Introduction to Shiny Apps

bmRn CSM: Building applications with R

Martin Frigaard

2020-12-13

`Shiny` = Web Applications with R

Load the packages

```r
install.packages(c("tidyverse", "inspectdf",
                   "shiny", "flexdashboard",
                   "reactable", "gtrendsR"))
library(tidyverse)
library(shiny)
library(inspectdf)
library(flexdashboard)
library(reactable)
library(gtrendsR)
```

Materials

Slides

https://mjfrigaard.github.io/intro-to-shiny/Index.html

Exercises

coming soon!

RStudio Project

https://rstudio.cloud/project/2021718

Outline

Shiny app anatomy

- ui
- server
- run

User Interface (UI)

- build layout (`fluidPage()`)
- define `inputs`

Server

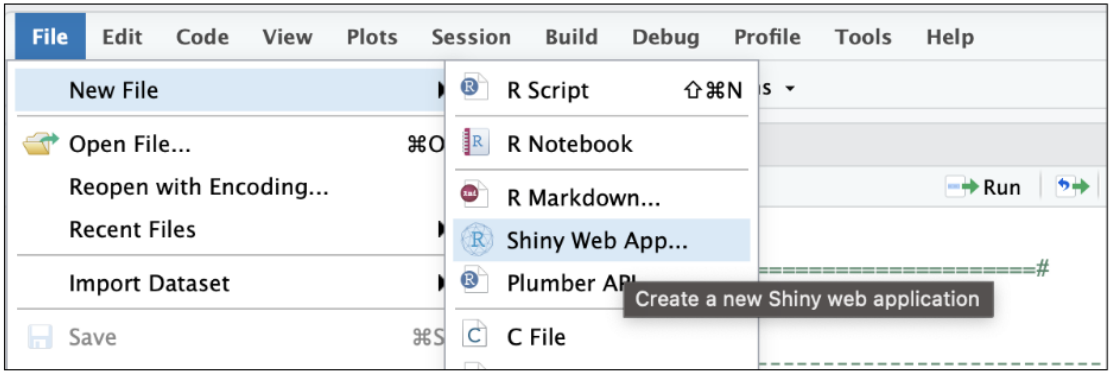
- build `reactive()`
- use `inputs` with `output`

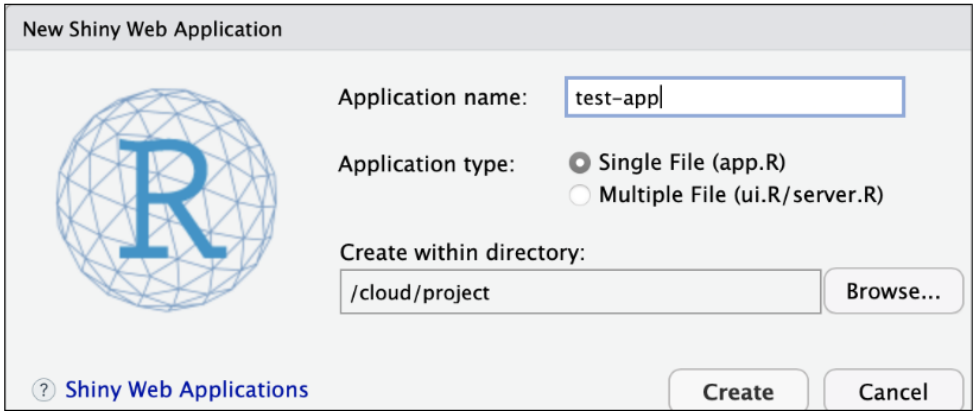
`flexdashboard`

- Uses R Markdown
- Convert from static to shiny app

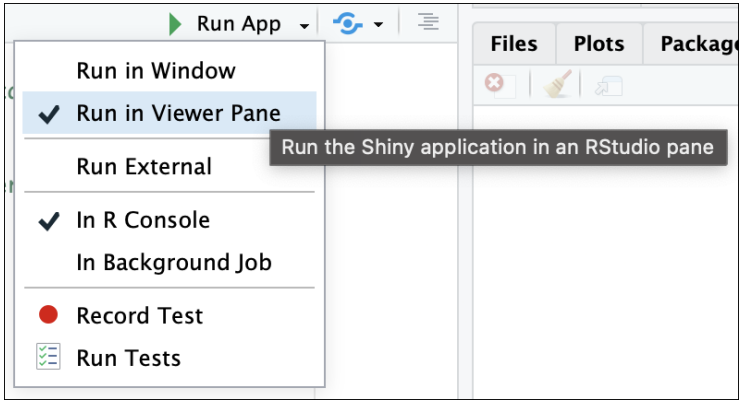
Create a new app

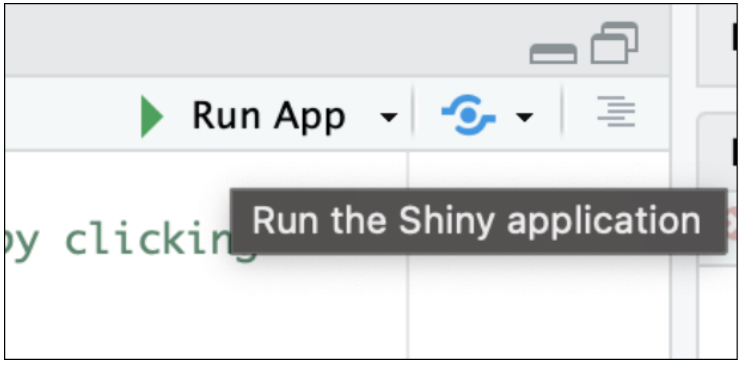
File > New File > Shiny Web App





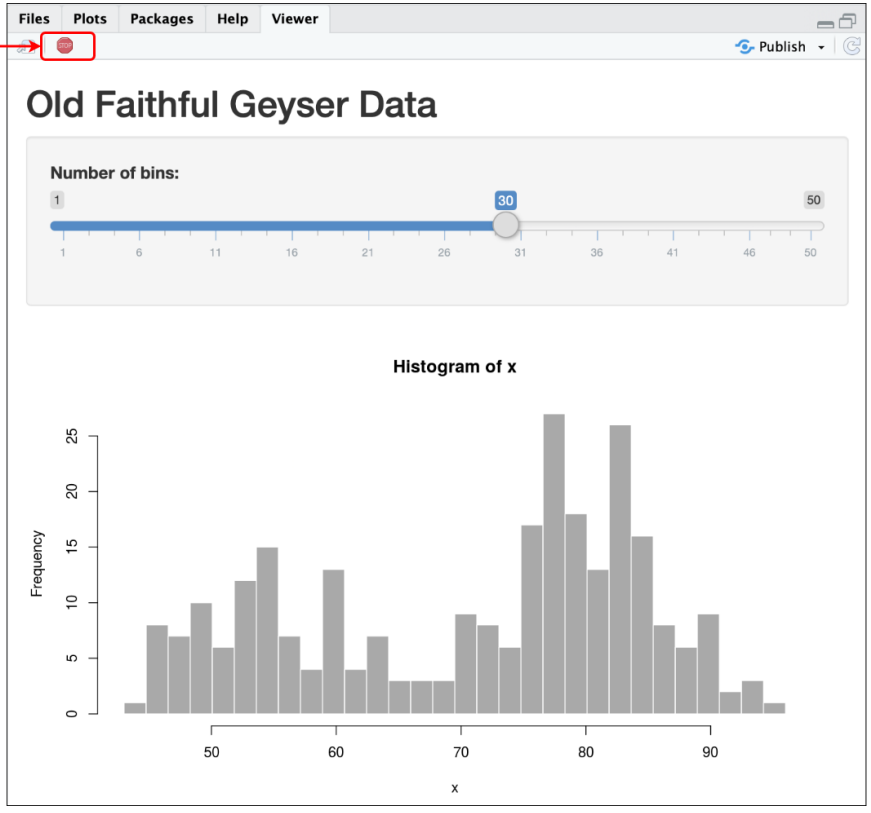
Run app


Run App
Run in Window
Run in Viewer Pane
Run External
In R Console
In Background Job
Record Test
Run Tests
Run the Shiny application in an RStudio pane
Files
Plots
Packag


Run App
Run the Shiny application

Stop running app

Click to stop running



Anatomy of a shiny app

Shiny app internals

User interface (UI)

```
ui <- fluidPage()
```

Server

```
ui <- fluidPage()
server <- function(input, output) {}
```

Run

```
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

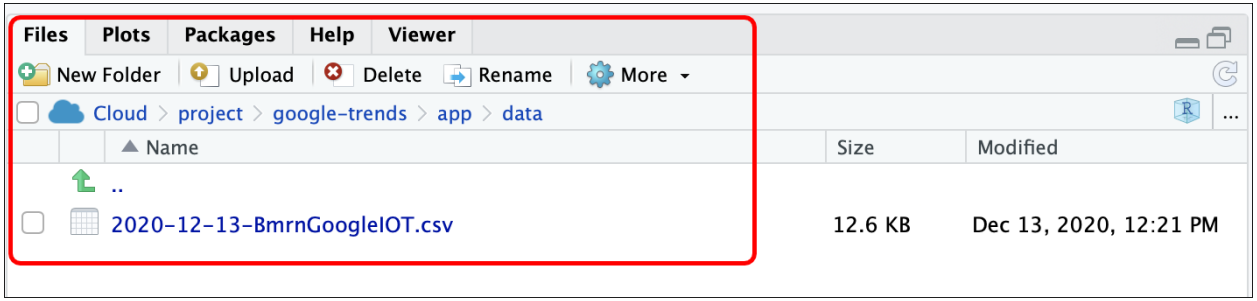
Example shiny app

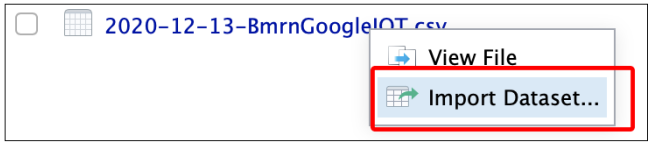
Open the app.R in `google-trends/`

```
google-trends/
            └── app
                ├── app.R # application
                └── data
                    └── 2020-12-13-BmrnGoogleIOT.csv # data
```

`google-trends/` data input

Locate the .csv file in `google-trends/app/data`





google-trends/ data input

Import the .csv file in

Import Text Data

File/URL:

/cloud/project/google-trends/app/data/2020-12-13-BmrnGoogleIOT.csv

Update

Data Preview:

date (double)	hits (double)	keyword (character)	geo (character)	time (character)	gprop (character)	category (double)	Location (character)
2019-12-15	5	BioMarin	US	today 12-m	web	0	United States
2019-12-22	3	BioMarin	US	today 12-m	web	0	United States
2019-12-29	5	BioMarin	US	today 12-m	web	0	United States
2020-01-05	11	BioMarin	US	today 12-m	web	0	United States
2020-01-12	8	BioMarin	US	today 12-m	web	0	United States
2020-01-19	9	BioMarin	US	today 12-m	web	0	United States
2020-01-26	10	BioMarin	US	today 12-m	web	0	United States
2020-02-02	3	BioMarin	US	today 12-m	web	0	United States
2020-02-09	9	BioMarin	US	today 12-m	web	0	United States
2020-02-16	11	BioMarin	US	today 12-m	web	0	United States
2020-02-23	14	BioMarin	US	today 12-m	web	0	United States
2020-03-01	10	BioMarin	US	today 12-m	web	0	United States

Previewing first 50 entries.

Import Options:

Name: X2020_12_13_BmrnGoogleIOT

Skip: 0

First Row as Names

Trim Spaces

Open Data Viewer

Delimiter: Comma

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```
library(readr)
X2020_12_13_BmrnGoogleIOT <- read_csv("data/2020-12-13
-BmrnGoogleIOT.csv")
View(X2020_12_13_BmrnGoogleIOT)
```

Reading rectangular data using readr

Import

Cancel

`google-trends/` data input

View the imported .csv file

app.R | X2020_12_13_BmrnGoogleIOT

Filter

	date	hits	keyword	geo	time	gprop	category	Location
1	2019-12-15	5	BioMarin	US	today 12-m	web	0	United States
2	2019-12-22	3	BioMarin	US	today 12-m	web	0	United States
3	2019-12-29	5	BioMarin	US	today 12-m	web	0	United States
4	2020-01-05	11	BioMarin	US	today 12-m	web	0	United States
5	2020-01-12	8	BioMarin	US	today 12-m	web	0	United States
6	2020-01-19	9	BioMarin	US	today 12-m	web	0	United States
7	2020-01-26	10	BioMarin	US	today 12-m	web	0	United States
8	2020-02-02	3	BioMarin	US	today 12-m	web	0	United States
9	2020-02-09	9	BioMarin	US	today 12-m	web	0	United States
10	2020-02-16	11	BioMarin	US	today 12-m	web	0	United States
11	2020-02-23	14	BioMarin	US	today 12-m	web	0	United States
12	2020-03-01	10	BioMarin	US	today 12-m	web	0	United States

Showing 1 to 13 of 208 entries, 8 total columns

The User Interface (ui)

UI: `fluidPage()`

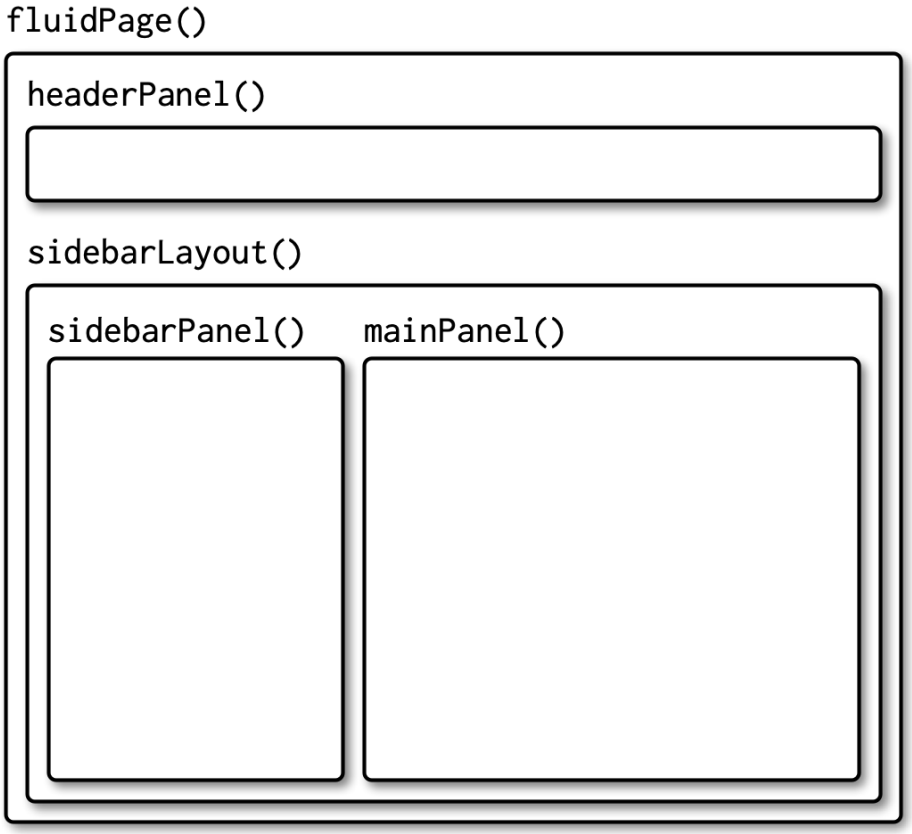


Image by Hadley Wickham

fluidPage: `sidebarLayout()`

This consists of a `sidebarPanel()` and `mainPanel()`

```
ui <- fluidPage(
            titlePanel(title = "BioMarin Google Trends"),
                sidebarLayout(
                    sidebarPanel(

                    ),
                    mainPanel(

                    )
                )
            )
```

fluidPage: `sidebarPanel()`

The `selectInput()` and `dateRangeInput()` are in the `sidebarLayout()`

```
sidebarLayout(
    sidebarPanel(
      # Select trend term to plot
      selectInput(inputId = "key",
                  label = strong("Trend Term"),
                  choices = unique(BmrnGoogleIOT$keyword),
                  selected = "BioMarin"),

      # Select date range to be plotted
      dateRangeInput(inputId = "date", strong("Date range"),
                     start = "2019-12-15", end = "2020-12-06",
                     min = "2019-12-15", max = "2020-12-06"),
    ), # note the comma!
```

fluidPage: `sidebarPanel()`

This consists of a `sidebarPanel()` and `mainPanel()`

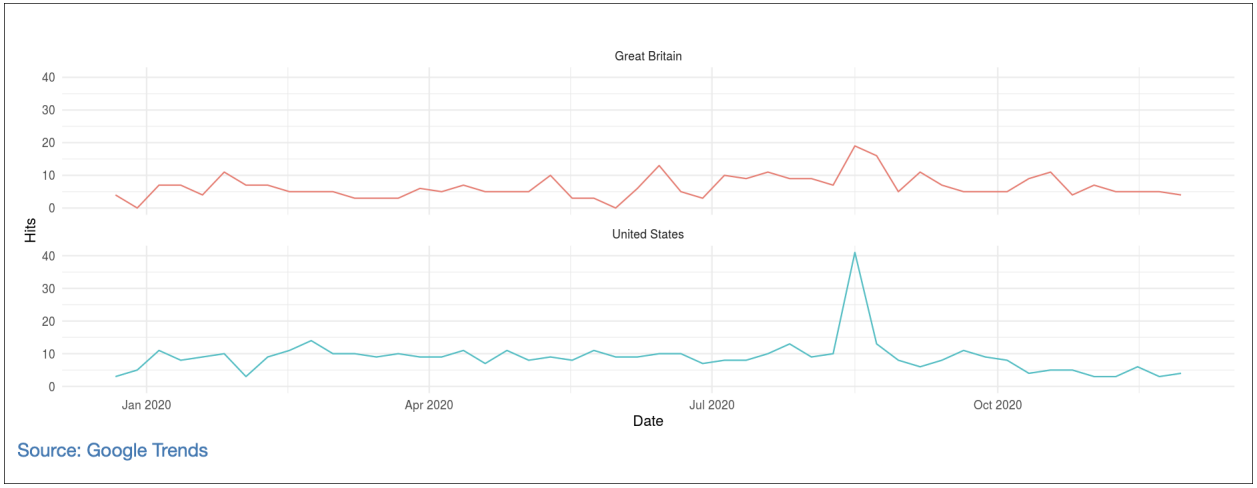
fluidPage: `mainPanel()`

The `plotOutput()` and text (`tags`) output will going into the `mainPanel()`

```
    sidebarLayout(
), # note the comma!
      mainPanel(
        plotOutput(outputId = "lineplot",
                   height = "300px"),
        tags$a(href = "https://www.google.com/",
               "Source: Google Trends",
               target = "_blank"))
      )
    )
```

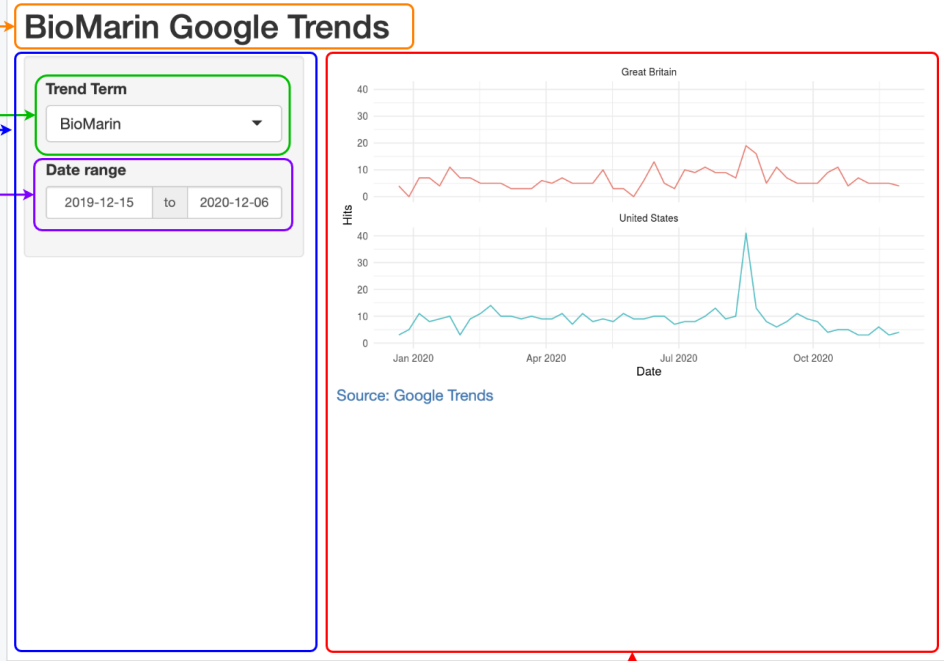
fluidPage: `mainPanel()`

The `plotOutput()` and text (`tags`) output will going into the `mainPanel()`



UI layout

```r
# Load data ---------------------------------------------------------------
BmrnGoogleIOT <- read_csv("data/2020-12-13-BmrnGoogleIOT.csv")


# Define UI ---------------------------------------------------------------
ui <- fluidPage(
  titlePanel(title = "BioMarin Google Trends"),

  sidebarLayout(
    sidebarPanel(

      # Select trend term to plot
      selectInput(inputId = "key",
                  label = strong("Trend Term"),
                  choices = unique(BmrnGoogleIOT$keyword),
                  selected = "BioMarin"),

      # Select date range to be plotted
      dateRangeInput("date", strong("Date range"),
                     start = "2019-12-15", end = "2020-12-06",
                     min = "2019-12-15", max = "2020-12-06"),
    ),

    # Output: Description, lineplot, and reference
    mainPanel(
      plotOutput(outputId = "lineplot",
                 height = "300px"),

      tags$a(href = "https://www.google.com/",
             "Source: Google Trends",
             target = "_blank")
    )
  )
)
```



The Server

`server`

The `server` is comprised of the `input` & `output`

```
function(input, output) {
  # data input
  # plot output
}
```

Elements from the `ui` (`inputId`, `outputId`) get passed to the `server`

Build `reactive` dataset (1)

Require `date` variable `input` from from `dateRangeInput()`:

```
selected_trends <- reactive({
    req(input$date)
```

Print errors if `date`s from `dateRangeInput()` are selected incorrectly

```
selected_trends <- reactive({
    req(input$date)
    validate(need(!is.na(input$date[1]) & !is.na(input$date[2]),
        "Error: Please provide both a start and an end date."))
    validate(need(input$date[1] < input$date[2],
         "Error: Start date should be earlier than end date."))
```

Build `reactive` dataset (2)

Filter these data by the `key` from the `selectInput()` and the `date` from `dateRangeInput()`

```
function(input, output) {
  selected_trends <- reactive({
    #....
    #....
    BmrnGoogleIOT %>%
      filter(keyword == input$key,
        date > as.POSIXct(input$date[1]) &
          date < as.POSIXct(input$date[2]))
  })
}
```

Build `reactive` dataset (3)

1) Use `reactive({})` to build data

```
selected_trends <- reactive({
    # inputs from ui are passed are used to filer .csv file
})
```

2) Call `selected_trends()` in `server`

Now whenever we need to use the dataset, we can refer to it using **`selected_trends()`**

```
selected_trends() %>%
  ggplot(aes(x = date, y = hits))
```

Build plot with `renderPlot({})` (1)

- Use `output$lineplot`...
  - which matches `plotOutput(outputId = "lineplot")`

```
output$lineplot <- renderPlot({
```

Each `outputId` in the `ui` can be used in the `server` with `output`.

Build plot with `renderPlot({})` (2)

Use **`selected_trends()`** (the reactive dataset) to build the `ggplot2` object:

```
    selected_trends() %>%
      ggplot(aes(x = date, y = hits,
                 color = Location, group = keyword)) +
      geom_line(aes(group = Location),
                show.legend = FALSE) +
      facet_wrap(. ~ Location, nrow = 2) +
      theme_minimal() +
      labs(x = "Date", y = "Hits")
  })
```

Server layout

```r
)

# Define server function ----------------------------------------
server <- function(input, output) {
  # filter data
  selected_trends <- reactive({
    req(input$date)
    validate(need(!is.na(input$date[1]) & !is.na(input$date[2]),
            "Error: Please provide both a start and an end date."))
    validate(need(input$date[1] < input$date[2],
             "Error: Start date should be earlier than end date."))
    BmrnGoogleIOT %>%
      filter(keyword == input$key,
        date > as.POSIXct(input$date[1]) &
          date < as.POSIXct(input$date[2]
          ))
  })

  output$lineplot <- renderPlot({
    selected_trends() %>%
      ggplot(aes(x = date, y = hits,
                 color = Location,
                 group = keyword)) +
      geom_line(aes(group = Location),
                show.legend = FALSE) +
      facet_wrap(. ~ Location,
                 nrow = 2) +
      theme_minimal() +
      labs(x = "Date",
           y = "Hits")
  })
}
# Create the Shiny app object -----------------------------------
shinyApp(ui = ui, server = server)
```

Run the app!

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

Define ui with fluidPage()

```
ui <- fluidPage(
  titlePanel(title = "BioMarin Google Trends"),
  sidebarLayout(
    sidebarPanel(
      # Select trend term to plot
      selectInput(inputId = "key",
                  label = strong("Trend Term"),
                  choices = unique(BmrnGoogleIOT$keyword),
                  selected = "BioMarin"),
      # Select date range to be plotted
      dateRangeInput(inputId = "date", strong("Date range"),
                     start = "2019-12-15", end = "2020-12-06",
                     min = "2019-12-15", max = "2020-12-06"),
    ),
    mainPanel(
      plotOutput(outputId = "lineplot",
                 height = "300px"),

      tags$a(href = "https://www.google.com/",
             "Source: Google Trends",
             target = "_blank")
    )
  )
)
```

Define server

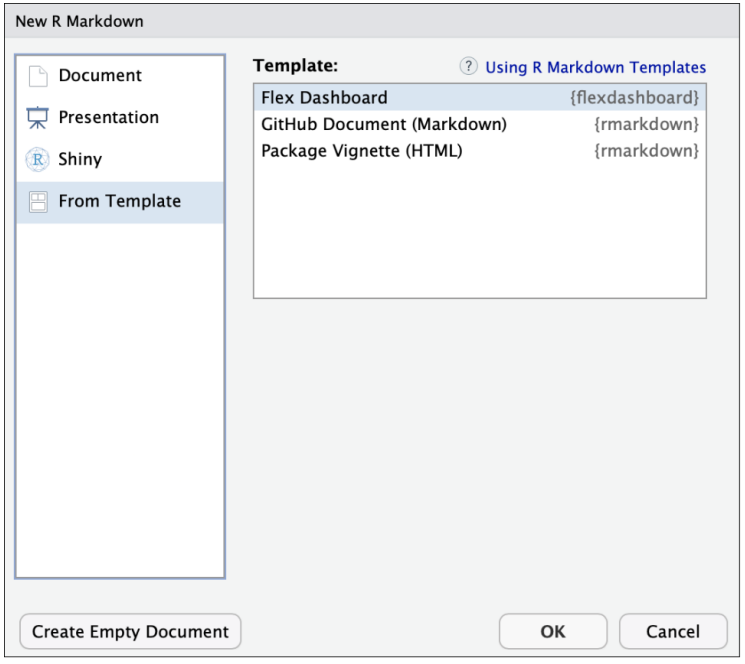
```
server <- function(input, output) {
  # filter data
  selected_trends <- reactive({
    req(input$date)
    validate(need(!is.na(input$date[1]) & !is.na(input$date[2]),
             "Error: Please provide both a start and an end date."))
    validate(need(input$date[1] < input$date[2],
              "Error: Start date should be earlier than end date."))
    BmrnGoogleIOT %>%
      filter(keyword == input$key,
        date > as.POSIXct(input$date[1]) &
          date < as.POSIXct(input$date[2]
          ))
  })

  output$lineplot <- renderPlot({
    selected_trends() %>%
      ggplot(aes(x = date, y = hits,
                 color = Location, group = keyword)) +
      geom_line(aes(group = Location),
                show.legend = FALSE) +
      facet_wrap(. ~ Location, nrow = 2) +
      theme_minimal() +
      labs(x = "Date", y = "Hits")
  })
}
```

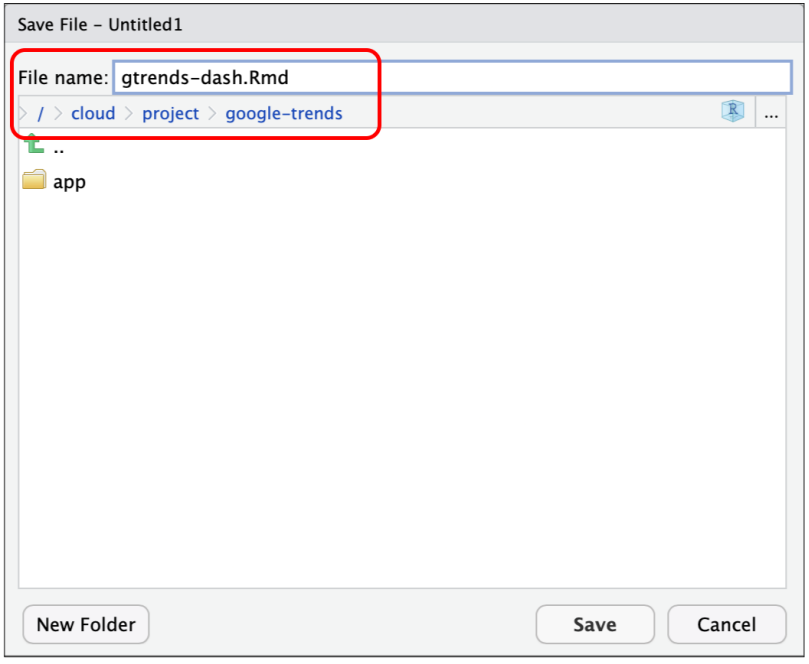
shinyApp(ui = ui, server = server)

`flexdashboard` and `shiny`

Create new `flexdashboard` file

File > New File > R Markdown





Add `runtime: shiny` to YAML

```yaml
---
title: "gtrendsR BioMarin Dashboard"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill

runtime: shiny
---
```

Add `setup` chunk

```{r setup, include=FALSE}
library(flexdashboard)

# Load packages ----------------------------------------------------
library(shiny)
library(tidyverse)

# Load data --------------------------------------------------------
BmrnGoogleIOT <- read_csv("app/data/2020-12-13-BmrnGoogleIOT.csv")
```

Define `inputs`

```{r define-inputs}
selectInput(inputId = "key",
            label = strong("Trend Term"),
            choices = unique(BmrnGoogleIOT$keyword),
            selected = "BioMarin")
dateRangeInput(inputId = "date", strong("Date range"),
               start = "2019-12-15", end = "2020-12-06",
               min = "2019-12-15", max = "2020-12-06")
```

Define `reactive`

```{r define-reactive}
selected_trends <- reactive({
    req(input$date)
    validate(need(!is.na(input$date[1]) & !is.na(input$date[2]),
            "Error: Please provide both a start and an end date."))
    validate(need(input$date[1] < input$date[2],
             "Error: Start date should be earlier than end date."))
    BmrnGoogleIOT %>%
      filter(keyword == input$key,
        date > as.POSIXct(input$date[1]) &
          date < as.POSIXct(input$date[2]
          ))
  })
```

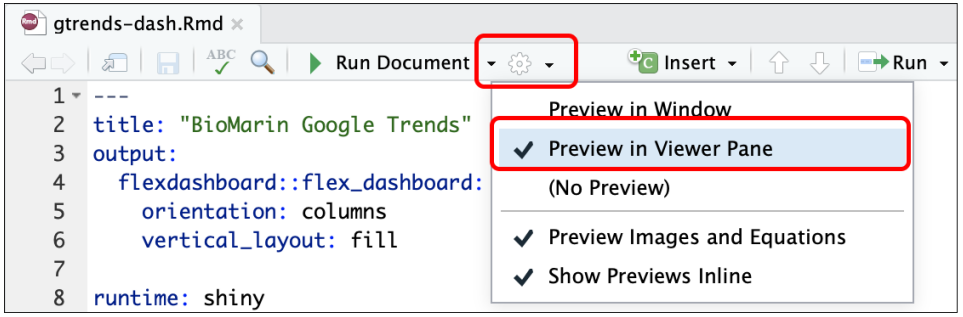
Build plot with `renderPlot()`

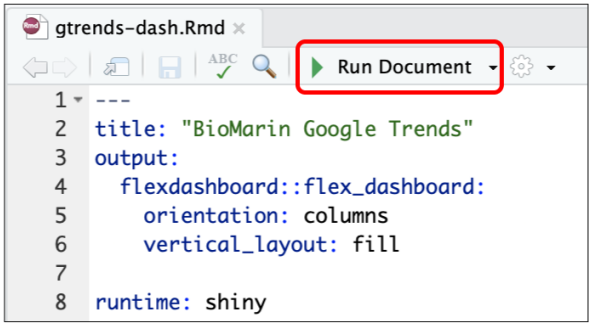
```
Column
-------------------------------------

### Google Trends

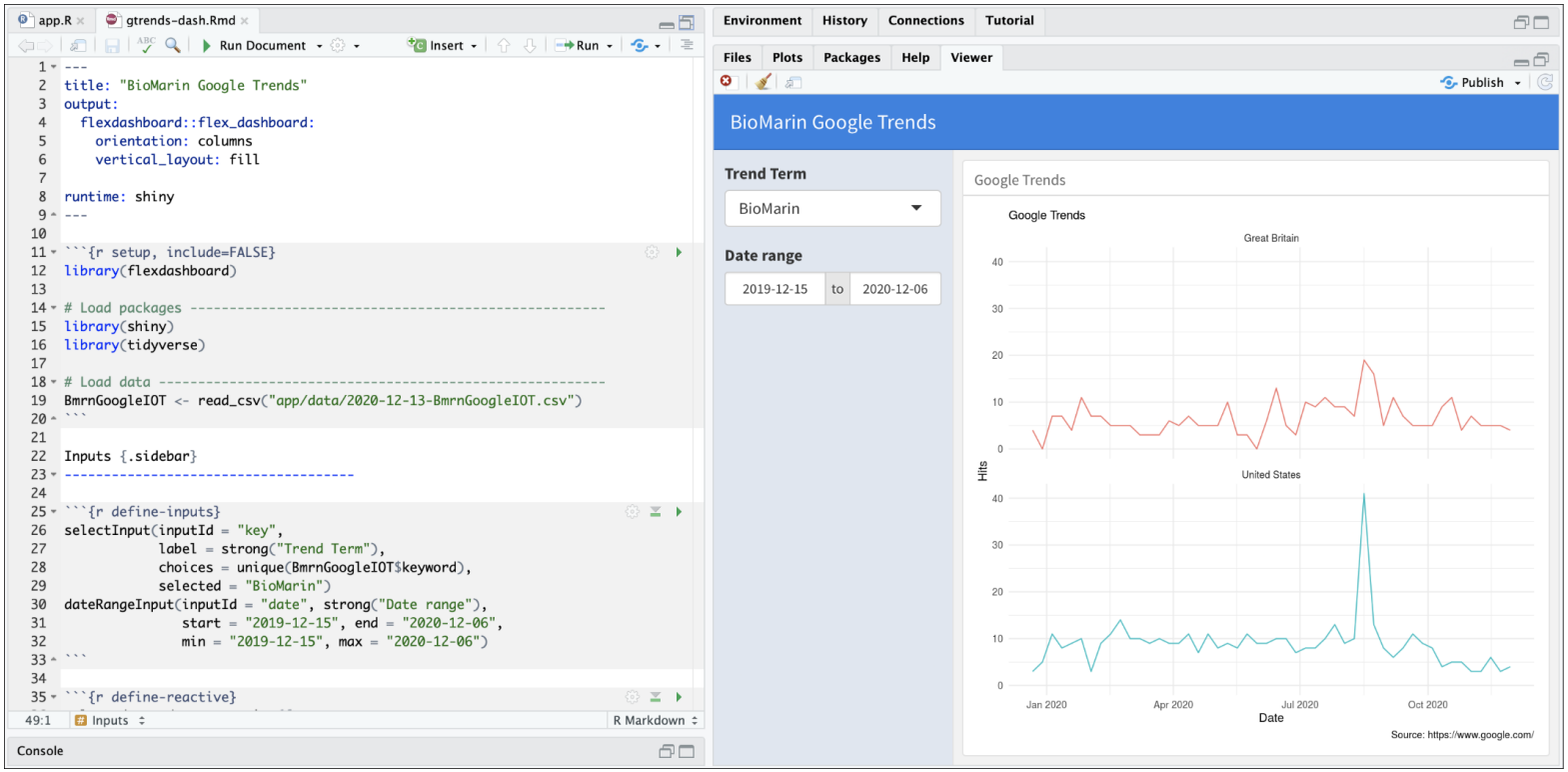
```{r renderPlot}
renderPlot({
    selected_trends() %>%
      ggplot(aes(x = date, y = hits,
                 color = Location, group = keyword)) +
      geom_line(aes(group = Location),
                show.legend = FALSE) +
      facet_wrap(. ~ Location, nrow = 2) +
      theme_minimal() +
      labs(x = "Date", y = "Hits",
           subtitle = "Google Trends",
           caption = "Source: https://www.google.com/")
  })
```
```

Save and Run Document





Our app as a `flexdashboard`!!



More Resources:

Shiny: RStudio resources

Mastering Shiny: A text from Hadley Wickham

Engineering Production-Grade Shiny Apps: A text from ThinkR