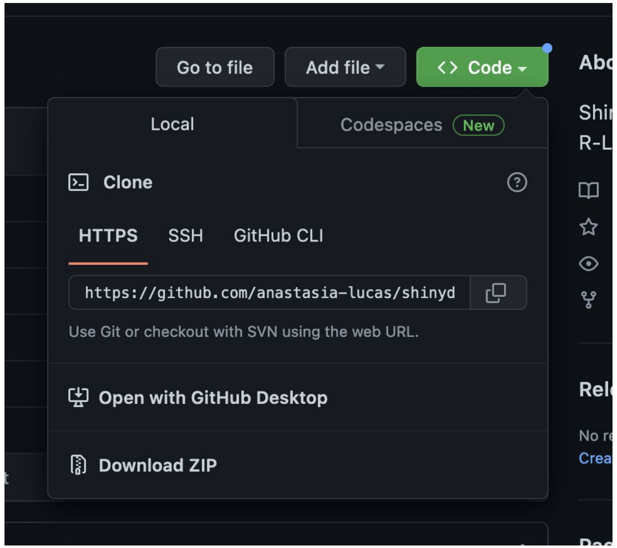
While we wait, please take the poll & download or clone the repo

Poll: https://forms.gle/Tm6mvEkgtWCEXV4y5

Exercises: https://github.com/anastasia-lucas/shinydash-rladies-dc

- Exercises in exercises/
- Solutions in solutions/
- Make sure packages in utils.R are installed!



```
https://github.com/anastasia-lucas/shinyd
```

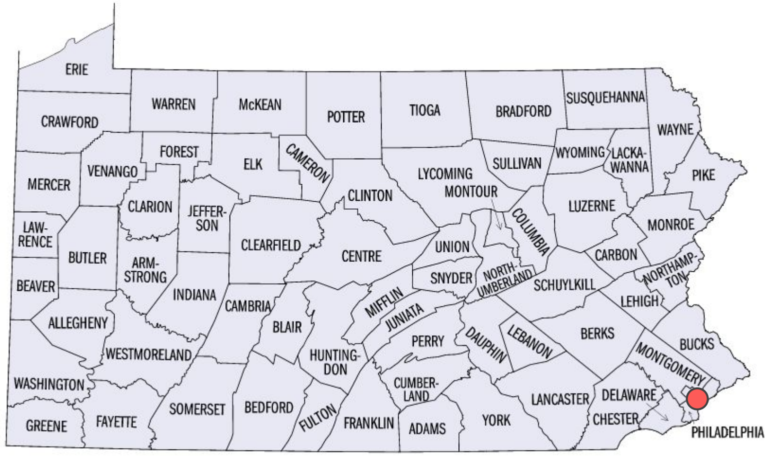
Use Git or checkout with SVN using the web URL.

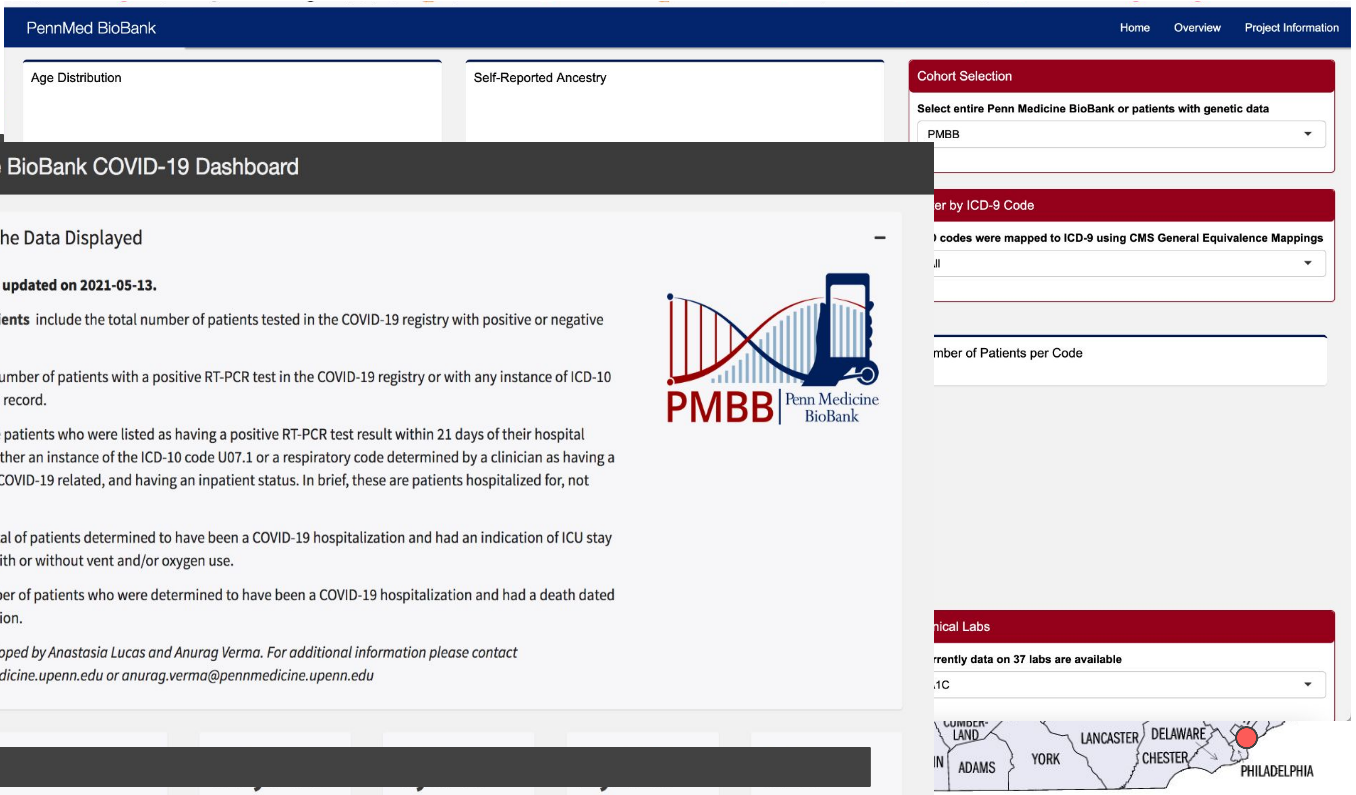
R-Ladies D.C.
Introduction to Shiny dashboards

Anastasia Lucas

Quick about me

- B.S. in Biostatistics
- Genetics data analysis
- Geisinger — EHR & genetics data integration
- EHR & genetics data integration
- Immunotherapy clinical & multi-omics data integration
- 1st year PhD student in Genomics & Computational Biology




PennMed BioBank

Age Distribution

Self-Reported Ancestry

Cohort Selection

Select entire Penn Medicine BioBank or patients with genetic data

PMBB

er by ICD-9 Code

) codes were mapped to ICD-9 using CMS General Equivalence Mappings

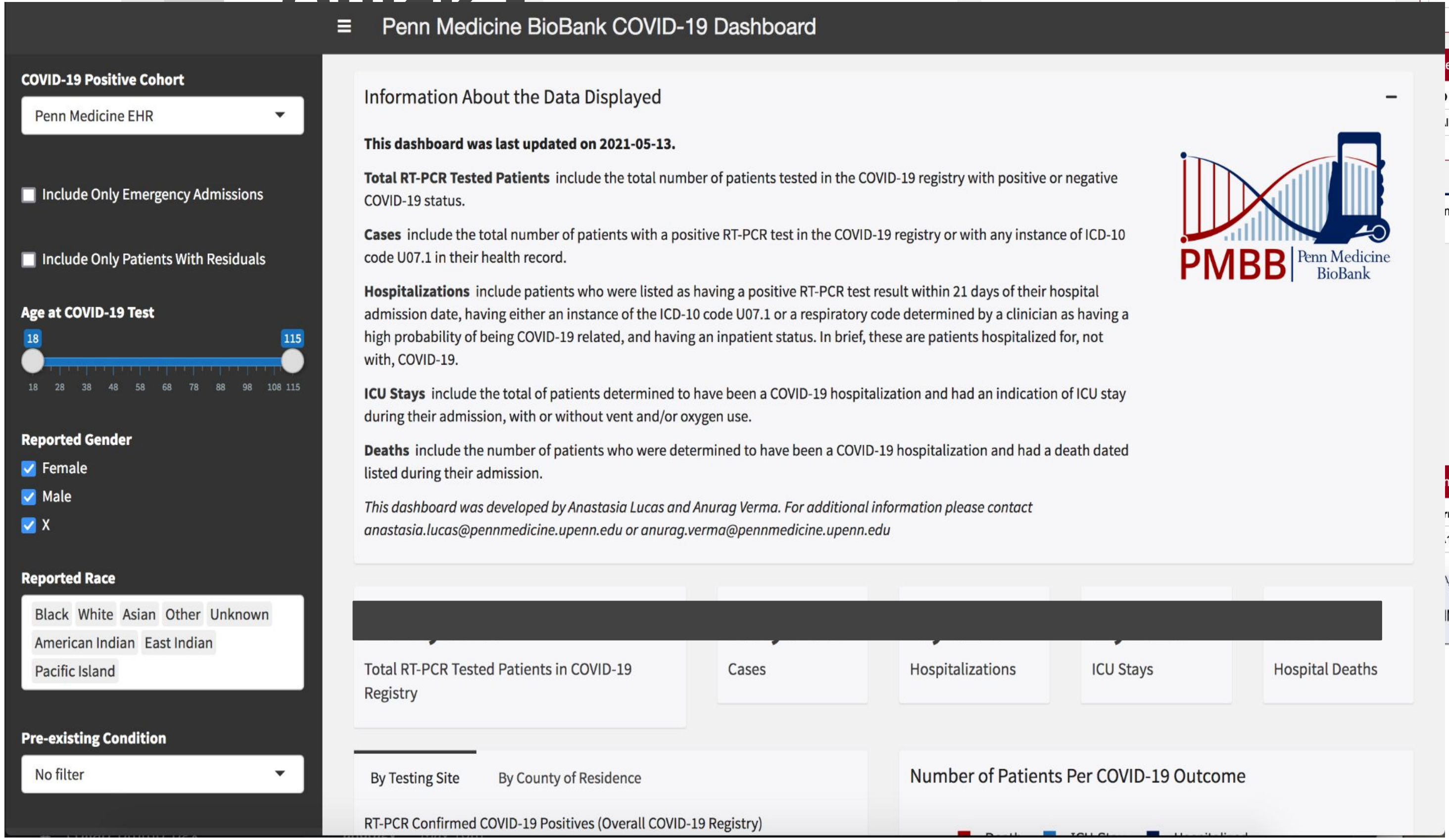
ll

mber of Patients per Code

ical Labs

rrently data on 37 labs are available

1C


Penn Medicine BioBank COVID-19 Dashboard

COVID-19 Positive Cohort

Penn Medicine EHR

☐ Include Only Emergency Admissions

☐ Include Only Patients With Residuals

Age at COVID-19 Test

18 – 115

Reported Gender

✓ Female
✓ Male
✓ X

Reported Race

Black White Asian Other Unknown American Indian East Indian Pacific Island

Pre-existing Condition

No filter

Information About the Data Displayed

This dashboard was last updated on 2021-05-13.

Total RT-PCR Tested Patients include the total number of patients tested in the COVID-19 registry with positive or negative COVID-19 status.

Cases include the total number of patients with a positive RT-PCR test in the COVID-19 registry or with any instance of ICD-10 code U07.1 in their health record.

Hospitalizations include patients who were listed as having a positive RT-PCR test result within 21 days of their hospital admission date, having either an instance of the ICD-10 code U07.1 or a respiratory code determined by a clinician as having a high probability of being COVID-19 related, and having an inpatient status. In brief, these are patients hospitalized for, not with, COVID-19.

ICU Stays include the total of patients determined to have been a COVID-19 hospitalization and had an indication of ICU stay during their admission, with or without vent and/or oxygen use.

Deaths include the number of patients who were determined to have been a COVID-19 hospitalization and had a death dated listed during their admission.

This dashboard was developed by Anastasia Lucas and Anurag Verma. For additional information please contact anastasia.lucas@pennmedicine.upenn.edu or anurag.verma@pennmedicine.upenn.edu

PMBB Penn Medicine BioBank

[redacted]

Total RT-PCR Tested Patients in COVID-19 Registry

Cases

Hospitalizations

ICU Stays

Hospital Deaths

By Testing Site | By County of Residence

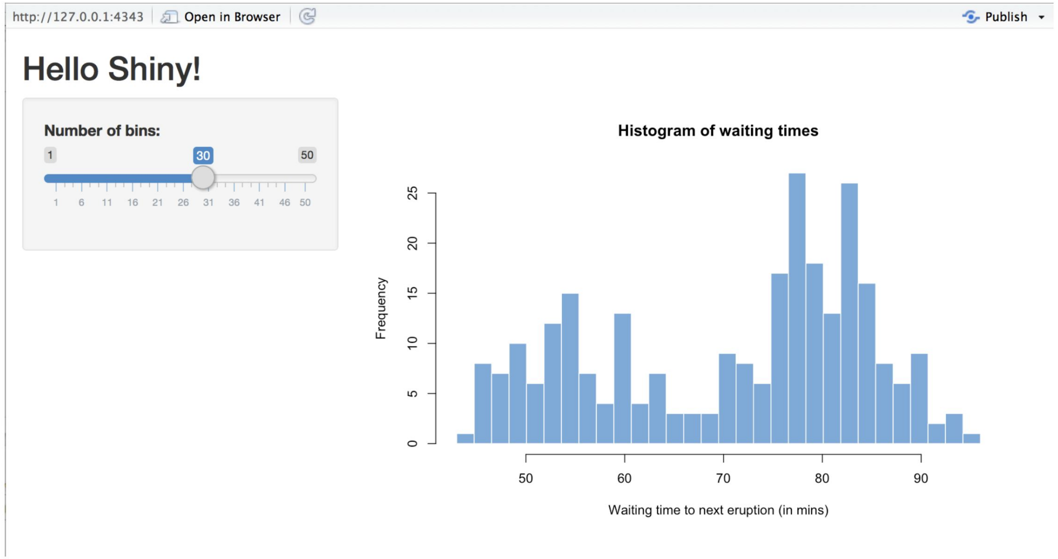
RT-PCR Confirmed COVID-19 Positives (Overall COVID-19 Registry)

Number of Patients Per COVID-19 Outcome

Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

- interactive web app
- built on HTML

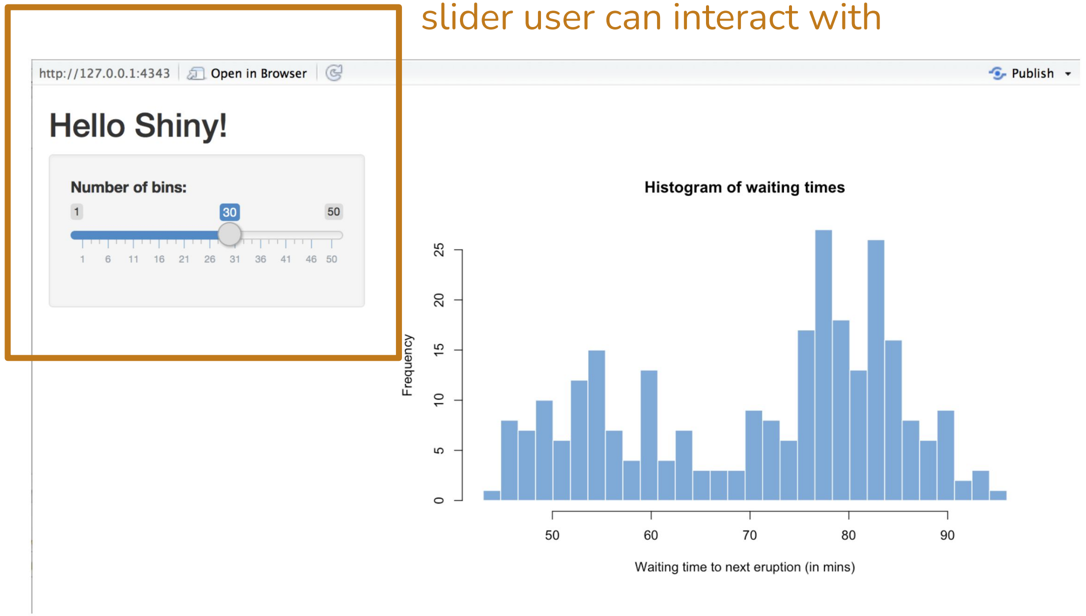


Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

interactive web app

slider user can interact with

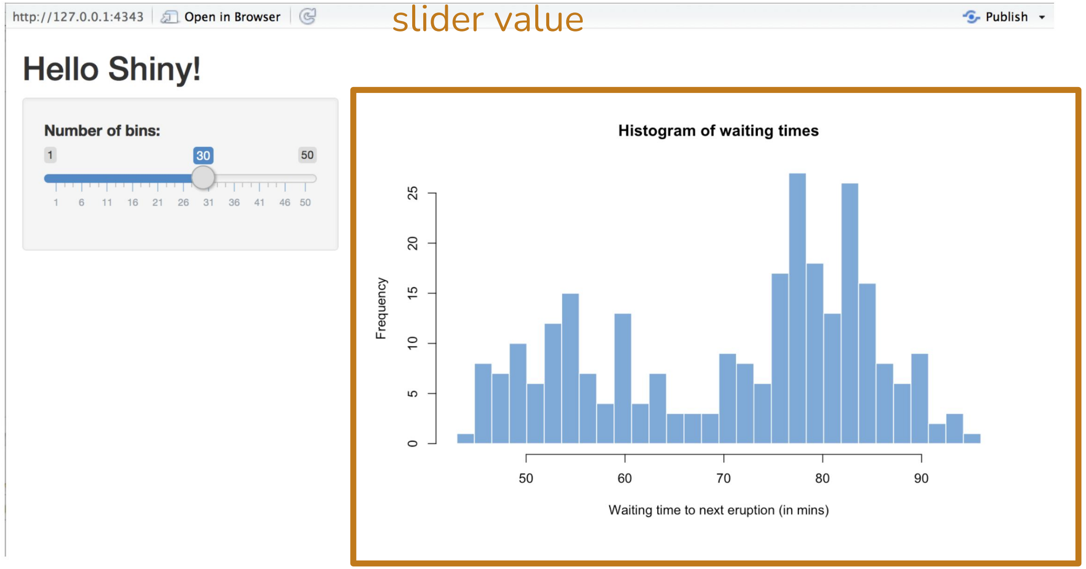


Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

interactive web app

histogram which changes based on slider value

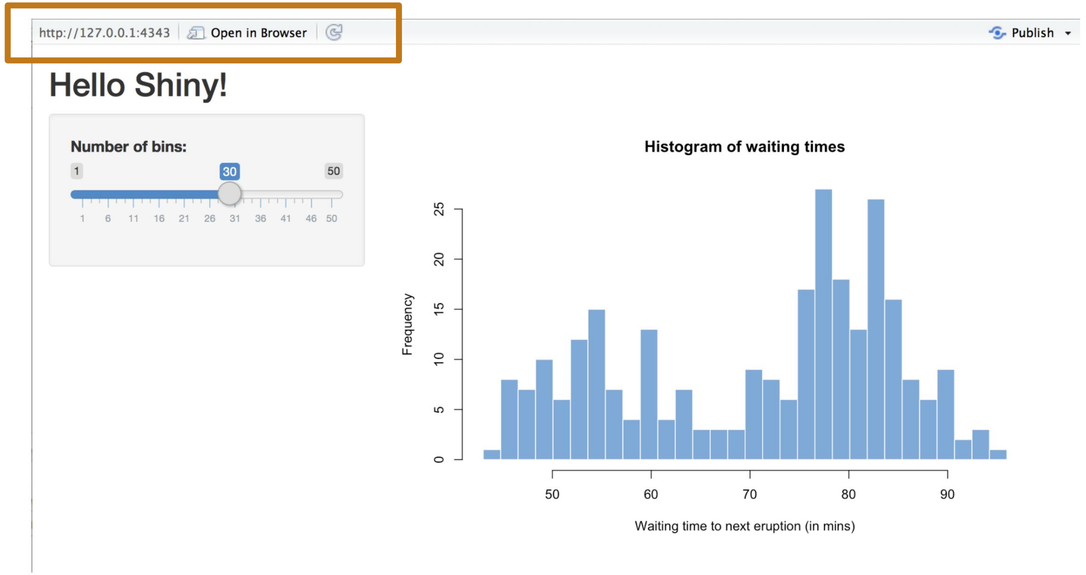


Before we get to Shiny dashboards, what is a Shiny app?

runs on a server

https://shiny.rstudio.com/

interactive web app



Shiny apps allow for interactivity and reactivity

- Interactivity: an element reacts to user actions
  - hover text, on-click actions, highlight on hover
  - {plotly}, {ggiraph}, {Shiny}

- Reactivity: data is updated from the server without refreshing the site
  - changing binwidth, filtering, changing parameters in models
  - {Shiny}, {shinydashboard}, {flexdashboard}

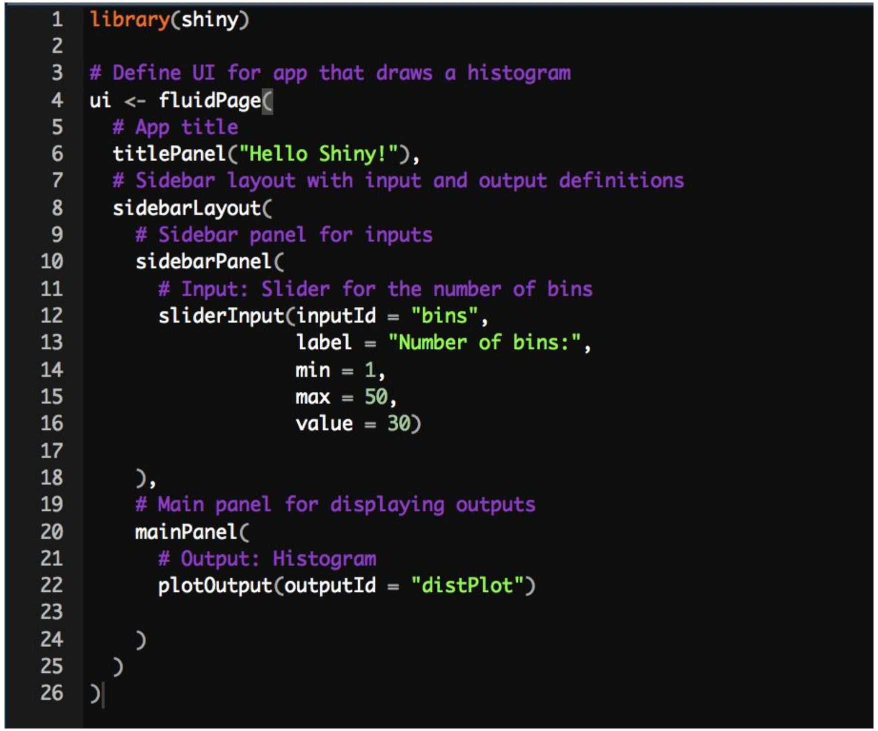
How would we make a Shiny app?

Shiny apps have three main components:

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object



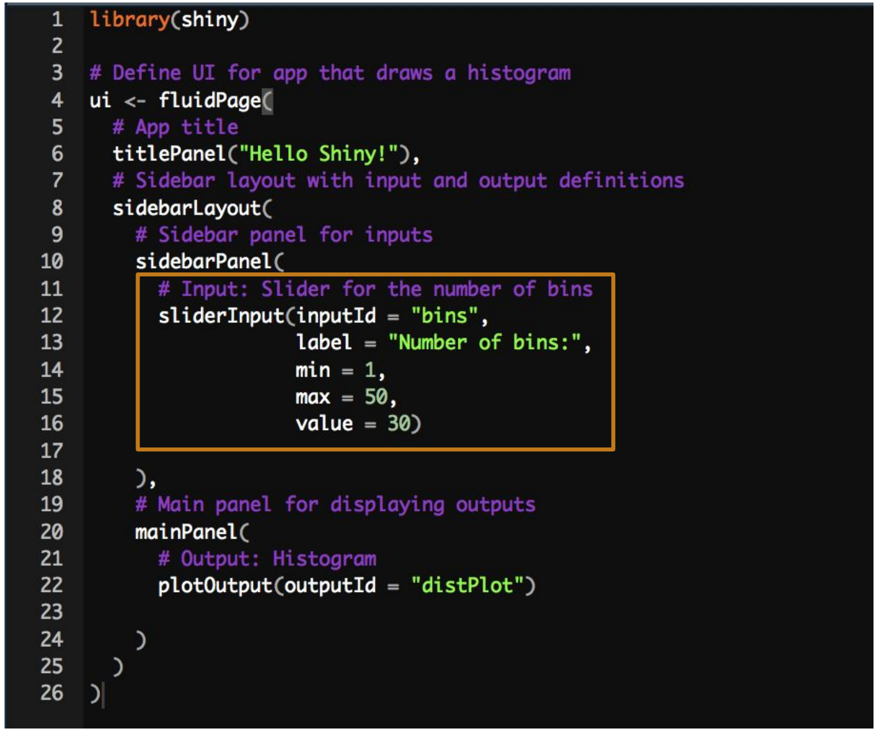
```r
library(shiny)

# Define UI for app that draws a histogram
ui <- fluidPage(
  # App title
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions
  sidebarLayout(
    # Sidebar panel for inputs
    sidebarPanel(
      # Input: Slider for the number of bins
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),
    # Main panel for displaying outputs
    mainPanel(
      # Output: Histogram
      plotOutput(outputId = "distPlot")

    )
  )
)
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

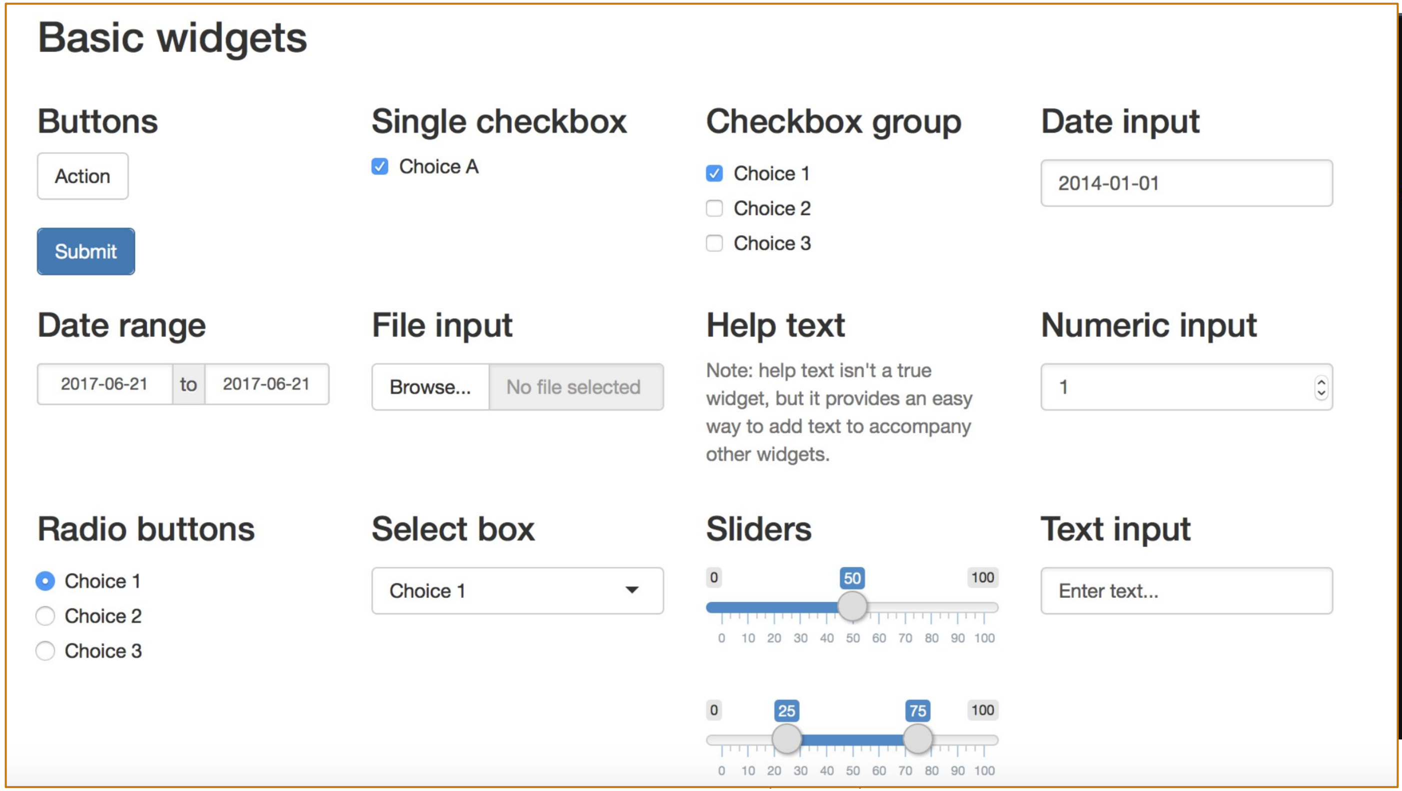
1. a user interface object



```r
library(shiny)

# Define UI for app that draws a histogram
ui <- fluidPage(
  # App title
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions
  sidebarLayout(
    # Sidebar panel for inputs
    sidebarPanel(
      # Input: Slider for the number of bins
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),
    # Main panel for displaying outputs
    mainPanel(
      # Output: Histogram
      plotOutput(outputId = "distPlot")

    )
  )
)
```

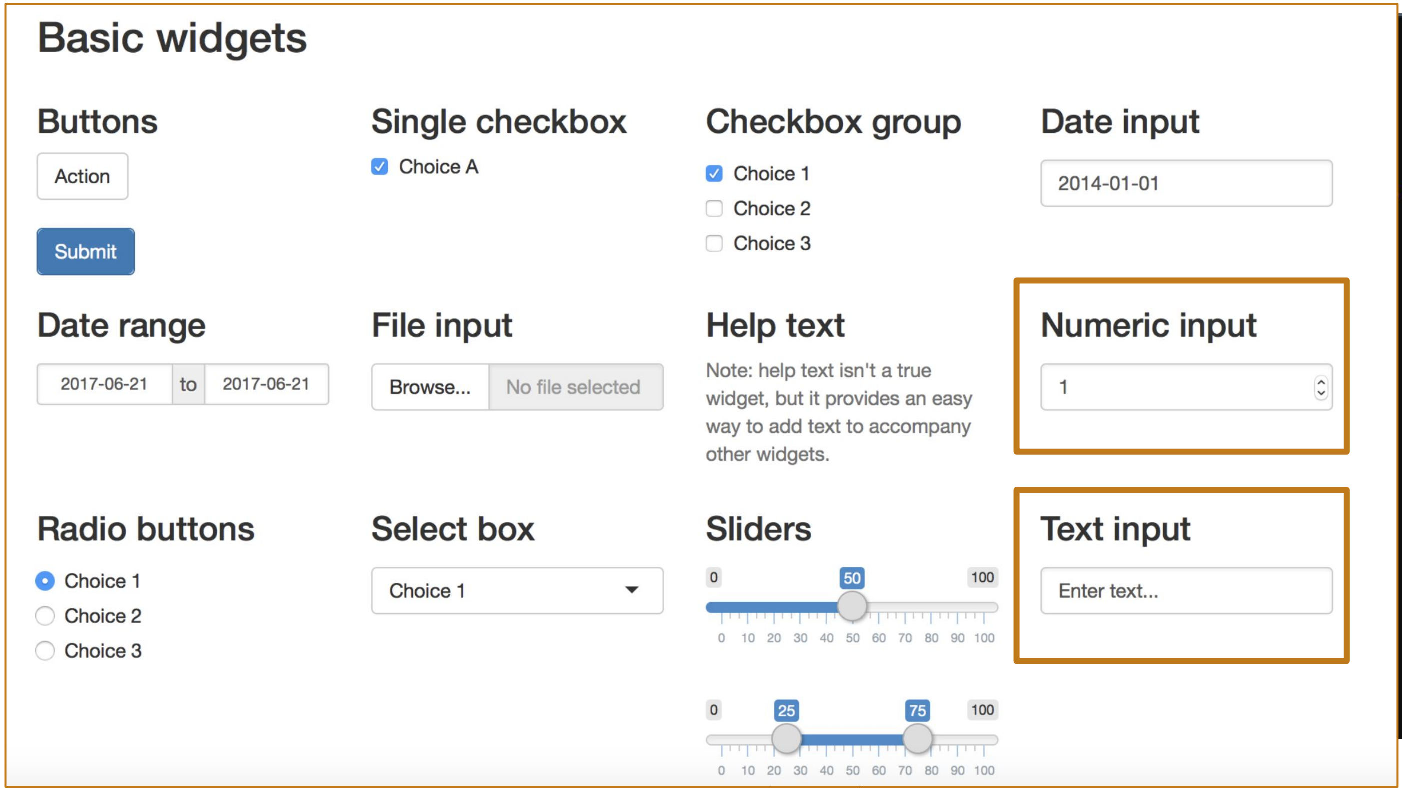
https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?



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

How would we make a Shiny app?

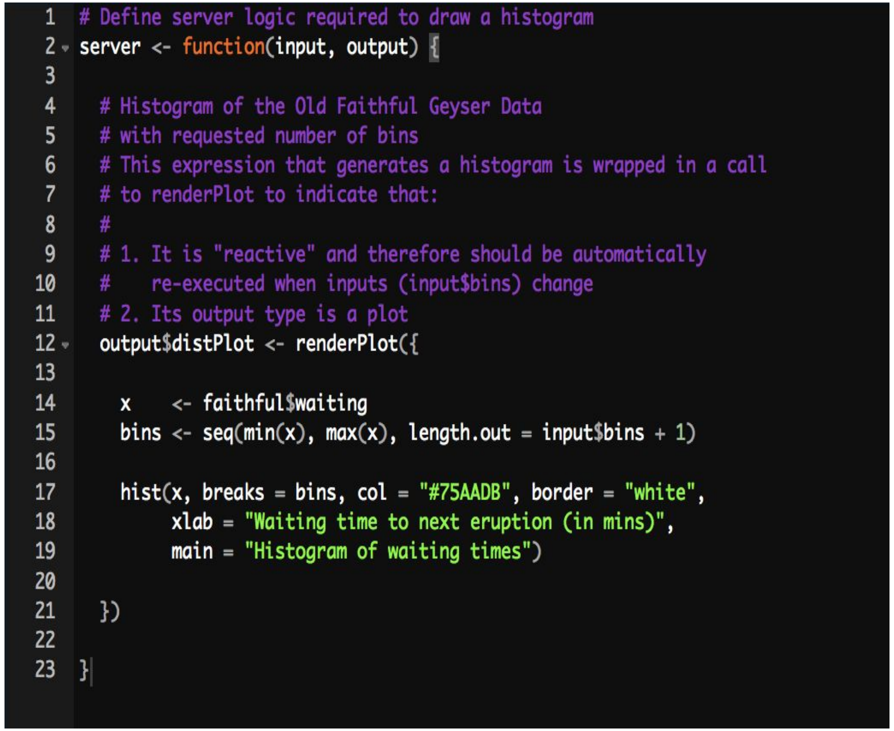


Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function



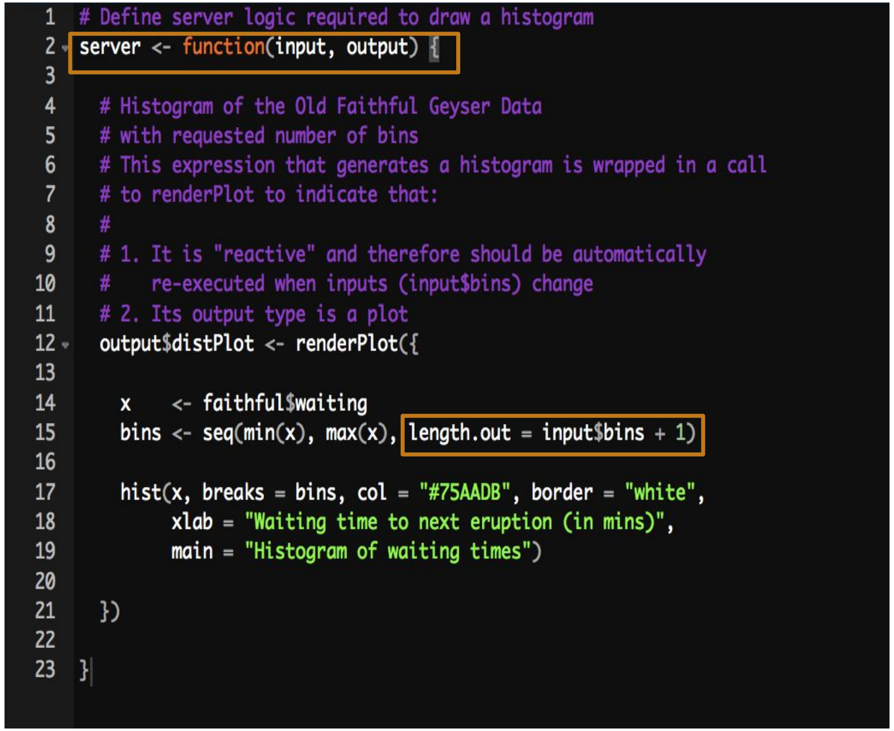
```r
# Define server logic required to draw a histogram
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function



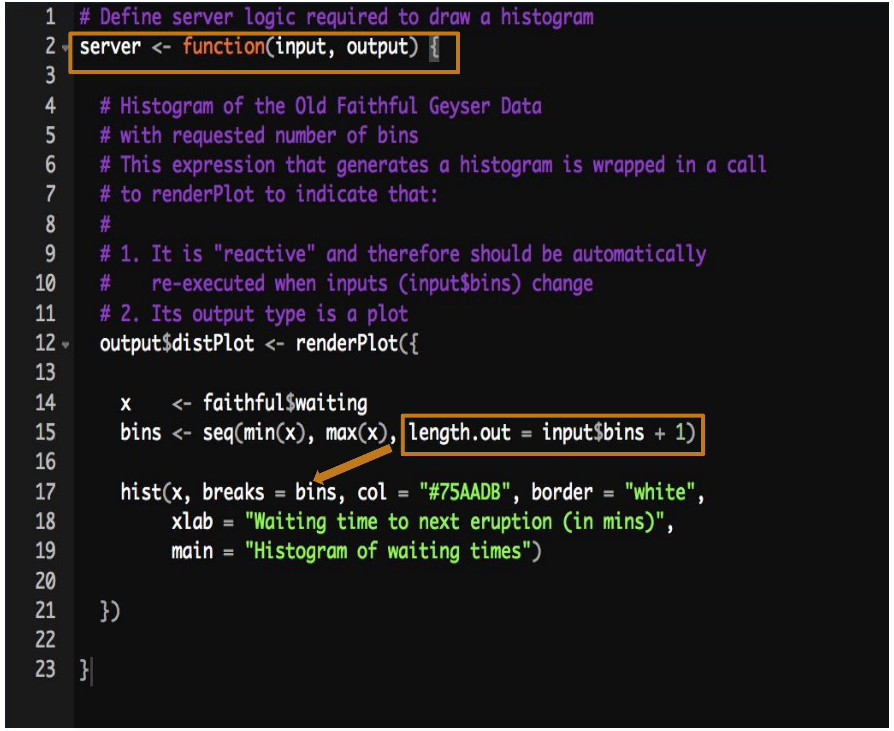
```r
# Define server logic required to draw a histogram
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function



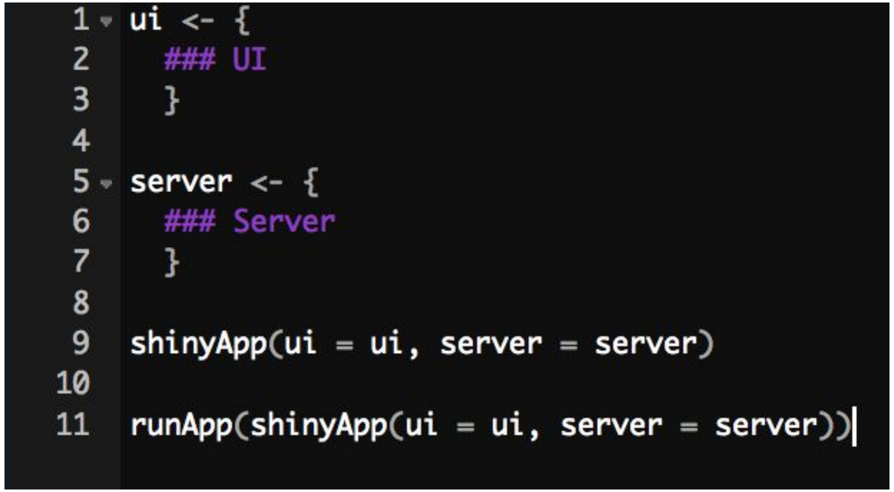
```r
# Define server logic required to draw a histogram
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function
3. a call to `ShinyApp()`



```
ui <- {
  ### UI
  }

server <- {
  ### Server
  }

shinyApp(ui = ui, server = server)

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

https://shiny.rstudio.com/articles/basics.html

So what is a Shiny dashboard?

- Simply put, a collection of Shiny apps
- More generally, a graphical interface for users to quickly visualize important metrics

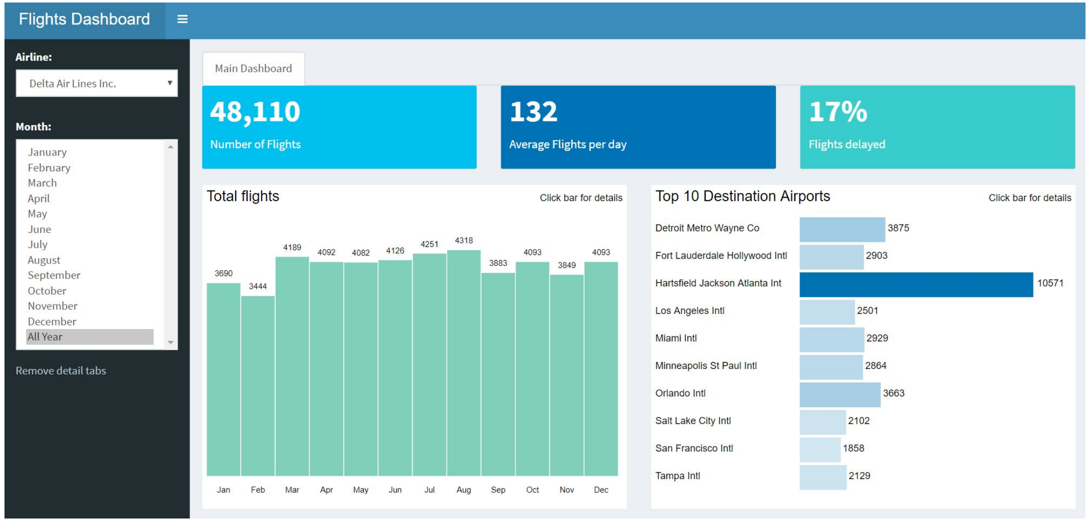


Image: https://db.rstudio.com/best-practices/dashboards/dashboard.png

Reasons to use Shiny dashboard

- Nice framework for making a professional looking product without wrangling with layouts
- You can do a lot without needing to know HTML and CSS
- If you already know flexdashboard, this may not provide a ton of advantages
  - shinydashboardPlus that provides even more features and is built off of shinydashboard
- Run analyses on the fly

Things to think about before we start

Before we start coding, I like to ask myself a few questions

1. Who will my users be?
2. What insights do we want our users to be able to gain?
3. Is this the best way to present the data given #1 & #2?

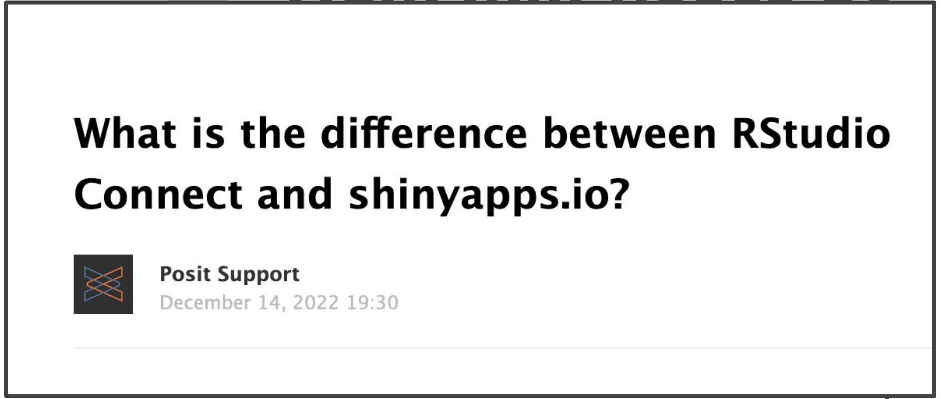
and one more...

How will users access the dashboard?

R provides a couple of hosting options:

1. Deploy to the cloud using Shinyapps.io
   a. Different pricing levels offer different features, limits on active users, etc.
2. Host on Shiny server
   a. Open source
3. Deploy with RStudio Connect
   a. Commercial software
   b. Not free

* We needed authentication, so we set up the free Shiny server and app on an AWS server, then set up our website which had authentication to be a proxy for the server.

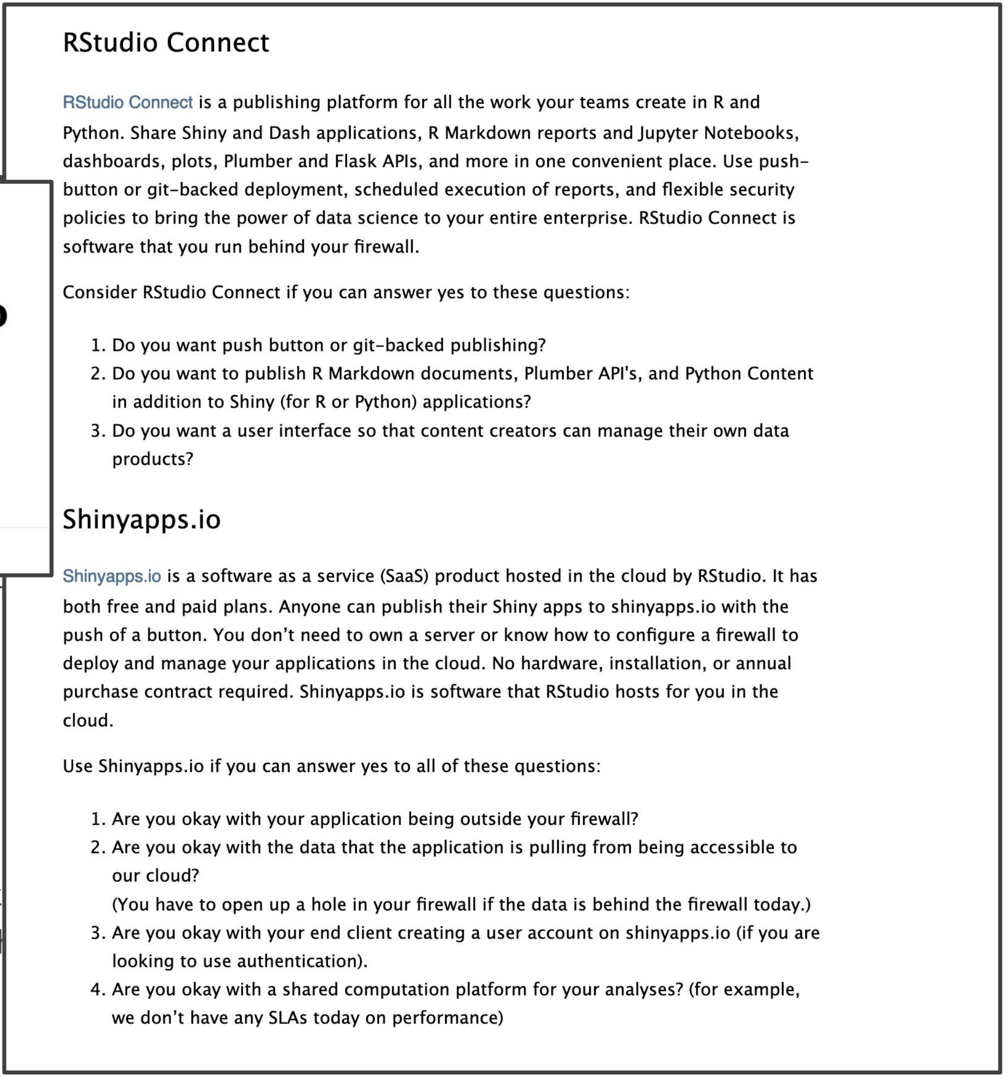


What is the difference between RStudio Connect and shinyapps.io?

Posit Support
December 14, 2022 19:30

   a. Different pricing levels offer dif
2. Host on Shiny server
   a. Open source
3. Deploy with RStudio Connect
   a. Commercial software
   b. Not free

* We needed authentication, so we set
then set up our website which had auth



RStudio Connect

RStudio Connect is a publishing platform for all the work your teams create in R and Python. Share Shiny and Dash applications, R Markdown reports and Jupyter Notebooks, dashboards, plots, Plumber and Flask APIs, and more in one convenient place. Use push-button or git-backed deployment, scheduled execution of reports, and flexible security policies to bring the power of data science to your entire enterprise. RStudio Connect is software that you run behind your firewall.

Consider RStudio Connect if you can answer yes to these questions:

1. Do you want push button or git-backed publishing?
2. Do you want to publish R Markdown documents, Plumber API's, and Python Content in addition to Shiny (for R or Python) applications?
3. Do you want a user interface so that content creators can manage their own data products?

Shinyapps.io

Shinyapps.io is a software as a service (SaaS) product hosted in the cloud by RStudio. It has both free and paid plans. Anyone can publish their Shiny apps to shinyapps.io with the push of a button. You don't need to own a server or know how to configure a firewall to deploy and manage your applications in the cloud. No hardware, installation, or annual purchase contract required. Shinyapps.io is software that RStudio hosts for you in the cloud.

Use Shinyapps.io if you can answer yes to all of these questions:

1. Are you okay with your application being outside your firewall?
2. Are you okay with the data that the application is pulling from being accessible to our cloud?
   (You have to open up a hole in your firewall if the data is behind the firewall today.)
3. Are you okay with your end client creating a user account on shinyapps.io (if you are looking to use authentication).
4. Are you okay with a shared computation platform for your analyses? (for example, we don't have any SLAs today on performance)

Ways to make your dashboard easier to deploy

- Have separate scripts for data preprocessing
- Use a utils script that installs all needed packages
- Only load libraries that you use a lot of functions from, e.g. shiny, dplyr, ggplot2
  Otherwise use package::function() notation
- Use Conventional Commits & tag repos

Now let’s do some exercises!

https://github.com/anastasia-lucas/shinydash-rladies-dc