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.

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

RStudio

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

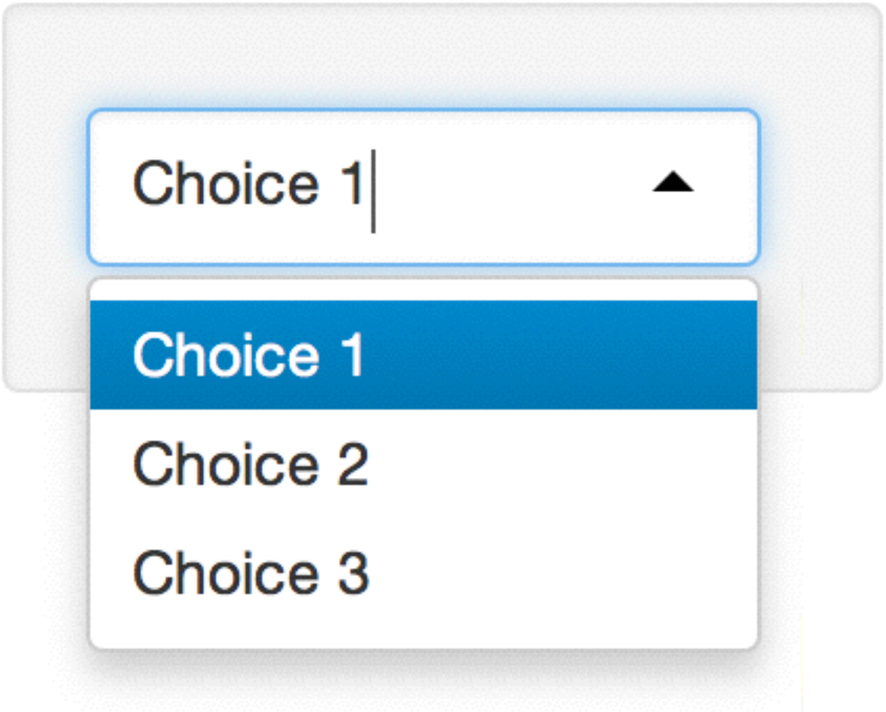
strataconf.com

#strataconf

#hadoopworld

Analytic Web Applications with Shiny

Introduction and possibilities



Garrett Grolemund

Data Scientist and Master Instructor
Email: garrett@rstudio.com
Follow @StatGarrett

garrett@rstudio.com
@StatGarrett

What is Shiny?

RStudio

Console ~/

```
> WorldPhones
     N.Amer Europe Asia S.Amer Oceania Africa Mid.Amer
1951  45939  21574 2876   1815    1646     89      555
1956  60423  29990 4708   2568    2366   1411      733
1957  64721  32510 5230   2695    2526   1546      773
1958  68484  35218 6662   2845    2691   1663      836
1959  71799  37598 6856   3000    2868   1769      911
1960  76036  40341 8220   3145    3054   1905     1008
1961  79831  43173 9053   3338    3224   2005     1076
>
>
> summary(iris$Sepal.Width)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.800   3.000   3.057   3.300   4.400
>
>
> lm(mpg ~ disp + hp, data = mtcars)

Call:
lm(formula = mpg ~ disp + hp, data = mtcars)

Coefficients:
(Intercept)         disp           hp
   30.73590     -0.03035     -0.02484

>
> hist(iris$Sepal.Width, border = "white", col = "darkgrey")
>
```

Environment | History

Import Dataset | Clear | Grid

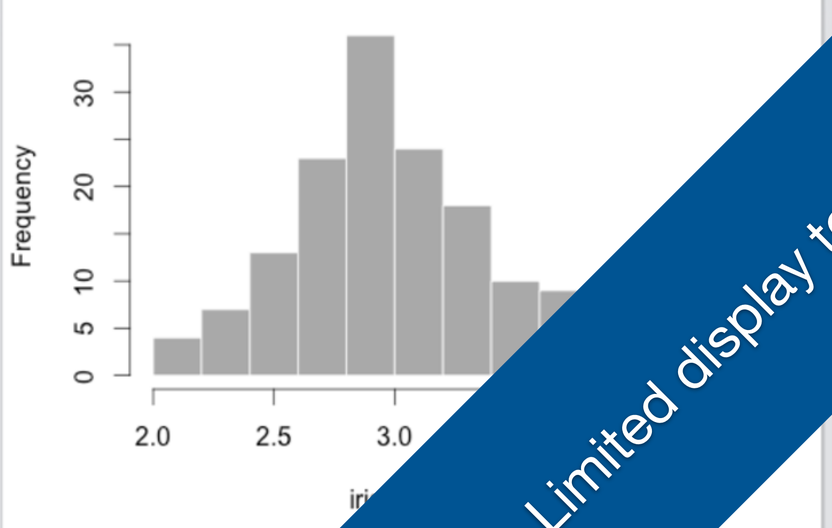
Global Environment

Name	Type	Length	Size	Value

Environment is empty

Files | Plots | Packages | Help | Viewer

Zoom | Export | Clear All



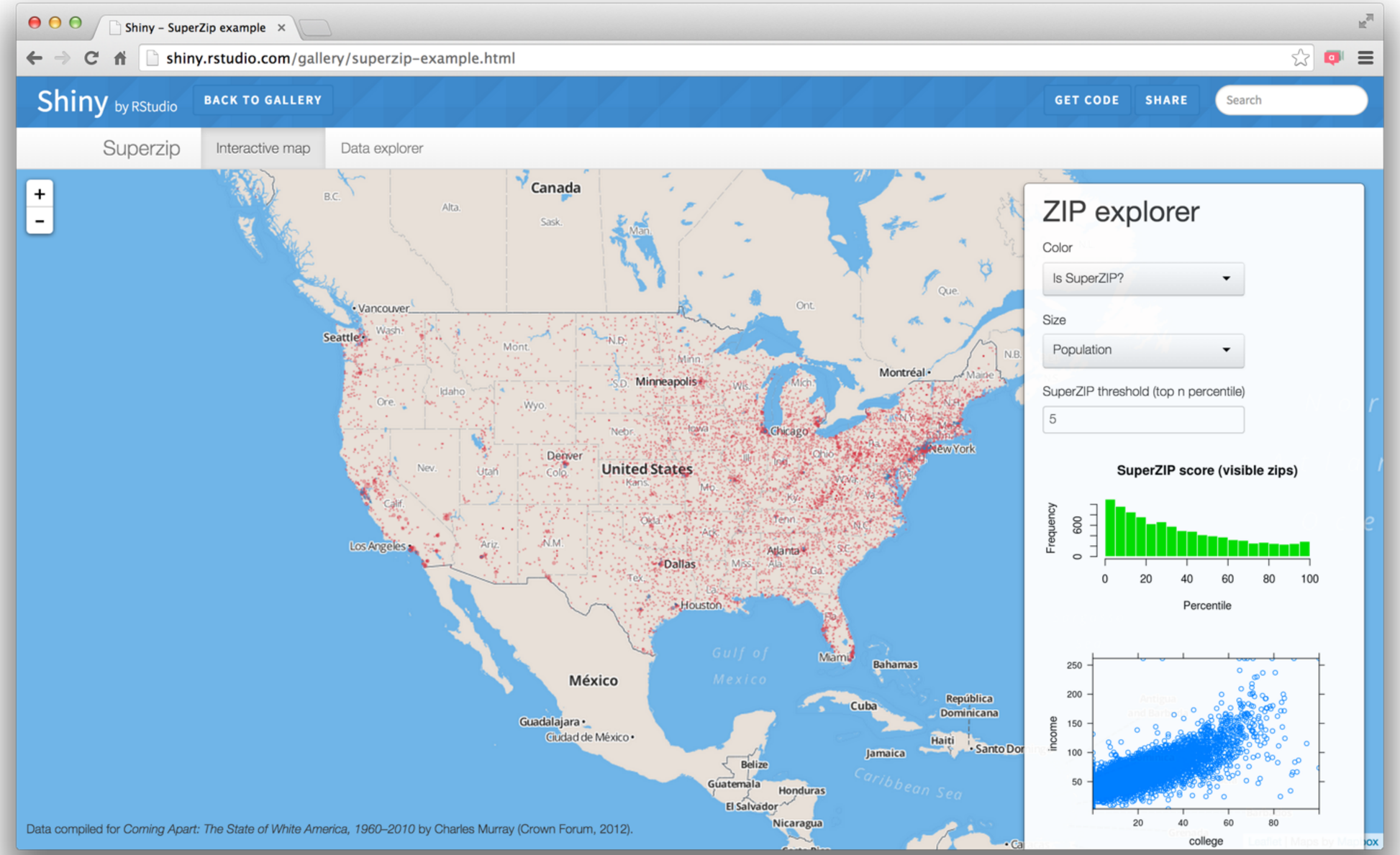
Limited display tools

Shiny

+

Analytic Power

Display abilities and interactivity



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

Data Products

2

Extend ggvis for EDA

3

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:

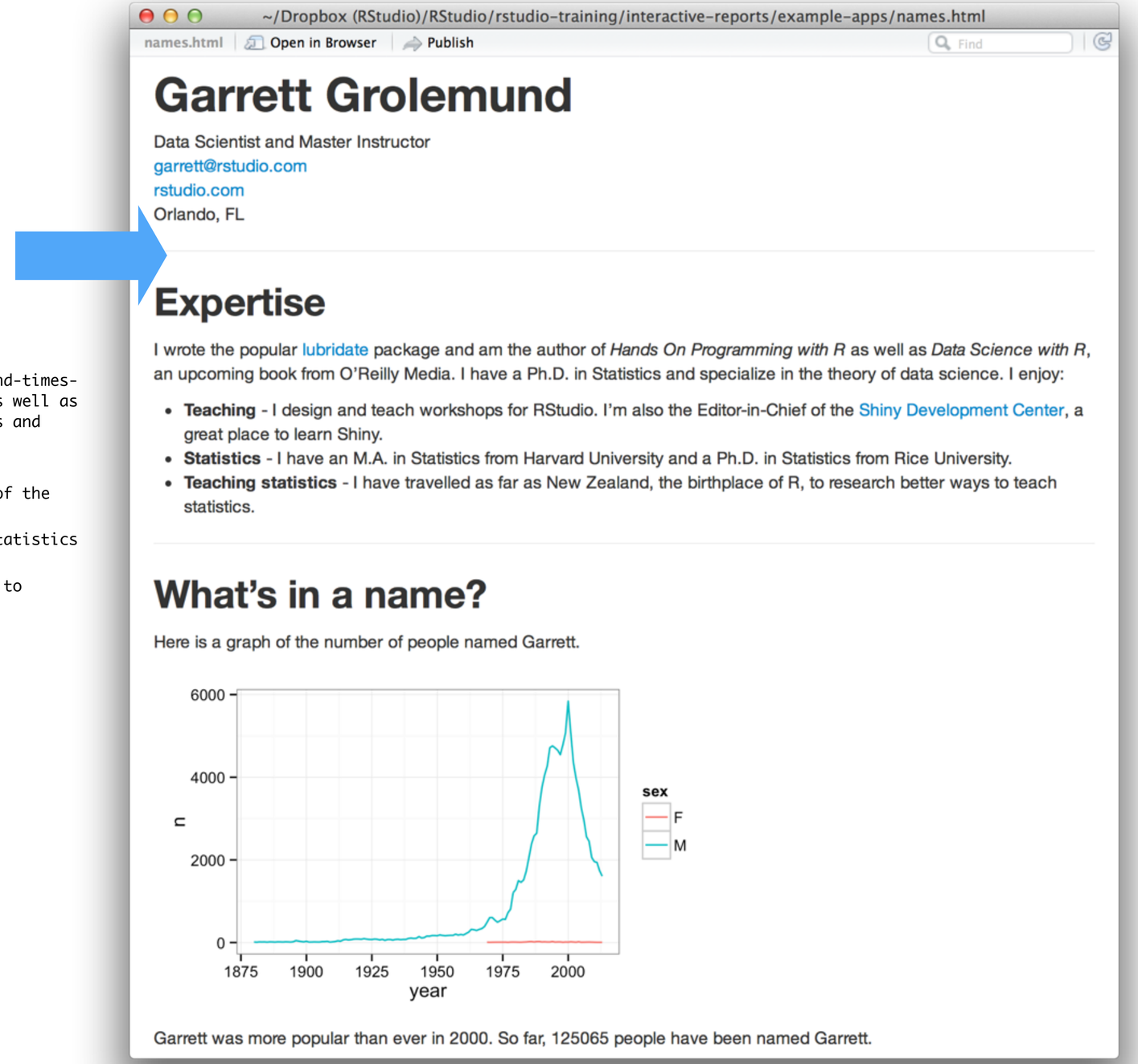
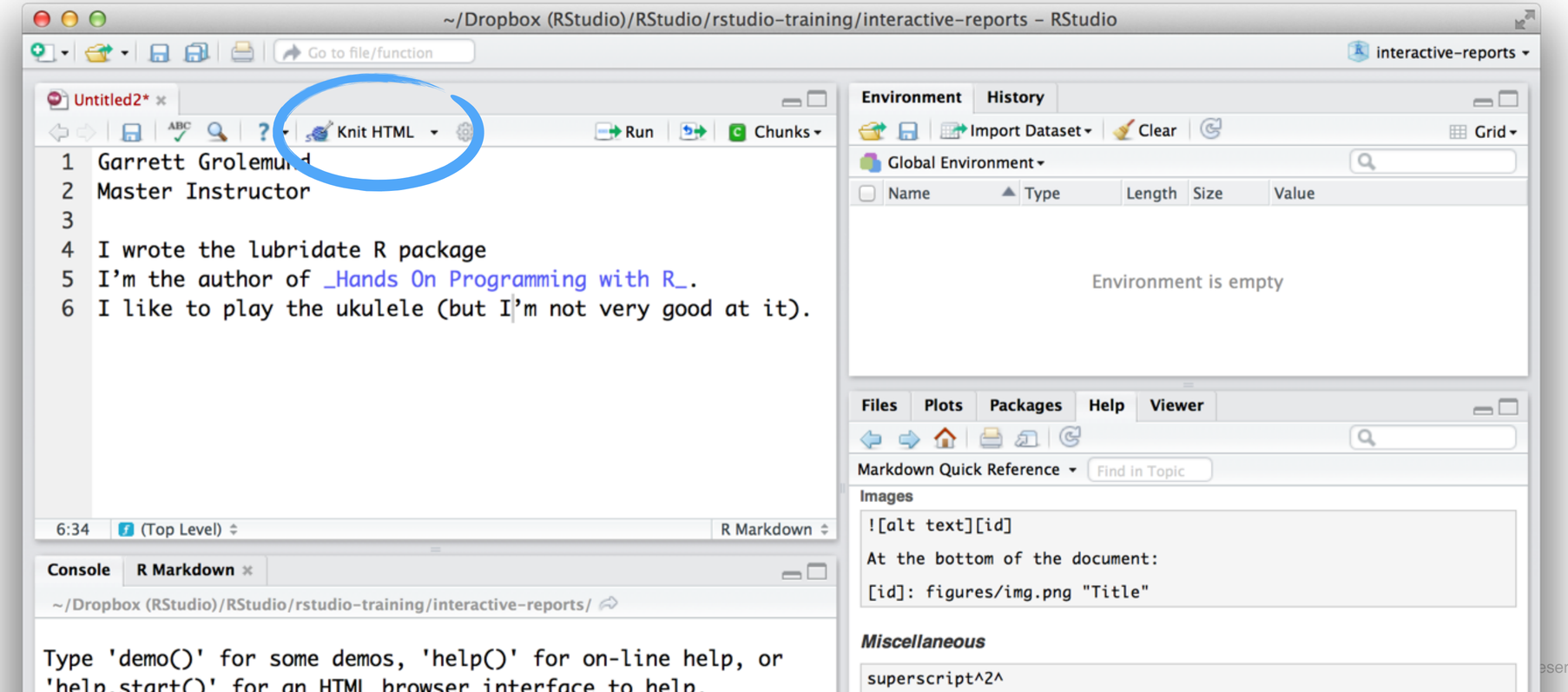
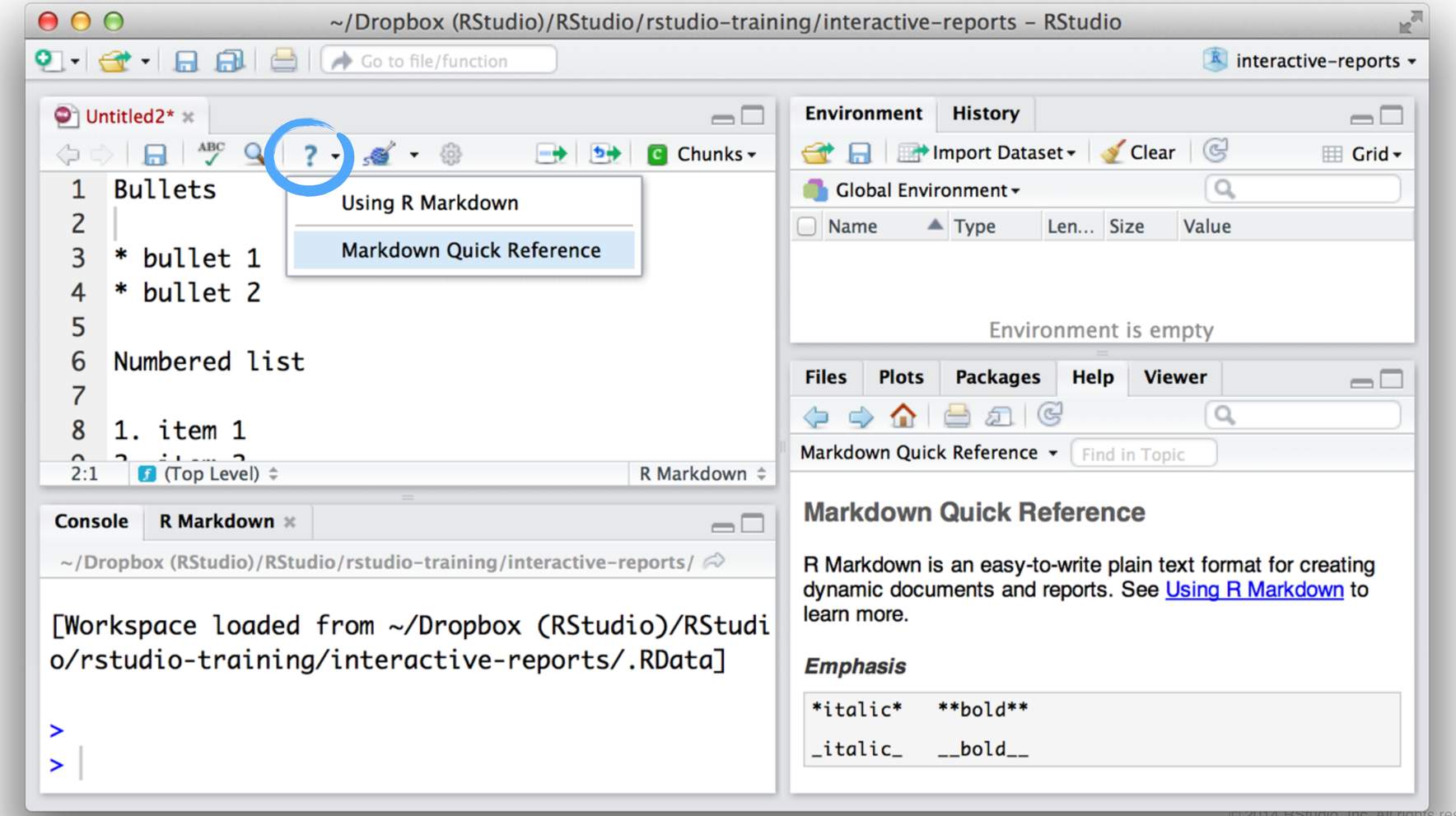
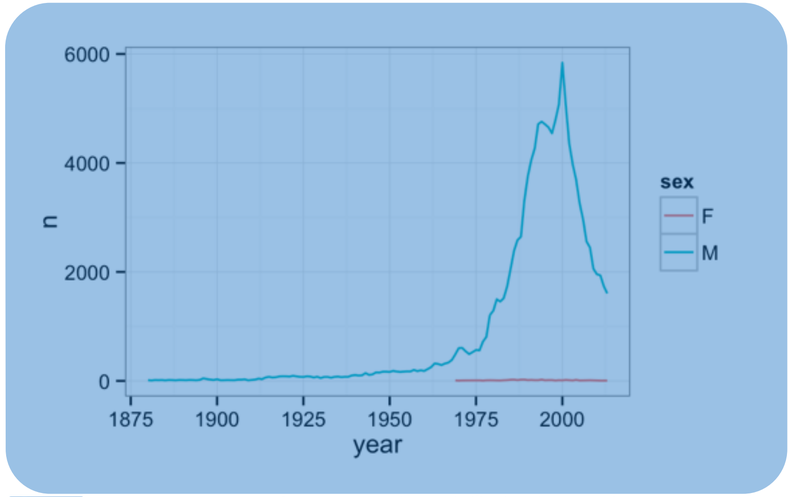
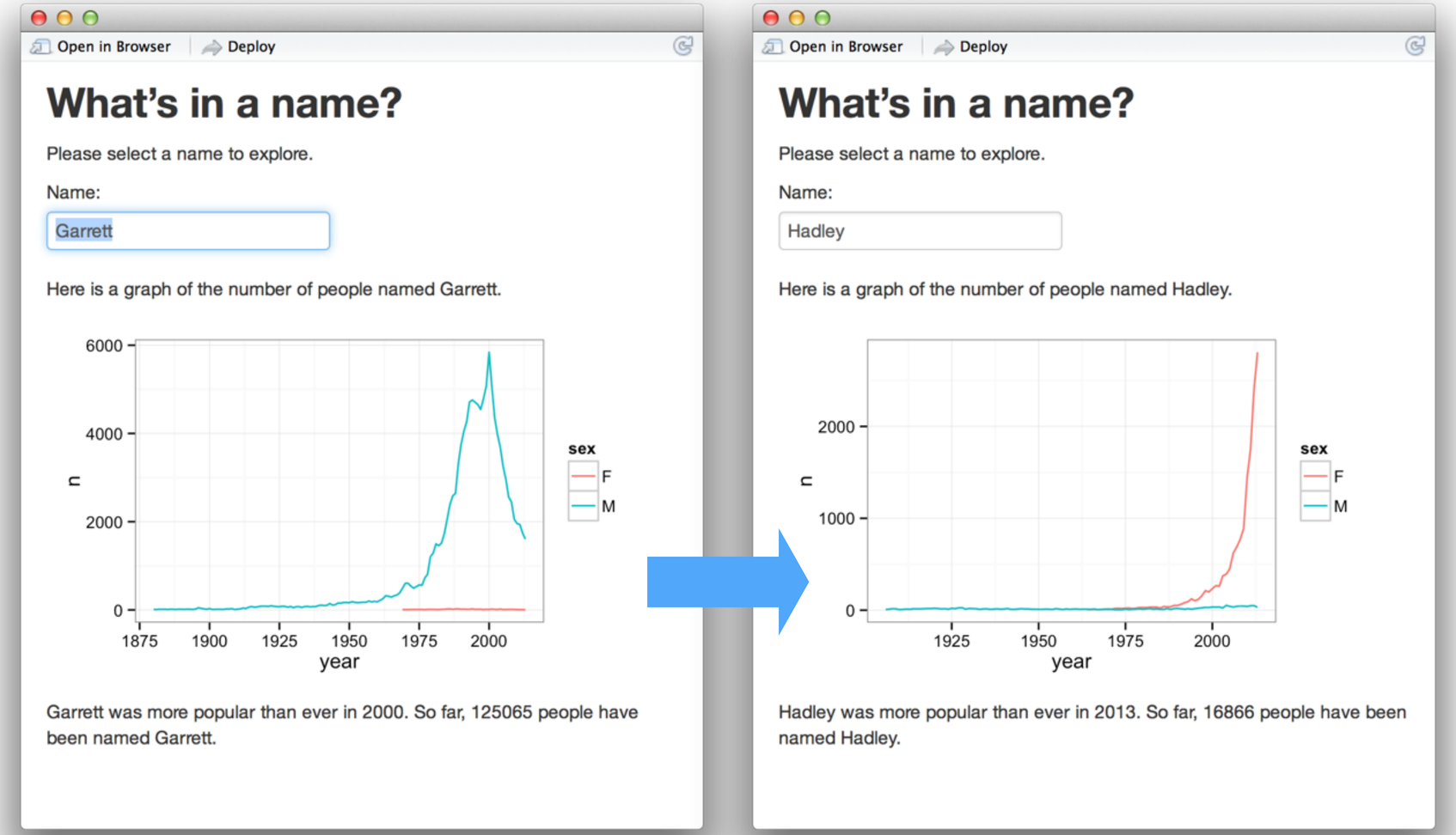
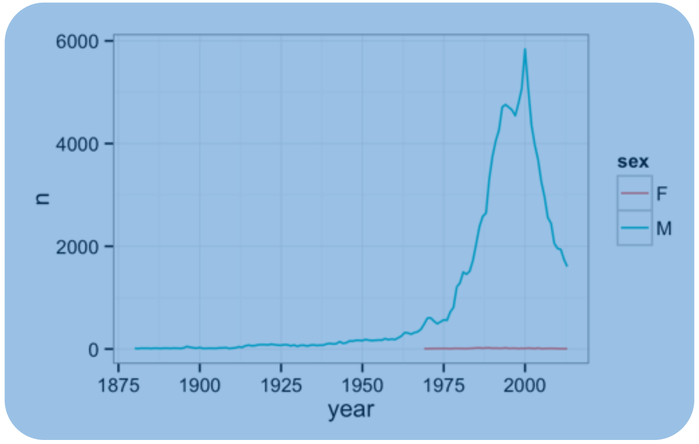
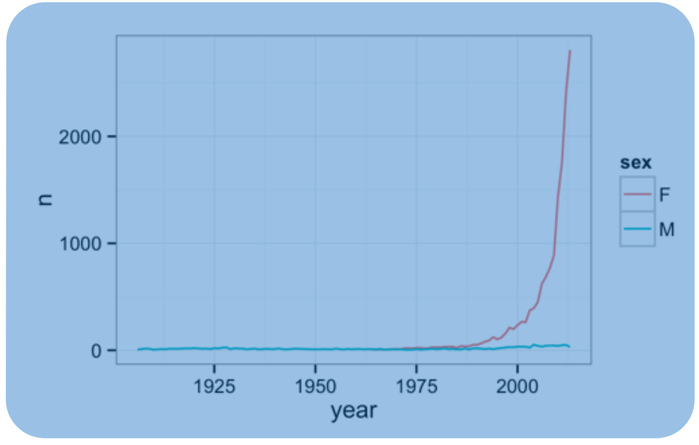
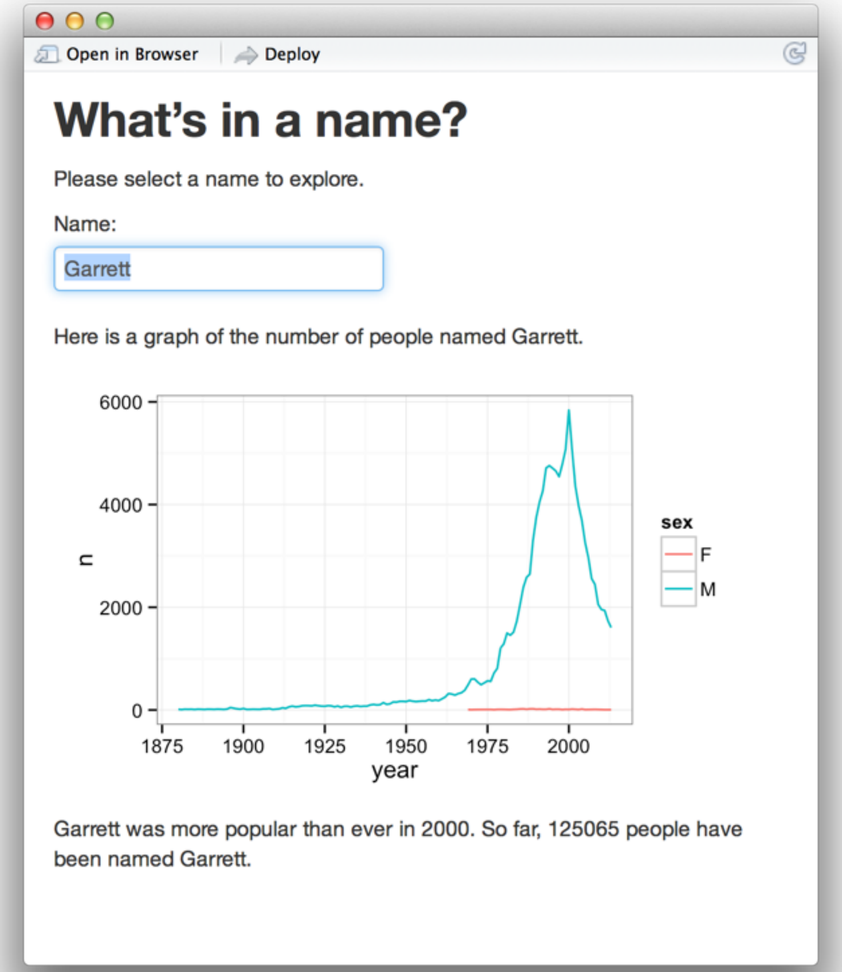
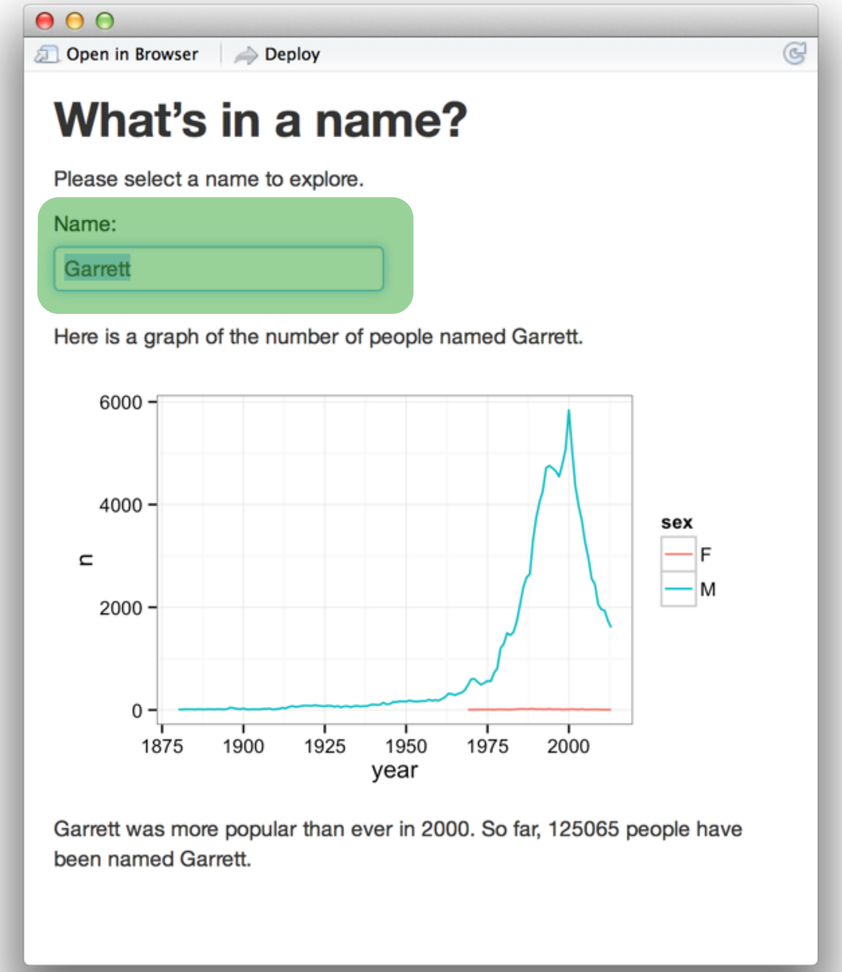
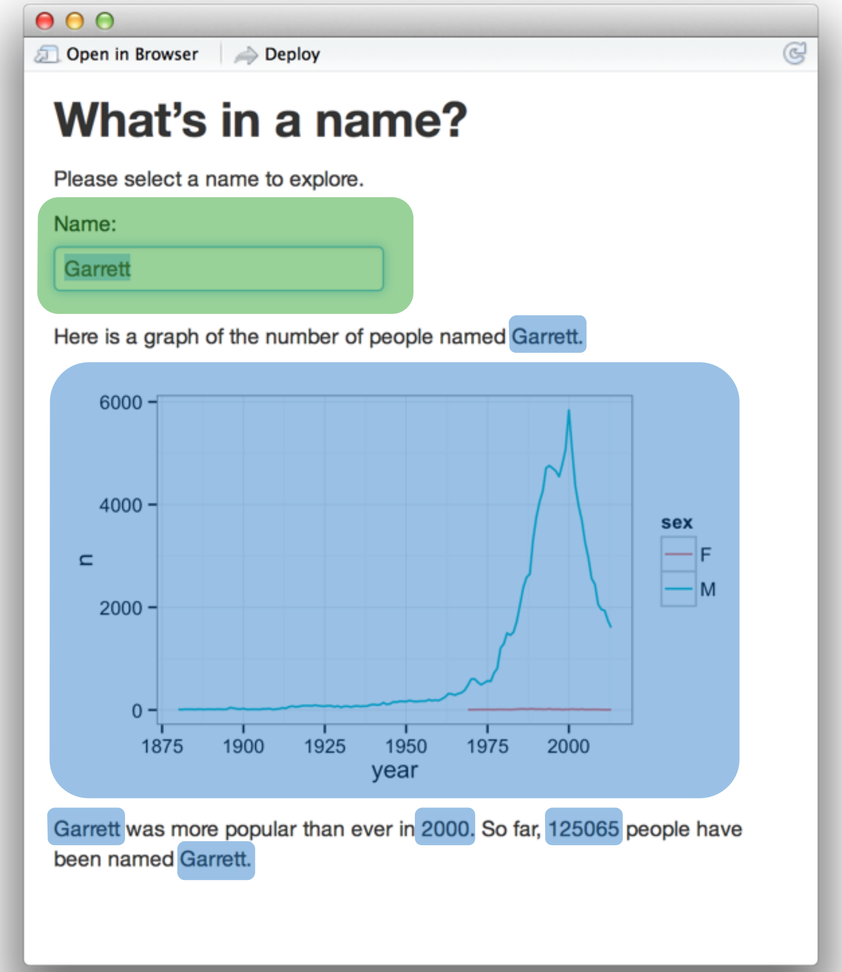
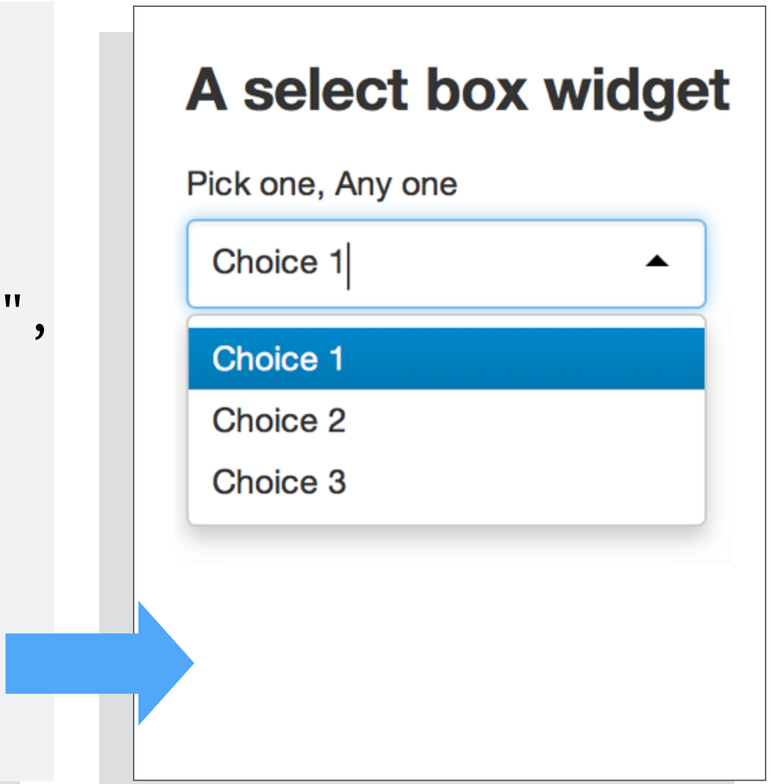
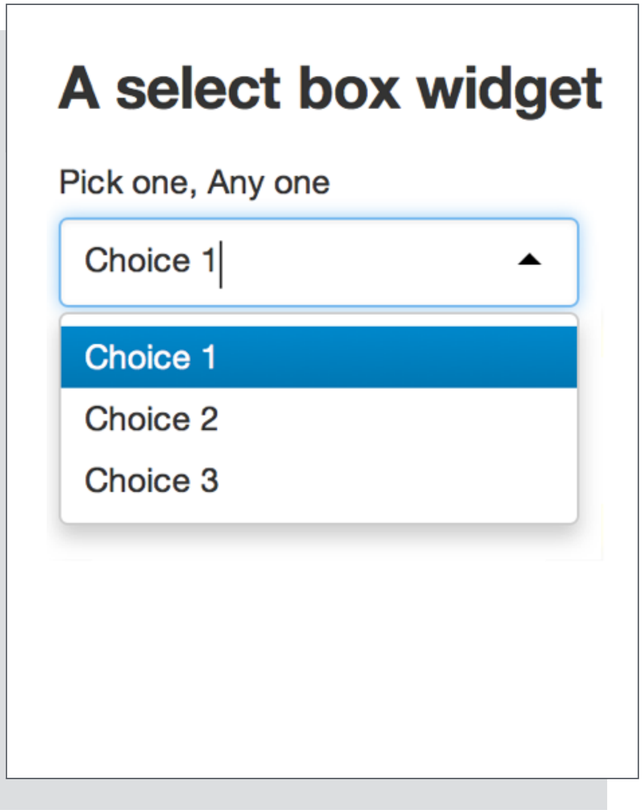
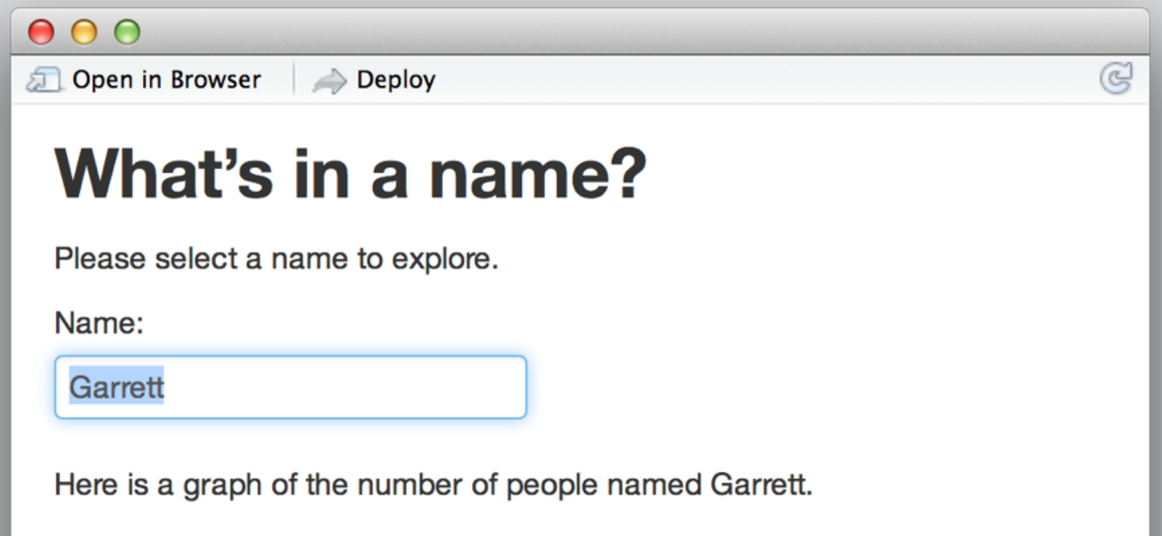
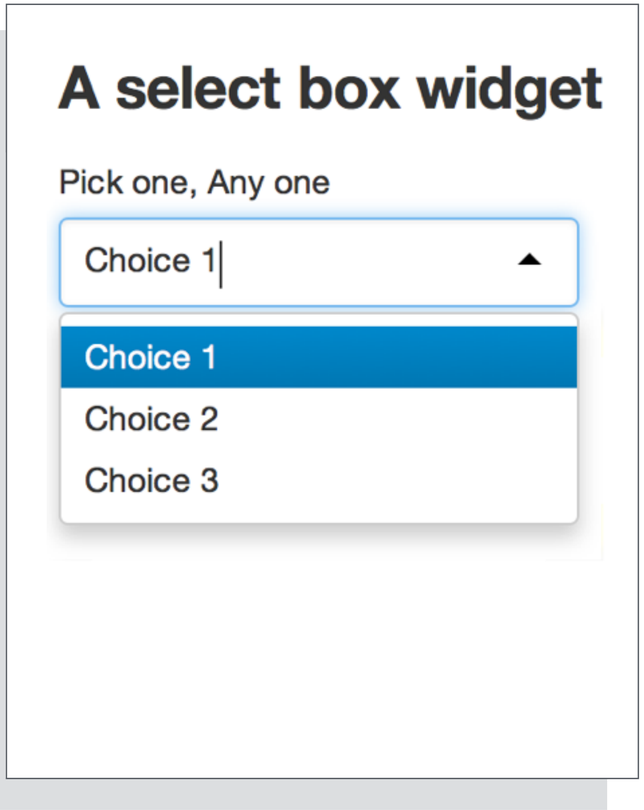
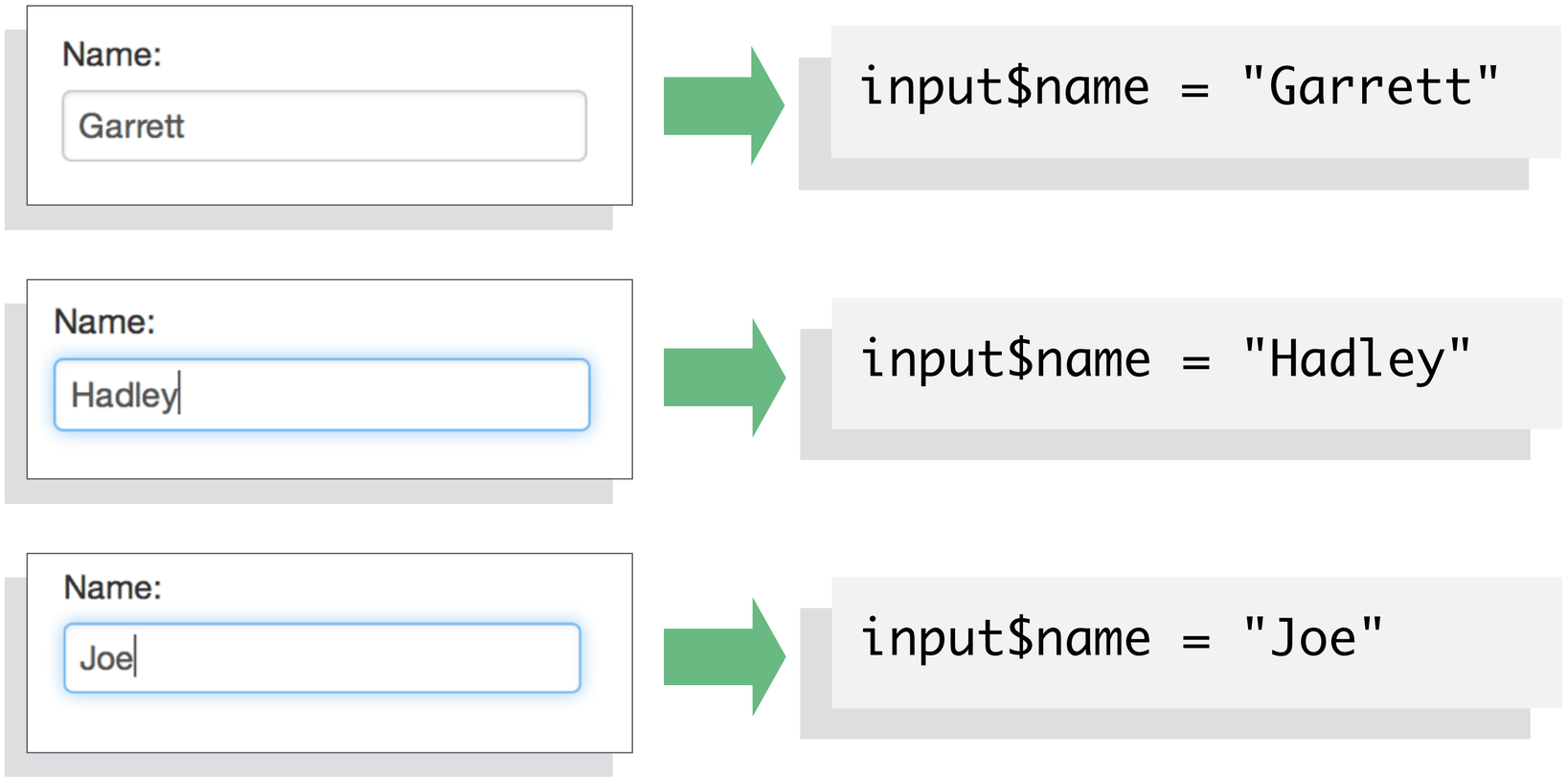
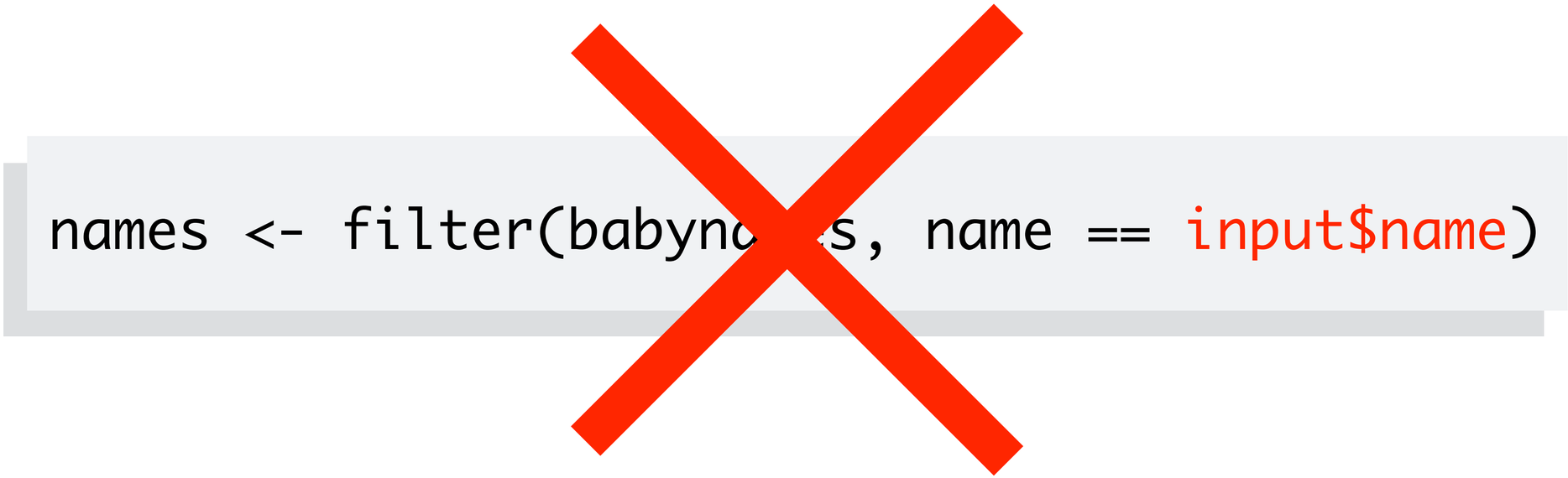
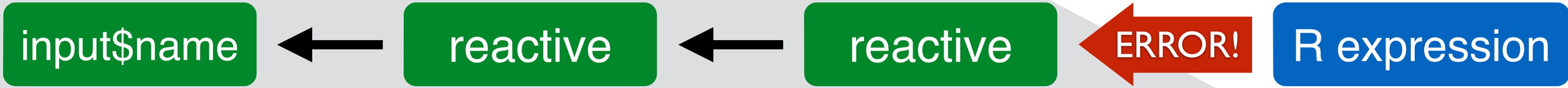
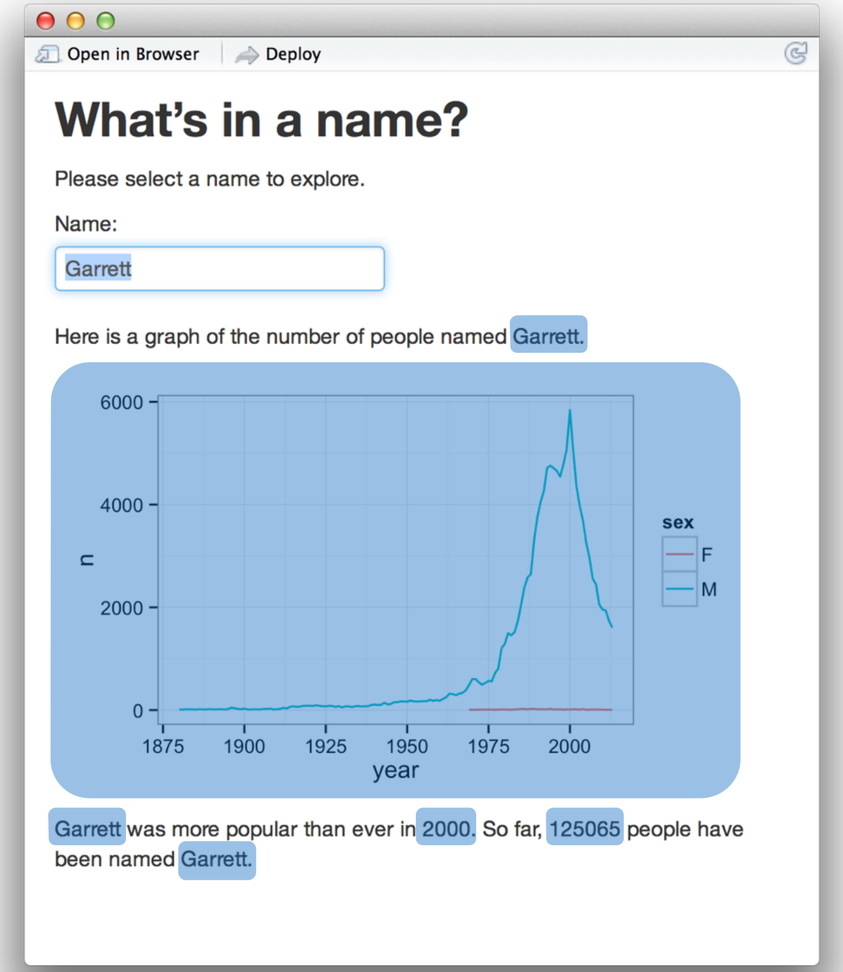
* __Teaching__ - I design and teach workshops for RStudio. I'm also the Editor-in-Chief of the [Shiny Development Center](http://shiny.rstudio.com), a great place to learn Shiny.
* __Statistics__ - I have an M.A. in Statistics from Harvard University and a Ph.D. in Statistics from Rice University.
*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

****

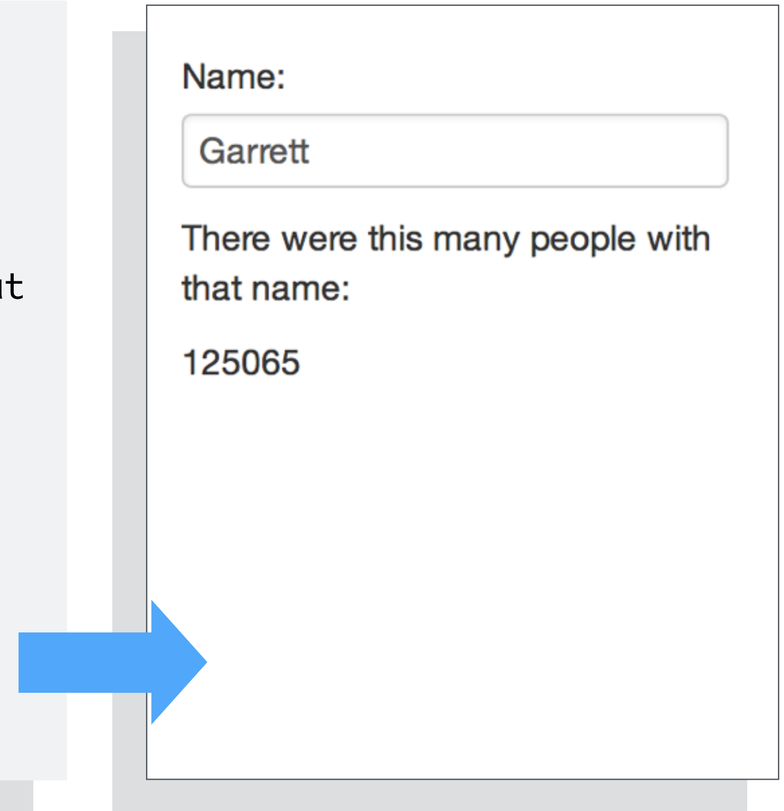
## What's in a name?

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

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.



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

What's in a name?

Please select a name to explore.

Name:

Garrett

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



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

Open in Browser | Deploy

What's in a name?

Please select a name to explore.

Name:

Hadley

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



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
---

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

****
```

Tell R Markdown to run the doc as a shiny app

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

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

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

Radio buttons

Choice 1
Choice 2
Choice 3

radioButtons

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



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*



Input values



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).



Standard values

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)})
```
````



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
---

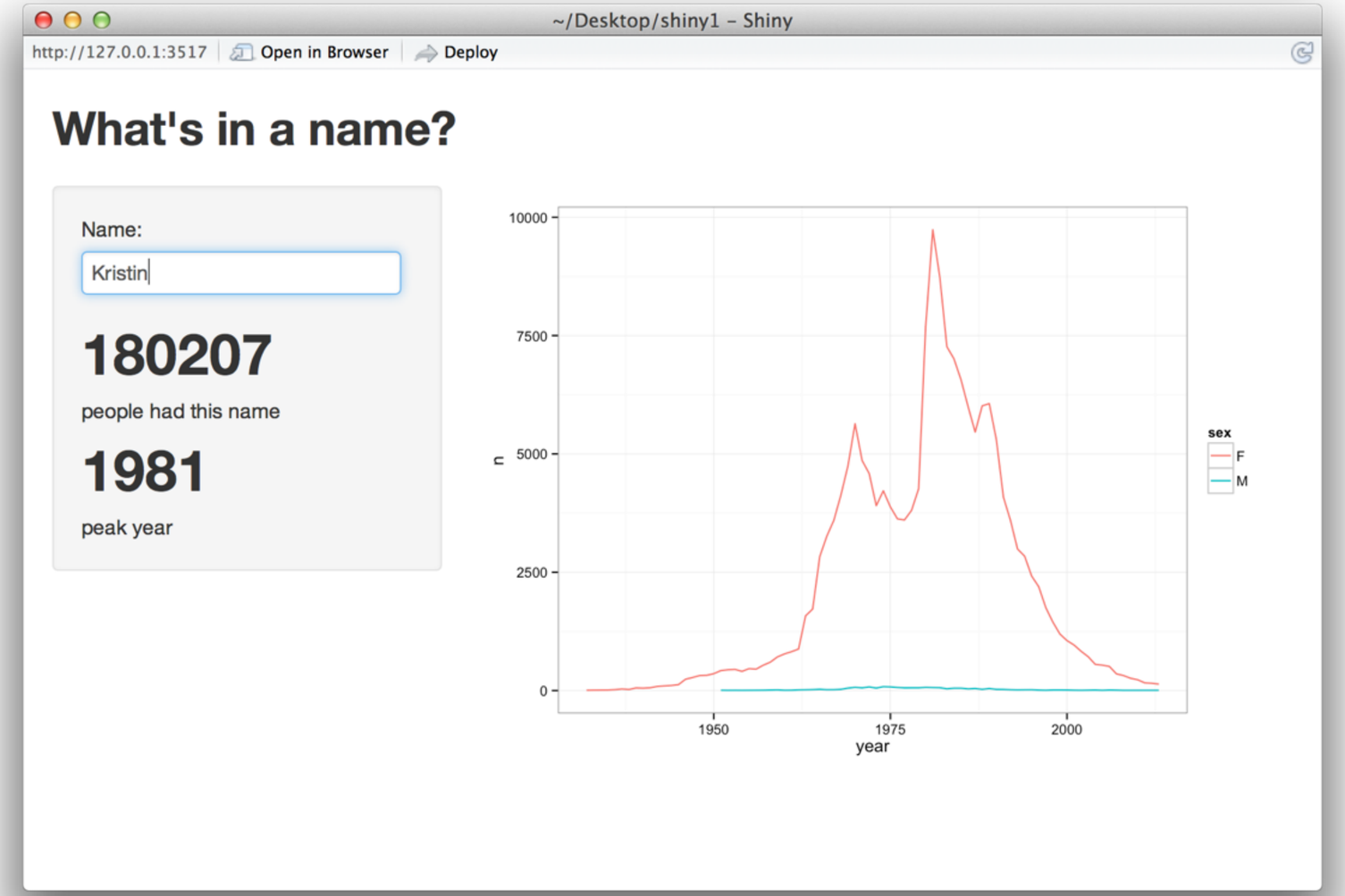
Please select a name to explore.
```{r echo=FALSE, message=FALSE}
library(babynames)
library(dplyr, warn.conflicts = FALSE)
library(ggplot2, warn.conflicts = FALSE)

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

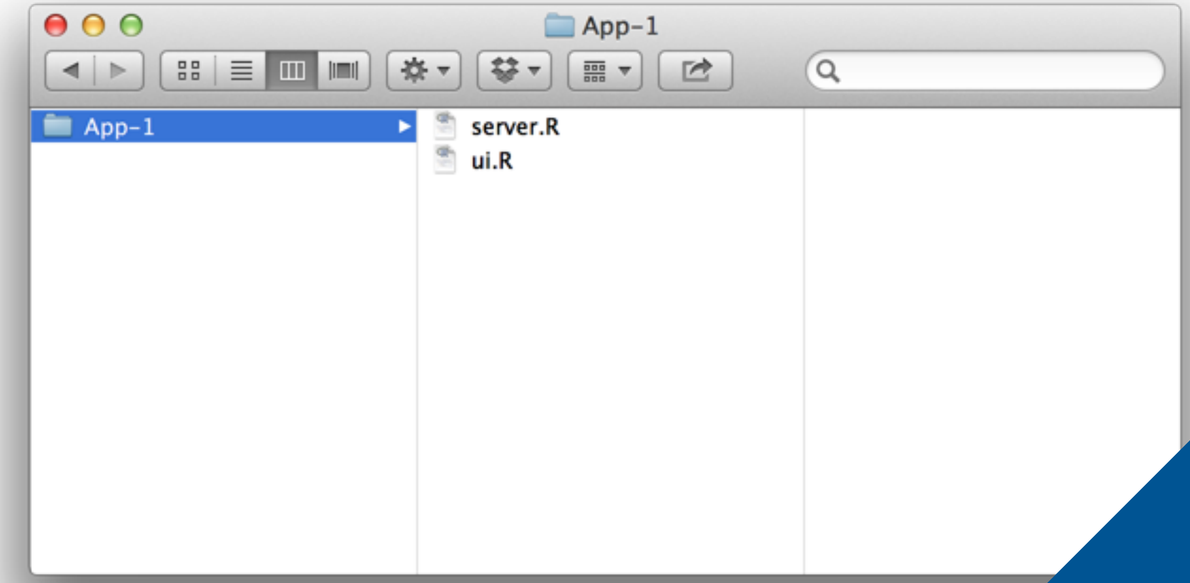
Web apps



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

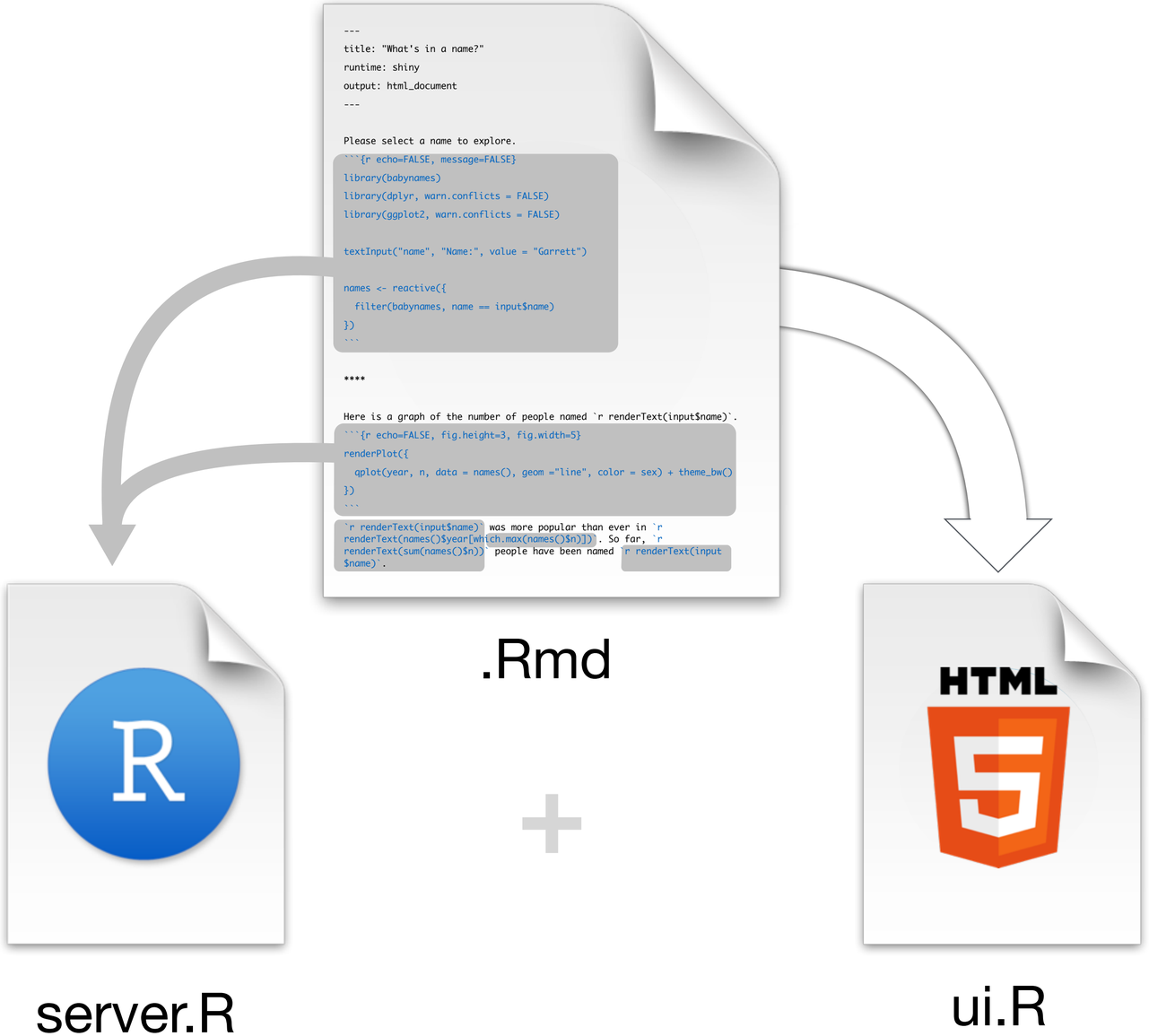
server.R

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
```

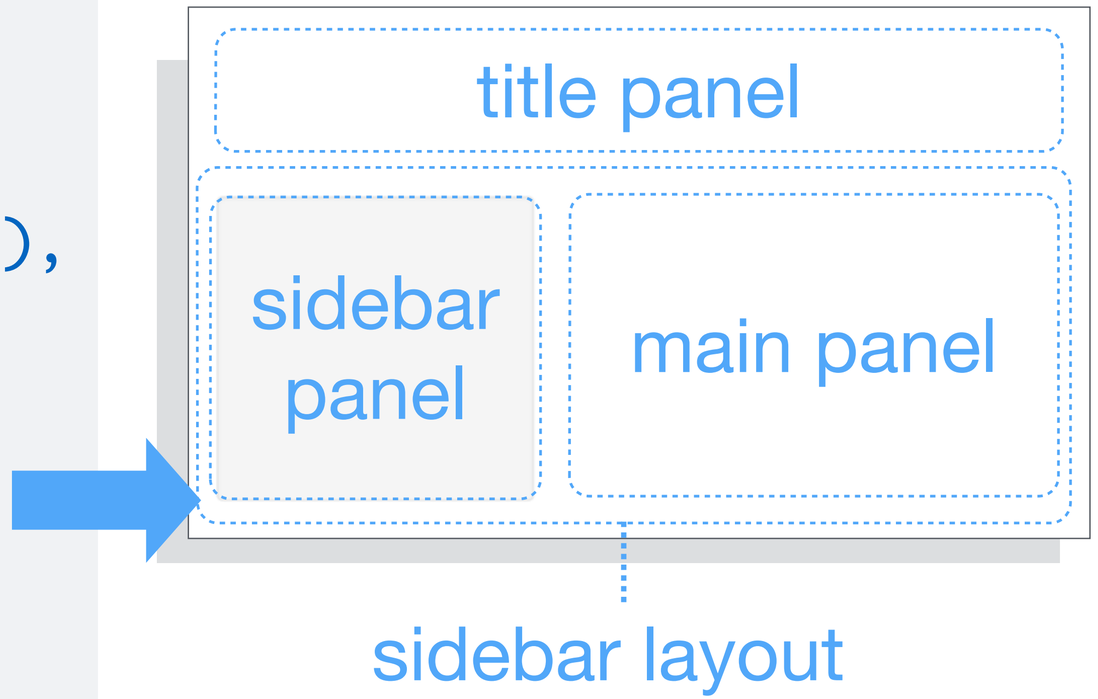
ui.R
(layouts)

The shortest viable ui.R file:

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

Place layout functions inside `fluidPage` to layout the app

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



text

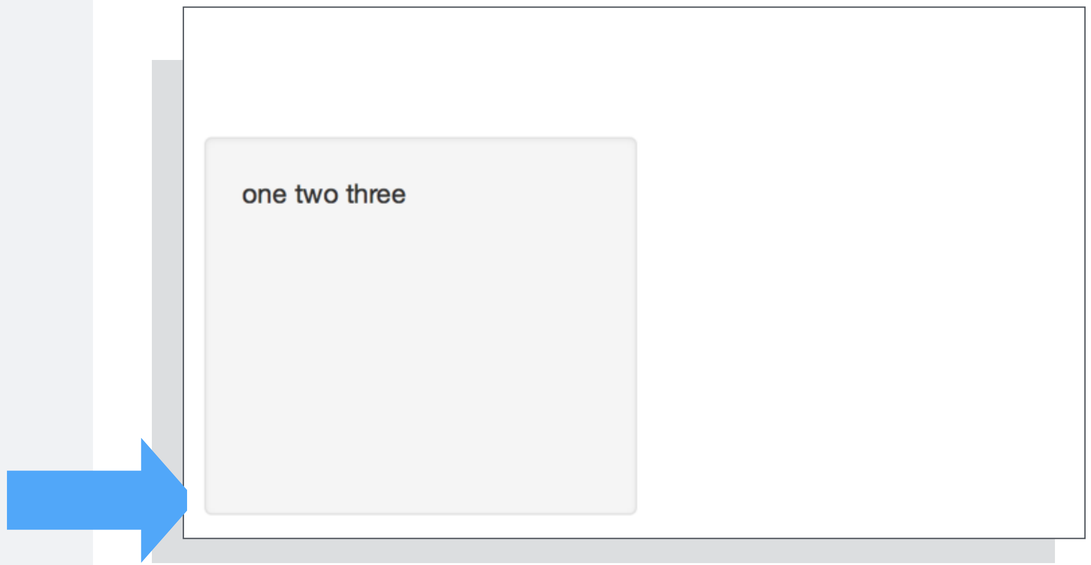
Place raw text into any panel you like.

```
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()
  )
))
```

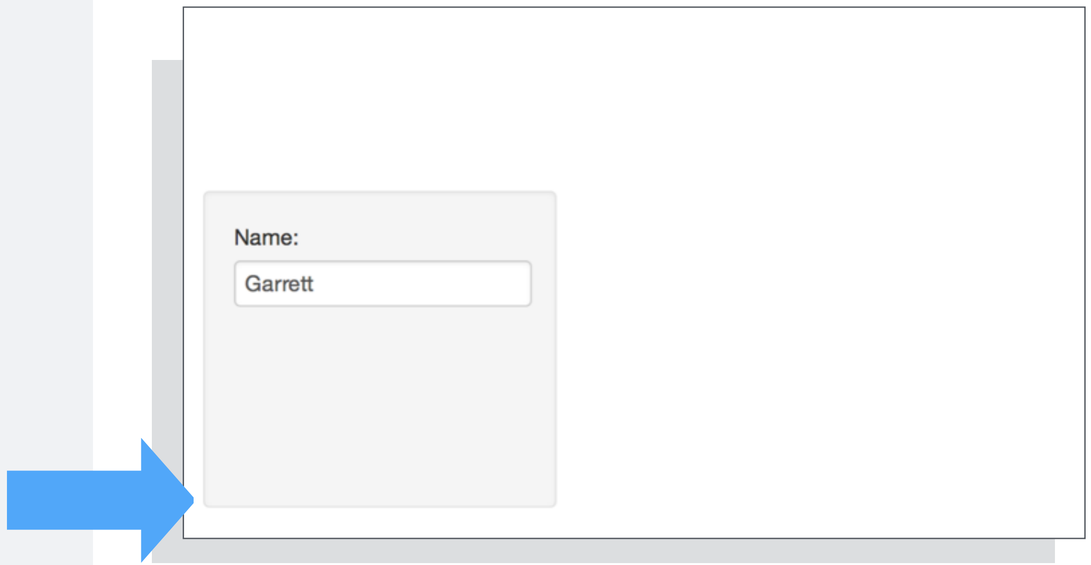


inputs

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()
  )
))
```

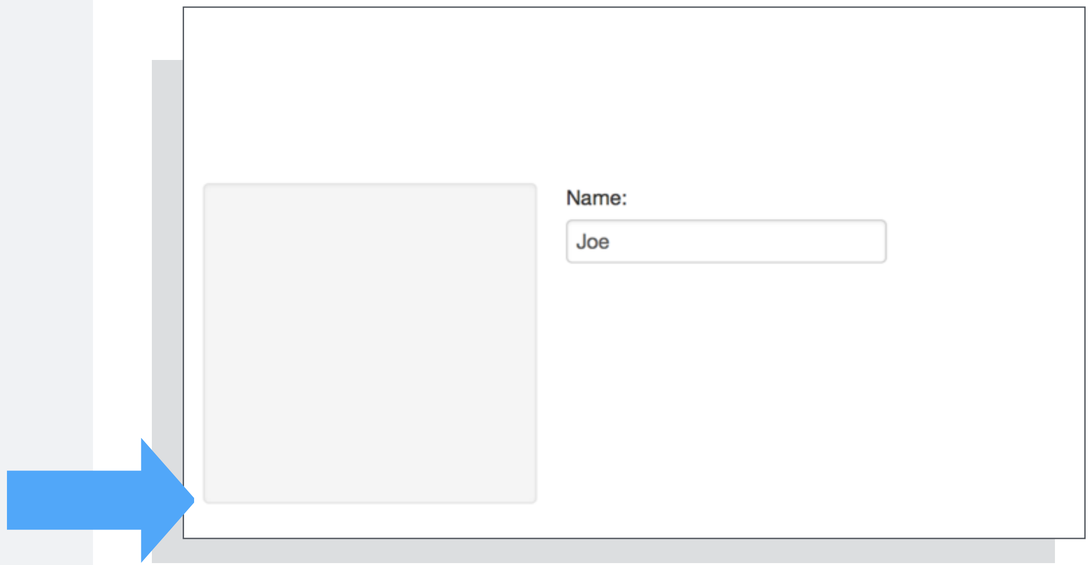


inputs

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)

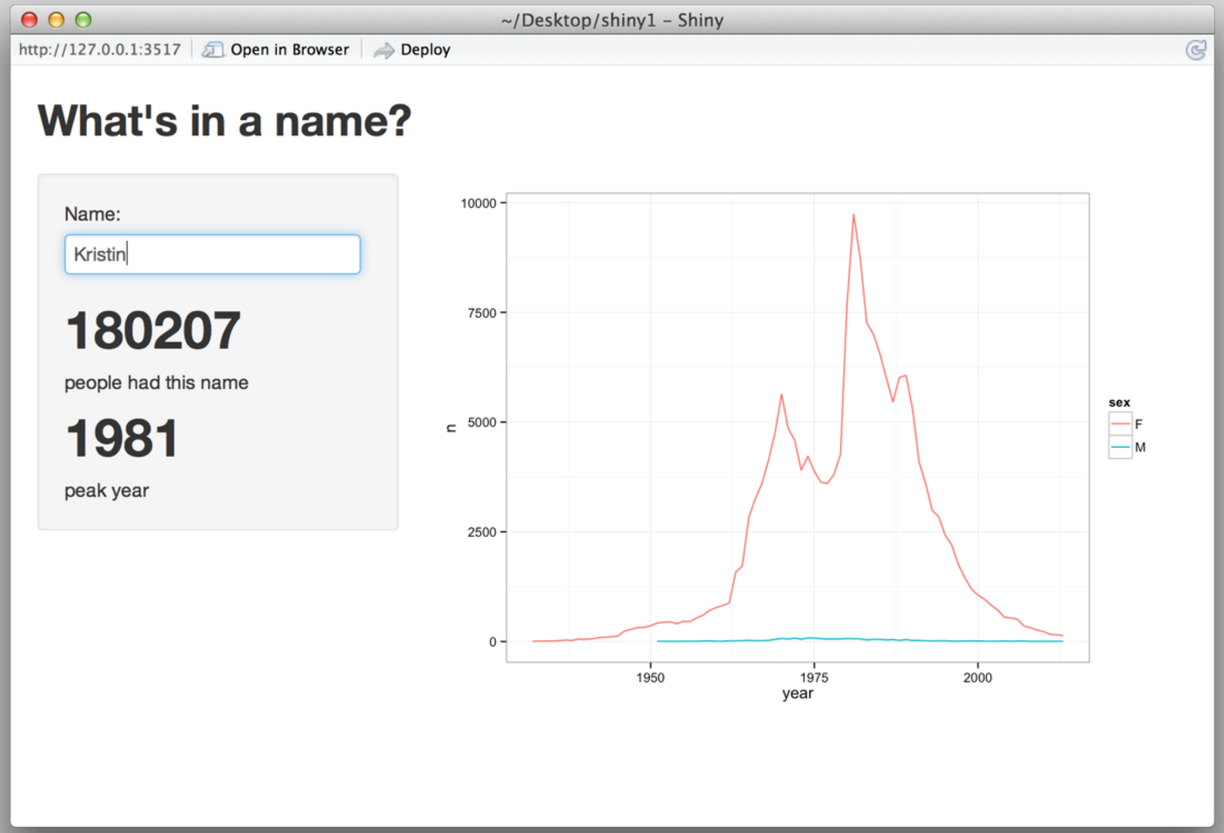
```
output$trend
```

```
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")
    )
  )
))
```

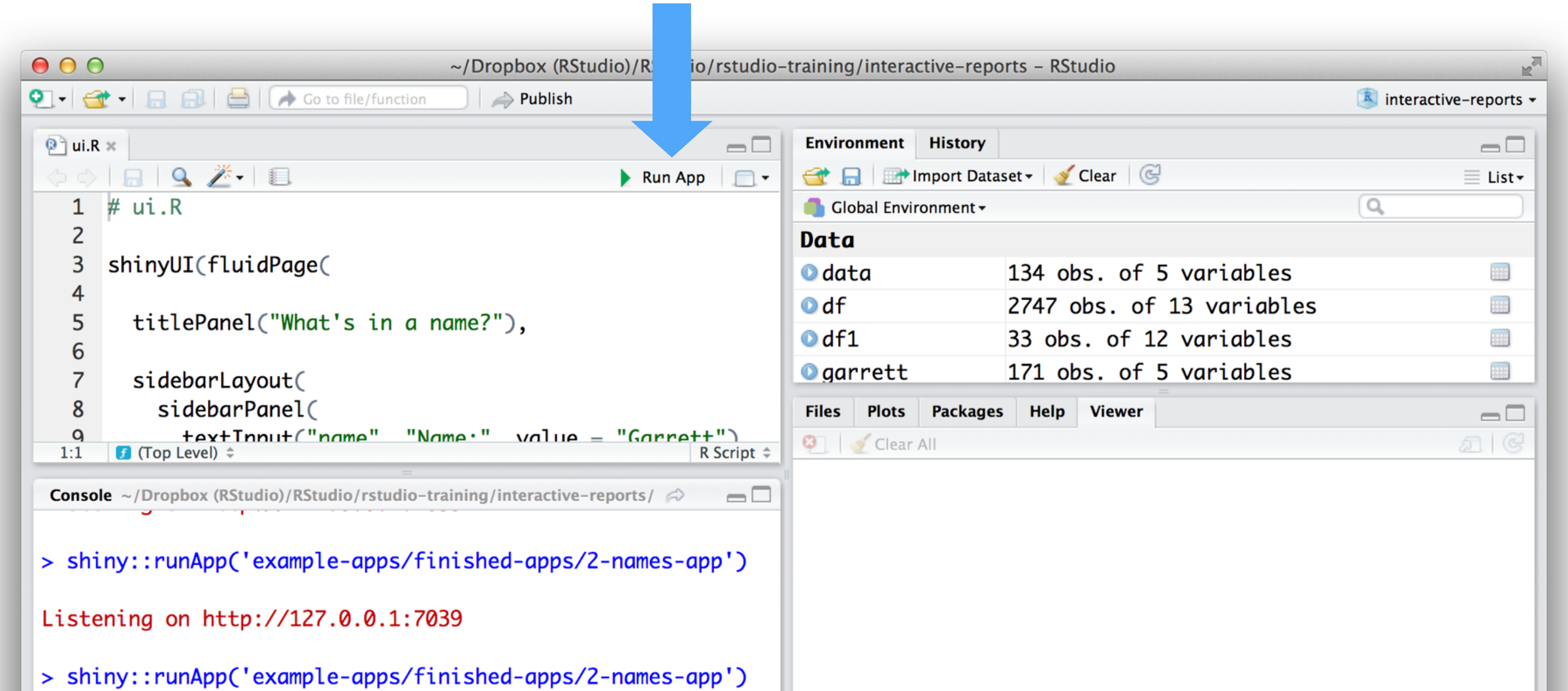
Customize your UI with HTML

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

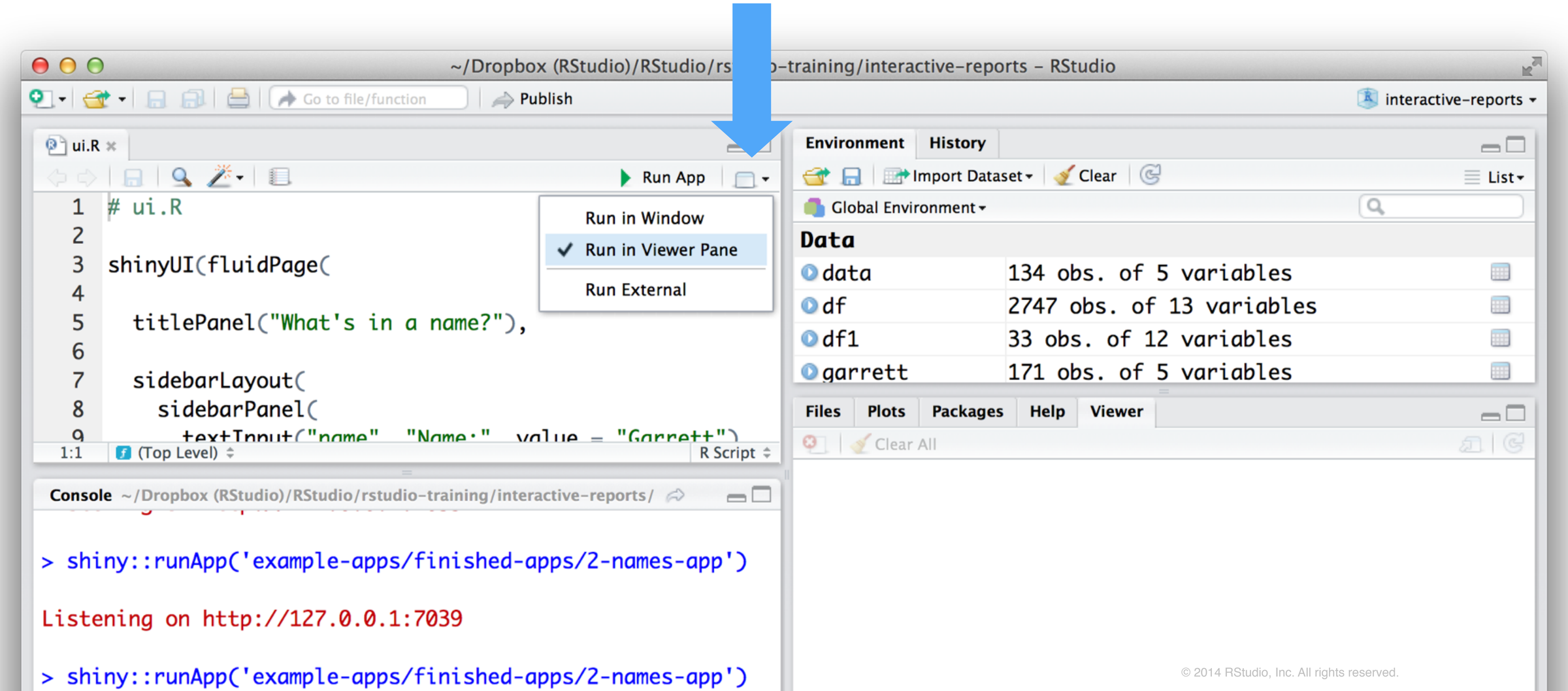
You can add any HTML element to your app with one of Shiny's `tags` function.

```
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"      "eventsource" "fieldset"
```

Run App

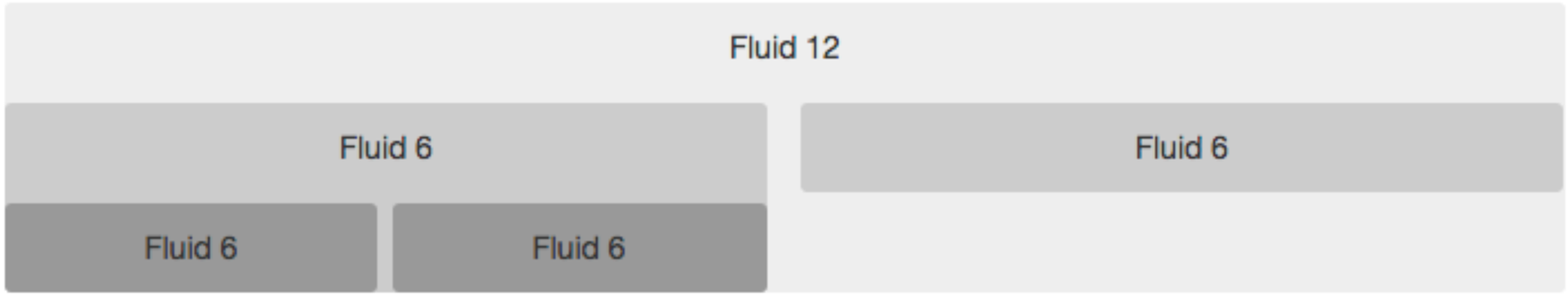

~/Dropbox (RStudio)/R io/rstudio-training/interactive-reports – RStudio
Go to file/function
Publish
interactive-reports
ui.R
Run App
1 # ui.R
2
3 shinyUI(fluidPage(
4
5 titlePanel("What's in a name?"),
6
7 sidebarLayout(
8 sidebarPanel(
1:1
(Top Level)
R Script
Environment
History
Import Dataset
Clear
List
Global Environment
Data
data 134 obs. of 5 variables
df 2747 obs. of 13 variables
df1 33 obs. of 12 variables
garrett 171 obs. of 5 variables
Files
Plots
Packages
Help
Viewer
Clear All
Console ~/Dropbox (RStudio)/RStudio/rstudio-training/interactive-reports/
> 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



Grid layouts

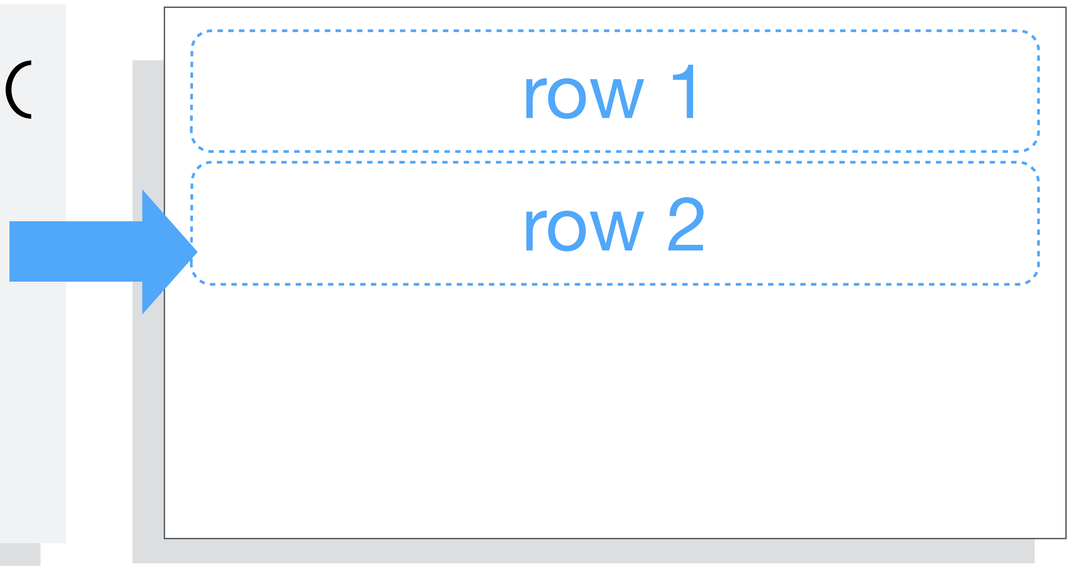
Alternatively, you can use fluidRow and column to layout your app with a grid system.



fluidRows

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

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

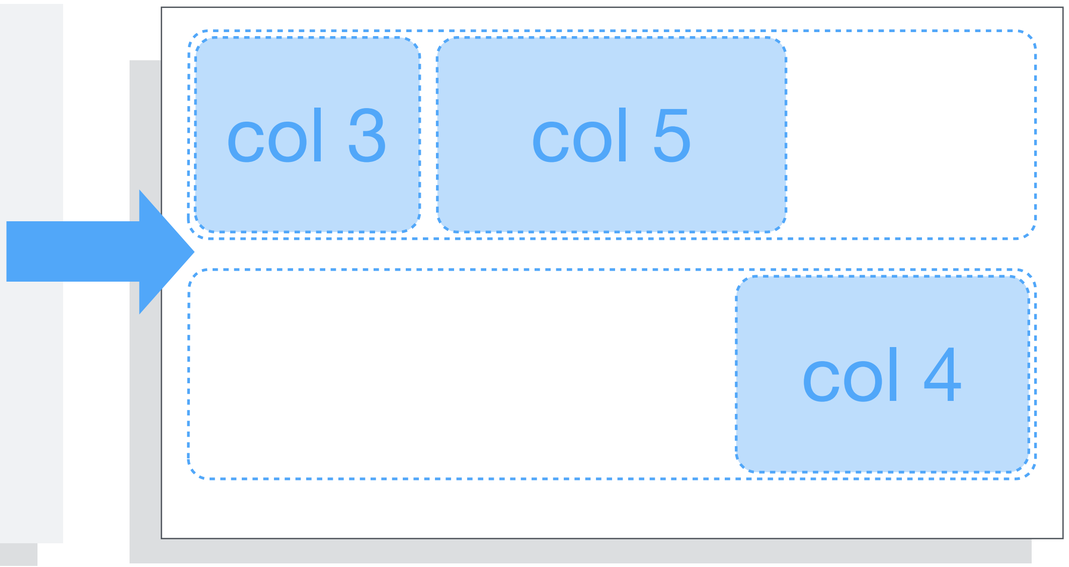


column

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

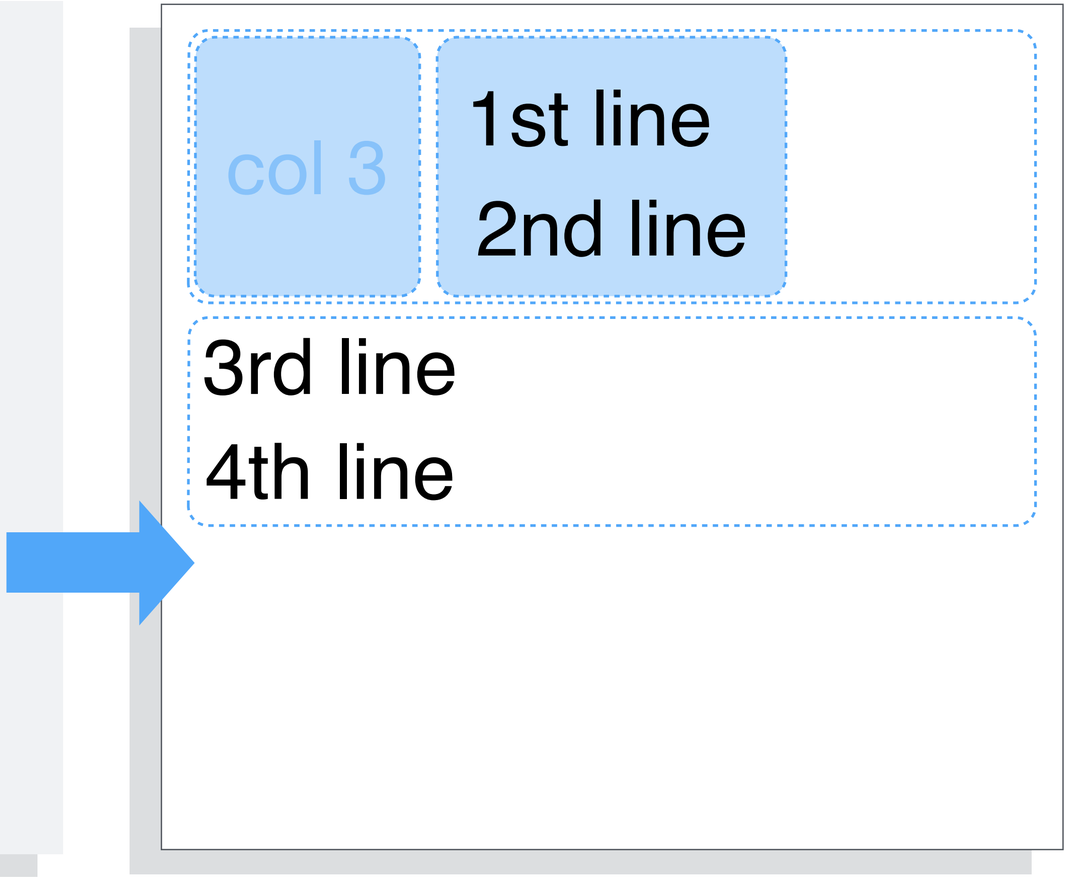
Specify the width and offset of each column out of 12

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



ggvis will fill columns and rows from top to bottom

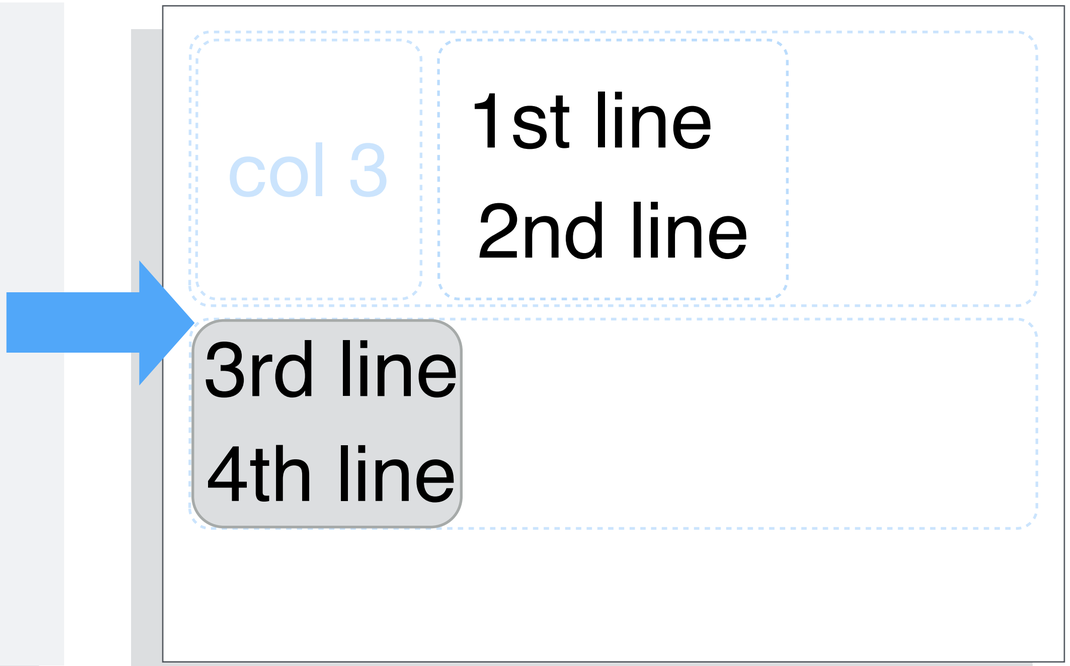
```
shinyUI(fluidPage(
  fluidRow(
   column(3),
   column(5,
     "1st line",
     "2nd line")),
  fluidRow(
   "3rd line",
   "4th line")
))
```



wellPanel

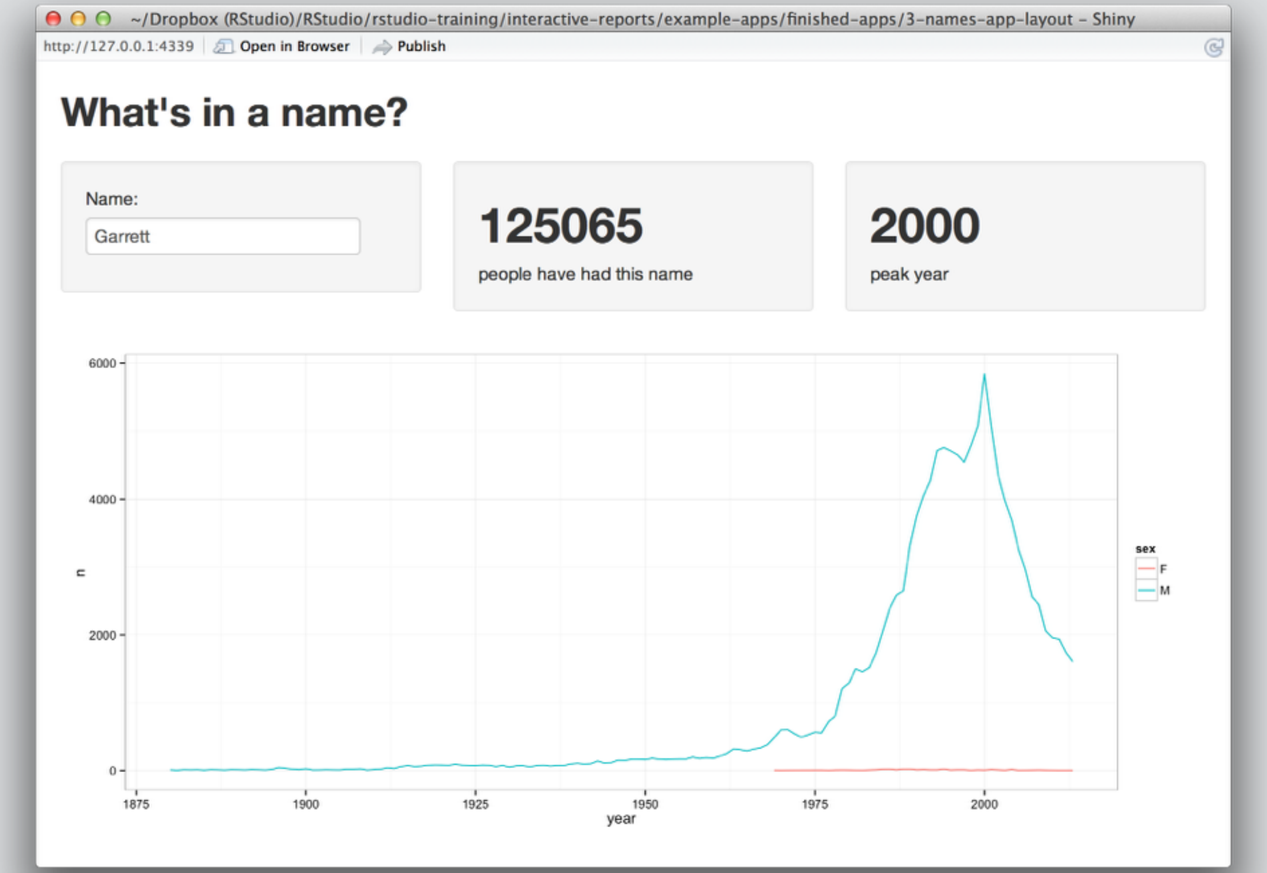
Recreate the grey background of the `sidebarPanel` by wrapping element(s) in `wellPanel`

```
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

05:00

```
shinyUI(fluidPage(

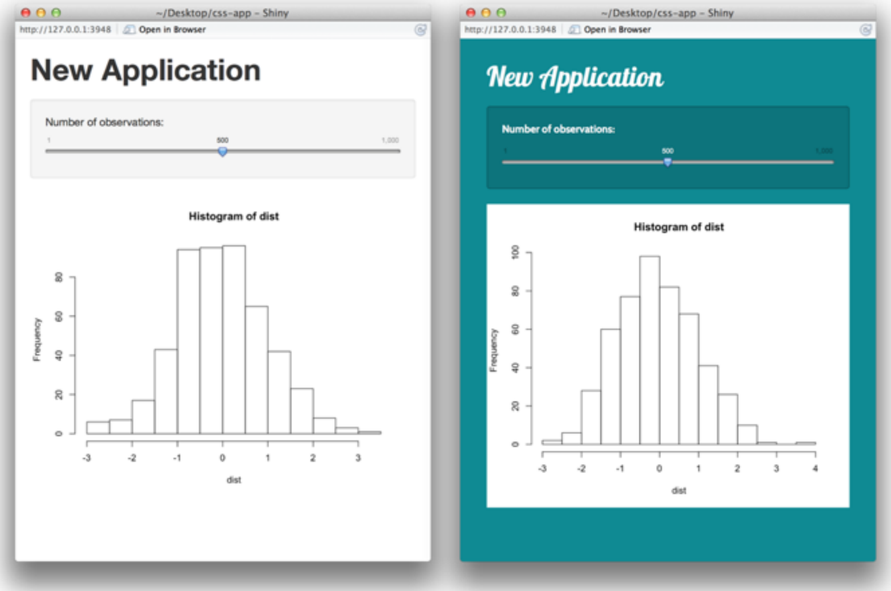
  titlePanel("What's in a name?"),

  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/

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

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

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

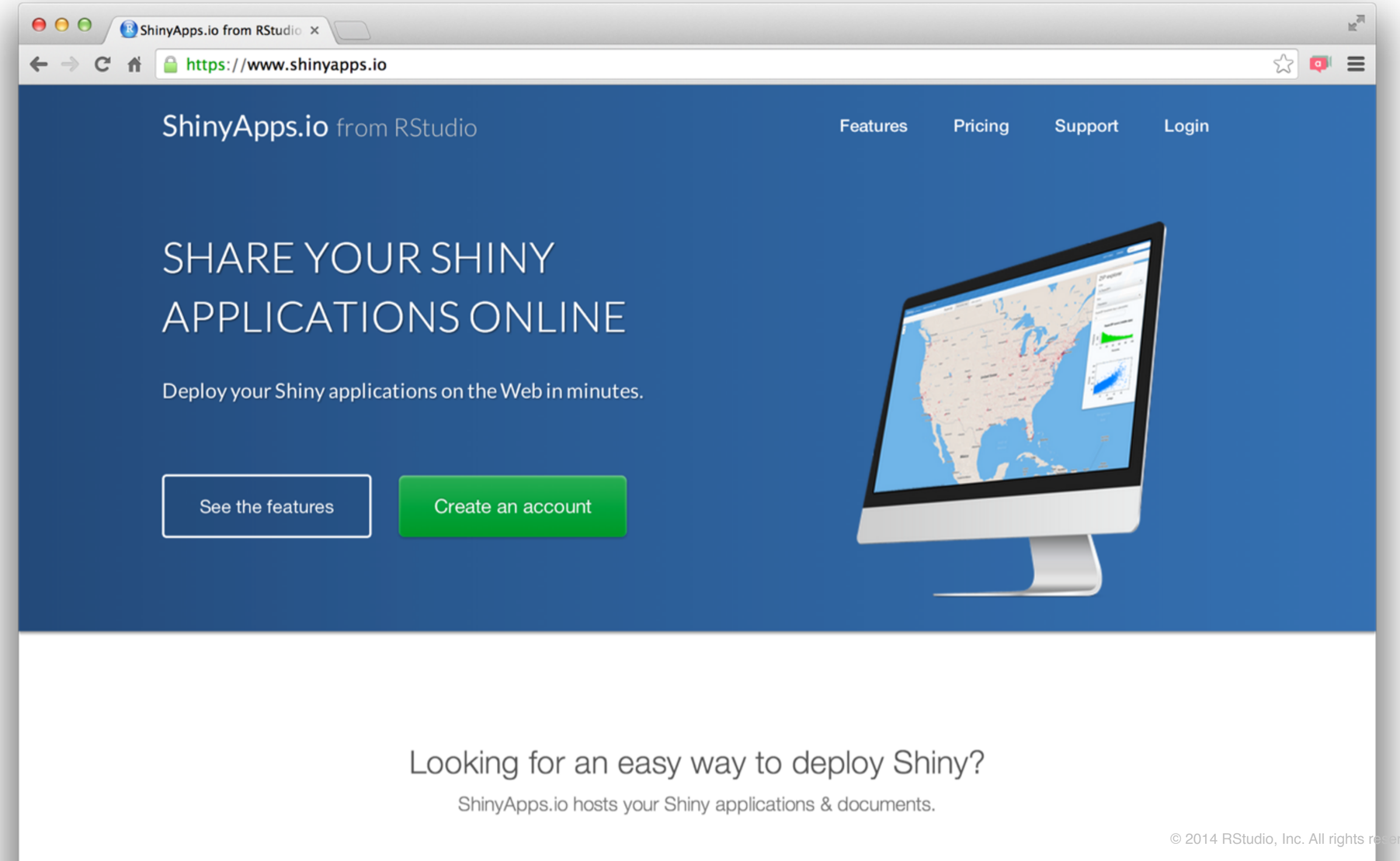
ShinyApps.io

Shiny Server

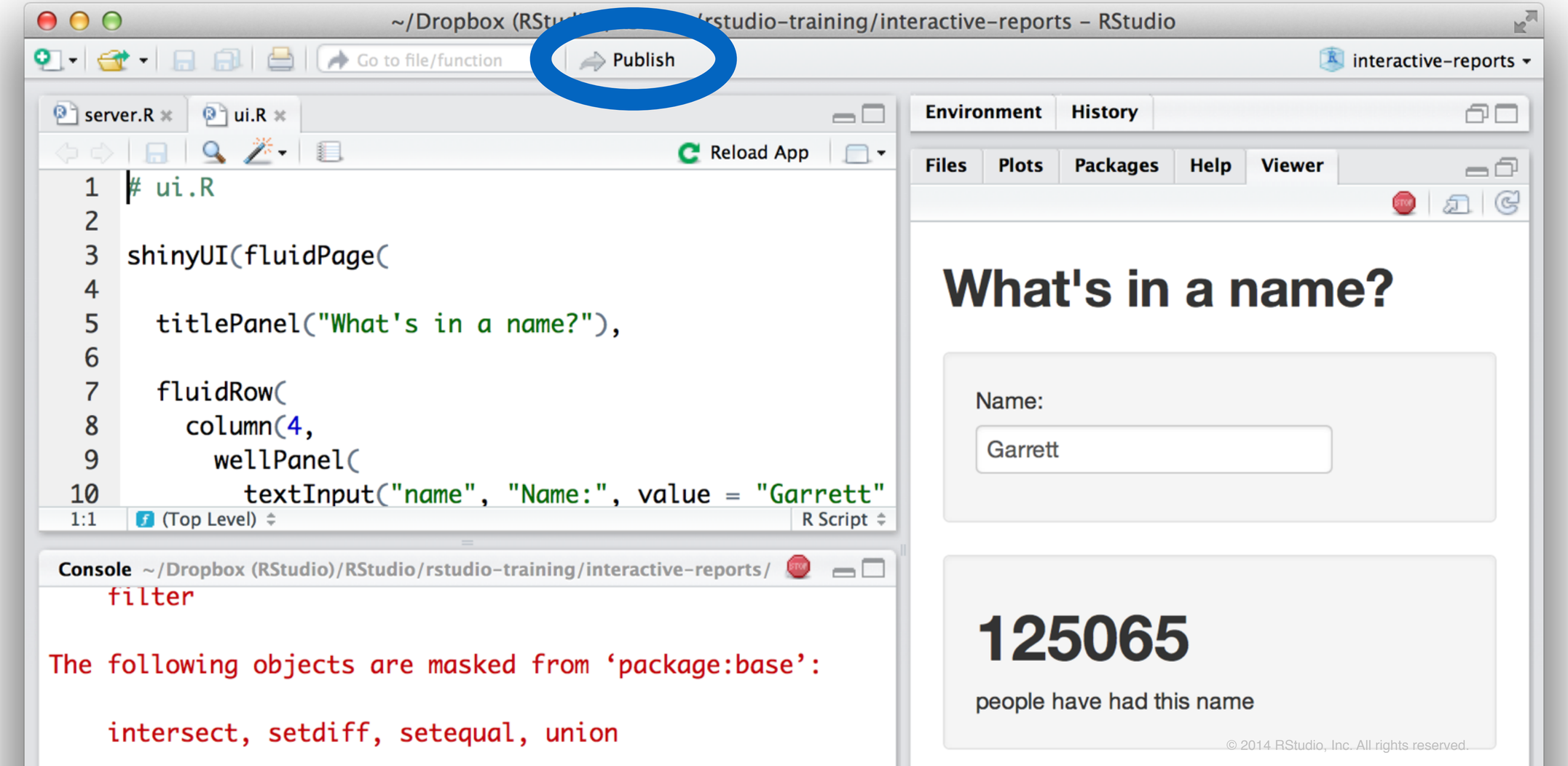
Shiny Server Pro

ShinyApps.io

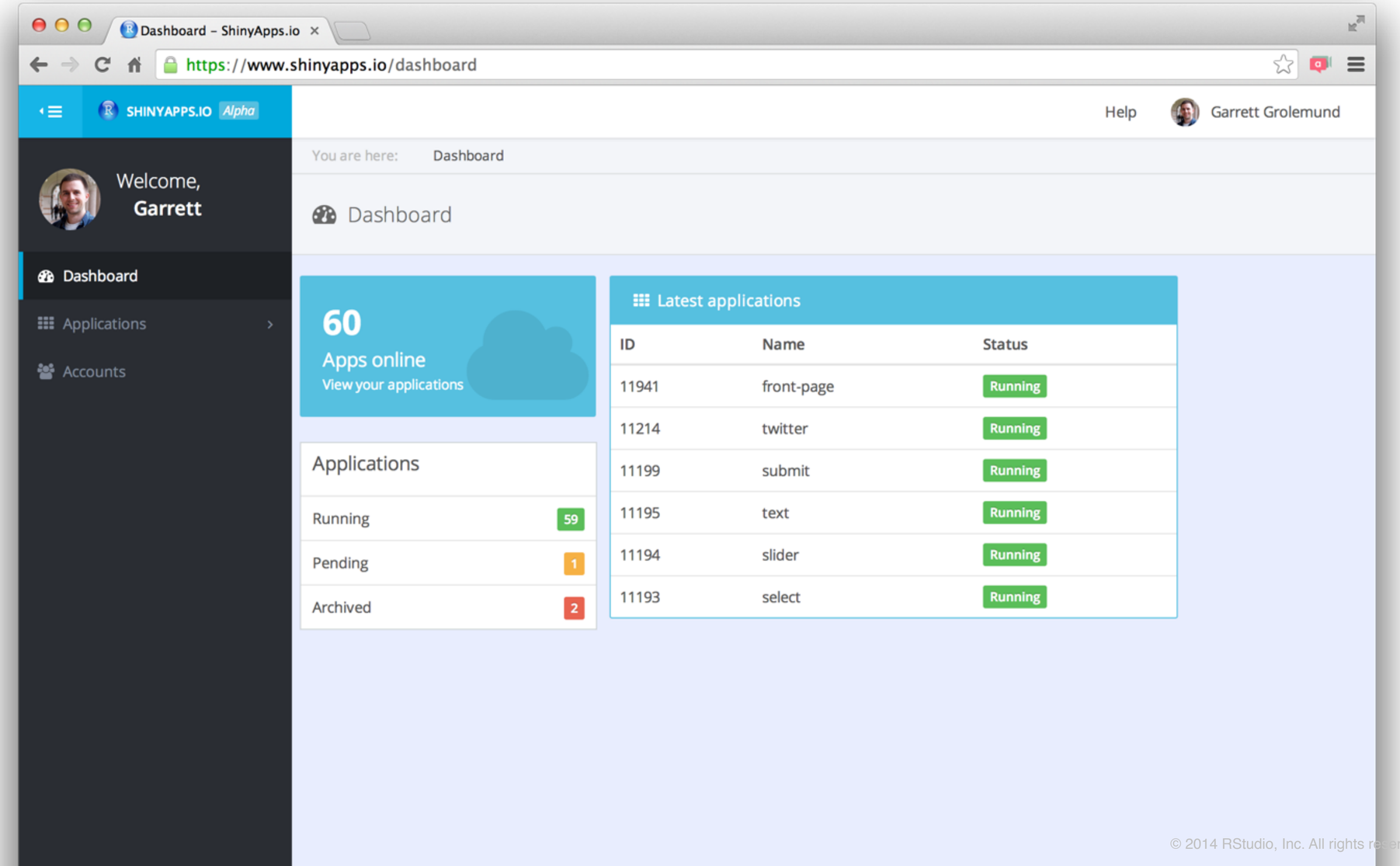
Hassle-free cloud hosting for Shiny



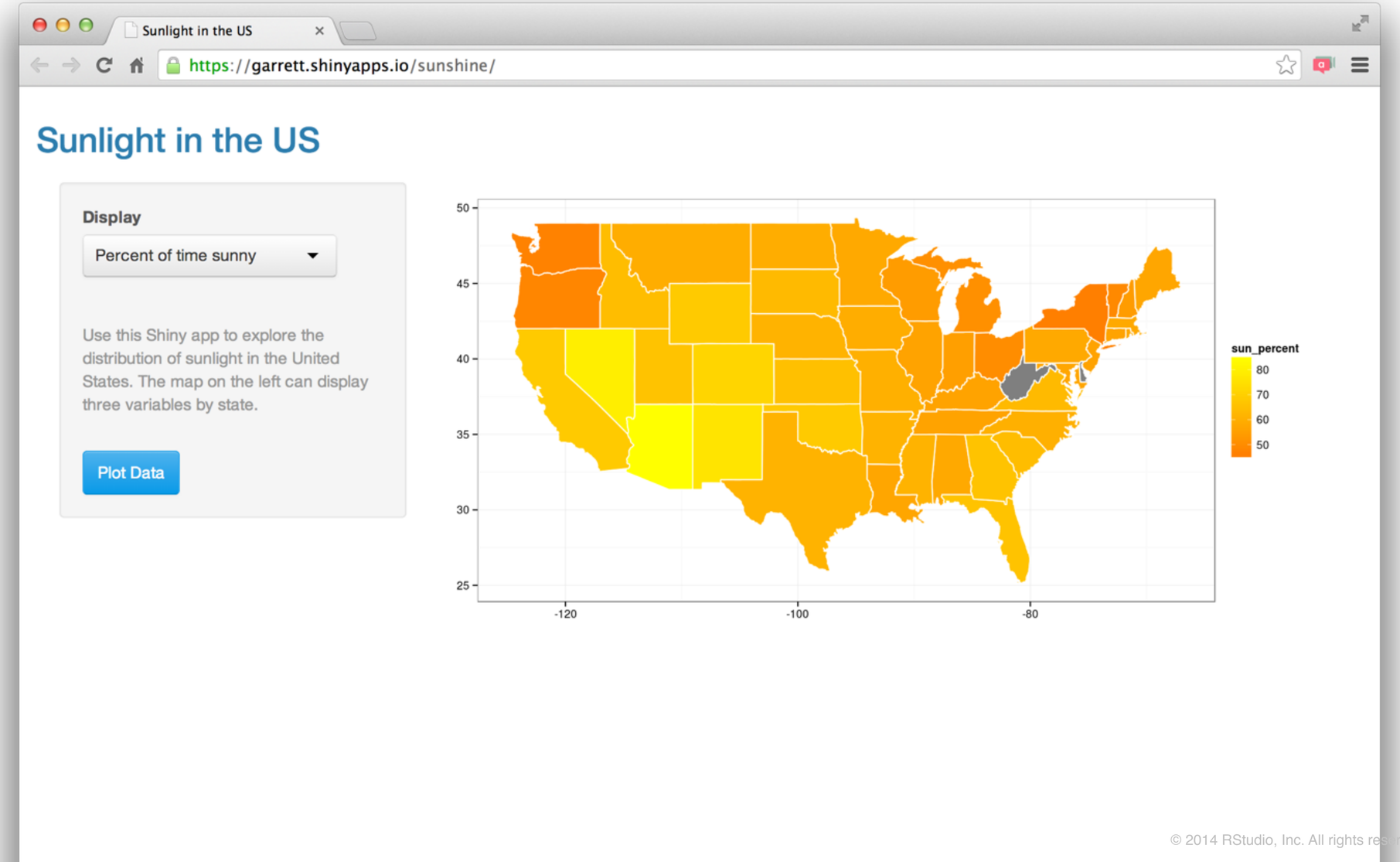
Deploy to web with one click



Basic admin tools



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

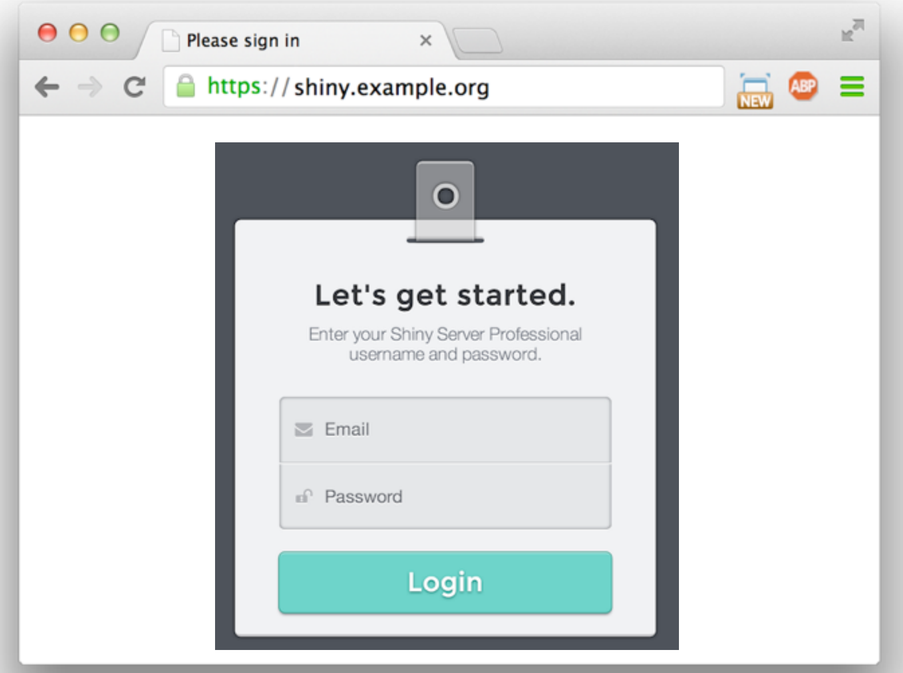
- Each app is hosted at its own URL
- Can deploy to internet, or within a controlled environment
- Starts app when user visits, closes app when user leaves
- Runs on a linux server

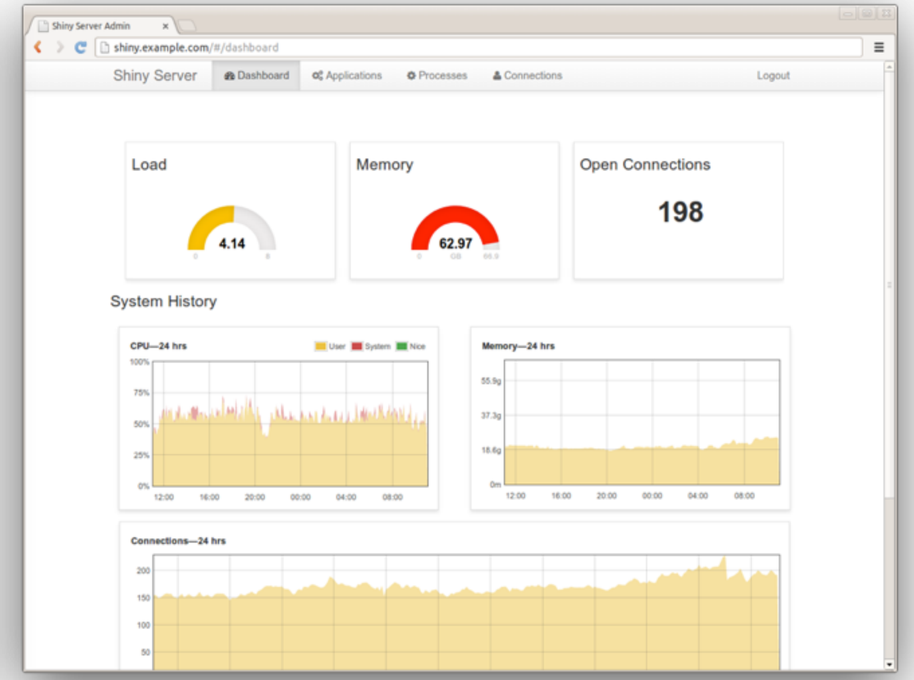
http://shiny.rstudio.com/articles/shiny-server.html

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)

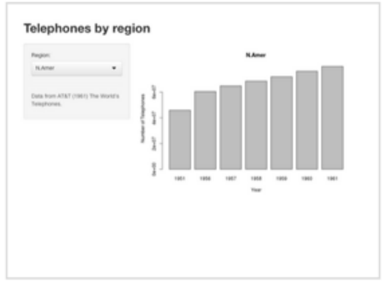
shiny.rstudio.com

Gallery

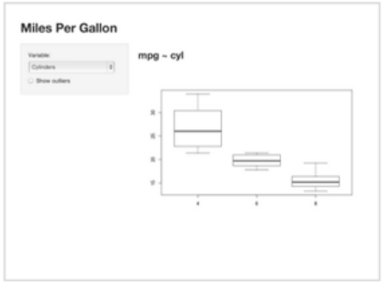
Start simple

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

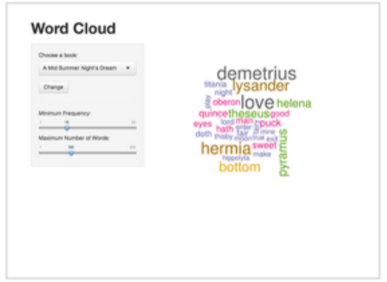
Kmeans example



Telephones by region



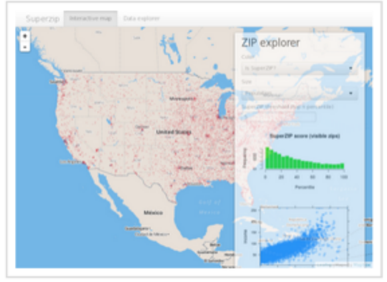
Miles per gallon



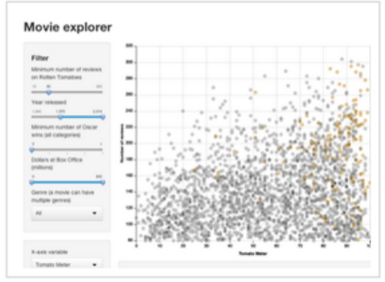
Word cloud

Interactive visualizations

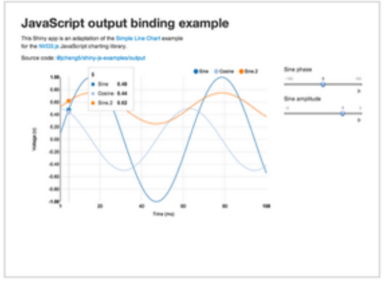
Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.



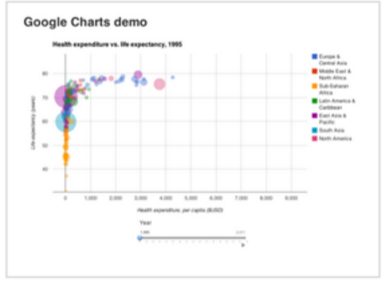
SuperZip example



Movie explorer



NVD3 line chart output

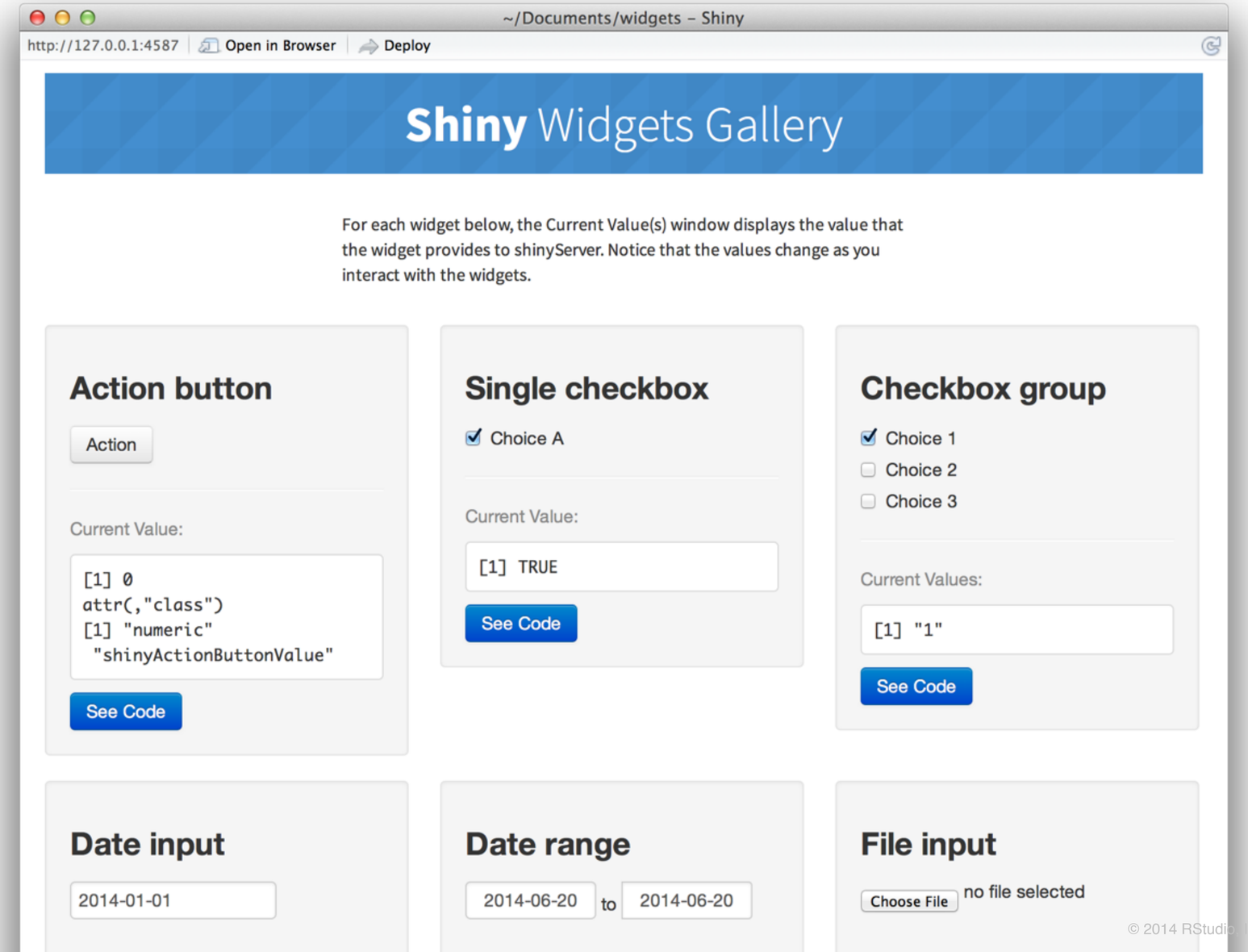


Google Charts

Widgets

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

shiny.rstudio.com/gallery/widgets



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

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

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
Tabsets
Customize your UI with HTML
Build your entire UI with HTML
Build a dynamic UI that reacts to user input
Shiny HTML Tags Glossary

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

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

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

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

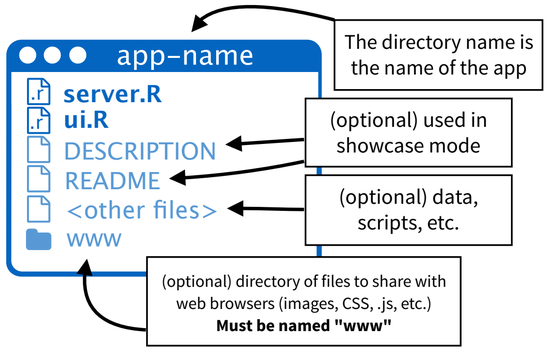
Shiny Cheat Sheet

learn more at shiny.rstudio.com

Shiny 0.10.0 Updated: 6/14

RStudio

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



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

Ⓐ Provide server.R with the minimum necessary code, **shinyServer(function(input, output) {})**

Ⓑ Define the R components for your app between the braces that follow **function(input, output)**

Ⓒ Save each R component in your UI as **output$<component name>**

Ⓓ Create each output component with a render* function.

Ⓔ 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.

Ⓕ Refer to widget values with **input$<widget name>**

server.R

```
# load libraries, scripts, data

shinyServer(function(input, output) {

  # make user specific variables

  output$text <- renderText({
    input$title
  })

  output$plot <- renderPlot({
    x <- mtcars[ , input$x]
    y <- mtcars[ , input$y]
    plot(x, y, pch = 16)
  })

})
```

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

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.

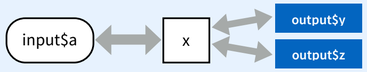
Run often - code placed within a render*, reactive, o… observe function will be run many times. Place h… code that the server needs to rebuild a UI com… widget changes.

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.

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


```
output$z <- renderTex…
  paste(
    isolate(input$…
    input$b
  )
)
```

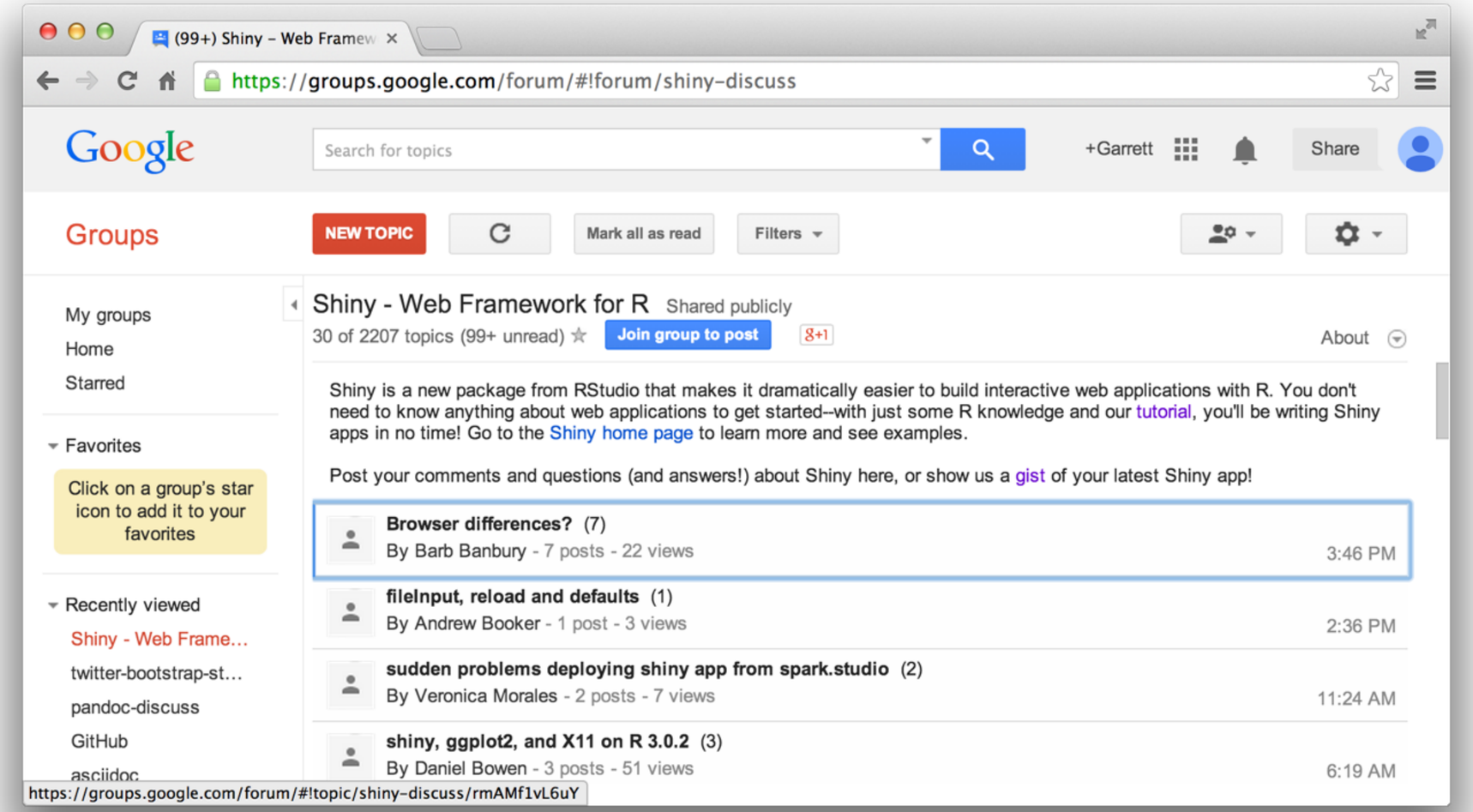
observe - use… that runs w… does no…

```
…un
```

in articles section

Shiny Discussion group

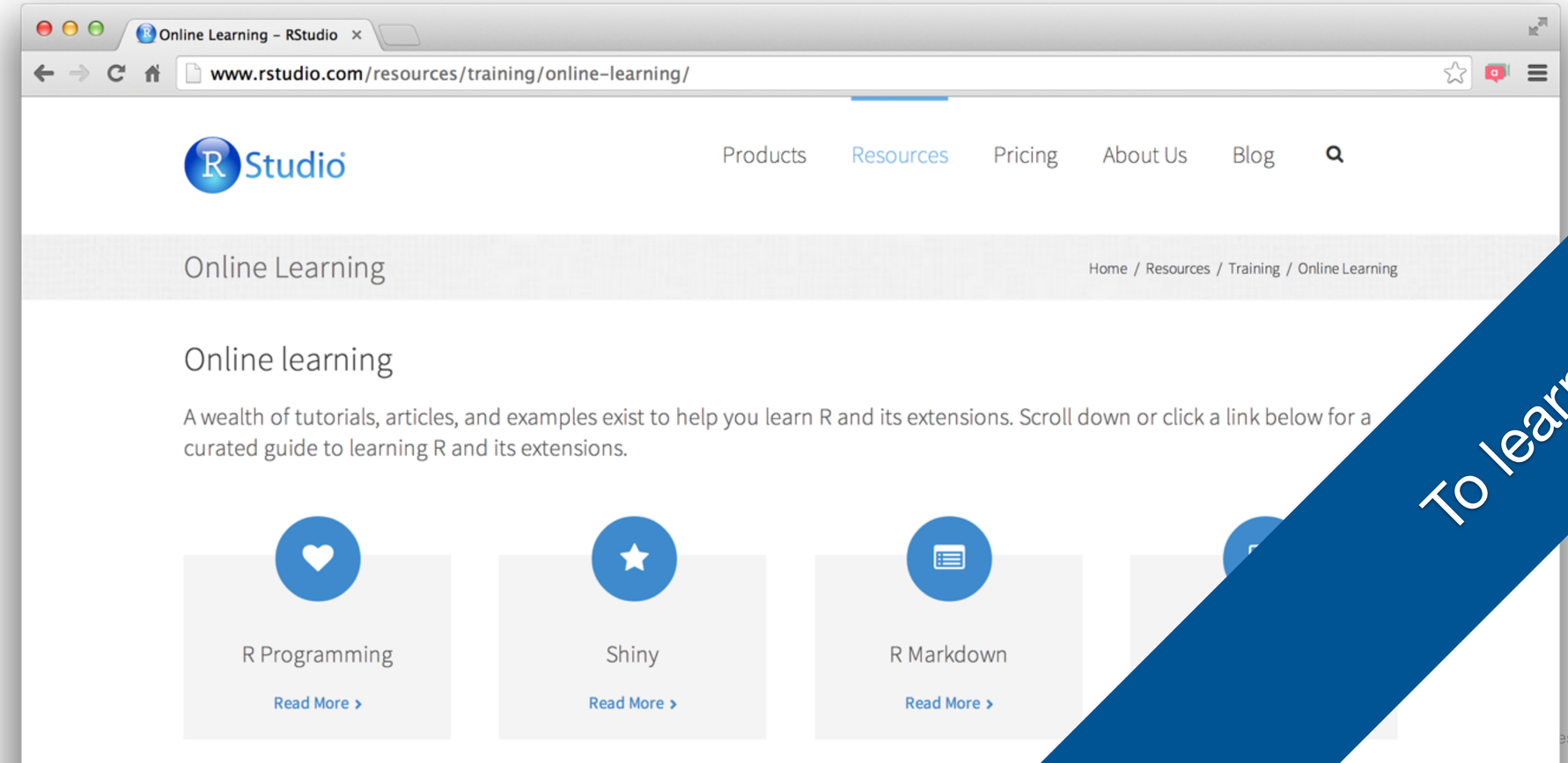
https://groups.google.com/forum/#!forum/shiny-discuss



RStudio training

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

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



To learn R