

A Brief Introduction to R-Shiny and Dashboards

Ryan Hafen & J. Hathaway

Agenda



1. Introduction to R-Shiny and Dashboards
2. Examples of R-Shiny
3. Making our first R-Shiny dashboard
4. A few rules for dashboards
5. How do I learn more?

Introduction to R-Shiny and Dashboards

:Dashboards:

Opinionated views for decision making that facilitate user input.



What is R-Shiny?

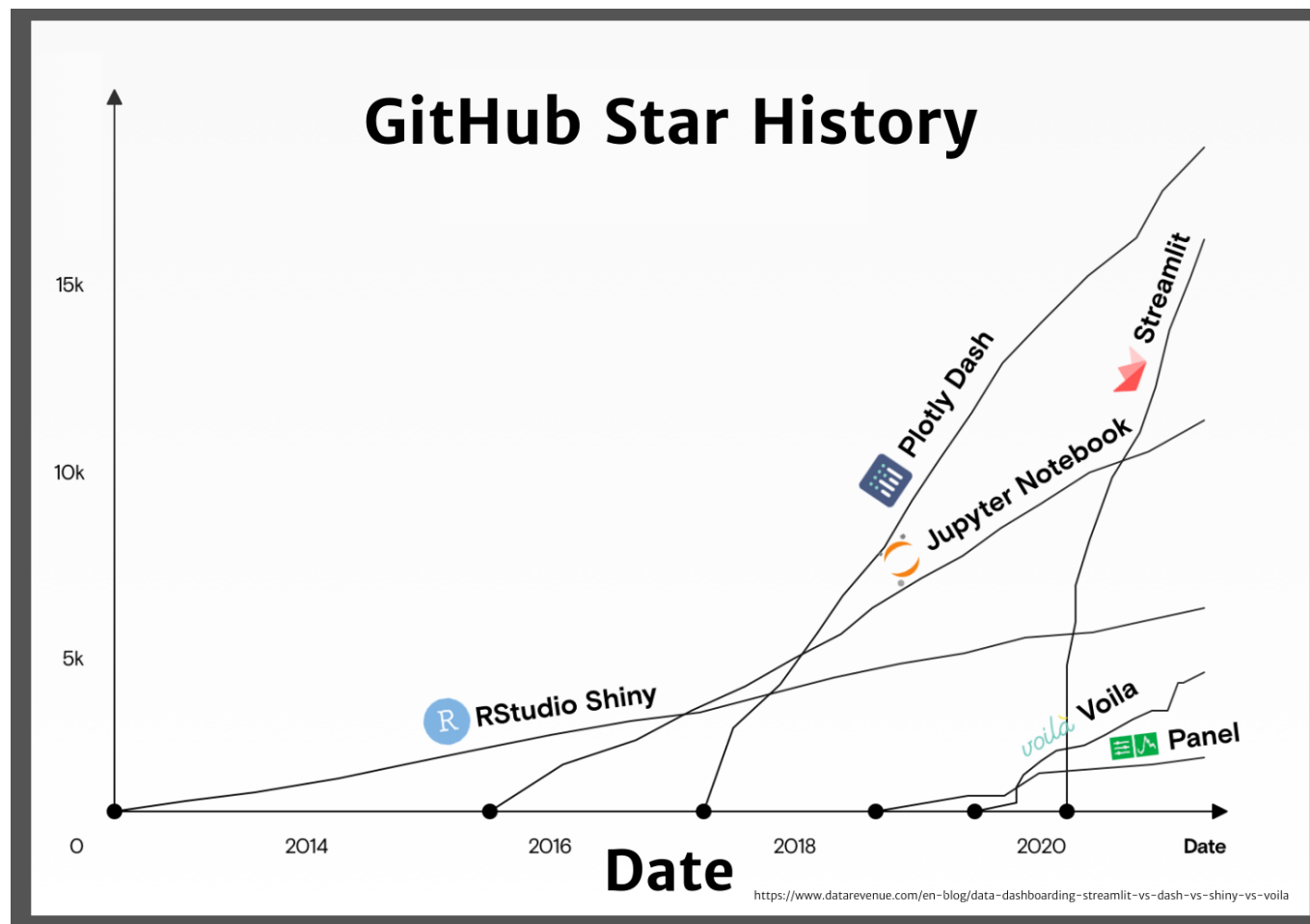
An interactive approach for narrating data stories using R.

Shiny is an R package that makes it easy to build interactive web apps straight from R.

- [RStudio Shiny website](https://www.rstudio.com/shiny/)



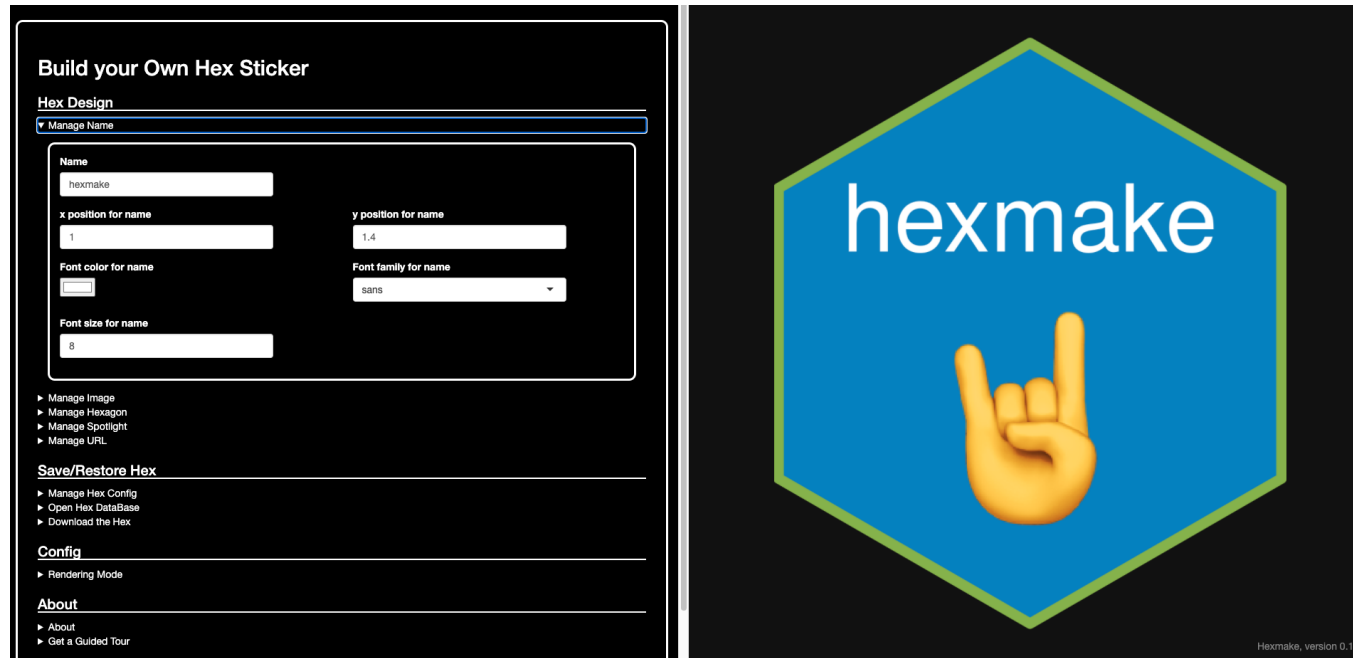
What are the alternatives to R-Shiny?



Examples of R-Shiny

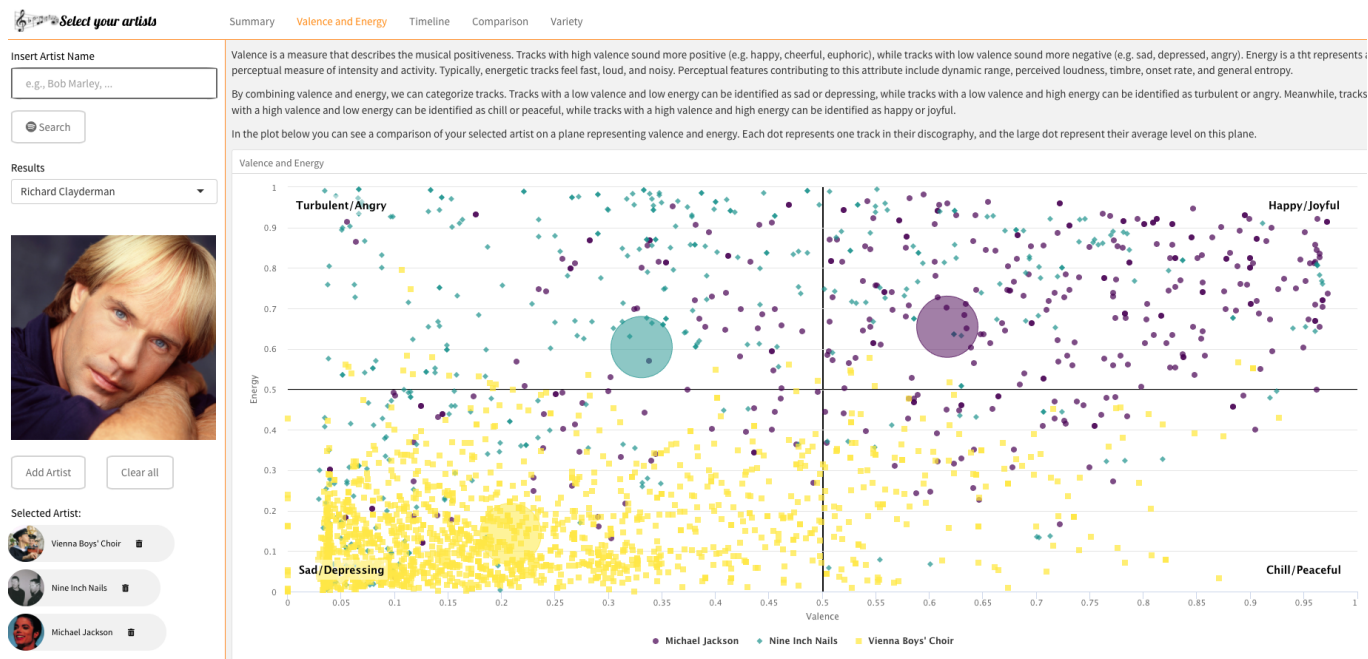
Build your Own Hex Sticker

- *The app's purpose is clear.*
- *Overwhelming inputs are hidden until they are needed.*
- [GitHub Repository](#)



Datify: Spotify Music

- *The search input invites use.*
- *Buttons are big, clear, and organized.*
- *Slow to refresh charts after API connection.*



Radiant: Interactive 'Software' for Modeling

- *Lot's of functionality through drop-down menus.*
- *General purpose application for modeling.*
- *Confusing entry for beginners.*
- [GitHub Repo](#)

The screenshot displays the Radiant software interface. The top navigation bar includes tabs for Data, Design, Basics, Model, Multivariate, and Report. The 'Multivariate' menu is open, showing options like Maps, (Dis)similarity, Attributes, Factor, Pre-factor, Cluster, Hierarchical, K-clustering, Conjoint, and Conjoint. The main workspace shows a dataset of diamond prices with columns: price, color, depth, table, x, y, z, and date. The dataset is titled 'Diamond prices' and contains 3,000 rows. The description states: 'A dataset containing the prices and other attributes of a sample of 3000 diamonds. The variables are as follows: Variables: price = price in US dollars (\$338-\$18,791), carat = weight of the diamond (0.2-3.00), clarity = a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, IF (best)), cut = quality of the cut (Fair, Good, Very Good, Premium, Ideal), color = diamond color, from J (worst) to D (best), depth = total depth percentage = 2 * max(depth, table) + 2 * avn * 100 / (carat * 2.75)'. The interface also includes a sidebar with options for loading and saving data, and a 'Show R-code' checkbox.

Making our first R-Shiny dashboard

Thanks to [shinyintro](#) by [Lisa Debruine](#)

- *We recommend her book for a smooth entrance into Shiny and ShinyDashboard.*
- *We will introduce dashboarding with Shiny using the ['shinydashboard'](#) package.*

Using the shinydashboard package

The shinydashboard package is a collection of functions that make it easy to create dashboards.

You can find the code for this example at our [RShinyDashboards Repository](#).

Our first R Shiny Dashboard

- **Our Goal:** Give our user the opportunity to see reported COVID-19 cases for selected states over a specified time window using the [New York Times COVID-19 data](#).



Starting with our chart (part 1)

- *Before we get into Shiny code, let's write the code for our key chart in the display.*

```
# load packages and format data for chart
library(ggplot2)
library(readr)
library(dplyr)
library(lubridate)

# Our data (updates daily)
url <- "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv"
nyt <- read_csv(url) %>%
  mutate(cases100k = cases / 1000)

# Our potential inputs
variable <- c("deaths", "cases100k")[2] # what is on our y axis.
daterange <- c(ymd("2020-01-01"), ymd("2022-12-31")) # filter for x-axis
states <- c("New York", "Texas", "Florida") # which states to plot.

# Create a filtered dataset for plotting.
plotdata <- nyt %>%
  filter(
    state %in% states,
    date >= daterange[1],
    date <= daterange[2])
```

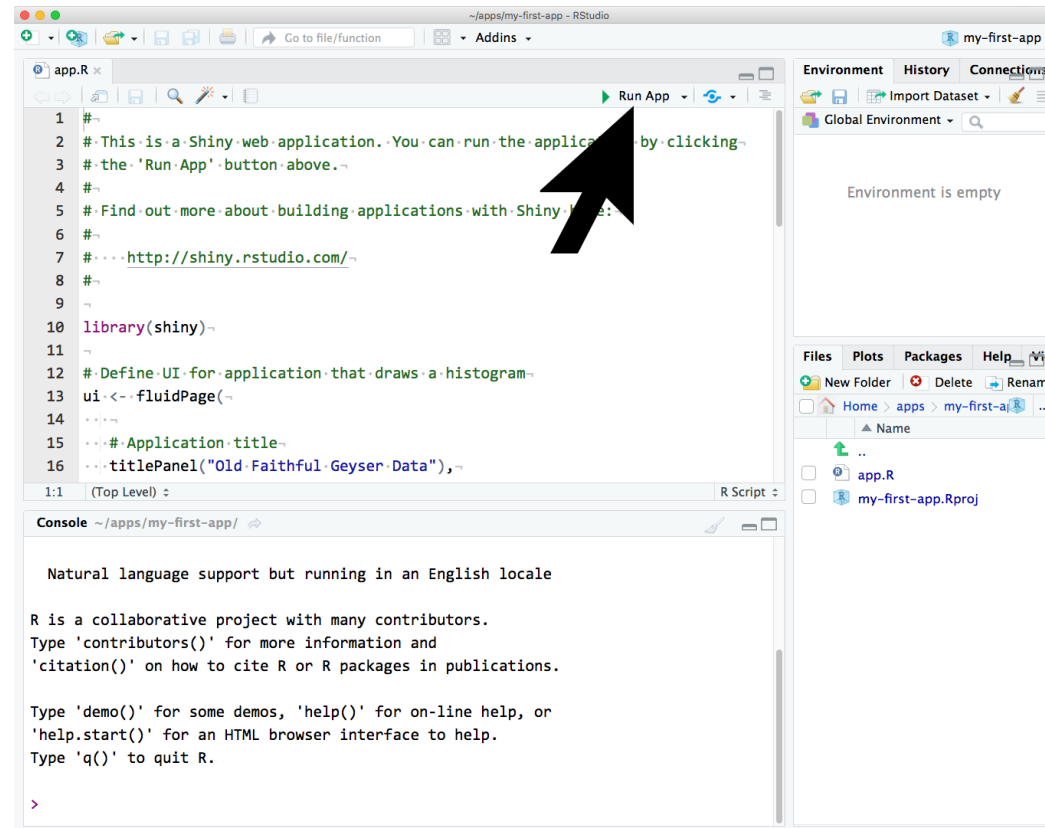
Starting with our chart (part 2)

- The plot code to use in a general R script. Download the [chart.R](#) script from the repository.

```
# create the plot
# Notice the use of .data[[variable]] to select the column to place on the y-axis
ggplot(data = plotdata, aes(x = date, y = .data[[variable]], color = state)) +
  geom_point(size = .5, alpha = .8) +
  geom_line() +
  theme_bw() +
  guides(color = guide_legend(
    override.aes = list(lty = NA, size = 5, shape = 15))) +
  labs(
    title = paste0("Progression of the pandemic from ",
      daterange[1], " to ", daterange[2]),
    x = "Date",
    y = ifelse(variable == "deaths", "Deaths", "Cases (1,000)",
    color = "State"
  )
)
```


Using R-Studio for app development

- *Start R-Studio and create a new script `app.R` in a new project or folder. Once we have our script built we will click on Run App.*



The Shiny app basics

- *Every Shiny app has three key elements: ui, server and the shinyApp(ui, server) call. When using shinydashboard our ui is a little different.*

```
header <- dashboardHeader(CODE FOR THE HEADER)
sidebar <- dashboardSidebar(CODE FOR THE INPUTS)
body <- dashboardBody(CODE FOR THE RESULTS)

ui <- dashboardPage(
  skin = 'purple',
  header,
  sidebar,
  body
)
```

```
server <- function(input, output) {
  THE CODE THAT CREATE CHARTS, TABLES, AND TEXT FROM INPUTS
}
```

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

Our ['app.R'](#) code for the dashboard is in the app folder of our [repository](#)

Building the interface: *Copy this code and paste it into app.R*

```
# Setup ----
library(shiny); library(shinydashboard); library(readr); library(dplyr); library(ggplot2)
# Define UI ----
header <- dashboardHeader(title = "COVID-19 NYT Data Dashboard", titleWidth = 400)

sidebar <- dashboardSidebar(width = 400,
  actionButton('load', label = "Import New York Times data"),
  radioButtons('variable', label = "Cases or Deaths?",
    choices = c("Number of Deaths" = "deaths", "Number of Cases" = "cases1000")),
  selectInput("state", "Which states?", state.name, multiple = TRUE, selected = "New York" ),
  dateRangeInput("daterange", "Date range:", start = "2020-01-01", end = "2022-12-31"),
  actionButton('makeplot', label = "Explore Visualization")
)

body <- dashboardBody(
  box(
    title = "Pandemic time-series (USA)", solidHeader = TRUE, width = 16, collapsible = TRUE,
    plotOutput("timeseries", height = 500, width = 'auto')
  ),
  box(title = "Total reported cases (USA)", solidHeader = TRUE, width = 6,
    div(style = "font-size:50px", textOutput("casetotal"))),
  box(title = "Total reported deaths (USA)", solidHeader = TRUE, width = 6,
    div(style = "font-size:50px", textOutput("deathtotal")))
)

# Run the application ----
ui <- dashboardPage(skin = "purple", header, sidebar, body)
server <- function(input, output) {}
shinyApp(ui, server)
```

Adding the body elements within server (Data & Totals).

- Include these [reactiveVal\(\)](#) assignments before the server function.

```
nyt <- reactiveVal()  
plotdata <- reactiveVal()
```

- Now include our data import event within the `server <- function(input, output) { }`.
- We can Reload App to see if the 'Import New York Times data' button works.

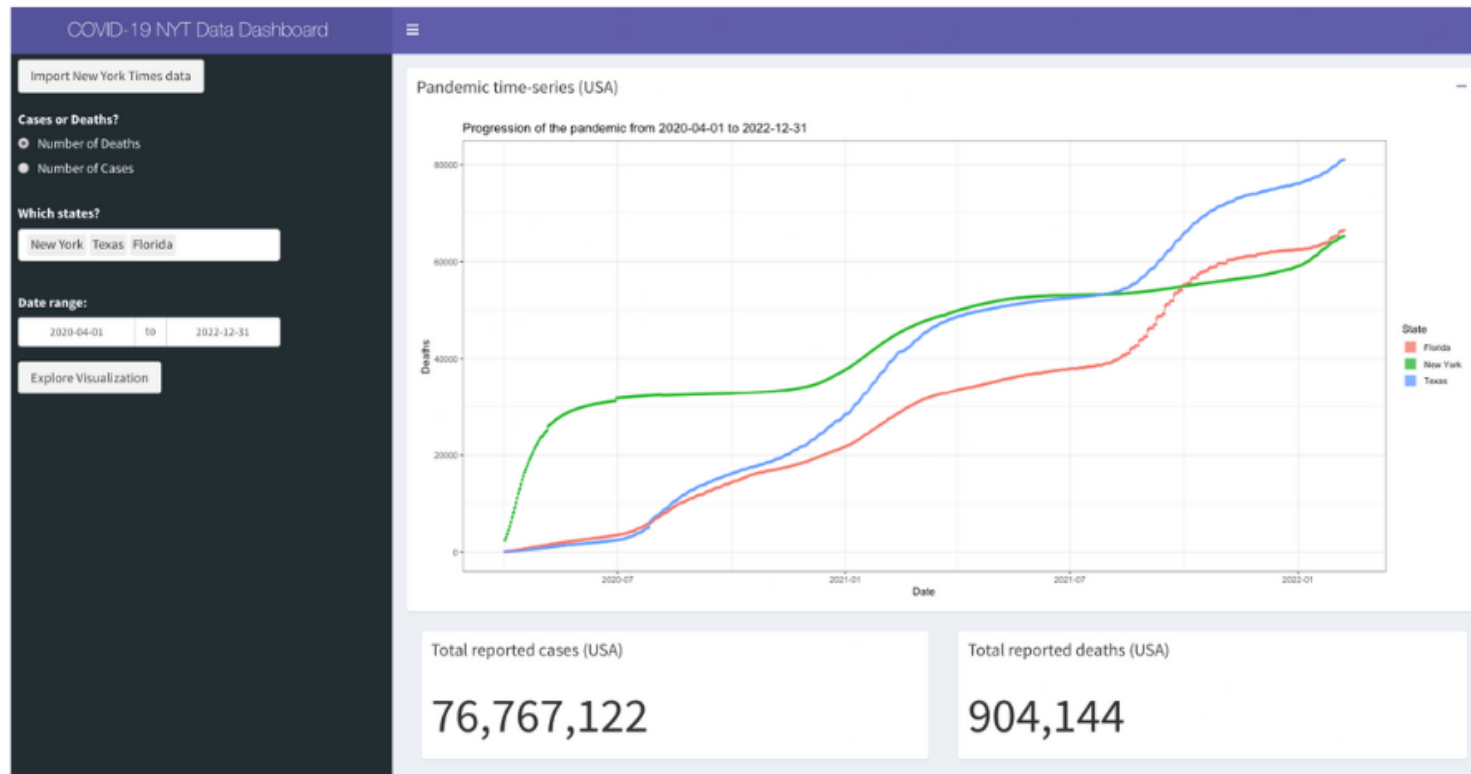
```
# Data ingestion and total number calculations. Set to work after `input$load` occurs from clicking on Import Data.  
observeEvent(input$load, {  
  url <- "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv"  
  fulldata <- readr::read_csv(url) %>%  
    mutate(cases100k = cases/1000)  
  
  nyt(fulldata) # This reactive item now is composed of the NYT COVID-19 data.  
  # assigning them into the output object allows them to be called by the `ui`.  
  output$deathtotal <- renderText(  
    fulldata %>% filter(date == max(date)) %>%  
    pull(deaths) %>% sum() %>% format(big.mark = ",")  
  
  output$casetotal <- renderText(  
    fulldata %>% filter(date == max(date)) %>%  
    pull(cases) %>% sum() %>% format(big.mark = ",")  
  })  
})
```

Adding the body elements within server (Chart).

```
observeEvent(input$makeplot, {  
  # notice the use of `input$` to pull the user input values into the function.  
  # if statement stops the outputs from being created if the user hasn't clicked import  
  if (length(nyt()) != 0) {  
    newdat <- nyt() %>%  
      filter(state %in% input$state, date >= input$daterange[1], date <= input$daterange[2])  
  
    plotdata(newdat) # This reactive item now has is composed of the filtered data.  
    # notice the use of plotdata() as the data object to signal a reactive dataset.  
    output$timeseries <- renderPlot({  
      ggplot(data = plotdata(), aes(date, .data[[isolate(input$variable)]], color = state)) +  
        geom_point(size = .8, alpha = .8) +  
        geom_line() +  
        theme_bw() +  
        labs(  
          title = paste0("Progression of the pandemic from ",  
            input$daterange[1], " to ", input$daterange[2]),  
          x = "Date",  
          y = ifelse(isolate(input$variable) == "deaths", "Deaths", "Cases (1,000)",  
            color = "State") +  
          guides(color = guide_legend(override.aes = list(lty = NA, size = 5, shape = 15)))  
        })  
    })  
  }  
})
```

Our final dashboard

- You can see the complete script at [app/app.R](#)



A few rules for dashboards

“Know^{the}
user,
they are not programmers

Define your production goals

- How often will it be used?
- How reliable does it need to be?
- What is the impact if it is inaccurate?



Design with a purpose

The purpose of design is much more closely linked to strategy than aesthetics. Design is the process of intentionally creating something while simultaneously considering it's objective, function, economics, AND aesthetics.

Kim Schaefer

How do I learn more?

Online Reading Material

- [Mastering Shiny](#)
- [Learn Shiny](#)
- [Get Started w/ shinydashboard](#)
- [Use shinydashboard](#)
- [Engineering Production-Grade Shiny Apps](#)
- [shinyjs](#)
- [shinydashboards](#)
- [Building Web Apps with R Shiny](#)
- [stat545 Shiny Tutorial](#)
- [R markdown: The Definitive Guide - Shiny](#)
- [YouTube: A Gentle Introduction to Creating R Shiny Webb Apps](#)
- [YouTube: Dynamic Dashboards with Shiny](#)

Short Courses

- [Building Web Applications with Shiny in R](#)
- [Shiny Fundamentals with R](#)
- [Building Data Apps with R and Shiny: Essential Training](#)
- [Creating Interactive Presentations with Shiny and R](#)
- [Interactive Visualization with R](#)

Shinyverse of R packages

There are so many packages available. [awesome-shiny](#) has an extensive list. Use the below list to start your journey.

- [shinydashboard](#)
- [shinydashboardPlus](#)
- [Shiny Themes](#)
- [shinymanager](#)
- [shiny.semantic](#)
- [shiny.react](#)
- [shiny.fluent](#)
- [shiny_sense](#)

Thanks