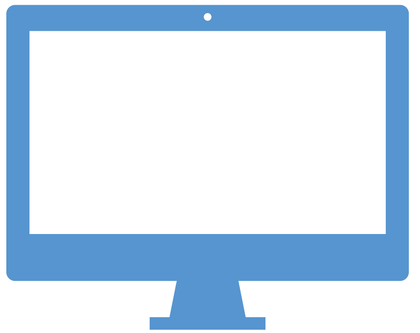
Getting started with Shiny

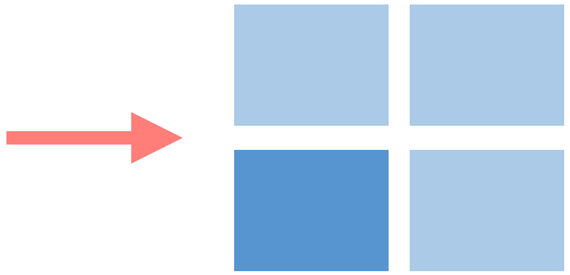
Mine Çetinkaya-Rundel

@minebocek
mine-cetinkaya-rundel
cetinkaya.mine@gmail.com

goog-index/app.R



DEMO



YOUR TURN

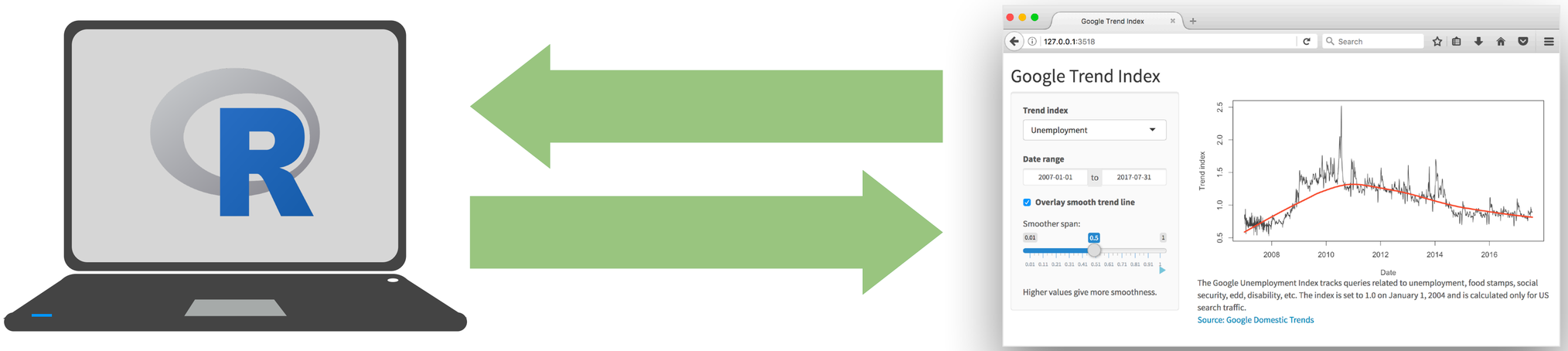
- Go to the goog-index folder in the RStudio Cloud project
- Launch the app by opening `app.R` and clicking Run App
- Close the app by clicking the stop icon
- Select view mode in the drop down menu next to Run App

3m 00s

High level view

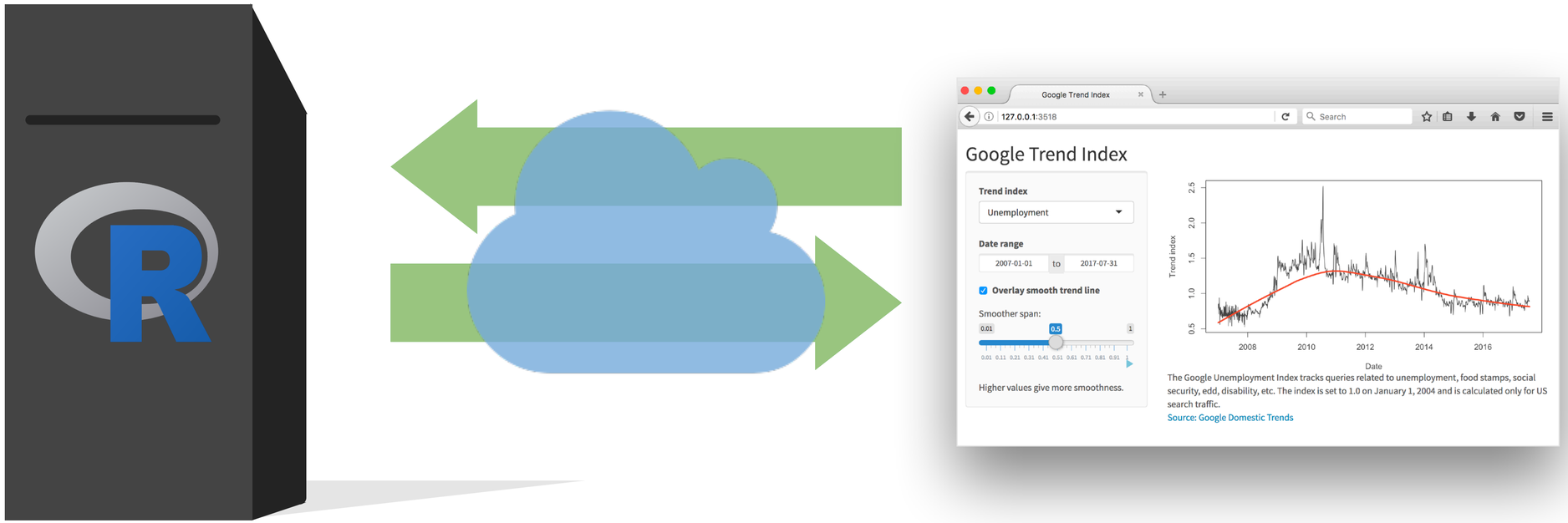
Every Shiny app has a webpage that the user visits, and behind this webpage there is a computer that serves this webpage by running R.

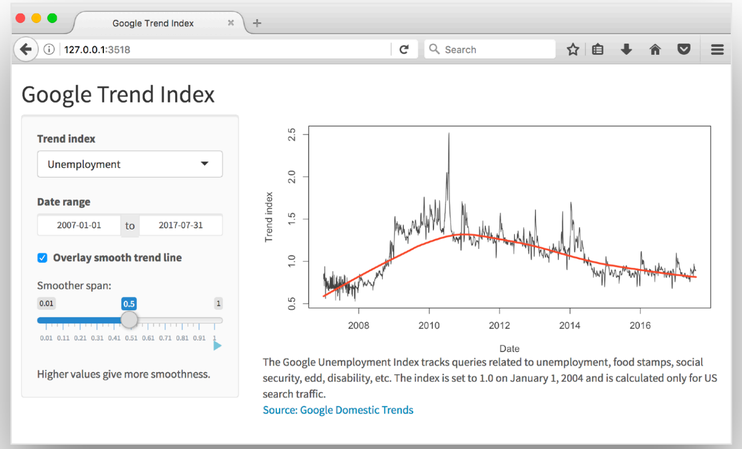
When running your app locally, the computer serving your app is your computer.



When your app is deployed,

the computer serving your app is a web server.





Server instructions

User interface

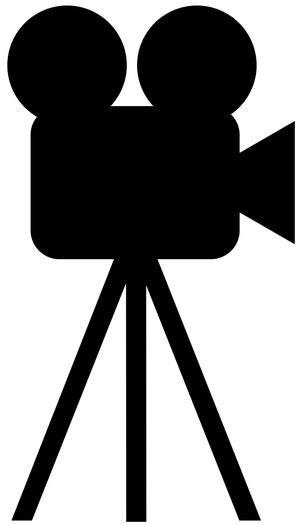
Anatomy of a Shiny app

What's in an app?

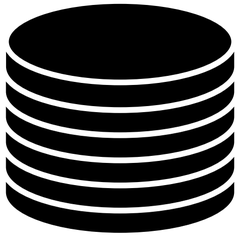
```
library(shiny)

ui <- fluidPage()


server <- function(input, output) {}


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

User interface
controls the layout and appearance of app

Server function
contains instructions needed to build app



Let’s build a simple movie browser app!



`movies-apps/data/movies.Rdata`

Data from IMDB and Rotten Tomatoes on random sample of 651 movies released in the US between 1970 and 2014

~/Dropbox (RStudio)/shiny-training/apps/movies - Shiny

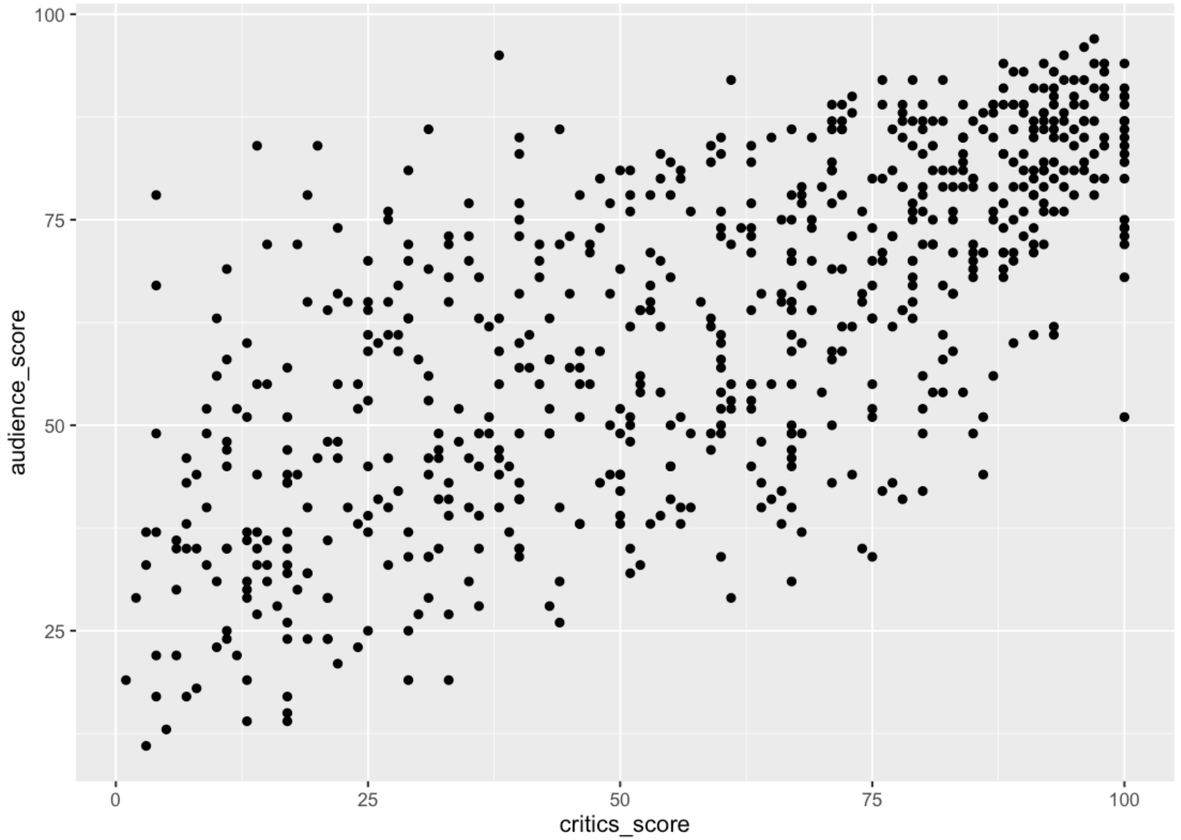
http://127.0.0.1:5877 | Open in Browser | Publish

Y-axis:

audience_score

X-axis:

critics_score



App template

```
library(shiny)
library(tidyverse)
load("data/movies.Rdata")
ui <- fluidPage()

server <- function(input, output) {}

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

Dataset used for this app

User interface

```
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create fluid page layout

```r
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

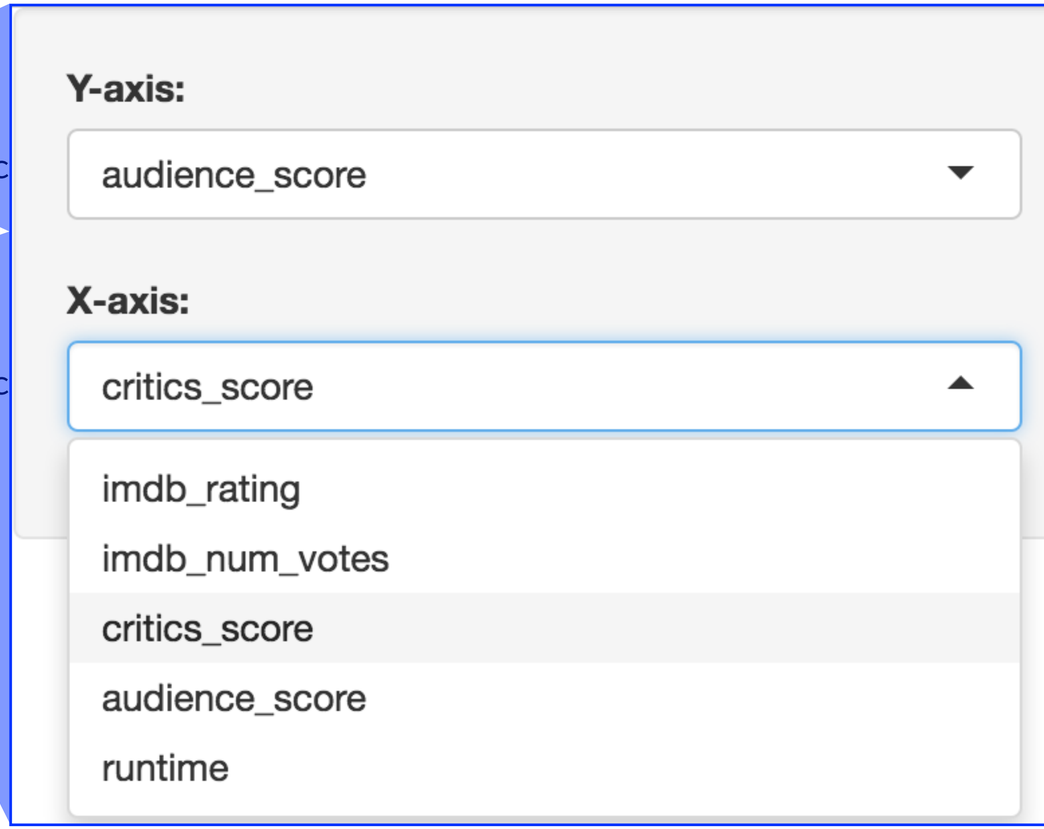
```r
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a layout with a sidebar and main area

```r
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a sidebar panel containing **input** controls that can in turn be passed to `sidebarLayout`

```r
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "c
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "c
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```



```r
# Define UI
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a main panel containing **output** elements that get created in the server function can in turn be passed to `sidebarLayout`

Server

```
# Define server function
server <- function(input, output) {

  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```

```
# Define server function
server <- function(input, output) {

  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```

Contains instructions needed to build app

```
# Define server function
server <- function(input, output) {


  # Create the scatterplot object the plotOutput fun
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y
      geom_point()
  })
}
```

Renders a **reactive** plot that is suitable for assigning to an output slot

```
# Define server function
server <- function(input, output) {


  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```

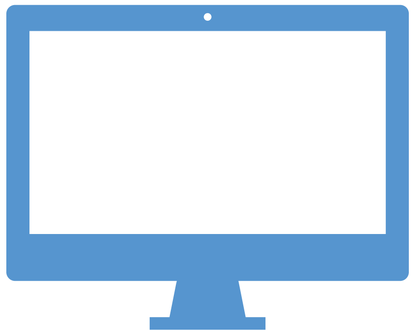
Good ol' ggplot2 code, with **input**s from UI

UI + Server

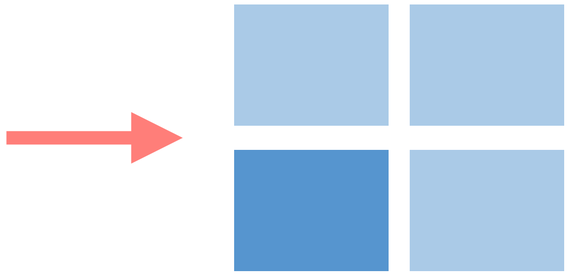
```
# Create the Shiny app object
shinyApp(ui = ui, server = server)
```

Putting it all together...

`movies-apps/movies-01.R`



DEMO



YOUR TURN

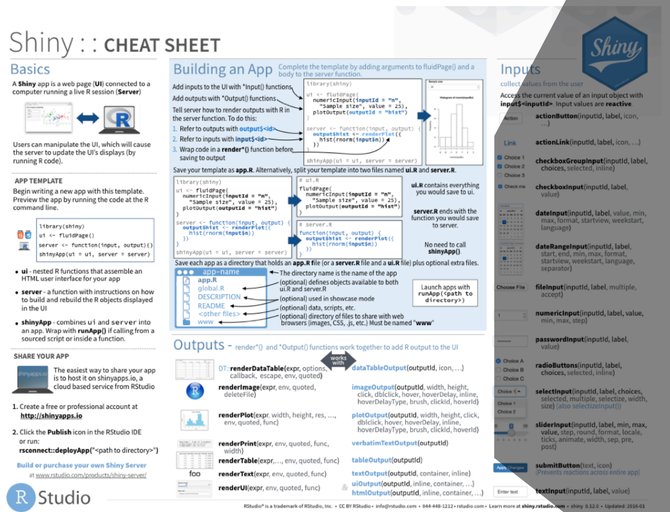
- Start with `movies-apps/movies-01.R`
- Add new select menu to color the points by
  - `inputId = "z"`
  - `label = "Color by:"`
  - `choices = c("title_type", "genre", "mpaa_rating", "critics_rating", "audience_rating")`
  - `selected = "mpaa_rating"`
- Use this variable in the aesthetics of the `ggplot` function as the color argument to color the points by

5m 00s

movies-apps/movies-02.R

SOLUTION

Inputs



Action — **actionButton**(inputId, label, icon, …)

Link — **actionLink**(inputId, label, icon, …)

Choice 1 / Choice 2 / Choice 3 — **checkboxGroupInput**(inputId, label, choices, selected, inline)

Check me — **checkboxInput**(inputId, label, value)

2015-06-08 — **dateInput**(inputId, label, value, min, max, format, startview, weekstart, language)

2015-06-08 to 2015-06-08 — **dateRangeInput**(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)

Choose File — **fileInput**(inputId, label, multiple, accept)

1 — **numericInput**(inputId, label, value, min, max, step)

•••••••• — **passwordInput**(inputId, label, value)

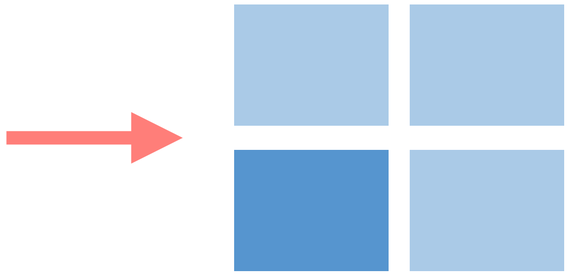
Choice A / Choice B / Choice C — **radioButtons**(inputId, label, choices, selected, inline)

Choice 1 (Choice 1 / Choice 2) — **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)

0 5 10 (0 2 4 6 8 10) — **sliderInput**(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)

Apply Changes — **submitButton**(text, icon)
(Prevents reactions across entire app)

Enter text — **textInput**(inputId, label, value)



YOUR TURN

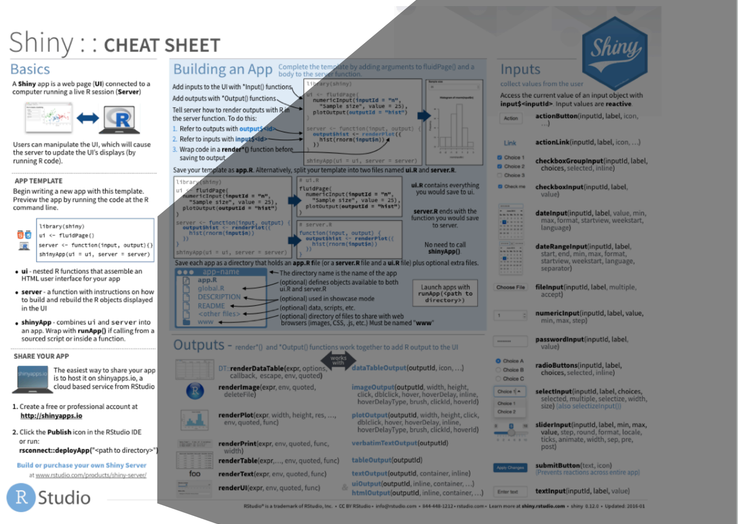
- Start with `movies-apps/movies-02.R`
- Add new input variable to control the alpha level of the points
  - This should be a `sliderInput`
    - See shiny.rstudio.com/reference/shiny/latest/ for help
  - Values should range from 0 to 1
  - Set a default value that looks good
- Use this variable in the geom of the `ggplot()` function as the alpha argument

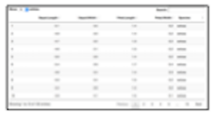
10m 00s

movies-apps/movies-03.R

SOLUTION

Outputs





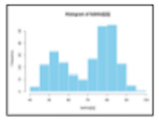
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

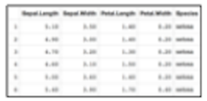


renderPlot(expr, width, height, res, …, env, quoted, func)

plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

renderPrint(expr, env, quoted, func, width)

verbatimTextOutput(outputId)



renderTable(expr,…, env, quoted, func)

tableOutput(outputId)

foo

renderText(expr, env, quoted, func)

textOutput(outputId, container, inline)

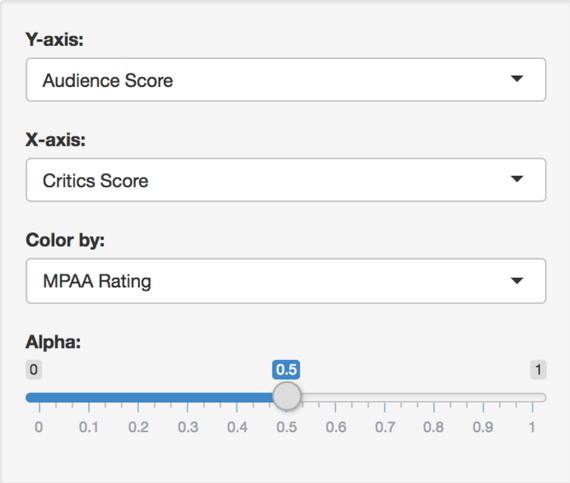
renderUI(expr, env, quoted, func)

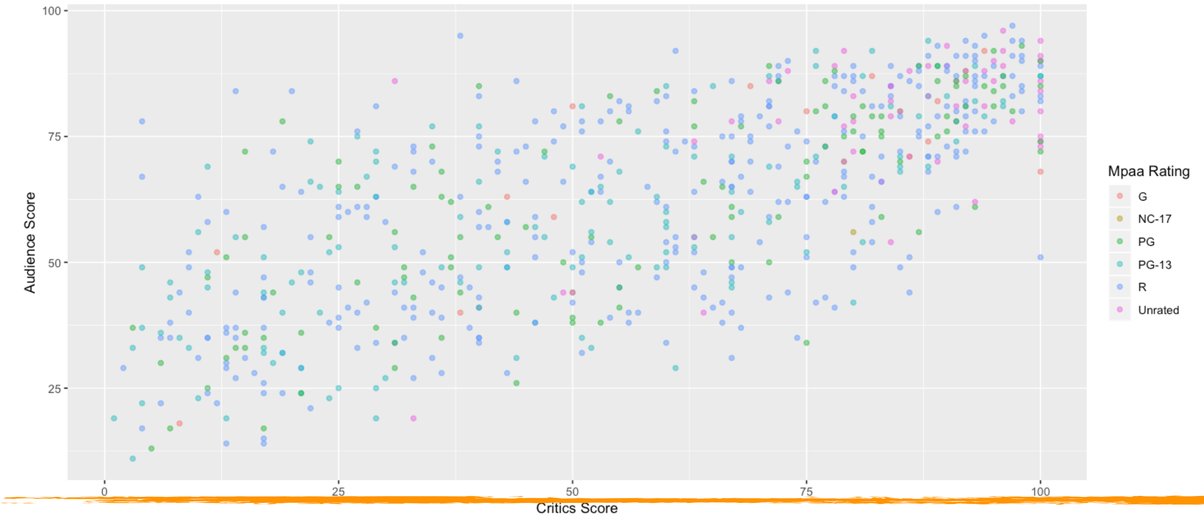
uiOutput(outputId, inline, container, …)
& **htmlOutput**(outputId, inline, container, …)

Which input output duo can we use to get this table?

shiny.rstudio.com/reference/shiny/latest/

Type your answer in the chat





Show 10 entries

Search:

title	title_type	genre	runtime	mpaa_rating	studio	thtr_rel_year
Filly Brown	Feature Film	Drama	80	R	Indomina Media Inc.	2013
The Dish	Feature Film	Drama	101	PG-13	Warner Bros. Pictures	2001
Waiting for Guffman	Feature Film	Comedy	84	R	Sony Pictures Classics	1996
The Age of Innocence	Feature Film	Drama	139	PG	Columbia Pictures	1993
Malevolence	Feature Film	Horror	90	R	Anchor Bay Entertainment	2004

```
library(shiny)
library(tidyverse)
load(“data/movies.Rdata")
ui <- fluidPage(
    DT::dataTableOutput()
)

server <- function(input, output) {
    DT::renderDataTable()
}

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

Over the break, if you like...

- Start with `movies-apps/movies-03.R`
- Create a new output item using `DT::renderDataTable()`.
- Show first seven columns of movies data, show 10 rows at a time, and hide row names, e.g.
  - `data = movies[, 1:7]`
  - `options = list(pageLength = 10)`
  - `rownames = FALSE`
- Add a `DT::dataTableOutput()` to the main panel
- Run the app in a new Window

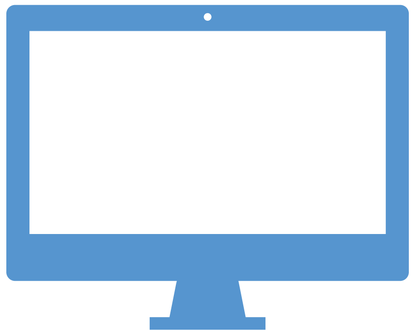
10m 00s

movies-apps/movies-04.R

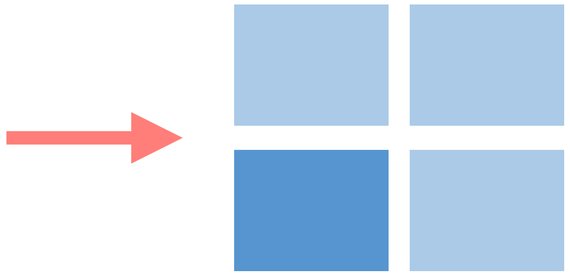
SOLUTION

Add a checkbox to show/hide the data table

`movies-apps/movies-05.R`



DEMO



YOUR TURN

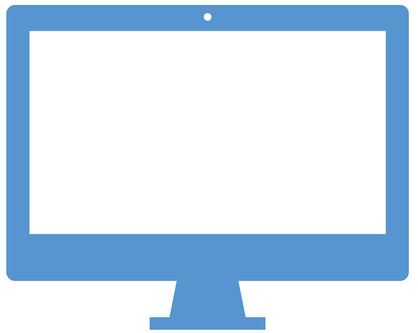
- Start with `movies-apps/movies-05.R`
- Add a title to your app with `titlePanel`, which goes before the `sidebarLayout`
- Prettify the variable names shown as input choices. Hint:
  - `choices = c("IMDB rating" = "imdb_rating", …)`
- Prettify the axis and legend labels of your plot. Hint: You might use
  - `stringr::str_replace_all()` (loaded with tidyverse)
  - `tools::toTitleCase()`

10m 00s

movies-apps/movies-06.R

SOLUTION

Helper functions



DEMO

movies-apps/movies-07.R

Execution

Execution

Where you place code in your app will determine how many times they are run (or re-run), which will in turn affect the performance of your app, since Shiny will run some sections your app script more often than others.

```
library(shiny)
library(tidyverse)
load("movies.Rdata")

ui <- fluidPage(
...
)

server <- function(input, output) {

    output$x <- renderPlot({
    ...
    })

}

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

Execution

```
library(shiny)
library(tidyverse)
load("movies.Rdata")

ui <- fluidPage(
...
)

server <- function(input, output) {

    output$x <- renderPlot({
    ...
    })

}

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

Run once each time a user visits the app

Execution

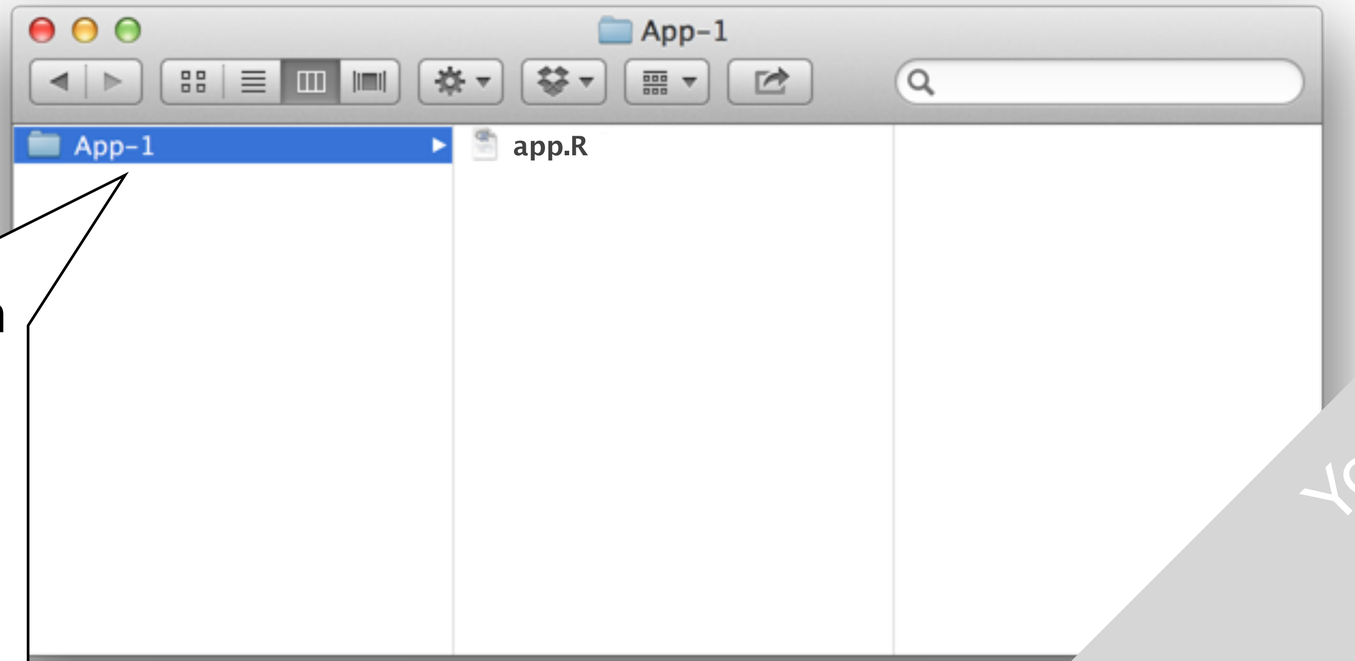
```
library(shiny)
library(tidyverse)
load("movies.Rdata")

ui <- fluidPage(
...
)

server <- function(input, output) {

    output$x <- renderPlot({
    ...
    })

}

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

Run once
each time a user
changes a widget that
output$x depends on

File structure

Single file

- One directory with every file the app needs:
  - `app.R` - your script which ends with a call to `shinyApp()`
  - datasets, images, css, helper scripts, etc.

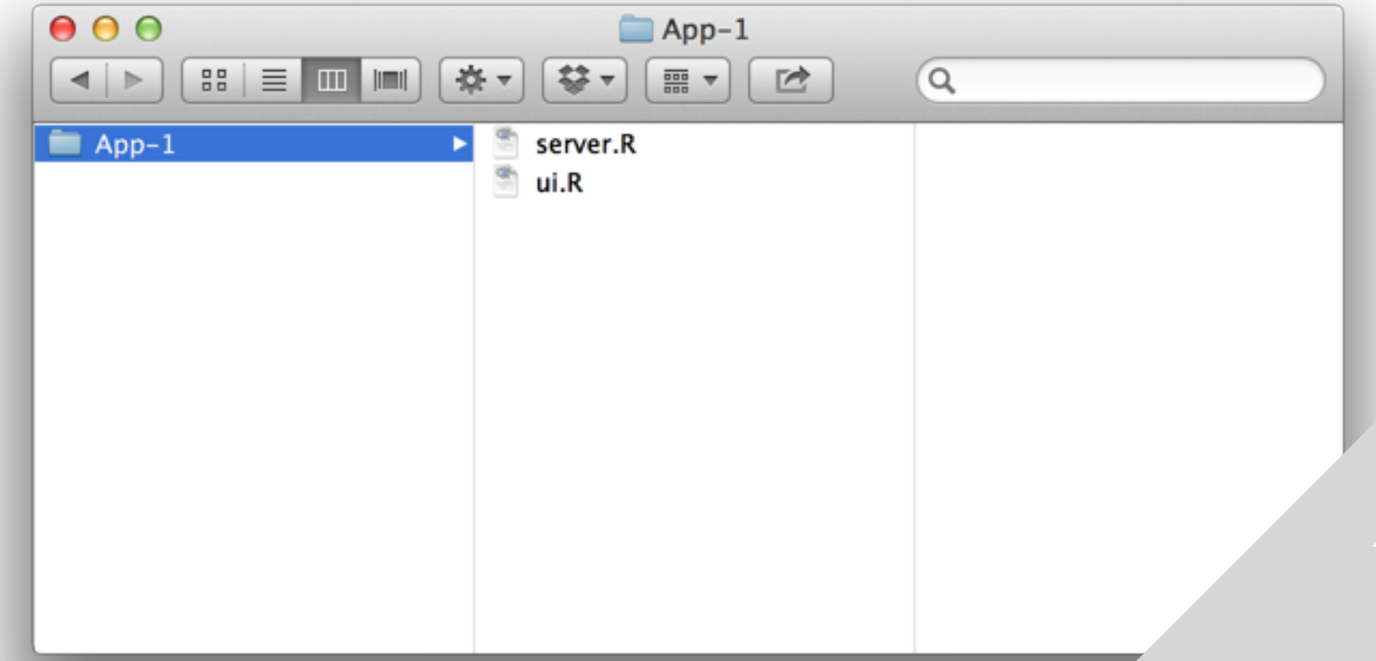


We will focus on the single file format throughout the workshop

You must use this exact name (**app.R**) for deploying the app

Multiple files

- One directory with every file the app needs:
  - `ui.R` and `server.R`
  - datasets, images, css, helper scripts, etc.

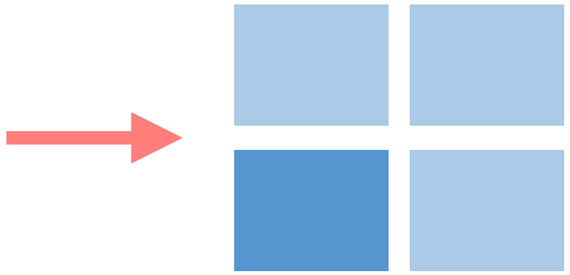


You must use these exact names

Deploying your app

shinyapps.io

- A server maintained by RStudio
- Easy to use, secure, and scalable
- Built-in metrics
- Free tier available



YOUR TURN

- Go to shinyapps.io and log in or create a free account
- In RStudio Cloud:
  - Open `movies-explorer/app.R`
  - Run the app — this is the last app we worked on, saved in a new folder where the folder name is the name of the app we want to deploy and the filename is changed to app
  - Follow the instructions and deploy your first app!
- See https://shiny.rstudio.com/tutorial/written-tutorial/lesson7/ for more

10m 00s

Shiny Server

- Free and open source
- Deploy Shiny apps to the internet
- Run on-premises: move computation closer to the data
- Host multiple apps on one server
- Deploy inside the firewall

RStudio Connect

- Secure access and authentication
- Performance: fine tune at app and server level
- Management: monitor and control resource use
- Direct priority support