Shiny Web Applications in R

Byeong-Hak Choe

R Shiny Setup

What is shiny and what is it good for?

Interactive tables and graphs

User interface (UI)

Where to go from here

R Shiny Setup

Required software

There are a few things you need to install in order to follow along with the examples and exercises.

- **shiny, rsconnect, plotly, leaflet, and DT packages**:
  `install.packages(c("shiny", "rsconnect", "plotly", "leaflet", "DT"))`

What is shiny and what is it good for?

A web application framework in R

- Shiny is a web application framework in R.

A web application framework in R

- Shiny is a web application framework in R.
- You can build web apps without knowing HTML/CSS/Javascript!

A web application framework in R

- Shiny is a web application framework in R.
- You can build web apps without knowing HTML/CSS/Javascript!
- The whole R package ecosystem is available for use in your web app!

Use cases and examples

- Exploratory data analysis:
  https://jjames.shinyapps.io/shinyHome/
- Teaching and learning:
  http://www.statstudio.net/free-tools/dists/
- Shiny Gallery: https://shiny.rstudio.com/gallery/
- Some built-in examples accessible via `runExample()`

Sharing and deploying

- Shiny apps can be run locally using the `shiny` R package.

Sharing and deploying

- Shiny apps can be run locally using the `shiny` R package.

- Apps can be deployed to the freemium https://shinyapps.io service.

Sharing and deploying

- Shiny apps can be run locally using the `shiny` R package.

- Apps can be deployed to the freemium https://shinyapps.io service.

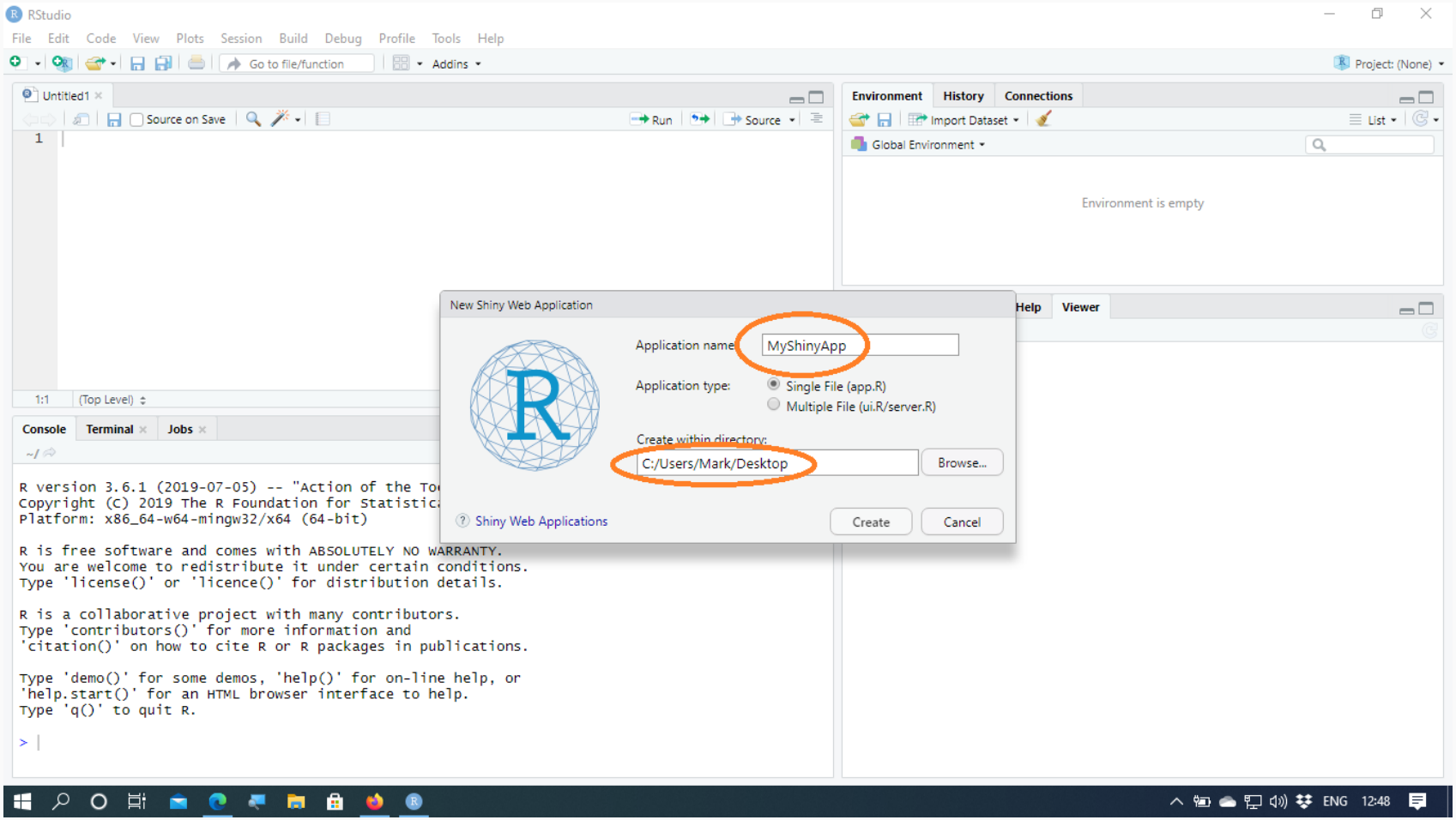
- You (or your organization) can run a free version of Shiny Server on your own system, or pay for the full-featured *RStudio Connect* platform.

Sharing and deploying—Step 1

- Code your web shiny app with RStudio and save it to your computer.
    - Open RStudio and a new Shiny App File by following the path "File>New File> Shiny Web App...".

Sharing and deploying—Step 1

- R will save your app in a folder you name and a directory you choose.



Sharing and deploying—Step 1

- Write your code for your application in RStudio.

```
library(shiny)
# Define UI for application that draws a histogram
ui <- fluidPage(
        sidebarLayout(
        sidebarPanel(sliderInput("samplesize",
                                 "Sample Size:",
                                 min = 100,
                                 max = 10000,
                                 value = 1000)),
        mainPanel(plotOutput("distPlot"))
    )
)
```

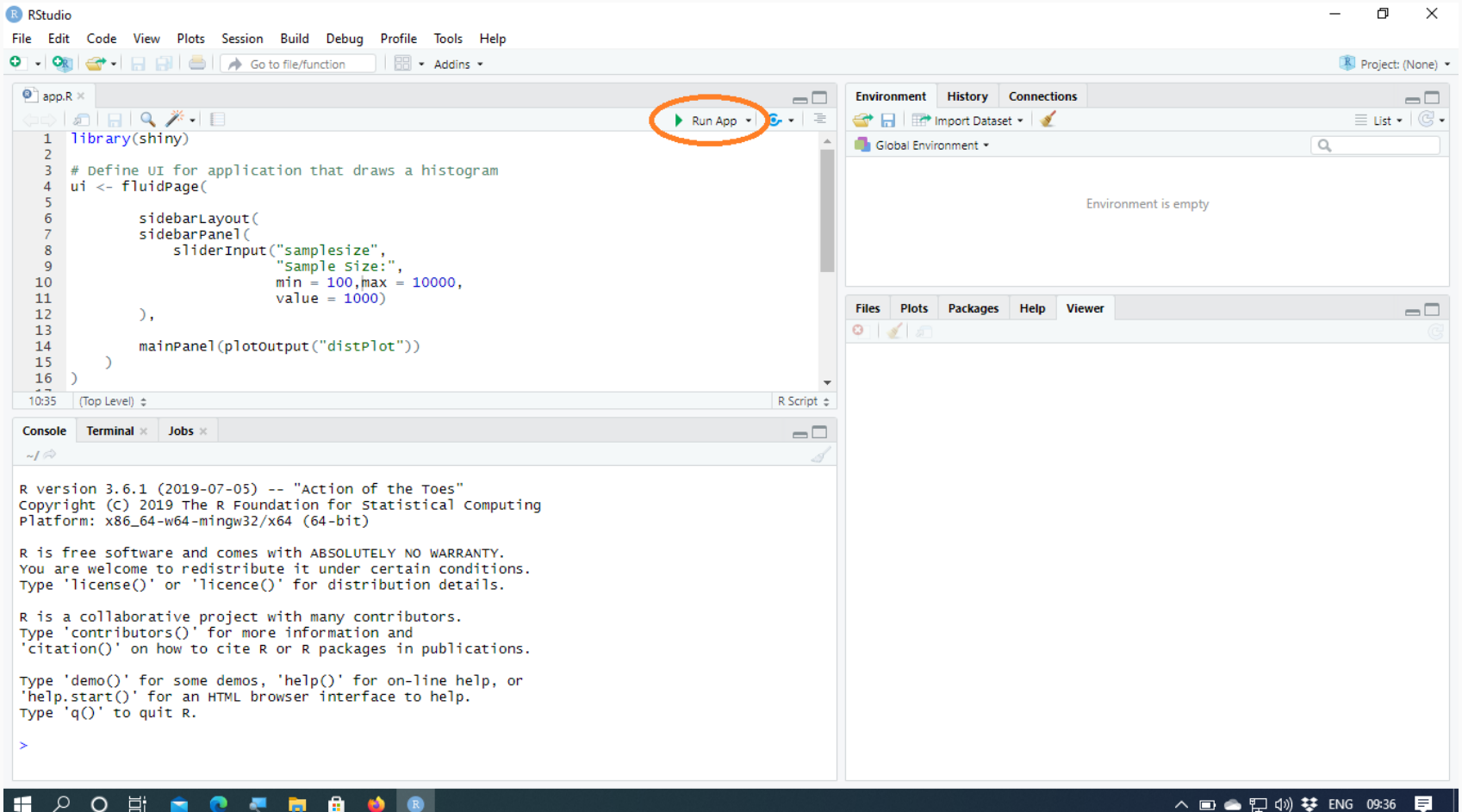
Sharing and deploying—Step 1

- Copy and paste all the codes in a single R file.

```
# Define server logic required to draw a histogram
server <- function(input, output) {
    output$distPlot <- renderPlot({
        hist(rnorm(input$samplesize),
             col='darkorchid',
             xlab="Sample",
             main="Normally Distributed Sample")},
        height=300
    )
}
# Run the application
shinyApp(ui = ui, server = server)
```

Sharing and deploying—Step 2

- Click the “Run App” button.
  - to run the app on your computer in order to be sure that there are no mistakes in the code.



Sharing and deploying—Step 3

- Go to https://www.shinyapps.io/ and sign up.
  - Enter your email address and construct a new password.
  - You are also allowed to sign up with your Google or GitHub account.
  - This platform will enable us to deploy our R shiny app and we can access and use it online and also share it with other users.
  - The free version allows you to deploy up to 5 applications.

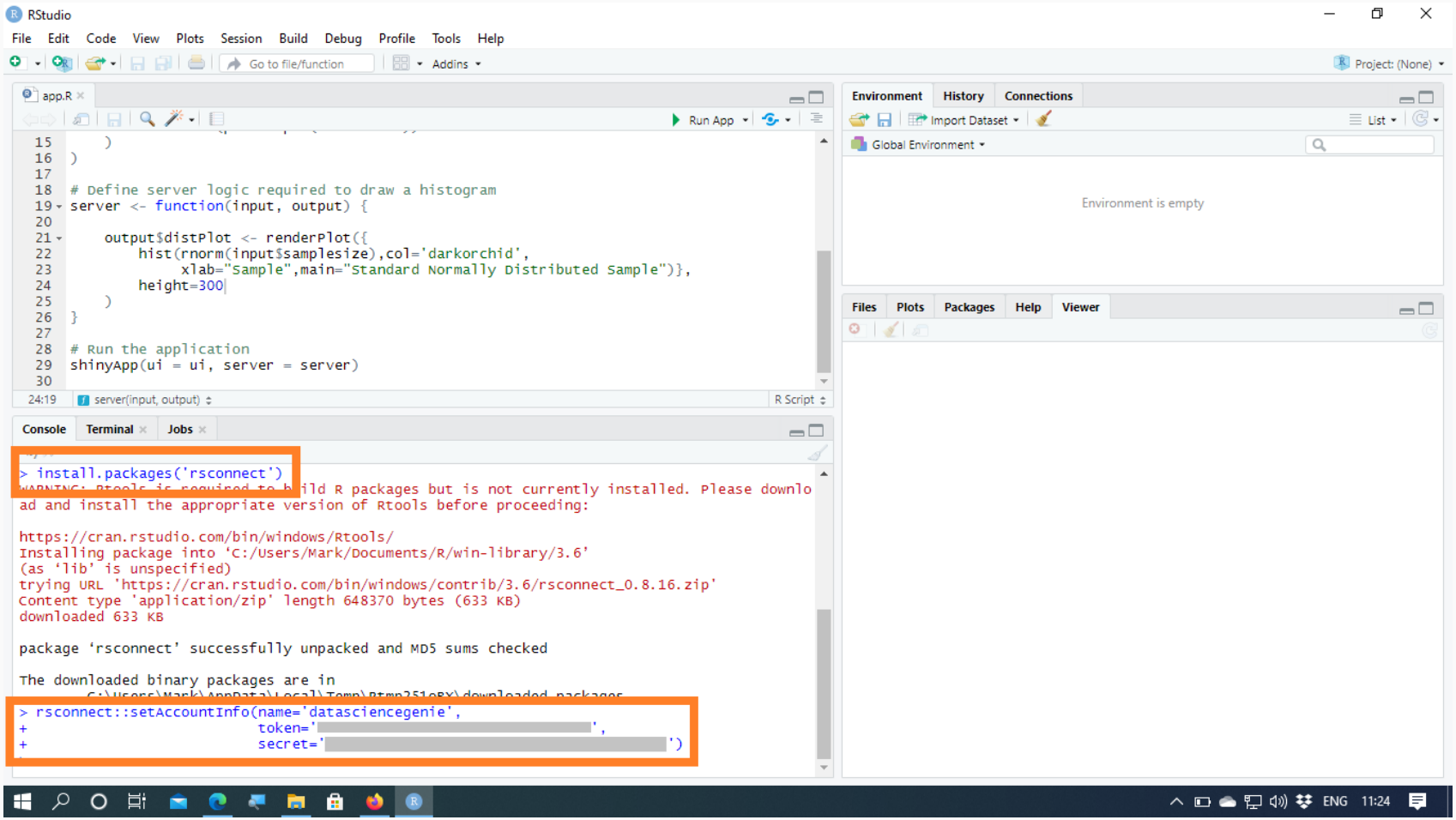
Sharing and deploying—Step 4

- Choose your username.
  - Immediately after signing up, you are requested to choose your own username.

Sharing and deploying—Step 5

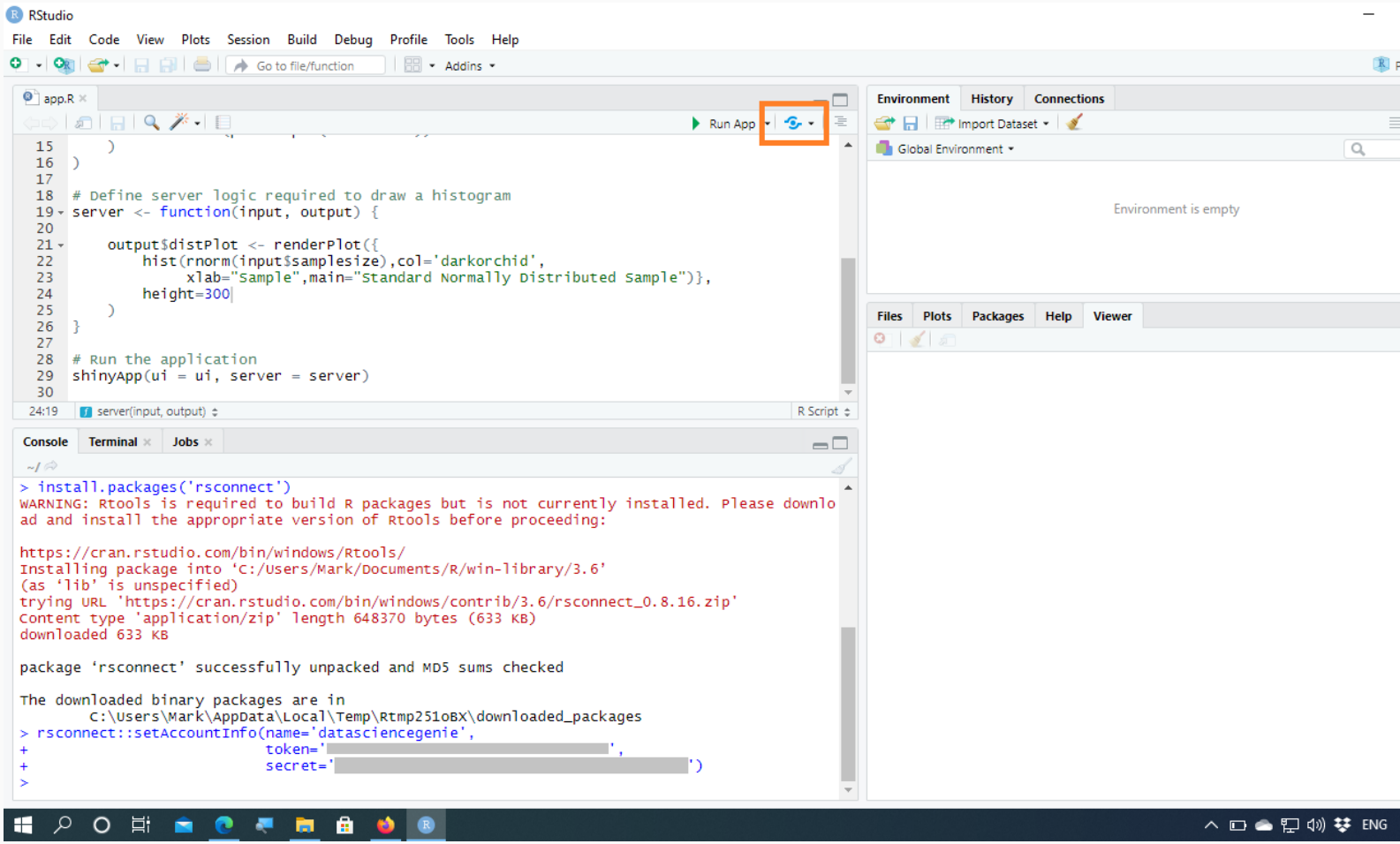
- Connect your RStudio to your online platform.
  - Make sure that you click the "Show secret" button and copy the two code lines to your R console.

Sharing and deploying—Step 5



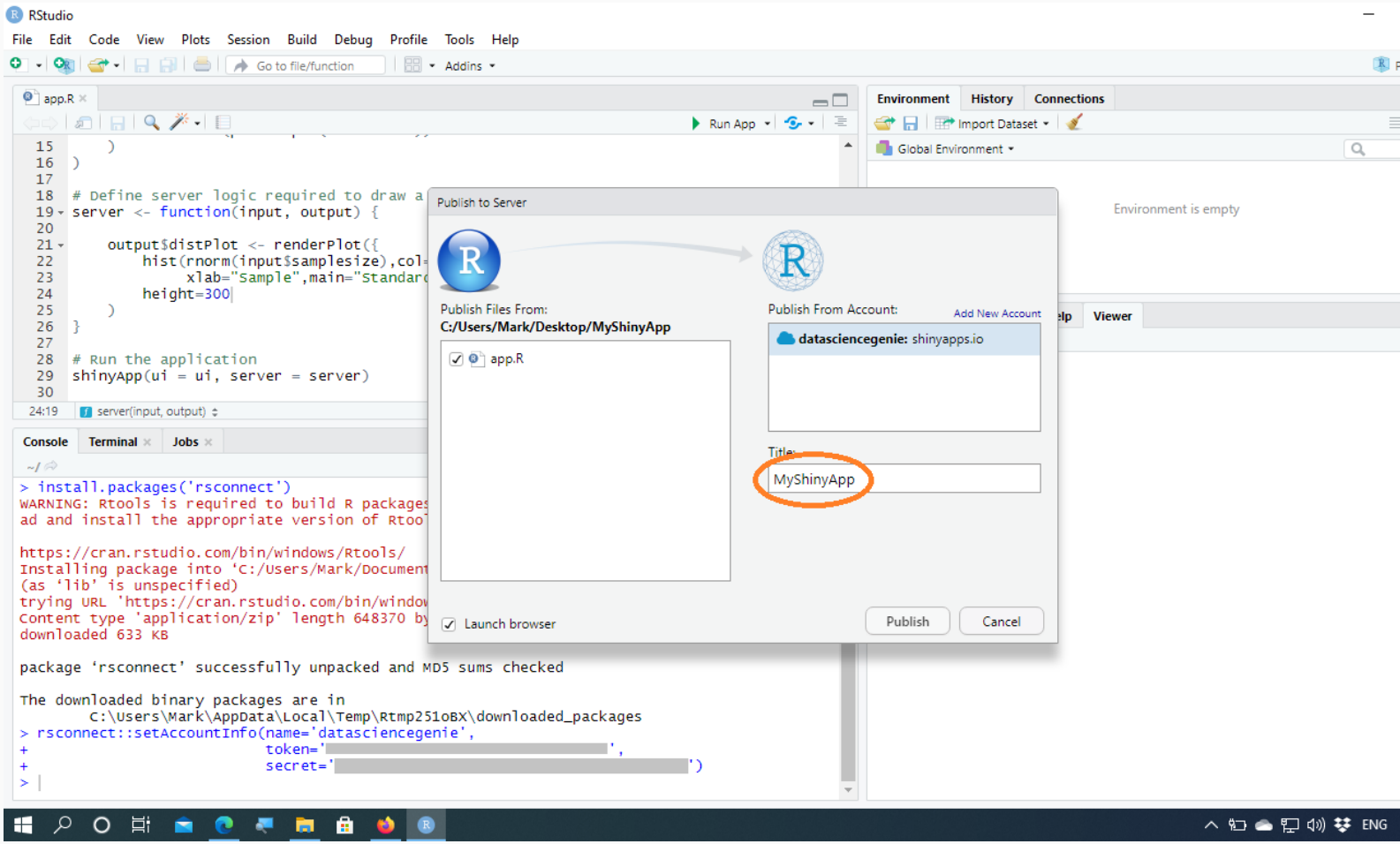
Sharing and deploying—Step 6

- Publish your App online
  - Now click the blue “Publish” button.



Sharing and deploying—Step 6

- Publish your App online
  - Choose a name for your app and click “Publish”.



Sharing and deploying—Step 7

- Embed the App on your personal website.
  - Now the app has been uploaded online on shinyapp.io.
  - You are given a URL which involves your username and the name of the app.
  - You can copy this URL and send it to other people whom you would like to use your app.

Sharing and deploying—Step 7

- Embed the App on your personal website using R Markdown

````
---
title: "TITLE"
subtitle: "SUBTITLE"
---
````{=html}
<style>iframe{height: 1200px; width: 800px}</style>

<iframe height="100%" width="100%" frameborder="no"
src="https://YOUR_USERNAME.shinyapps.io/APP_NAME/">
</iframe>
```` #FOUR BACKTICKS HERE TO CLOSE THE HTML CHUNK
````

UI and server elements

- The `ui` specifies the elements of your application and their arrangement.
  - Common elements include *inputs*, *outputs*, and *descriptive text*.
  - Elements can be easily arranged in panels and tabs.
- The `server` is responsible for all computation and output rendering.
  - The `server` monitors *inputs* and other *reactive* values.
  - When inputs change, rendered outputs are created or updated.

Outputs and Renderers

Outputs are the way generated content produced by R is displayed in shiny. Examples include:

- `textOutput()`
- `plotOutput()`
- `tableOutput()`

You can use

```
help.search("Output", package = "shiny")
```

to find other output functions in shiny.

Outputs and Renderers

Each `*Output()` function has a corresponding `render*()` function. For example:

- `textOutput()` →`renderText()`

Outputs and Renderers

Each `*Output()` function has a corresponding `render*()` function. For example:

- `textOutput()` $\rightarrow$`renderText()`
- `plotOutput()` $\rightarrow$`renderPlot()`

Outputs and Renderers

Each `*Output()` function has a corresponding `render*()` function. For example:

- `textOutput()` →`renderText()`
- `plotOutput()` →`renderPlot()`
- `tableOutput()` →`renderTable()`

Example: render outputs

Start *examples/02_render_output.R*

Finished *examples/02_render_output_finished.R*

Exercise 2: Display number of storms by year.

Open the exercise file: *File →Open File →exercises/02_storms_by_year.R*

1. Add a table output element to the ui and a corresponding renderer to the server. The table should display the number of named storms in each year.

2. Add a plot output element to the ui, and a corresponding renderer to the server. The plot should display the number of named storms in each year.

Creating input elements

Inputs are form elements like check boxes, text fields, and sliders. Examples include:

- `textInput()`
- `selectInput()`
- `fileInput()`

You can use

```
help.search("Input", package = "shiny")
```

to find other input functions in shiny.

Accessing inputs

- Inputs are accessed in the *server* function via the *input* argument.
- Inputs are *reactive*, meaning that changes trigger updates.
- It is often helpful to `print` or use `str` to examine inputs; `str(reactiveValuesToList(input))` will show the current input names and values.

Example: create and use input

Start *examples/03_input_output.R*

Finished *examples/03_input_output_finished.R*

Exercise 3: Display storms for a user-selected year.

Open the exercise file: *File →Open File →exercises/03_storms_filtered.R*

1. Add a sliderInput element to the ui .
2. Modify the renderTable expression to filter the year displayed to the one selected by the user.

Interactive tables and graphs

Interactive tables and graphs

Javascript is the language of the web, and many of the most popular javascript libraries can be used directly from R.

Objectives:

- Discover available html widgets that can be used in shiny applications.
- Learn how html widgets interact with shiny.
- Practice using html widgets as inputs and outputs in shiny apps.

Interactive tables with DT

The DataTables javascript library can be used to create interactive tables directly from R. Features inlude:

- searching,
- pagination,
- sorting.

Interacting with tables updates `input`, enabling integration with Shiny.

See https://shiny.rstudio.com/articles/datatables.html for more.

Interactive graphs with plotly

Plotly is a robust javascript data visualization library with an excellent R package. Features include:

- easy converson of `ggplot` graphs,
- hover, click, pan and zoom
- support for plots, scatter plots, error bars, box plots, heatmaps and much more.

Plotly includes an `event_data` function for retrieving values produced by interacting with the plot. This enables deep integration with Shiny.

See https://plot.ly/r/shiny-coupled-events/ for more.

Interactive maps with leaflet

Leaflet is a popular javascript library for producing interactive maps.

Like DataTables, interacting with leaflet maps updates `input`, enabling interacting with shiny.

See http://rstudio.github.io/leaflet/shiny.html for more.

Example: html widgets DT and Plotly

Start *examples/04_htmlwidgets.R*

Finished *examples/04_htmlwidgets_finished.R*

Exercise 4: Interactive storms map

Open the exercise file: *File →Open File →exercises/04_interactive_storms_map.R*

1. Run the app and click on the blue markers. Examine the output in the console to determine the input corresponding to the row number of the clicked observation.

2. Use the slice function to select the row corresponding to the clicked marker from the storms data frame and return this row to the stormDetails output on line 65.

User interface (UI)

User interface (UI)

Objectives:

- Learn the high-level layout features provided by shiny.
- Learn how to include html tags in your shiny app.
- Explore available shiny themes.
- Practice creating apps using different layouts.

Page layouts

Our storms app is functional, but not much to look at.

Top-level page layout functions in shiny include:

- `fluidPage()`
- `navbarPage()`
- `fixedPage()`
- `fillPage()`
- `bootstrapPage()`

`fluidPage` is the most commonly used, and `navbarPage` is useful for more complex apps with many elements.

See https://shiny.rstudio.com/articles/layout-guide.html for more.

Layout functions

Other layout functions include:

- `fluidRow()` / `fixedRow()`
- `column()`,
- `sidebarLayout()`
- `splitLayout()`
- `verticalLayout()`

These are often used inside the page layout functions.

See https://shiny.rstudio.com/articles/layout-guide.html for more.

HTML tags

In addition to the high-level layout functions you can also use low-level functions to generate html tags.

Directly available tag functions include:

- `h1()` .. `h6()` (headers)
- `p()` (paragraph)
- `a()` (link)
- `br()` (line break)
- `div()` (division)

See `?builder` for more.

Shiny themes

Shiny uses bootstrap (https://getbootstrap.com/) under the hood, providing easy access to a range of themes.

To use other themes, install the `shinythemes` package and use the `theme` argument to your page layout function.

More information at https://rstudio.github.io/shinythemes/

Example: Page and element layout

Start *examples/05_layout_appearance.R*

Finished *examples/05_layout_appearance_finished.R*

Exercise 5: Spiff up the storms app

Open the exercise file: *File →Open File →exercises/05_storms_pretty.R*

1. Lay out this application using `navbarPage` and ‘tabPanel. See https://shiny.rstudio.com/articles/layout-guide.html for examples.
2. Use html tags (e.g., `h2()`, `p()`, `a()`) to add some descriptive text to your application.
3. Use the `shinythemes` package (install if needed) to change the theme used by your app. See https://rstudio.github.io/shinythemes/ for examples.

Where to go from here

Learning resources

A number of excellent tutorials and other resources are available, including:

- https://mastering-shiny.org/
- https://shiny.rstudio.com/tutorial/
- http://shiny.rstudio.com/articles/
- https://shiny.rstudio.com/images/shiny-cheatsheet.pdf
- https://www.linkedin.com/learning/building-data-apps-with-r-and-shiny-essential-training

References

- Institute for Quantitative Social Science: https://github.com/IQSS/