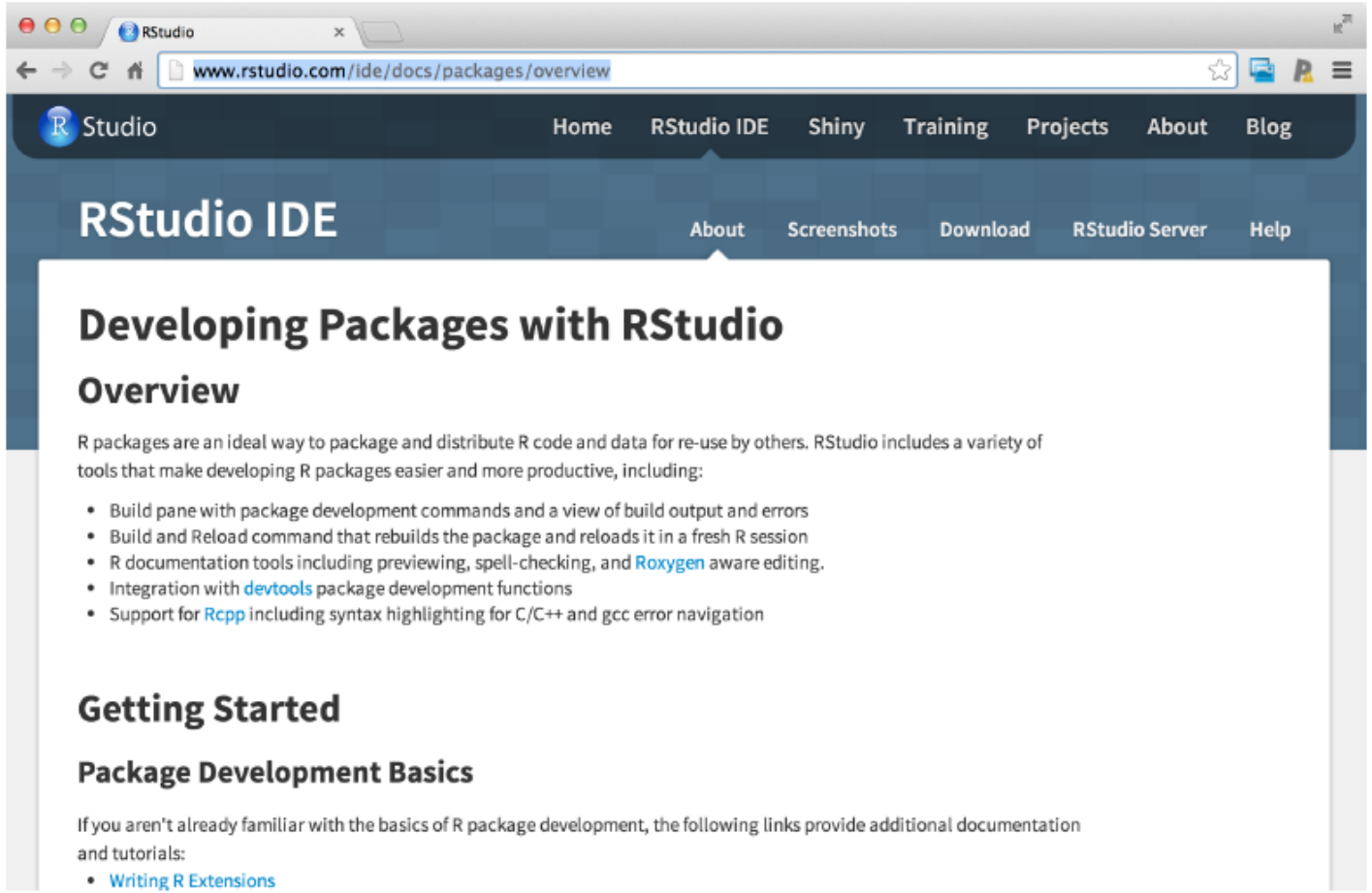
Building Data Products Overview

Building Data Products Content

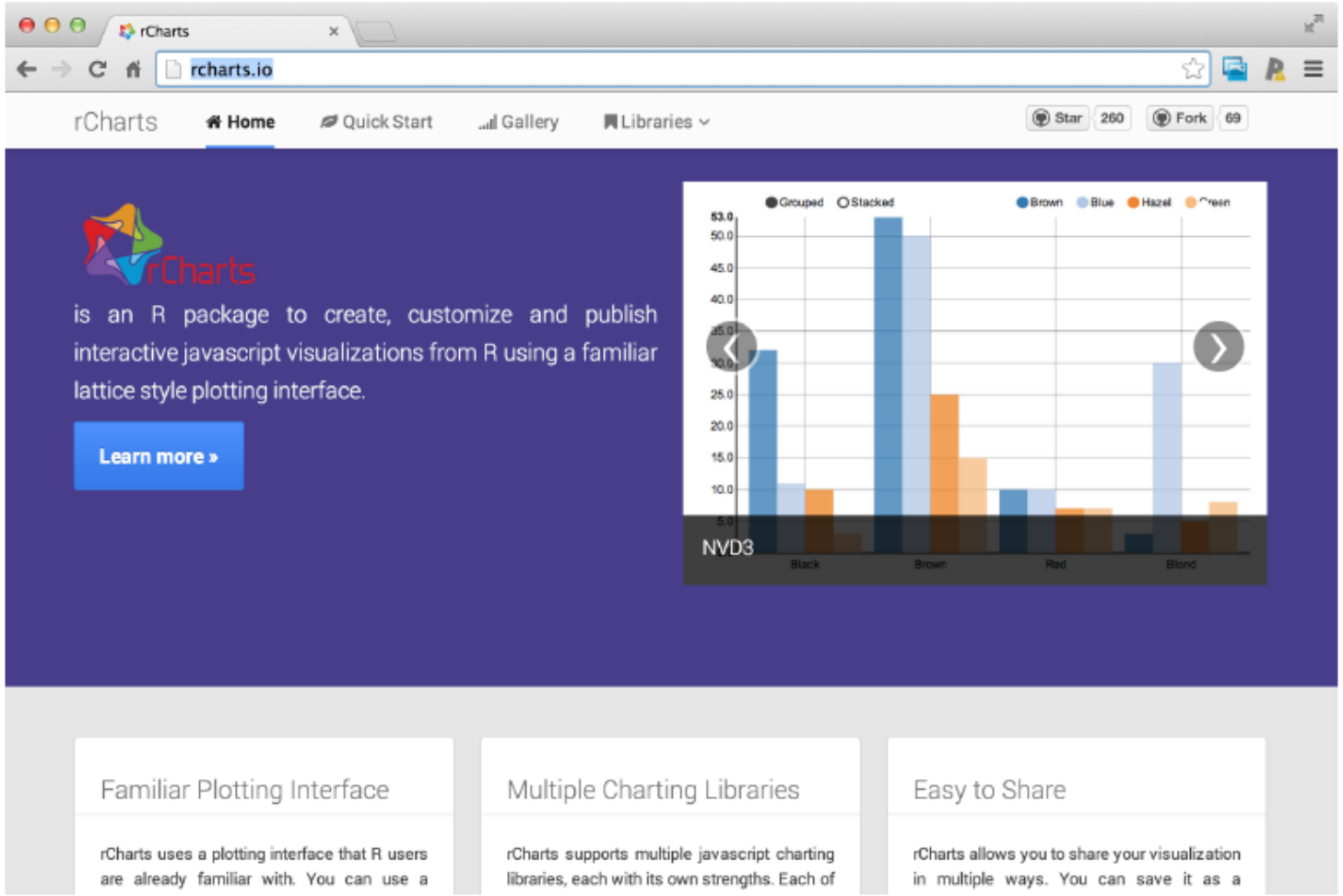
- R packages
  - devtools
  - roxygen
  - testthat
- rCharts
- Slidify
- Shiny

R packages - for the engineers



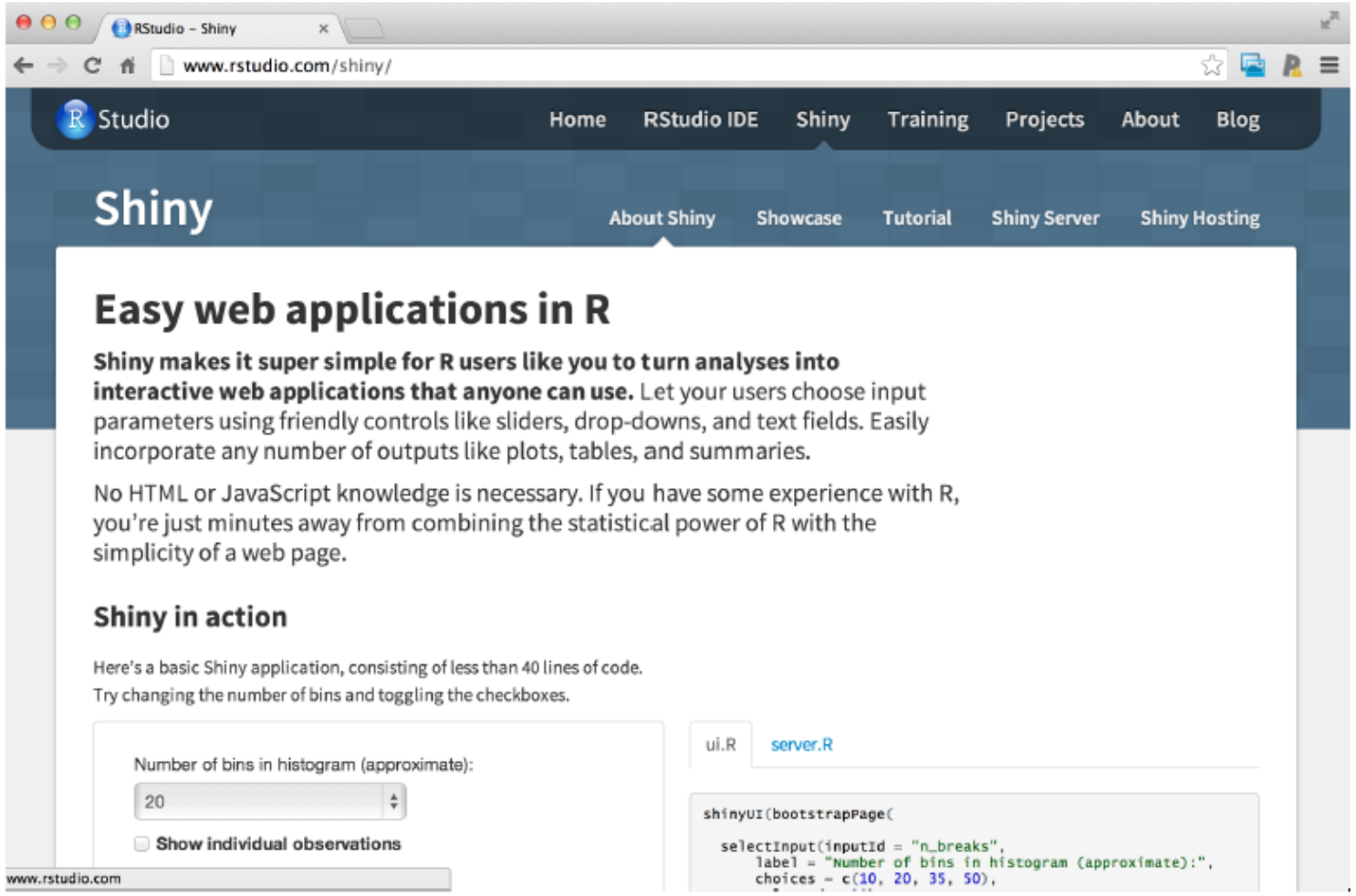
http://cran.r-project.org/web/packages/ http://www.rstudio.com/ide/docs/packages/overview

rCharts - for marketing



http://rcharts.io/ http://ramnathv.github.io/rChartsNYT/

Shiny - for your users



http://www.rstudio.com/shiny/ http://www.rstudio.com/shiny/showcase/

What is Shiny?

- Shiny is a platform for creating interactive R programs embedded into a web page.
- Suppose that you create a prediction algorithm, with shiny you can very easily create web input form that calls R and thus your prediction algorithm and displays the results.
- Using Shiny, the time to create simple, yet powerful, web-based interactive data products in R is minimized.
- Shiny is made by the fine folks at R Studio. · · · - However, it lacks the flexibility of full featured (and more complex) solutions.

Some mild prerequisites

Shiny doesn't really require it, but as with all web programming, a little knowledge of html, css and js is very helpful

- html gives a web page structure and sectioning as well as markup instructions
- css gives the style
- js for interactivity

There are too many tutorials online to count for getting basic proficiency in these topics to count.
Shiny uses bootstrap (no relation to the statistics bootstrap) style, which (to me) seems to look nice and renders well on mobile platforms

What else is out there?

- Creating any solution requiring fairly deep knowledge of web client/server programming

- OpenCPU by Jerome Ooms, is a really neat project providing an API for calling R from web documents

- And he even hosts an OpenCPU server, but you can create your own

Context

You created a novel prediction algorithm to predict risk for developing diabetes.

- You want to create a web site so that users can input the relevant predictors and obtain their prediction.

Your prediction algorithm

· You're hoping patients and caregivers will be able to enter their data and, if needed, take preventative measures.

- diabetesRisk <- function(glucose) glucose/200

Getting started

- Make sure you have the latest release of R installed
- If on windows, make sure that you have Rtools
- installed install.packages("shiny")
- library(shiny)
- Great tutorial at http://rstudio.github.io/shiny/tutorial/
- Basically, this lecture is walking through that tutorial offering some of my insights
- Note, some of the proposed interactive plotting uses of Shiny could be handled by the very simple manipulate function rstudio manipulate

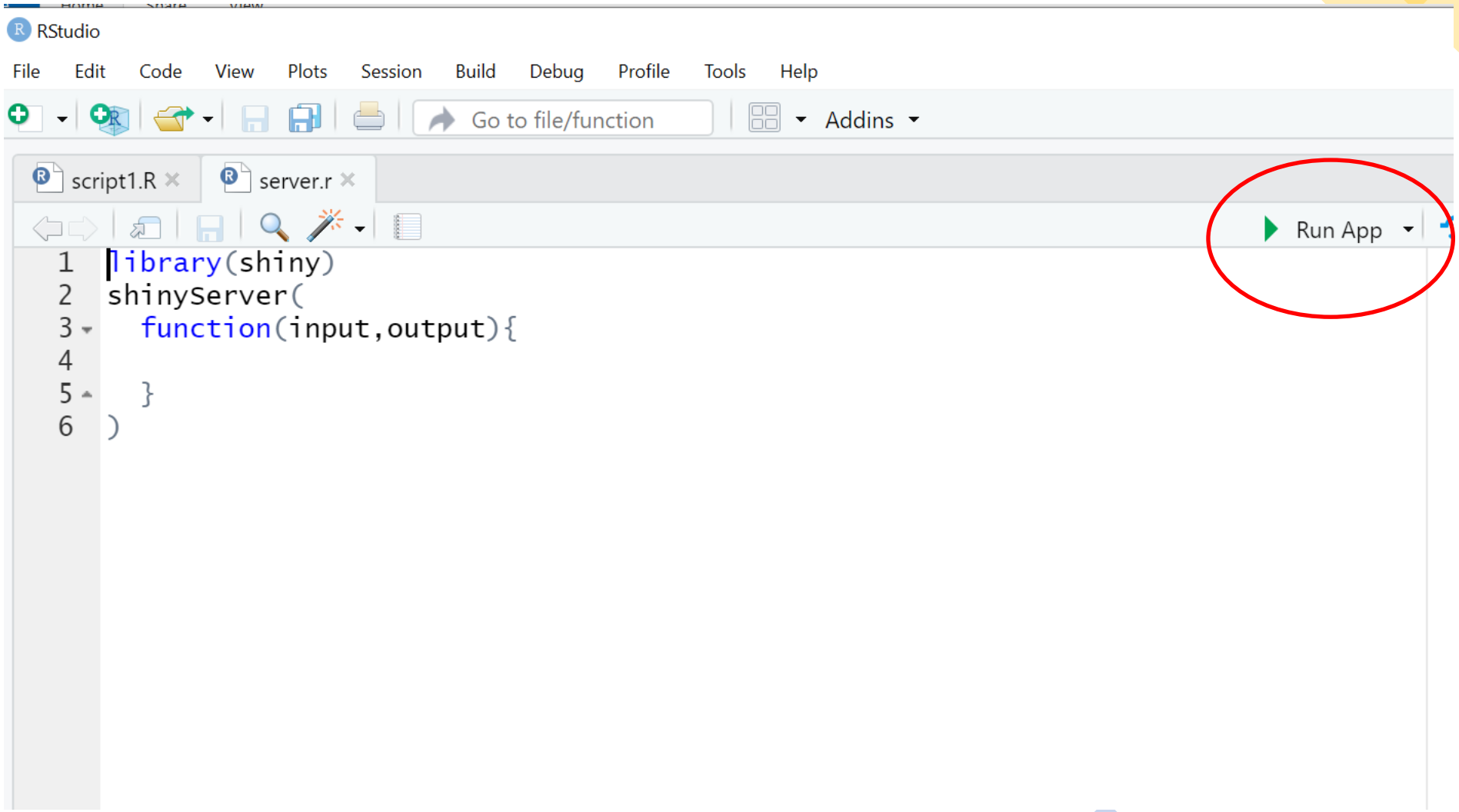
A Shiny project

- A shiny project is a directory containing at least two parts:
  - One named ui.R (for user interface) controls how it looks.
  - One named server.R that controls what it does.

To run it

- In R, change to the directories with these files and type runApp() or put the path to the directory as an argument. It should open an browser window with the app running.

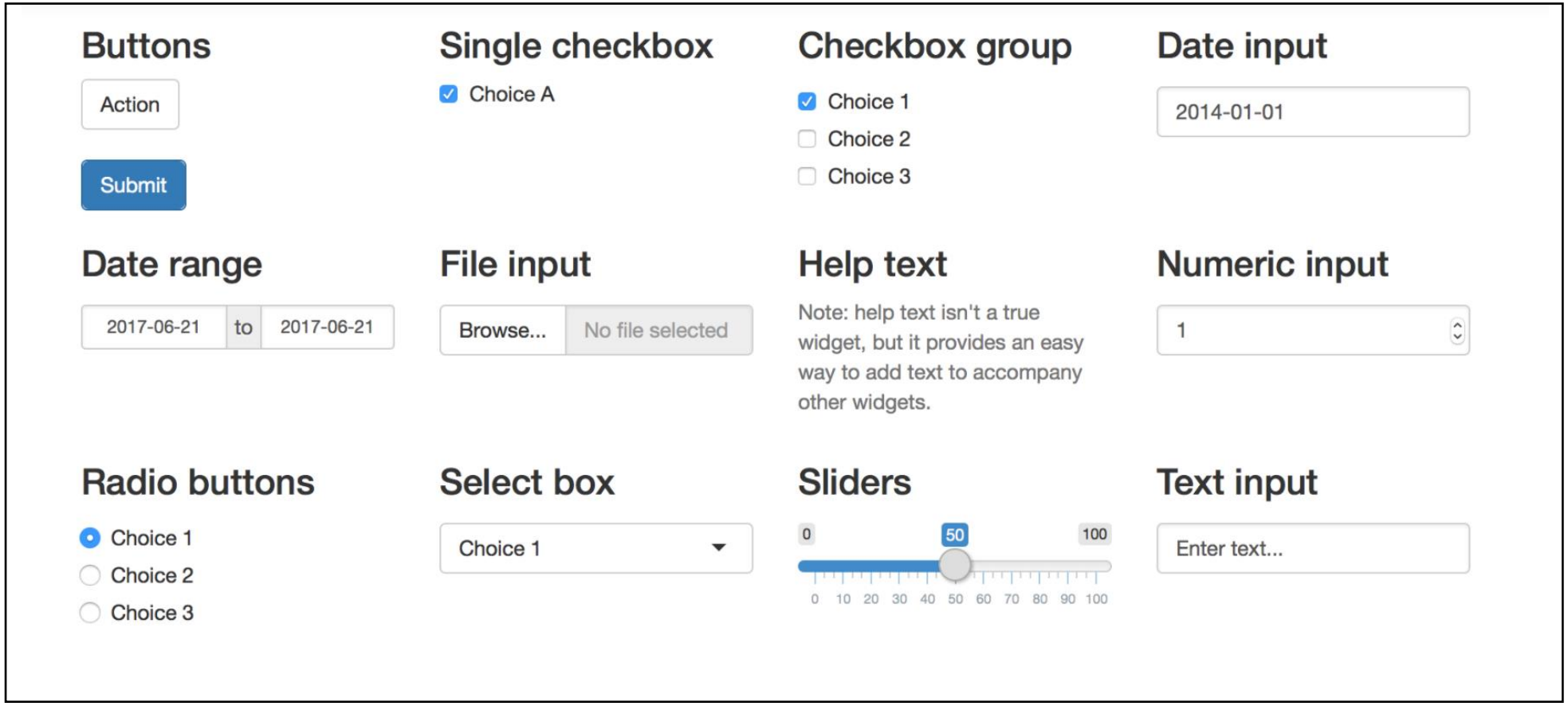
To run it


RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Help
Go to file/function
Addins
script1.R
server.r
Run App
library(shiny)
shinyServer(
function(input,output){
}
)

HTML Content

shiny function	HTML5 equivalent	creates
p	<p>	A paragraph of text
h1	<h1>	A first level header
h2	<h2>	A second level header
h3	<h3>	A third level header
h4	<h4>	A fourth level header
h5	<h5>	A fifth level header
h6	<h6>	A sixth level header
a	<a>	A hyper link
br	 	A line break (e.g. a blank line)
div		A division of text with a uniform style
span		An in-line division of text with a uniform style
pre	<pre>	Text ‘as is’ in a fixed width font
code	<code>	A formatted block of code
img	<img>	An image
strong		Bold text
em		Italicized text
HTML		Directly passes a character string as HTML code

Basic widgets



The standard Shiny widgets are:

function	widget
actionButton	Action Button
checkboxGroupInput	A group of check boxes
checkboxInput	A single check box
dateInput	A calendar to aid date selection
dateRangeInput	A pair of calendars for selecting a date range
fileInput	A file upload control wizard
helpText	Help text that can be added to an input form
numericInput	A field to enter numbers
radioButtons	A set of radio buttons
selectInput	A box with choices to select from
sliderInput	A slider bar
submitButton	A submit button
textInput	A field to enter text

Display reactive output

Two steps
You can create reactive output with a two step process.

1. Add an R object to your user interface.
2. Tell Shiny how to build the object in the server function. The object will be reactive if the code that builds it calls a widget value.

Step 1: Add an R object to the UI

Output function	Creates
dataTableOutput	DataTable
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	raw HTML
verbatimTextOutput	text

Step 2: Provide R code to build the object.

render function	**creates**
renderDataTable	DataTable
renderImage	images (saved as a link to a source file)
renderPlot	plots
renderPrint	any printed output
renderTable	data frame, matrix, other table like structures
renderText	character strings
renderUI	a Shiny tag object or HTML

Images

```
img(src = "my_image.png", height = 72, width = 72)
```

```
ui <- fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      img(src = "rstudio.png", height = 140, width = 400)
    )
  )
)
```

My Shiny App

Image example

- Let's build an example with an image
- How about we create a histogram of data
- Put a slider on so that the user has to guess the mean

ui.R

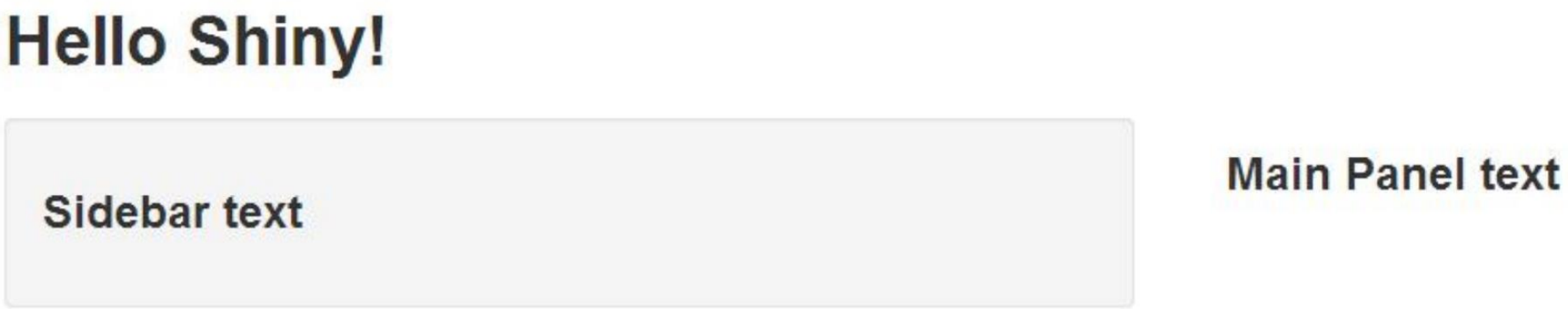
```
library(shiny)
shinyUI(pageWithSidebar(
  headerPanel("Data science FTW!"),
  sidebarPanel(
    h3('Sidebar text')
  ),
  mainPanel(
      h3('Main Panel text')
  )
))
```

h3(“Third level title”)

server.r

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

Output



R functions for HTML markup

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("Illustrating markup"),
  sidebarPanel(
      h1('Sidebar panel'),
      h1('H1 text'),
      h2('H2 Text'),
      h3('H3 Text'),
      h4('H4 Text')

  ),
  mainPanel(
      h3('Main Panel text'),
      code('some code'),
      p('some ordinary text')
  )
))
```

h1("First level title")

h3("Third level title")

Illustrating markup

Sidebar panel

H1 text

H2 Text

H3 Text

H4 Text

Main Panel text

`some code`

some ordinary text

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("Illustrating inputs"),
  sidebarPanel(
    numericInput('id1', 'Numeric input, labeled id1', 0, min = 0, max = 10, step = 1),
    checkboxGroupInput("id2", "Checkbox",
                       c("Value 1" = "1",
                         "Value 2" = "2",
                         "Value 3" = "3")),
    dateInput("date", "Date:")
  ),
  mainPanel(

  )
))
```

Illustrating inputs

Illustrating inputs

Numeric input, label id1

5

Checkbox

- [ ] Value 1
- [ ] Value 2
- [ ] Value 3

Date:

2021-06-06

Let’s build our prediction function

ui.R

```
shinyUI(
  pageWithSidebar(
    # Application title
    headerPanel("Diabetes prediction"),

    sidebarPanel(
      numericInput('glucose', 'Glucose mg/dl', 90, min = 50, max = 200, step = 5),
      submitButton('Submit')
    ),
    mainPanel(
        h3('Results of prediction'),
        h4('You entered'),
        verbatimTextOutput("inputValue"),
        h4('Which resulted in a prediction of '),
        verbatimTextOutput("prediction")
    )
  )
)
```

Server. R

```
diabetesRisk <- function(glucose) glucose / 200

shinyServer(
  function(input, output) {
    output$inputValue <- renderPrint({input$glucose})
    output$prediction <- renderPrint({diabetesRisk(input$glucose)})
  }
)
```

The result

Diabetes prediction

Glucose mg/dl

120

Submit

Results of prediction

You entered

```
[1] 120
```

Which resulted in a prediction of

```
[1] 0.6
```

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("Example plot"),
  sidebarPanel(
    sliderInput('mu', 'Guess at the mean',value = 70, min = 62, max = 74, step = 0.05,)
  ),
  mainPanel(
    plotOutput('newHist')
  )
))
```

server.R

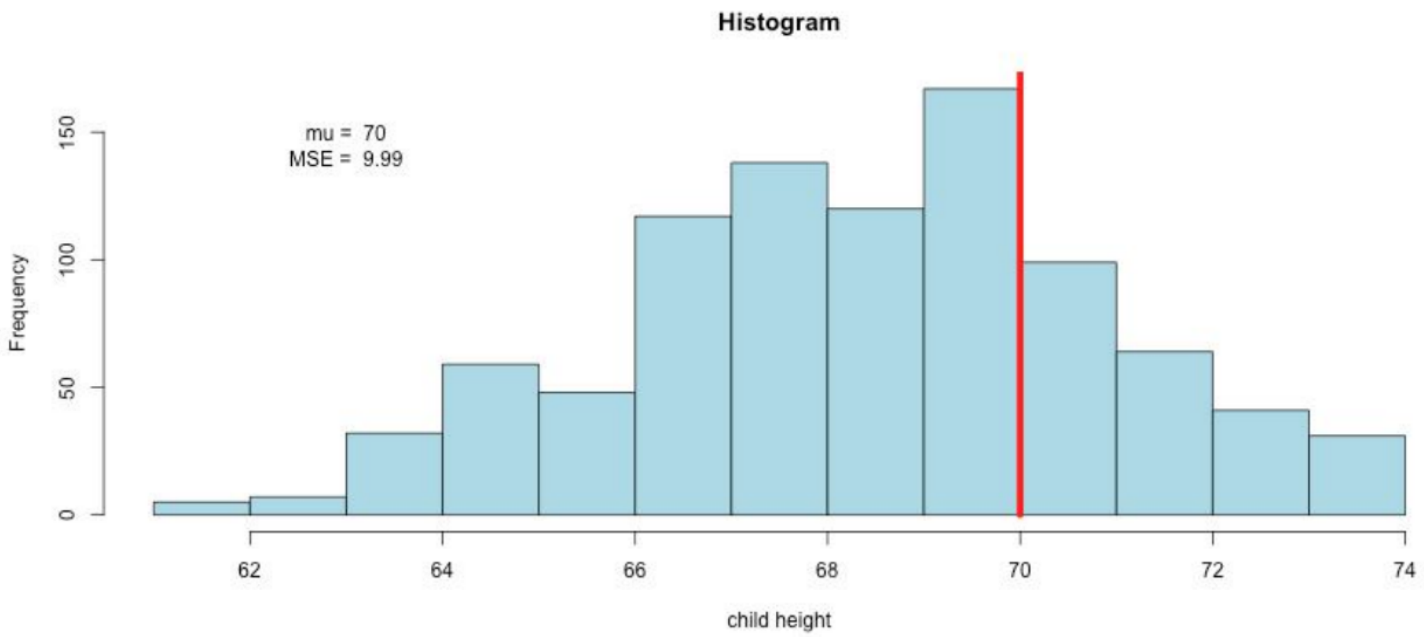
```
library(UsingR)
data(galton)

shinyServer(
  function(input, output) {
    output$newHist <- renderPlot({
      hist(galton$child, xlab='child height', col='lightblue',main='Histogram')
      mu <- input$mu
      lines(c(mu, mu), c(0, 200),col="red",lwd=5)
      mse <- mean((galton$child - mu)^2)
      text(63, 150, paste("mu = ", mu))
      text(63, 140, paste("MSE = ", round(mse, 2)))
      })

  }
)
```

The output

Example plot

Guess at the mean

62 70 74



Other things Shiny can do

- Allow users to upload or download files
- Have tabbed main panels Have editable data tables
- Have a dynamic UI User defined inputs and outputs
- Put a submit button so that Shiny only executes complex code after user hits submit

https://shiny.rstudio.com/gallery/

shiny.rstudio.com/gallery/

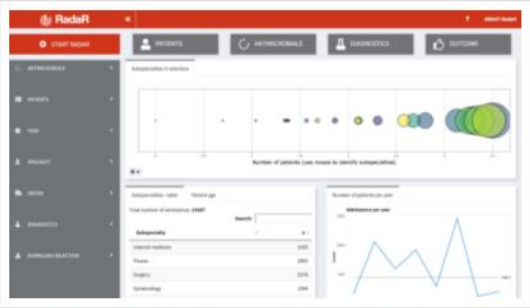
Shiny from RStudio

Get Started | Gallery | Articles | App Stories | Reference | Deploy | Help | Contribute

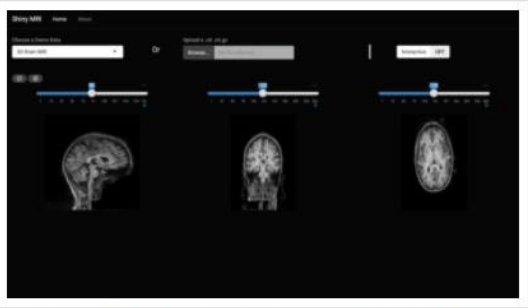
Compare the cycling speed of cities

Life sciences

COVID-19 tracker



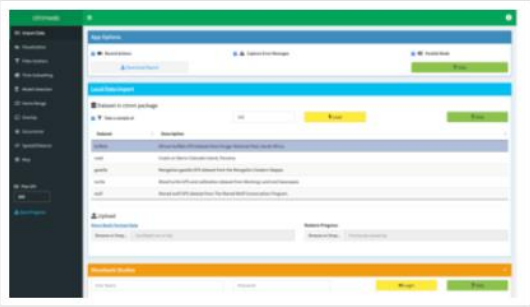
Exploring large hospital data for better use of antimicrobials



ShinyMRI - View MRI images in Shiny

Nutrition Calculator - calculate nutrition for recipes

A/B Testing Sample Size Calculator

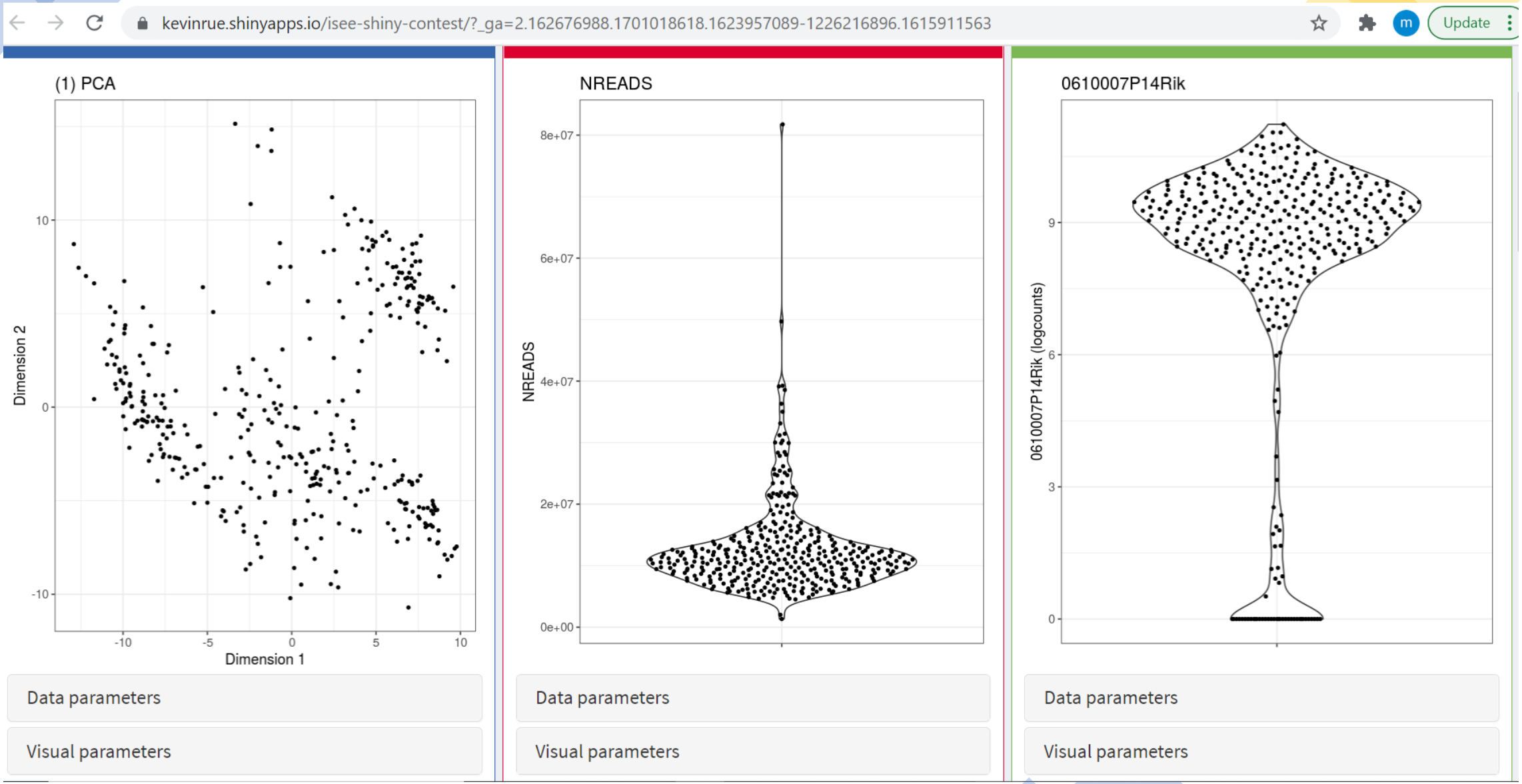


ctmmweb, a web app to analysis Animal tracking data

Visualizing Biodiversity in National Parks data

ExPanD: Explore Your Data Interactively

https://shiny.rstudio.com/gallery/



https://shiny.rstudio.com/gallery/

kevinrue.shinyapps.io/isee-shiny-contest/?_ga=2.162676988.1701018618.1623957089-1226216896.1615911563

Update

Column statistics table 1

Show 10 entries

Search:

	NREADS	driver_1_s	dissection_s	Core.Type	passes_qc_checks_s	sizeFactors(se)
		A	All	/	All	All
SRR2140028	13743900	Scnn1a-Tg3-Cre	L4	Intermediate	Y	0.998110767443503
SRR2140022	14078700	Scnn1a-Tg3-Cre	L4	Core	Y	1.24333131596158
SRR2140055	5842930	Scnn1a-Tg3-Cre	All	Intermediate	Y	0.452070689173832
SRR2140083	16784400	Scnn1a-Tg3-Cre	L4		N	1.04916797364396
SRR2139991	11558600	Scnn1a-Tg3-Cre	L4	Intermediate	Y	0.917782329483387
SRR2140067	6298380	Scnn1a-Tg3-Cre	All	Core	Y	0.456212939248591

Heat map 1

Manipulate

Data Products

Manipulate

- Suppose that you want to create a quick interactive graphic
  - You have to do it *now*
  - The intended users also use Rstudio
- `manipulate` is a really cool solution that is often all you need to quickly make interactive graphics

Documentation

- Manipulate is well documented at the Rstudio web site here
  - http://www.rstudio.com/ide/docs/advanced/manipulate
- From there, try this `library(manipulate) manipulate(plot(1:x), x = slider(1, 100))`
- You can create a slider, checkbox, or picker (drop down) and have more than one

Example

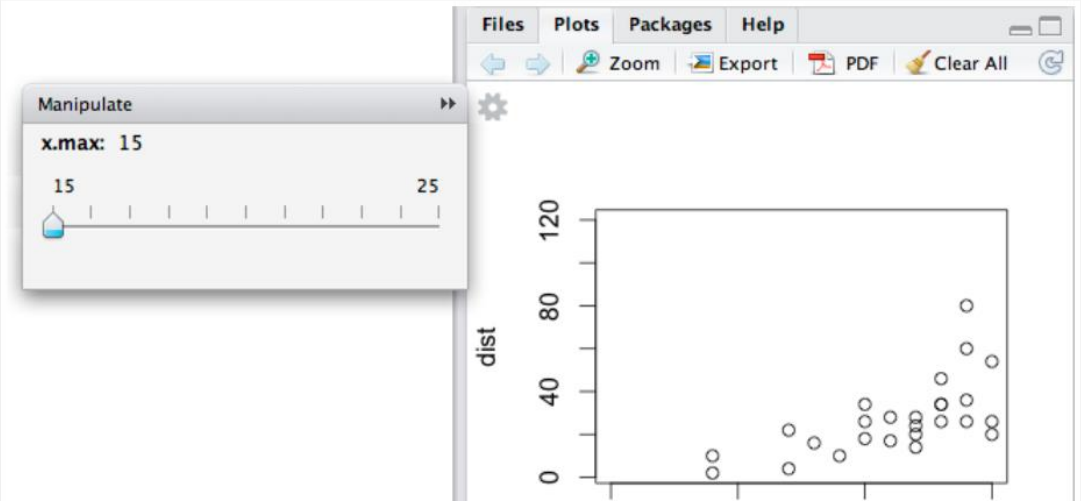
```
library(manipulate)
manipulate(plot(1:x), x = slider(1, 100))
```

Slider Control

The `slider` control enables manipulation of plot variables along a numeric range. For example:

```
manipulate(
  plot(cars, xlim=c(0,x.max)),
  x.max=slider(15,25))
```

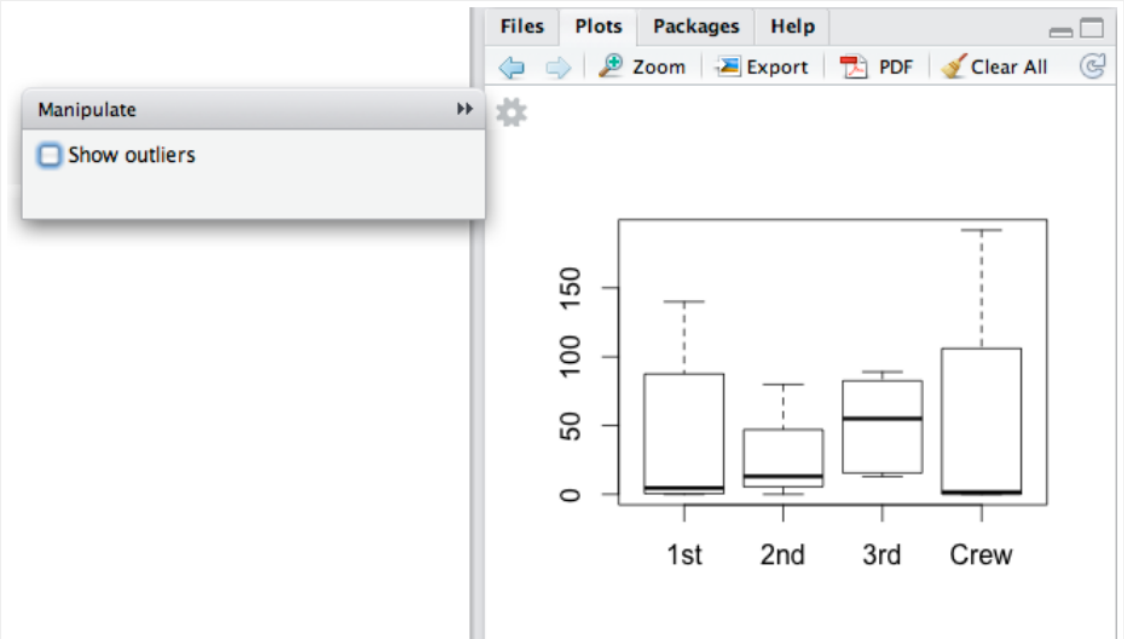
Results in this plot and manipulator:



The `checkbox` control enables manipulation of logical plot variables. For example:

```
manipulate(
  boxplot(Freq ~ Class, data = Titanic, outline = outline),
  outline = checkbox(FALSE, "Show outliers"))
```

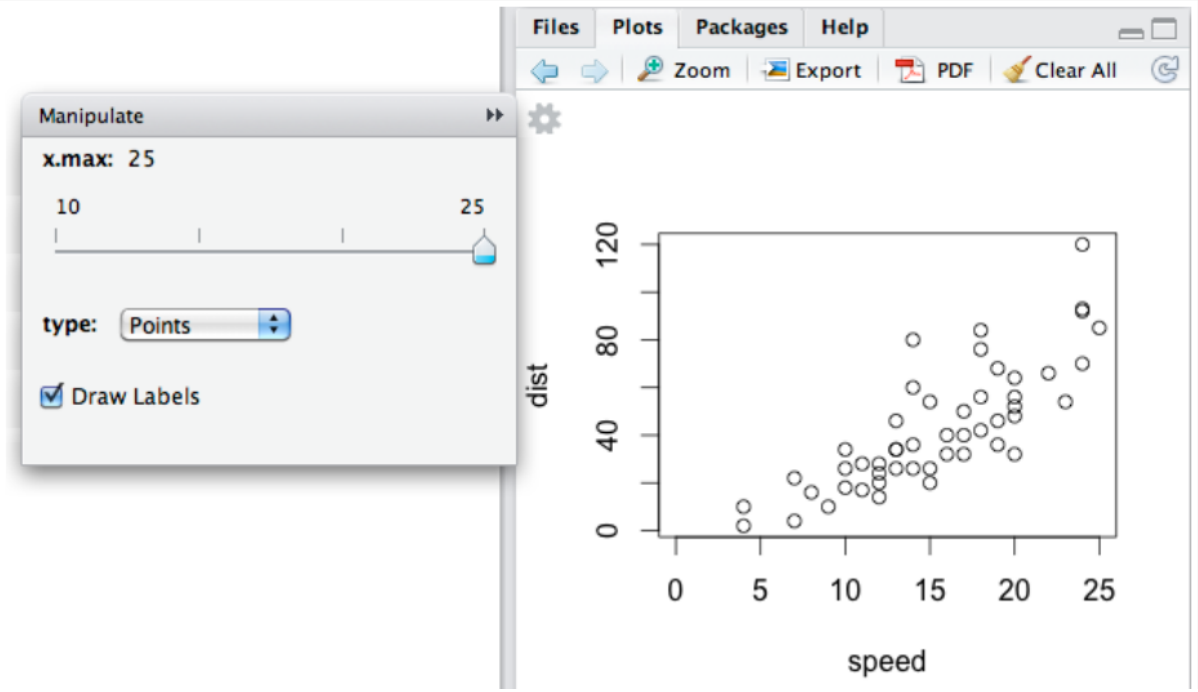
Results in this plot and manipulator:



Multiple controls can be combined within a single manipulator. For example:

```
manipulate(
  plot(cars, xlim = c(0, x.max), type = type, ann = label),
  x.max = slider(10, 25, step=5, initial = 25),
  type = picker("Points" = "p", "Line" = "l", "Step" = "s"),
  label = checkbox(TRUE, "Draw Labels"))
```

Results in this plot and manipulator:



Example

```
library(manipulate)
myHist <- function(mu){
  hist(galton$child,col="blue",breaks=100)
  lines(c(mu, mu), c(0, 150),col="red",lwd=5)
  mse <- mean((galton$child - mu)^2)
  text(63, 150, paste("mu = ", mu))
  text(63, 140, paste("MSE = ", round(mse, 2)))
}
manipulate(myHist(mu), mu = slider(62, 74, step = 0.5))
```