Introduction to Shiny

Fall 2022

October 24 2022

Shiny: Overview

- Shiny is an R package that provides a fairly high-level framework for creating interactive graphics
- Shiny web app allows us to build interactive dashboard that we will let Rstudio host for us with their servers
- Rstudio makes it easy to create and even upload these graphs to the web
- You can publish Shiny documents to the ShinyApps (https://shinyapps.io) hosted service

Interactive plots

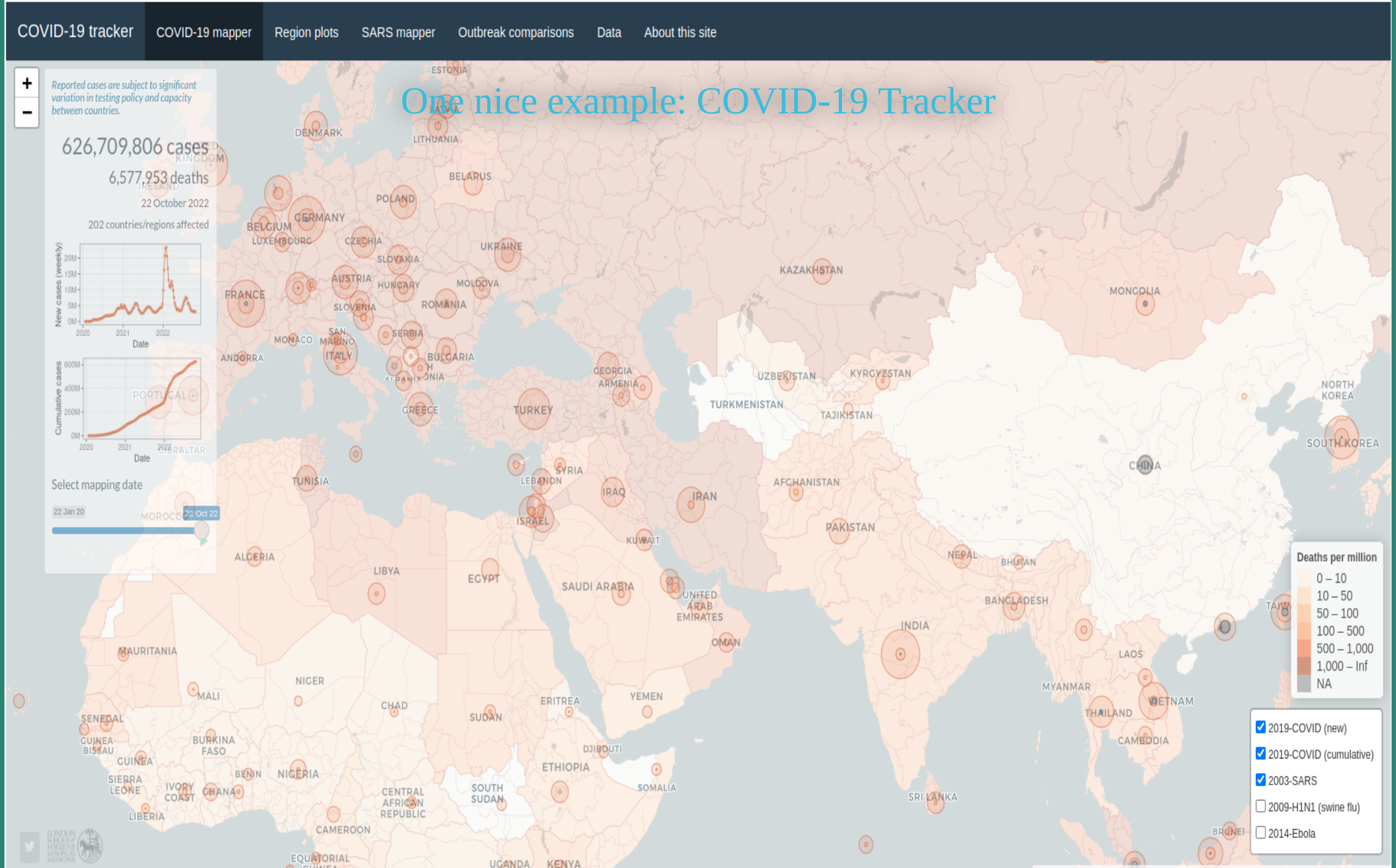
- Shiny uses reactive programming to automatically update outputs when inputs change
- Shiny applications have two components:
  - a user interface (UI) object
  - a server function
- These are passed as arguments to the `shinyApp` function that creates a Shiny app object from this UI/server pair

The YAML header of this R Markdown document has the line `runtime: shiny` so that RStudio understands this is a Shiny document.

Useful Resources

- RStudio Documentation
- Shiny Documents for further reading
- Learn Shiny with videos and written tutorials.
- Shiny Gallery with example demonstrations
- Mastering Shiny by Hadley Wickham.
- Shiny Cheatsheet

One nice example: COVID-19 Tracker



Shiny : : CHEAT SHEET

Building an App

A **Shiny** app is a web page (**ui**) connected to a computer running a live R session (**server**).

Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

Save your template as **app.R**. Keep your app in a directory along with optional extra files.

To generate the template, type **shinyapp** and press **Tab** in the RStudio IDE or go to **File > New Project > New Directory > Shiny Web Application**

In **ui** nest R functions to build an HTML interface

Tell the **server** how to render outputs and respond to inputs with R

```
# app.R
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)


server <- function(input, output, session) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}


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

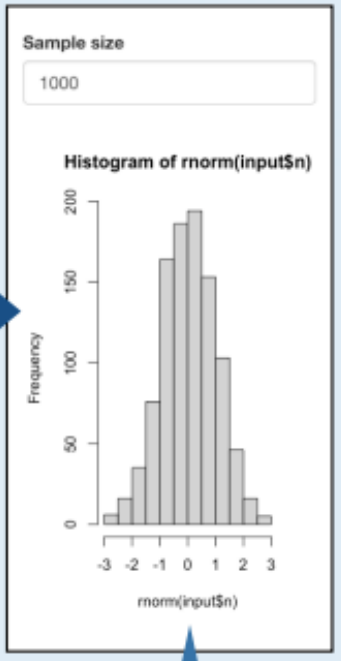
Customize the UI with **Layout Functions**

Add Inputs with ***Input()** functions

Add Outputs with ***Output()** functions

Wrap code in **render*()** functions before saving to output

Refer to UI inputs with **input$<id>** and outputs with **output$<id>**



Call **shinyApp()** to combine **ui** and **server** into an interactive app!

app-name
- app.R
- DESCRIPTION
- README
- R/
- www/

The directory name is the app name

(optional) used in showcase mode

(optional) directory of supplemental .R files that are sourced automatically, must be named "**R**"

(optional) directory of files to share with web browsers (images, CSS, .js, etc.), must be named "**www**"

Launch apps stored in a directory with **runApp(**<path to directory>**)**.

See annotated examples of Shiny apps by running **runExample(**<example name>**)**. Run **runExample()** with no arguments for a list of example names.

shinyapps.io ☰ http://shinyapps.io/ Help Account: deepbas

- Dashboard
- Applications »
- Account »
  - Profile
  - Tokens
  - Domains
  - Settings
  - Usage
  - Billing
  - Members

GETTING STARTED ×

Hi! You must be new here...

Thanks for trying out shinyapps.io! You'll need to install the `rsconnect` R package or `rsconnect-python` Python package to get started. These packages enable you to deploy and manage your Shiny applications directly from your local environment. To get started, fire up your favorite IDE, and follow the directions below.

Start with R | Start with Python

STEP 1 – INSTALL RSCONNECT

The `rsconnect` package can be installed directly from CRAN. To make sure you have the latest version run following code in your R console:

```
install.packages('rsconnect')
```

STEP 2 – AUTHORIZE ACCOUNT

The `rsconnect` package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Paste it into your R console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy Shiny for R applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='deepbas',
                          token='DF6C0142297452E0313D02AF1674AB03',
                          secret='<SECRET>')
```

Show secret

Copy to clipboard

In the future, you can manage your tokens from the Tokens page the settings menu.

STEP 3 – DEPLOY

Once the `rsconnect` package has been configured, you're ready to deploy your first application. If you haven't written any applications yet, you can also checkout the Getting Started Guide for instructions on how to deploy our demo application. Run the following code in your R console.

```
library(rsconnect)
    rsconnect::deployApp('path/to/your/app')
```

Tracking Covid in Minnesota

Pick a County

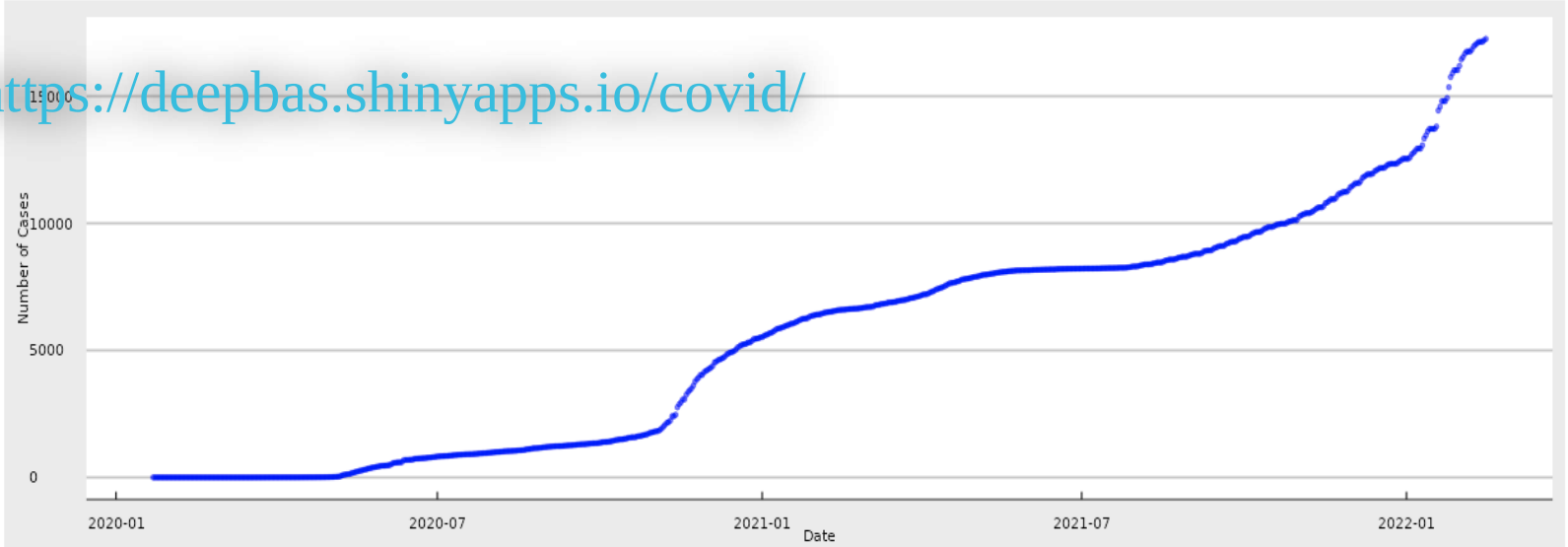
County

Rice

Date range

2020-01-22 to 2022-02-15

https://deepbas.shinyapps.io/covid/



Show 10 entries

Search:

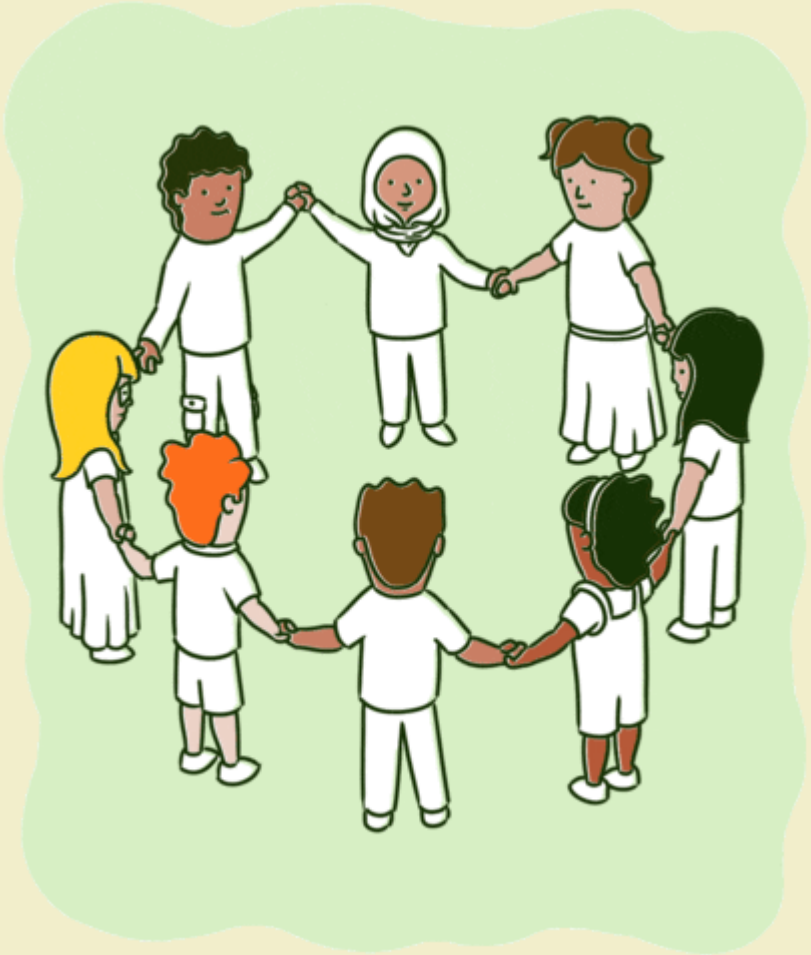
	Counties	dates	cases	month	year
1	Rice	2020-01-22	0	1	2020
2	Rice	2020-01-23	0	1	2020
3	Rice	2020-01-24	0	1	2020
4	Rice	2020-01-25	0	1	2020
5	Rice	2020-01-26	0	1	2020
6	Rice	2020-01-27	0	1	2020
7	Rice	2020-01-28	0	1	2020
8	Rice	2020-01-29	0	1	2020
9	Rice	2020-01-30	0	1	2020
10	Rice	2020-01-31	0	1	2020

Showing 1 to 10 of 756 entries

Previous 1 2 3 4 5 ... 76 Next

Group Activity 1

10:00



- Let's go over to class activity 18 .Rmd file and practice building a shiny app together
- Ask me questions