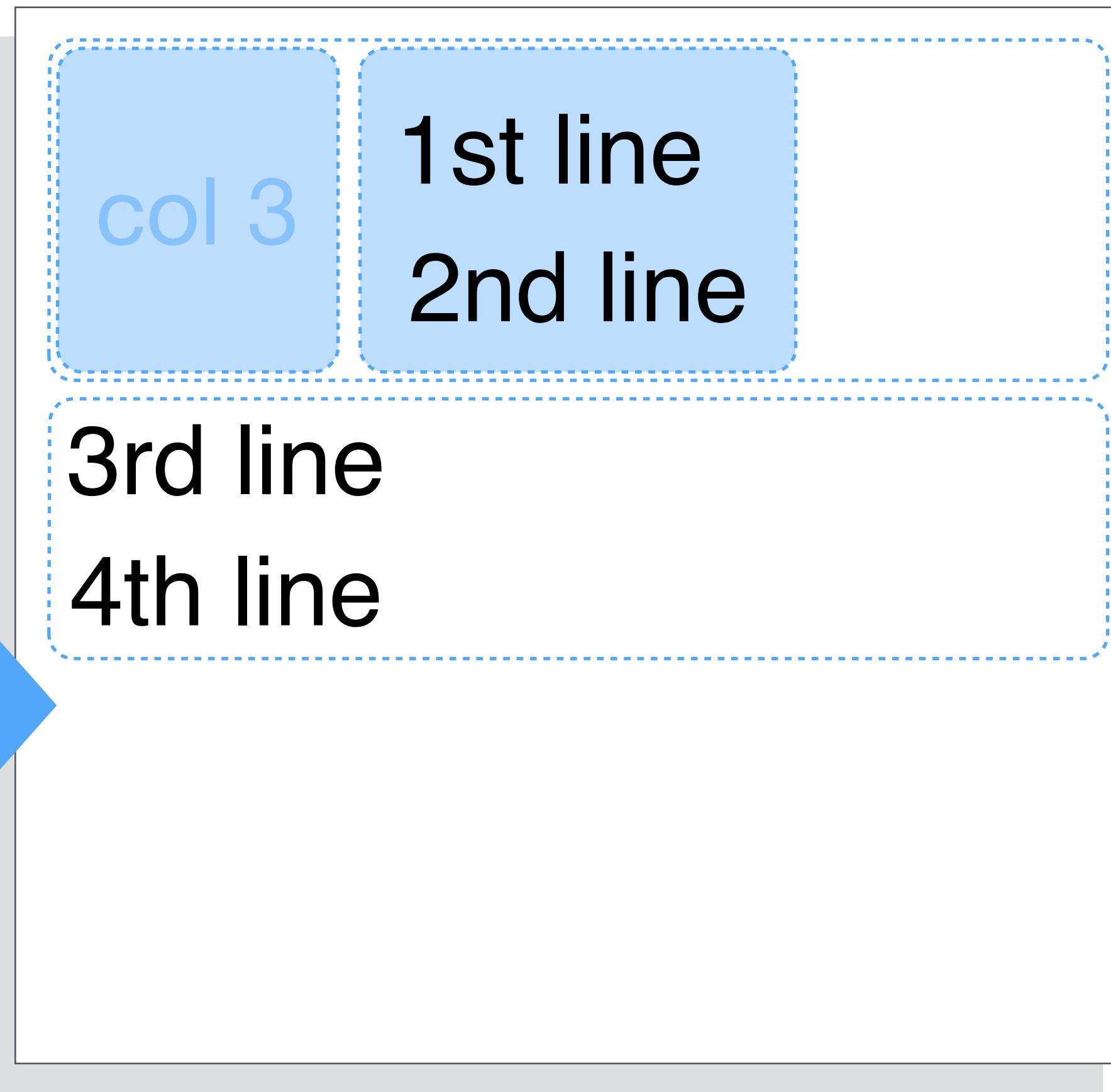
Specify the width and offset of each column out of 12

```
shinyUI(fluidPage(  
  fluidRow(  
    column(3),  
    column(5)),  
  fluidRow(  
    column(4, offset = 8)  
)
```



ggvis will fill columns and rows from top to bottom

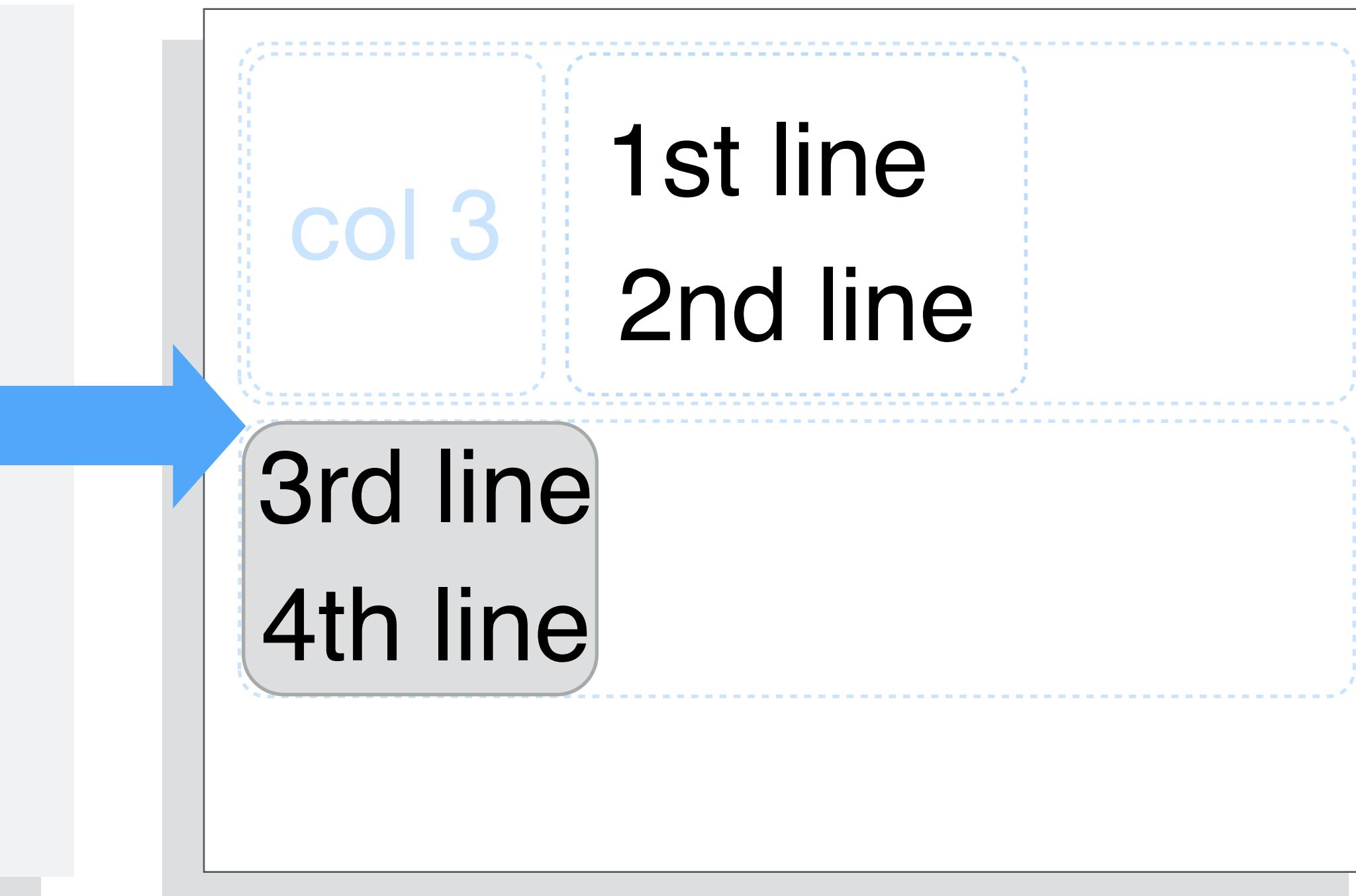
```
shinyUI(fluidPage(  
  fluidRow(  
    column(3),  
    column(5,  
      "1st line",  
      "2nd line")),  
  fluidRow(  
    "3rd line",  
    "4th line"))  
)
```



wellPanel

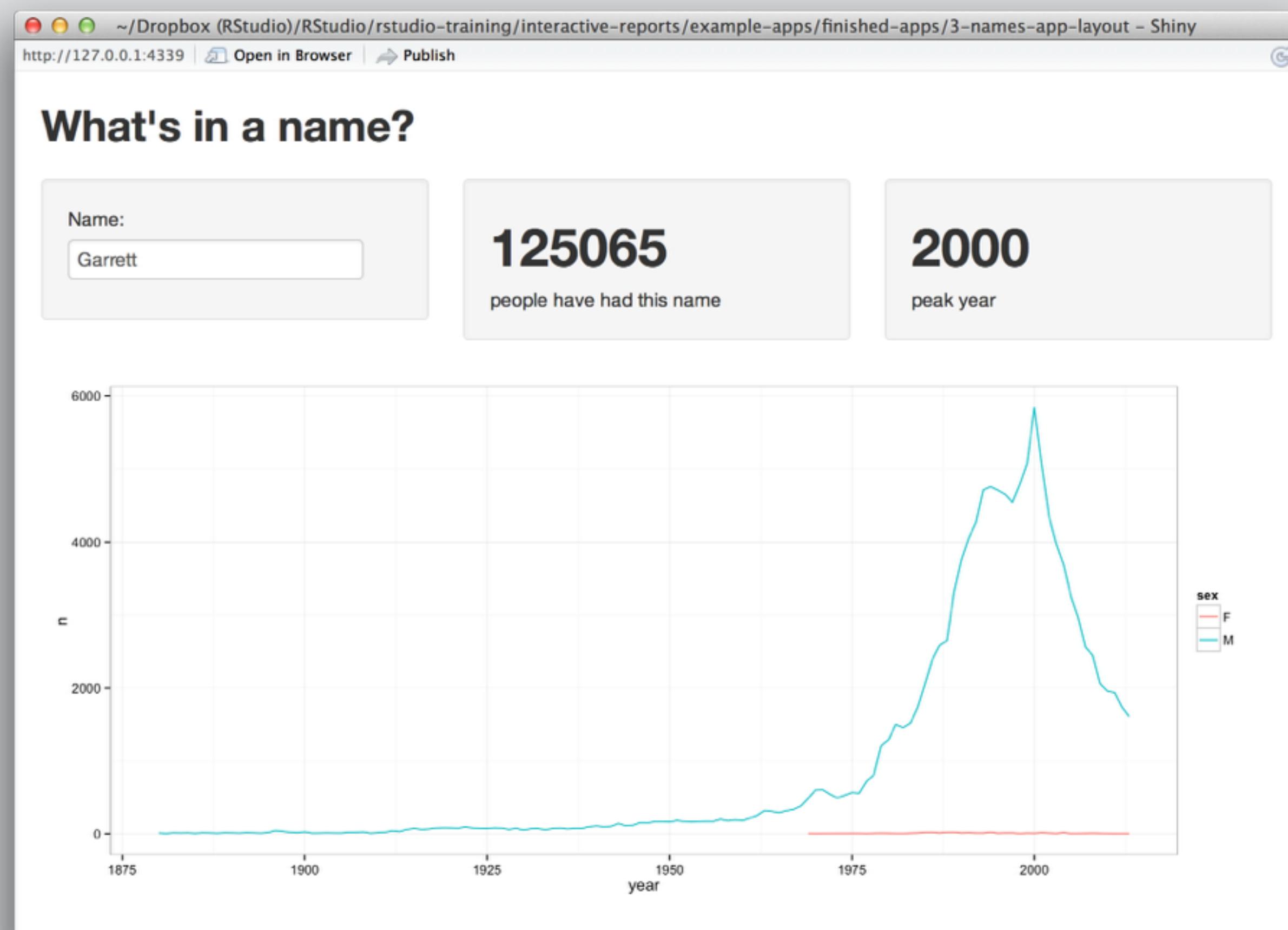
Recreate the grey background of the sidebarPanel by wrapping element(s) in wellPanel

```
shinyUI(fluidPage(  
  fluidRow(column(3),  
           column(5, "1st line",  
                  "2nd line")),  
  fluidRow(  
    wellPanel(  
      "3rd line",  
      "4th line"))  
)
```

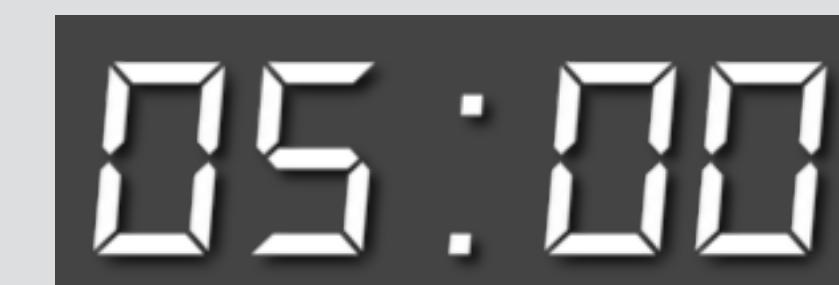


Your turn

Modify your ui.R file to recreate the app below.
Run the app.



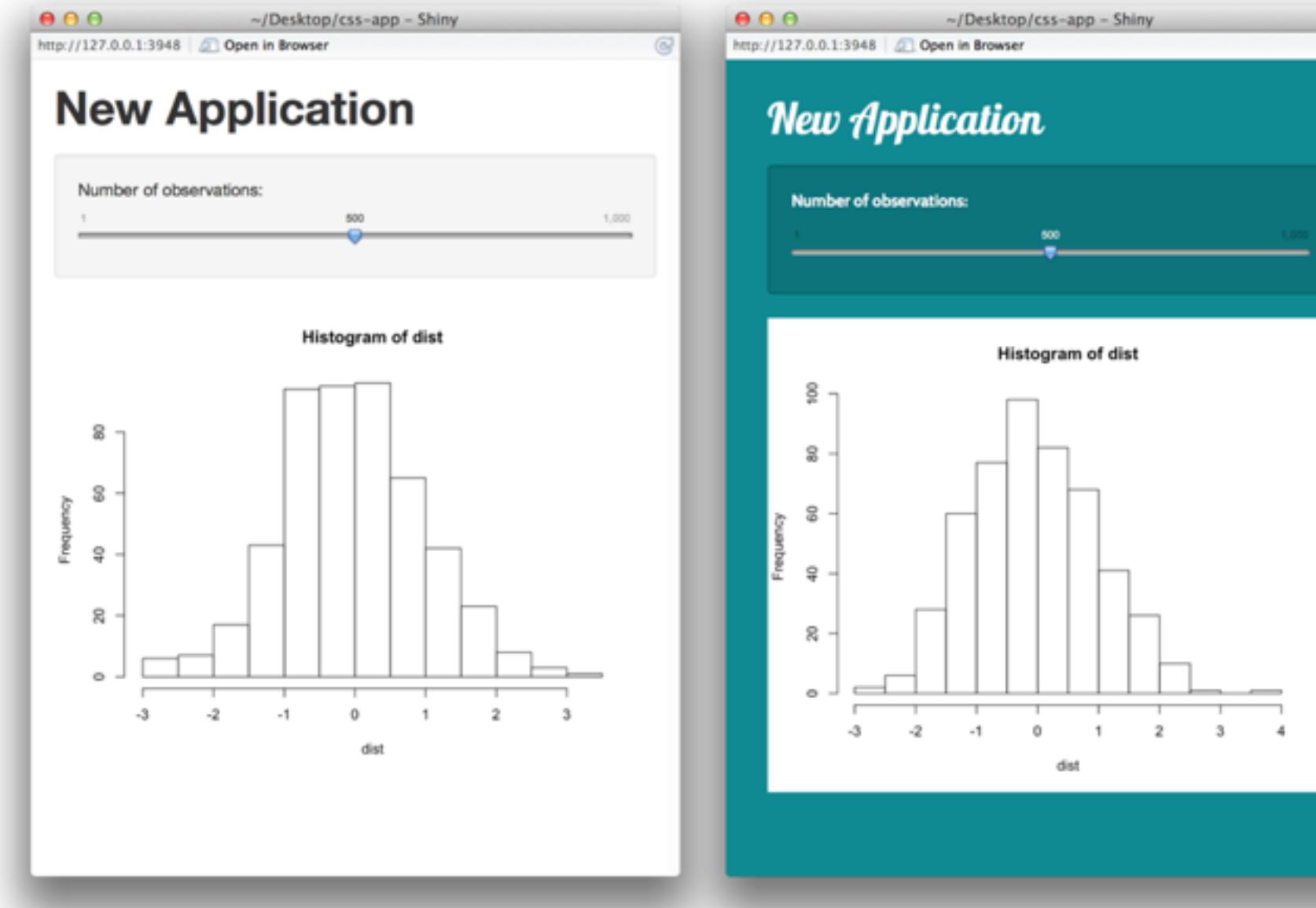
Tip: You can still add titlePanel() at the top of an app that uses the grid layout



```
shinyUI(fluidPage(  
  
  titlePanel("What's in a name?"),  
  
  fluidRow(  
    column(4,  
      wellPanel(  
       textInput("name", "Name:", value = "Garrett"))),  
    column(4,  
      wellPanel(  
        h1(textOutput("total")),  
        "people have had this name")),  
    column(4,  
      wellPanel(  
        h1(textOutput("peak")),  
        "peak year"))),  
  
  fluidRow(plotOutput("trend"))  
)
```

Customize your apps with HTML, CSS, and Javascript

<http://shiny.rstudio.com/articles/css.html>



You can pair any app with whatever web technologies you wish. The above guide explains how to style your app with CSS.

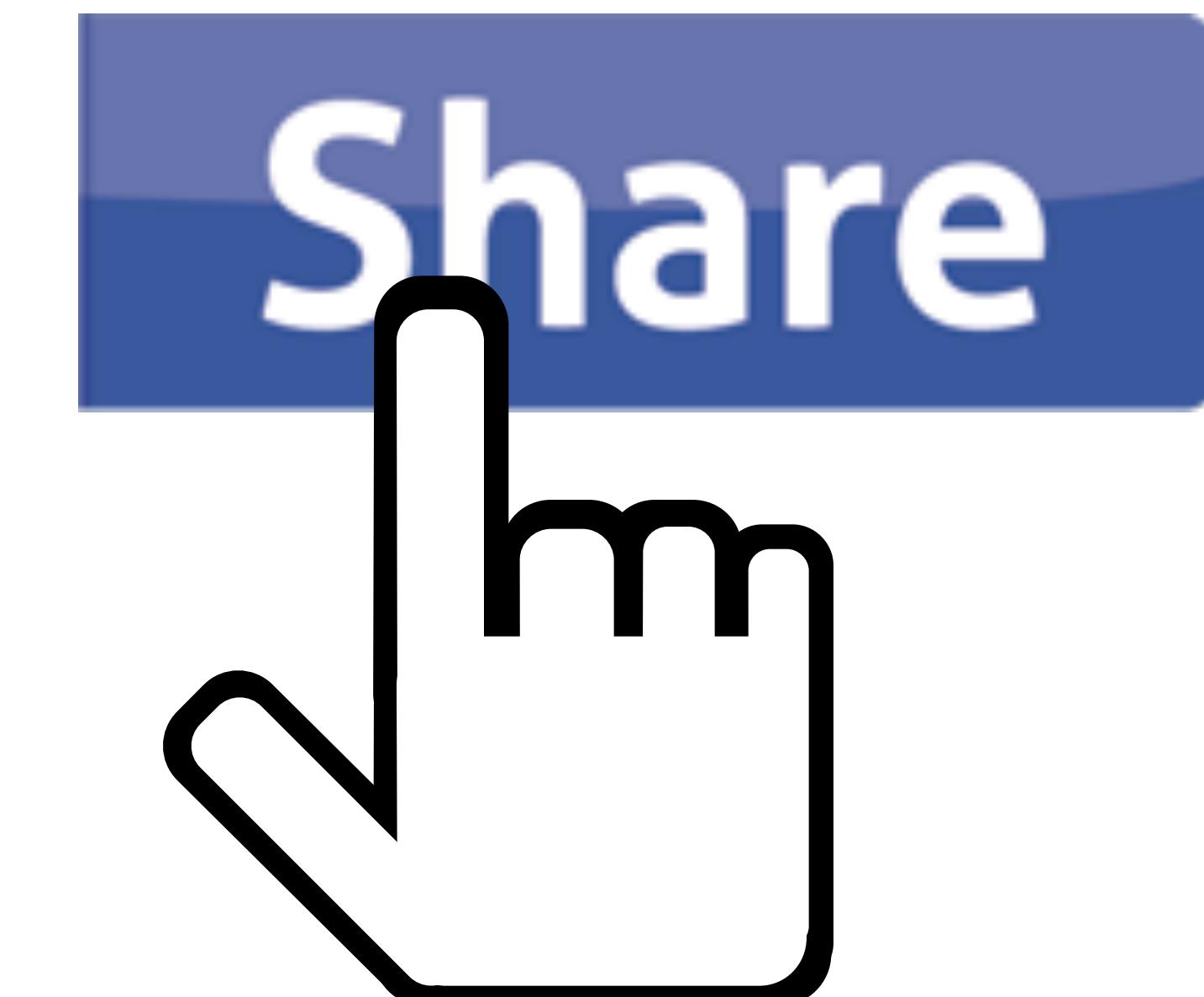
Share

These functions can help users run your server.R and ui.R files on their own computer.

runApp()

runGitHub()

runGist()

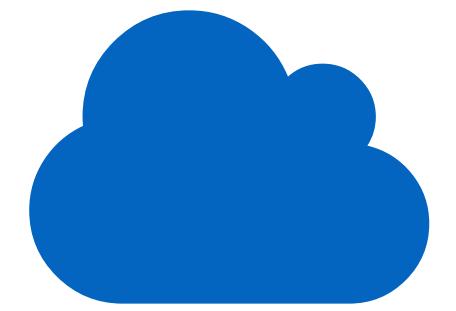


[https://github.com/rstudio/shiny_example/](https://github.com/rstudio/shiny_example)

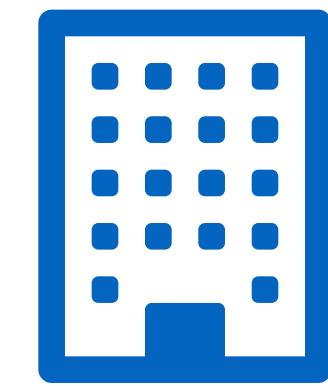
```
runGitHub("shiny_example", "rstudio")
```

<https://gist.github.com/jcheng5/3239667>

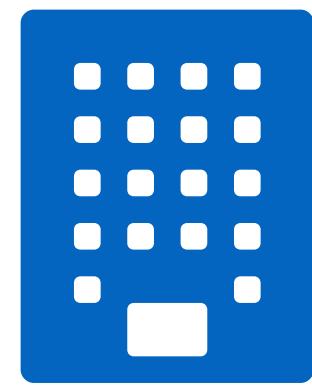
```
runGist("3239667")
```



[ShinyApps.io](#)



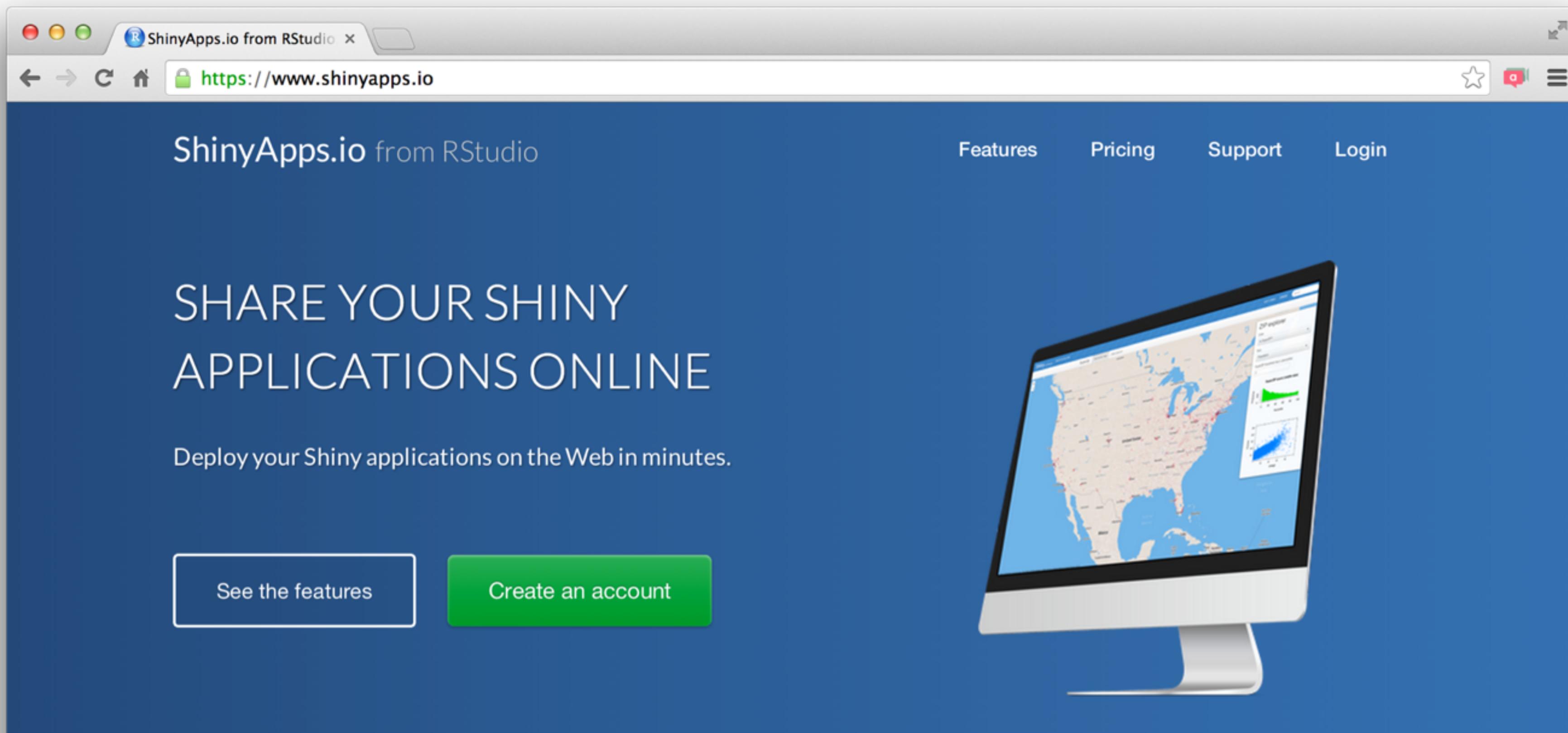
[Shiny Server](#)



[Shiny Server Pro](#)

ShinyApps.io

Hassle-free cloud hosting for Shiny



Looking for an easy way to deploy Shiny?

ShinyApps.io hosts your Shiny applications & documents.

Deploy to web with one click

The screenshot shows the RStudio interface with a blue circle highlighting the 'Publish' button in the top toolbar. The left pane displays an R script named 'ui.R' containing the code for a shiny application. The right pane shows the resulting shiny app with the title 'What's in a name?' and a text input field containing 'Garrett'. The console at the bottom shows the command 'filter' and a warning about masked objects.

```
# ui.R
shinyUI(fluidPage(
  titlePanel("What's in a name?"),
  fluidRow(
    column(4,
    wellPanel(
     textInput("name", "Name:", value = "Garrett")))))
#> Warning: package 'gridExtra' was built under R version 3.1.3
```

Console ~Dropbox (RStudio)/RStudio/rstudio-training/interactive-reports/ filter

The following objects are masked from ‘package:base’: intersect, setdiff, setequal, union

What's in a name?

Name:
Garrett

125065 people have had this name

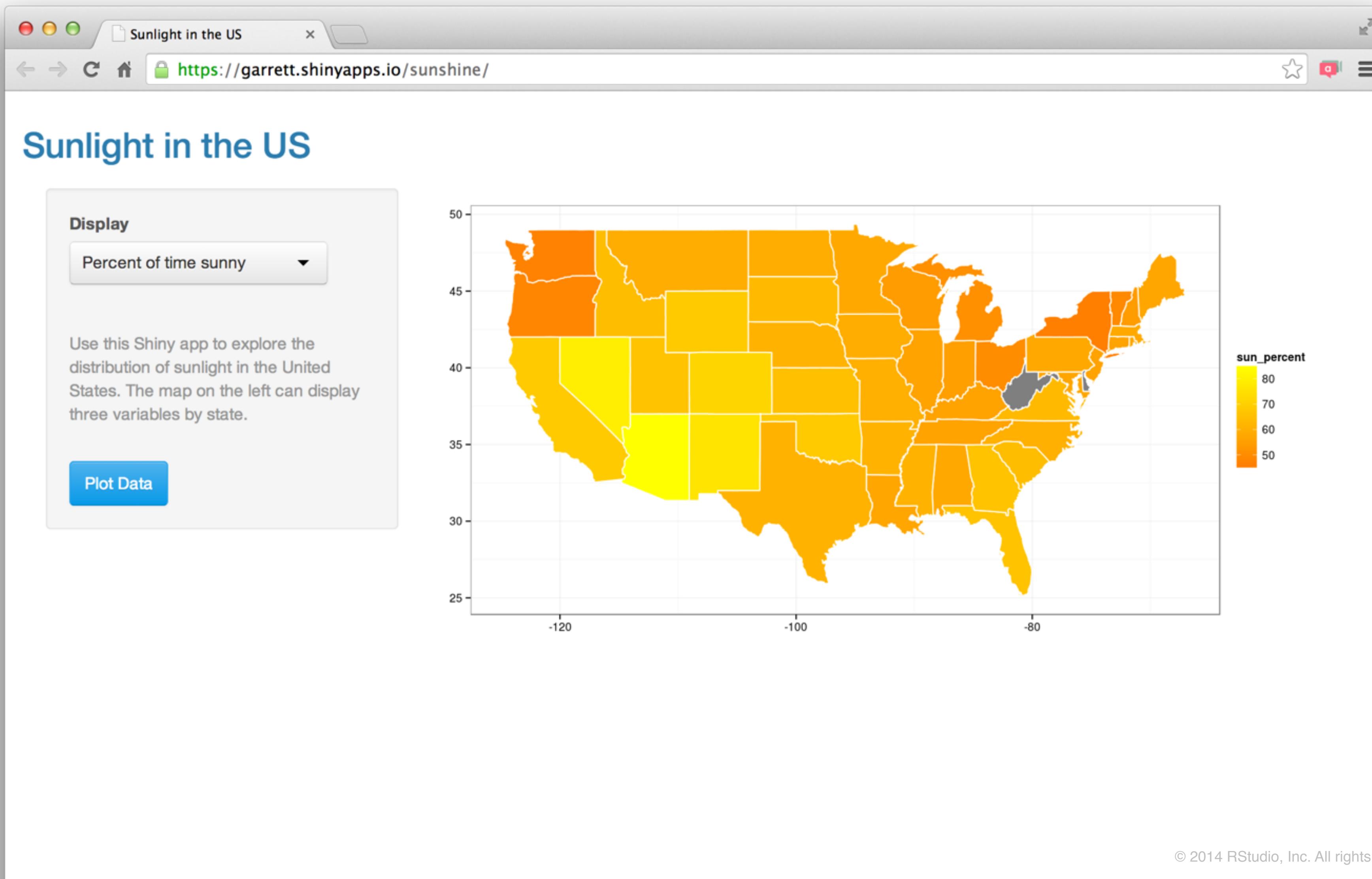
Basic admin tools

The screenshot shows the ShinyApps.io dashboard interface. On the left, a sidebar displays a user profile picture of Garrett Grolemund, the name "Garrett", and navigation links for "Dashboard", "Applications", and "Accounts". The main content area features a large blue box indicating "60 Apps online" with a "View your applications" button and a cloud icon. Below this, there's a table titled "Latest applications" listing six entries:

ID	Name	Status
11941	front-page	Running
11214	twitter	Running
11199	submit	Running
11195	text	Running
11194	slider	Running
11193	select	Running

At the bottom right, a footer note reads "© 2014 RStudio, Inc. All rights reserved."

Each app hosted at its own URL



Get Started

Guide at:

<http://shiny.rstudio.com/articles/shinyapps.html>

1. install `shinyapps` package
2. create account at www.shinyapps.io
3. Run `setAccountInfo()` at command line

`deployApp()` or click button to host app online

**Shiny
Server (Pro)**

Shiny Server

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

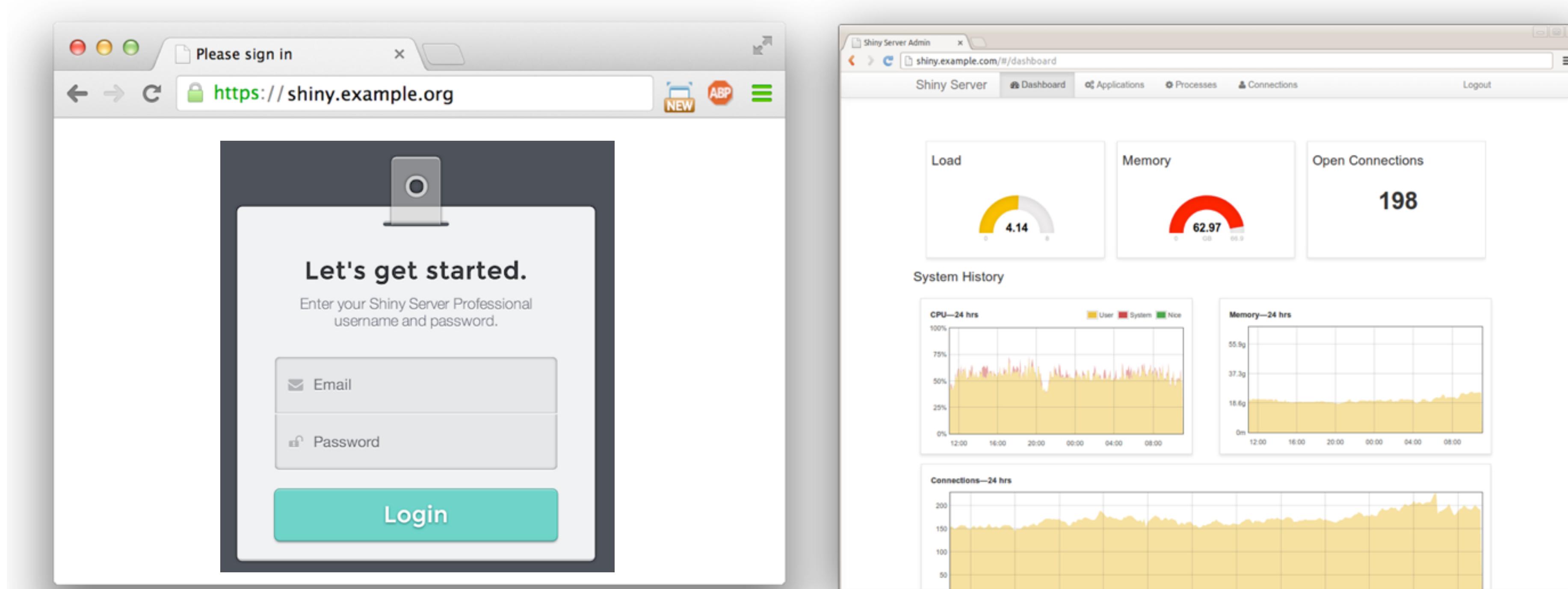
- Each app is hosted at its own URL
- Can deploy to internet, or within a controlled environment
- Starts app when user visits, closes app when user leaves
- Runs on a linux server

<http://shiny.rstudio.com/articles/shiny-server.html>

Shiny Server Pro

<http://www.rstudio.com/products/shiny-server-pro/>

- Password protect apps, LDAP, GoogleAuth, SSL, and more
- Administrative tools, priority support



<http://rstudio.com/shiny/server>

		Open Source Edition	Professional Edition
General	Deploy Shiny applications to the Internet	•	•
	Move computation close to the data	•	•
	Host multiple applications on a single server	•	•
	Deploy Shiny applications behind firewalls	•	•
	Custom page templates	•	•
Security and authentication	Password file authentication	•	
	LDAP and Active Directory authentication	•	
	Google authentication (OAuth2)	•	
	PAM authentication & sessions	•	
	Group based authorization	•	
	SSL support	•	
Tuning and scaling	Scale applications across multiple processes	•	
	View and manage active sessions	•	
	Allocate resources on a per application basis	•	
	Define application concurrency limits	•	
Server monitoring	System performance and resource metrics 	•	
	Per application performance and resource metrics 	•	
	Application usage metrics	•	
	Customizable health check end point	•	

* For volume discounts, OEMs, or additional capacity for larger audiences please email us at sales@rstudio.com.

How to learn more

Shiny

by RStudio

A web application framework for R

Turn your analyses into interactive web applications

No HTML, CSS, or JavaScript knowledge required

TUTORIAL

ARTICLES

GALLERY

REFERENCE

DEPLOY

HELP



Get inspired
(gallery)



Get started
(tutorial)



Go deeper
(articles)

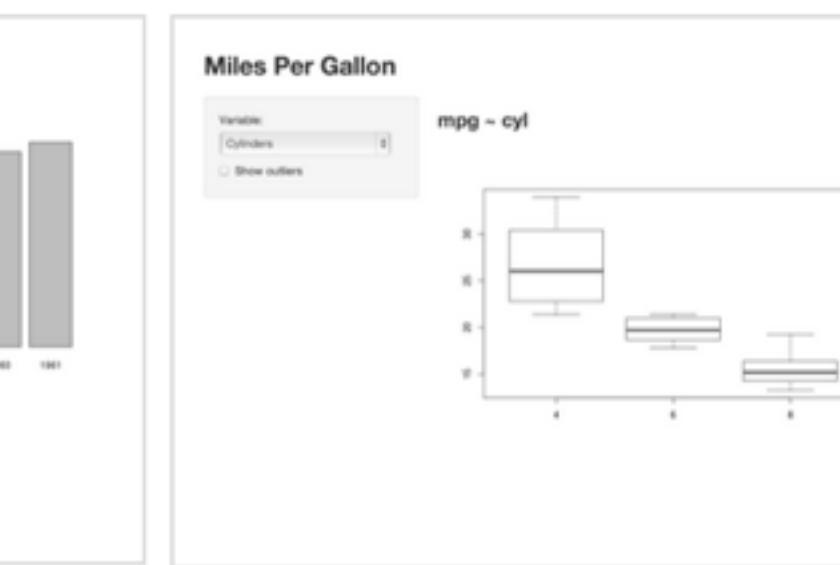
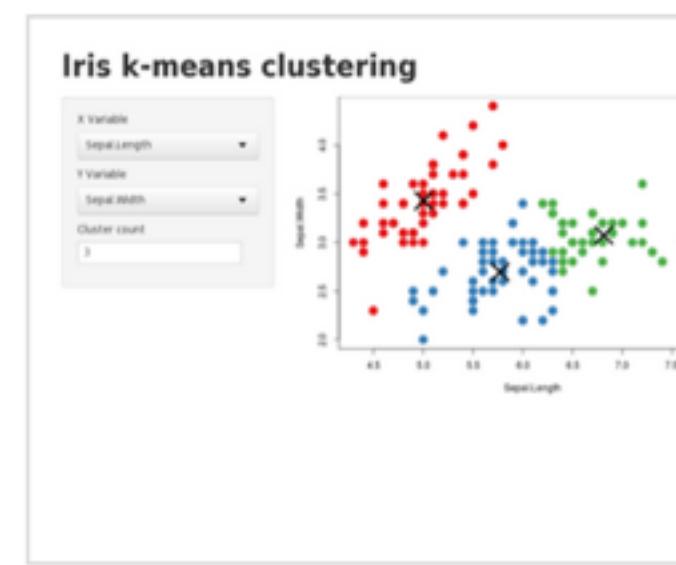
shiny.rstudio.com

[OVERVIEW](#)[TUTORIAL](#)[ARTICLES](#)[GALLERY](#)[REFERENCE](#)[DEPLOY](#)[HELP](#)

Gallery

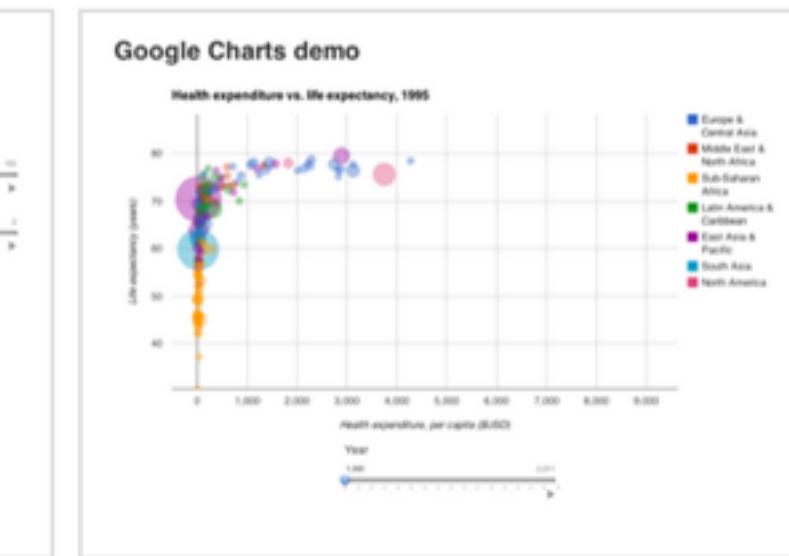
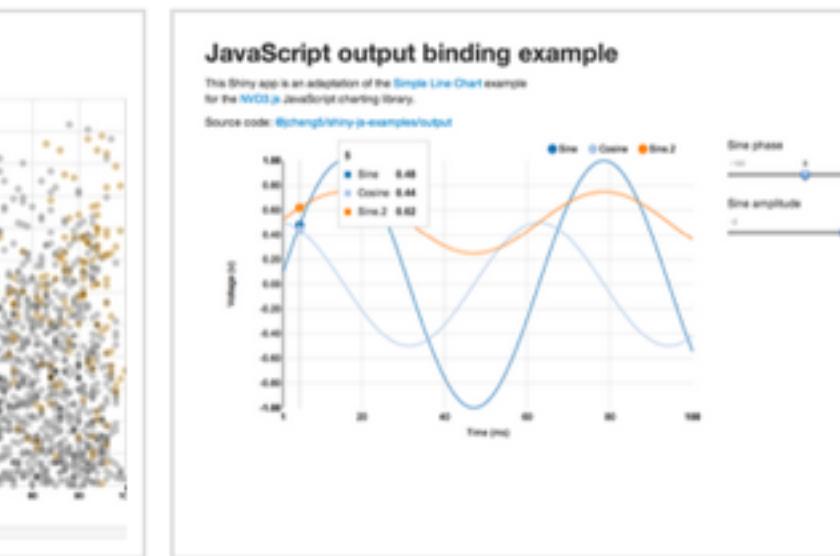
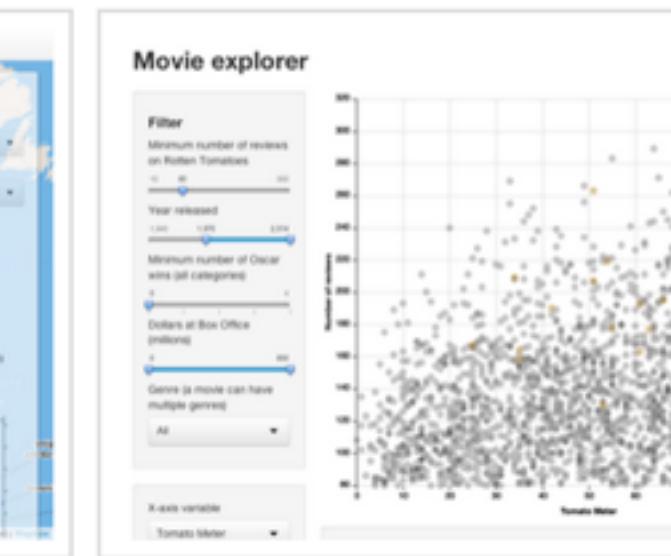
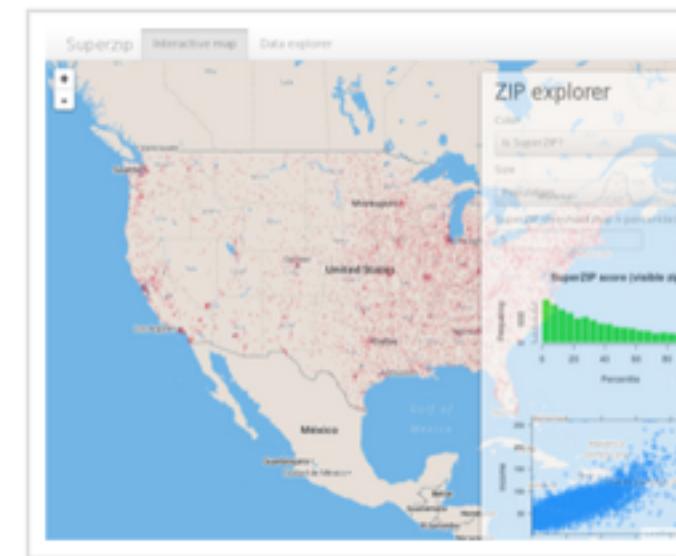
Start simple

If you're new to Shiny, these simple but complete applications are designed for you to study.

[Kmeans example](#)[Telephones by region](#)[Miles per gallon](#)[Word cloud](#)

Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like [d3](#), [Leaflet](#), and [Google Charts](#).

[SuperZip example](#)[Movie explorer](#)[NVD3 line chart output](#)[Google Charts](#)

Widgets

Get to know many of the input and output widgets that are available in Shiny with these examples.

The screenshot shows a Mac OS X window titled "~/Documents/widgets - Shiny" with the URL "http://127.0.0.1:4587". The window contains a title bar with standard OS X controls and a menu bar with "Open in Browser" and "Deploy" options. Below the title bar is a large blue header with the text "Shiny Widgets Gallery". The main content area displays seven examples of Shiny widgets:

- Action button**: A button labeled "Action". Below it, a "Current Value:" box shows the R code: [1] 0 attr(,"class") [1] "numeric" "shinyActionButtonValue". A "See Code" button is at the bottom.
- Single checkbox**: A checked checkbox labeled "Choice A". Below it, a "Current Value:" box shows the R code: [1] TRUE. A "See Code" button is at the bottom.
- Checkbox group**: Three checkboxes labeled "Choice 1" (checked), "Choice 2" (unchecked), and "Choice 3" (unchecked). Below them, a "Current Values:" box shows the R code: [1] "1". A "See Code" button is at the bottom.
- Date input**: A date input field showing "2014-01-01".
- Date range**: Two date input fields showing "2014-06-20" and "to" and "2014-06-20".
- File input**: A file input field with the placeholder "no file selected" and a "Choose File" button.

At the bottom right of the content area, there is a copyright notice: "© 2014 RStudio, Inc. All rights reserved."

[OVERVIEW](#)[TUTORIAL](#)[ARTICLES](#)[GALLERY](#)[REFERENCE](#)[DEPLOY](#)[HELP](#)

Teach yourself Shiny

Who should take the tutorial?

You will get the most out of this tutorial if you already know how to program in R, but not Shiny.

If R is new to you, you may want to check out the learning resources at www.rstudio.com/training before taking this tutorial. If you are not sure whether you are ready for Shiny, try our [quiz](#).

If you use Shiny on a regular basis, you may want to skip this tutorial and visit the articles section of the Development Center. In the articles section, we cover individual Shiny topics at an advanced level.

Get started with Shiny

With this seven lesson tutorial, you will move up from R programmer to Shiny developer. Each lesson takes about 15 minutes and teaches one new Shiny skill. By the end of the lessons, you will know how to build and deploy a Shiny app.

Each lesson includes an exercise. Don't skip the exercises, even if you are tempted to get to the next lesson. The learning occurs in the exercises. How do we know? Because we designed the tutorial and organized the material around the exercises.

Click the Lesson 1 button to get started and say hello to Shiny!

- [Lesson 1](#) - Welcome to Shiny
- [Lesson 2](#) - Layout the user interface
- [Lesson 3](#) - Add control widgets
- [Lesson 4](#) - Display reactive output
- [Lesson 5](#) - Use R scripts and data
- [Lesson 6](#) - Use reactive expressions
- [Lesson 7](#) - Share your apps

[Continue to lesson 1](#)

[OVERVIEW](#)[TUTORIAL](#)[ARTICLES](#)[GALLERY](#)[REFERENCE](#)[DEPLOY](#)[HELP](#)

Articles

The basics

If you've been through the [tutorial](#) and need a refresher, these articles are a good place to start. They describe the lay of the land.

[The basic parts of a Shiny app](#)[How to build a Shiny app](#)[How to launch a Shiny app](#)[How to get help](#)[The Shiny Cheat sheet](#)

Widgets

These articles describe Shiny's pre-built widgets and provide ideas on how to use them. (See also [Lesson 3](#) in the tutorial, and the Widgets section in the [gallery](#).)

[Using sliders](#)[Help users download data from your app](#)[Using selectize input](#)

Deploying apps

These articles describe the different ways to share your Shiny apps with users.

[Getting started with ShinyApps.io](#)[Introduction to Shiny Server](#)[Deploying Shiny apps over the web](#)[Sharing apps to run locally](#)

Layouts and UI

These articles explain how to control the layout, user-interface, and general appearance of your Shiny apps.

[Application layout guide](#)[Display modes](#)[Tabssets](#)[Customize your UI with HTML](#)[Build your entire UI with HTML](#)[Build a dynamic UI that reacts to user input](#)[Shiny HTML Tags Glossary](#)

Outputs

These articles show you how to create and use different output objects, the parts of your app that display results and react to user input.

[Render images in a Shiny app](#)[How to use DataTables in a Shiny App](#)

Reactive programming

These articles describe reactivity from a conceptual level. Understanding reactivity will help you build apps that are more efficient, robust, and correct.

[Reactivity: An overview](#)[Stop reactions with isolate\(\)](#)[Execution scheduling](#)

Shiny Cheat Sheet

learn more at shiny.rstudio.com

Shiny 0.10.0 Updated: 6/14



2. server.R

A set of instructions that build the R components of your app. To write server.R:

- A** Provide server.R with the minimum necessary code, `shinyServer(function(input, output) {})`
- B** Define the R components for your app between the braces that follow `function(input, output)`
- C** Save each R component in your UI as `output$<component name>`
- D** Create each output component with a render* function.
- E** Give each render* function the R code the server needs to build the component. The server will note any reactive values that appear in the code and will rebuild the component whenever these values change.
- F** Refer to widget values with `input$<widget name>`

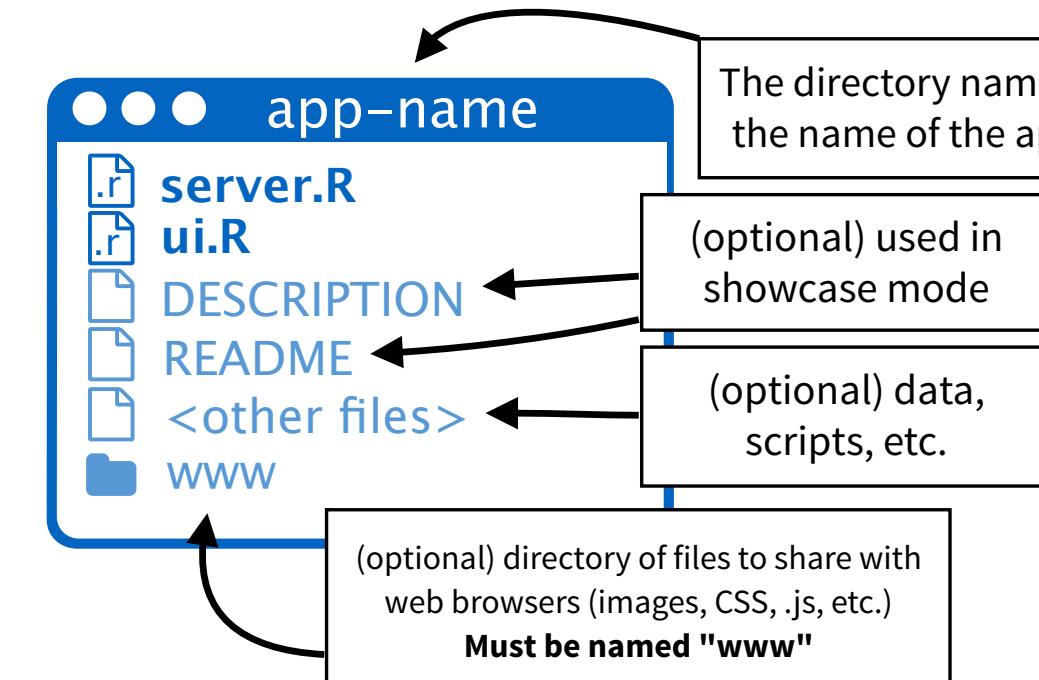
4. Reactivity

When an input changes, the server will rebuild each output that depends on it (even if the dependence is indirect). You can control this behavior by shaping the chain of dependence.

RStudio® and Shiny™ are trademarks of RStudio, Inc.
All rights reserved info@rstudio.com
844-448-1212 rstudio.com

1. Structure

Each app is a directory that contains a `server.R` file and usually a `ui.R` file (plus optional extra files)



render* functions

function	expects	creates
renderDataTable	any table-like object	DataTables.js table
renderImage	list of image attributes	HTML image
renderPlot	plot	plot
renderPrint	any printed output	text
renderTable	any table-like object	plain table
renderText	character string	text
renderUI	Shiny tag object or	UI element (HTML)

input values are reactive.

They must be surrounded with one of:

- render*** - creates a shiny UI component
- reactive** - creates a reactive expression
- observe** - creates a reactive observer
- isolate** - creates a non-reactive copy of a reactive object

server.R

```
# load libraries, scripts, data
A shinyServer(function(input, output) {
  # make user specific variables
  output$text <- renderText({
    input$title
  })
  C output$plot <- renderPlot({
    D x <- mtcars[, input$x] F
    E y <- mtcars[, input$y]
    plot(x, y, pch = 16)
  })
})
```

3. Execution

Place code where it will be run the minimum necessary number of times

Run once - code placed *outside* of `shinyServer` will be run once, when you first launch your app. Use this code to set up the tools that your server will only need one copy of.

Run once per user - code placed *inside* `shinyServer` will be run once each time a user visits your app (or refreshes his or her browser). Use this code to set up the tools that your server will need a unique copy of for each user.

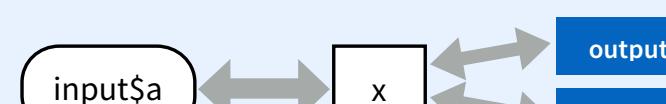
Run often - code placed within a `render*`, `reactive`, or `observe` function will be run many times. Place here any code that the server needs to rebuild a UI component whenever a widget changes.

render* - An output will automatically update whenever an input in its render* function changes.



```
output$z <- renderText({
  input$a
})
```

Reactive expression - use reactive to create objects that will be used in multiple outputs.



```
x <- reactive({
  input$a
})
output$y <- renderText({
  x()
})
output$z <- renderText({
  x()
})
```

isolate - use isolate to use an input without depending on it. Shiny will not rebuild the output when the isolated input changes.



```
output$z <- renderText(
  paste(
    isolate(input$b),
    input$b
  )
)
```

observe - use observe to run code that runs whenever an input value does not change.



```
observe(input$b, {
  output$z <- renderText(
    paste(
      isolate(input$b),
      input$b
    )
  )
})
```

in articles section

Shiny Discussion group

<https://groups.google.com/forum/#!forum/shiny-discuss>

The screenshot shows a web browser window titled '(99+) Shiny - Web Framework' displaying the Google Groups page for the 'shiny-discuss' forum. The URL in the address bar is <https://groups.google.com/forum/#!forum/shiny-discuss>. The page header includes the Google logo, a search bar, and user profile links for '+Garrett'. Below the header, there are buttons for 'Groups', 'NEW TOPIC', and 'Mark all as read'. On the left, a sidebar lists 'My groups', 'Home', 'Starred', and 'Favorites' (with a note to click stars to favorite). Under 'Recently viewed', items include 'Shiny - Web Frame...', 'twitter-bootstrap-st...', 'pandoc-discuss', 'GitHub', and 'asciidoc'. The main content area shows a topic titled 'Shiny - Web Framework for R' (Shared publicly) with 30 of 2207 topics (99+ unread). It features a 'Join group to post' button and a '8+1' badge. The topic description explains Shiny as a package for building web apps with R. Below the description, a message encourages users to post comments and share gists. A list of posts follows, with the first one highlighted:

Browser differences? (7)
By Barb Banbury - 7 posts - 22 views 3:46 PM

fileInput, reload and defaults (1)
By Andrew Booker - 1 post - 3 views 2:36 PM

sudden problems deploying shiny app from spark.studio (2)
By Veronica Morales - 2 posts - 7 views 11:24 AM

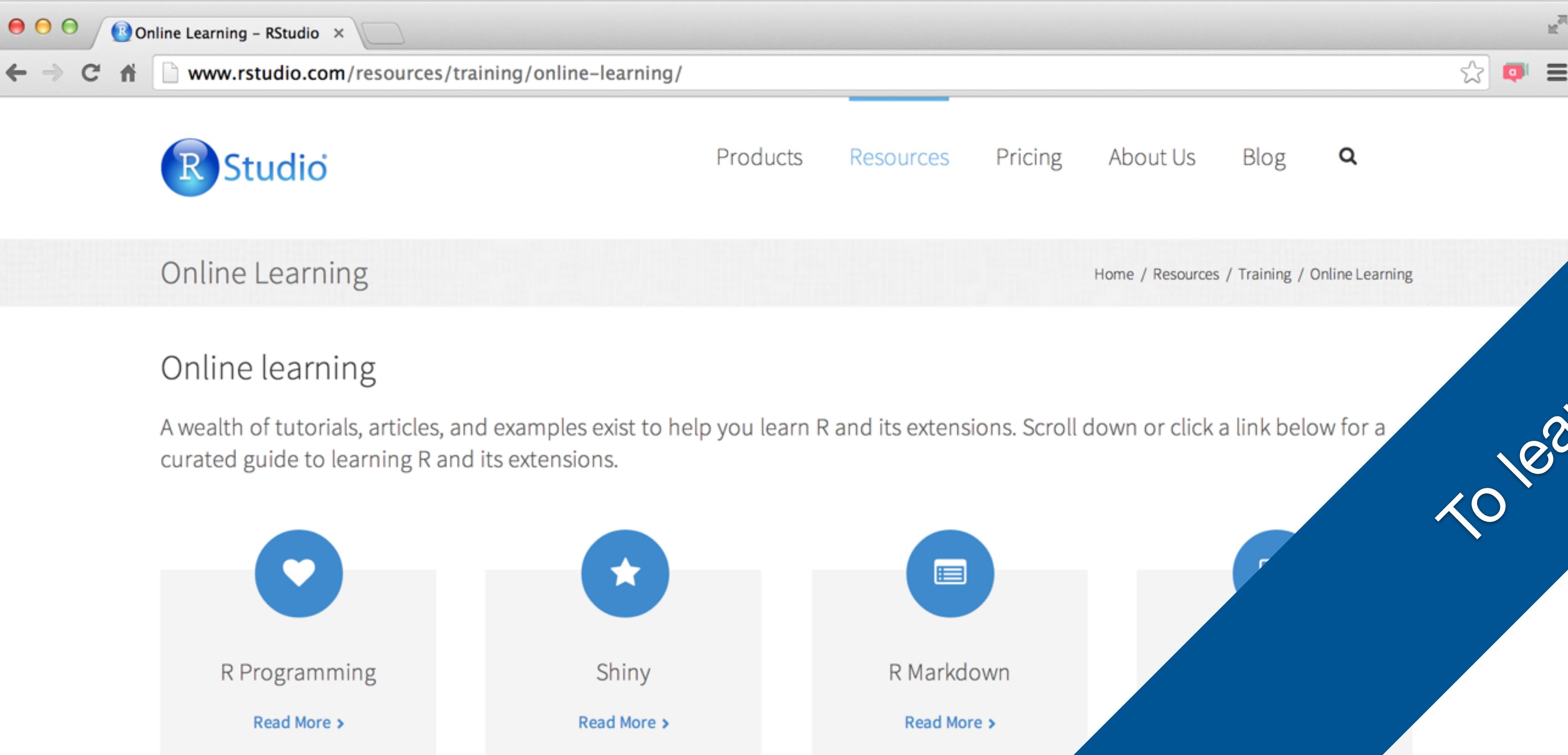
shiny, ggplot2, and X11 on R 3.0.2 (3)
By Daniel Bowen - 3 posts - 51 views 6:19 AM

At the bottom of the page is the URL <https://groups.google.com/forum/#!topic/shiny-discuss/rMAMf1vL6uY>.

RStudio training

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

Links to recommended tutorials, articles, examples
to help you learn R and its extensions



The screenshot shows a web browser window displaying the RStudio Online Learning page. The URL in the address bar is www.rstudio.com/resources/training/online-learning.html. The page features the RStudio logo and navigation links for Products, Resources, Pricing, About Us, and Blog. A search icon is also present. The main content area is titled "Online Learning" and includes a breadcrumb trail: Home / Resources / Training / Online Learning. Below this, a section titled "Online learning" describes the available resources. It states: "A wealth of tutorials, articles, and examples exist to help you learn R and its extensions. Scroll down or click a link below for a curated guide to learning R and its extensions." Three cards are visible: "R Programming" with a heart icon, "Shiny" with a star icon, and "R Markdown" with a document icon. Each card has a "Read More >" link at the bottom.

To learn R