Introduction to Shiny

What we'll cover today

- Basic Structure of a Shiny App
- Customizing the shiny UI
- Shiny Server Rules
- Saving & Publishing Your App

Getting Started

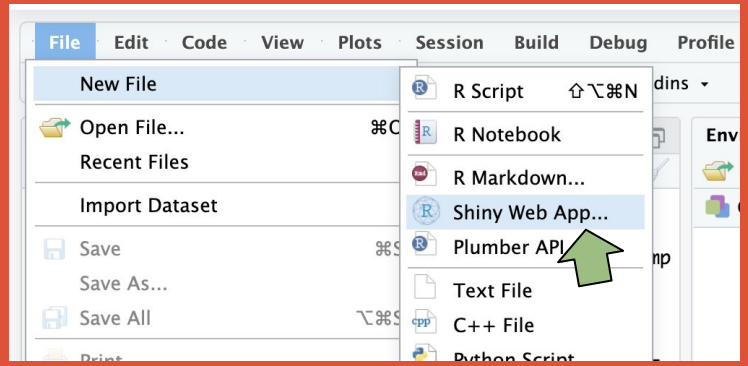
- We'll be using RStudio Cloud today:

  https://rstudio.cloud/spaces/104041/join?access_code=hdLt0g0eikzypCI6E3iX08899gdRLpObZCIA49Mo

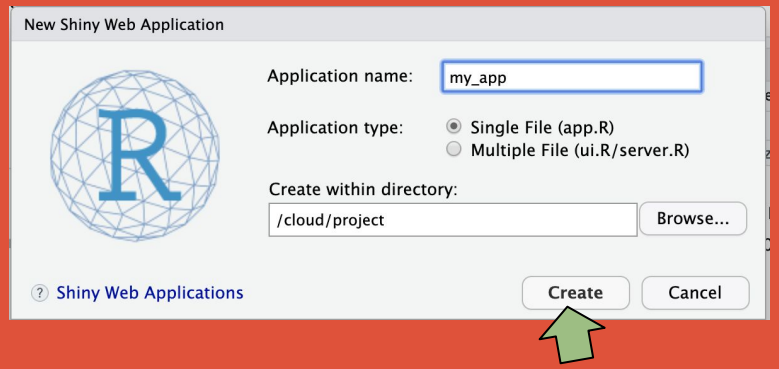
- Use the zoom chat for questions
- **Tip:** Set up your screens so Zoom is one half and RStudio Cloud is on the other

Let's launch our first app!

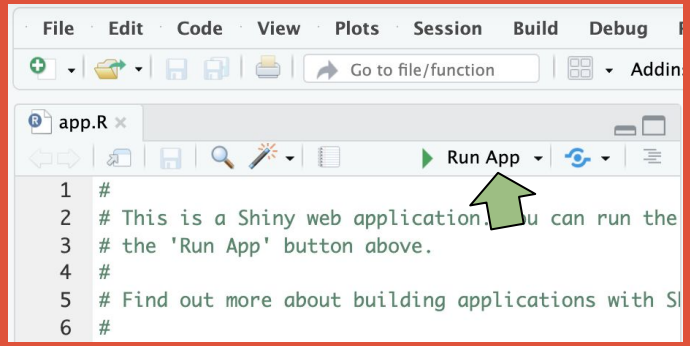
1.



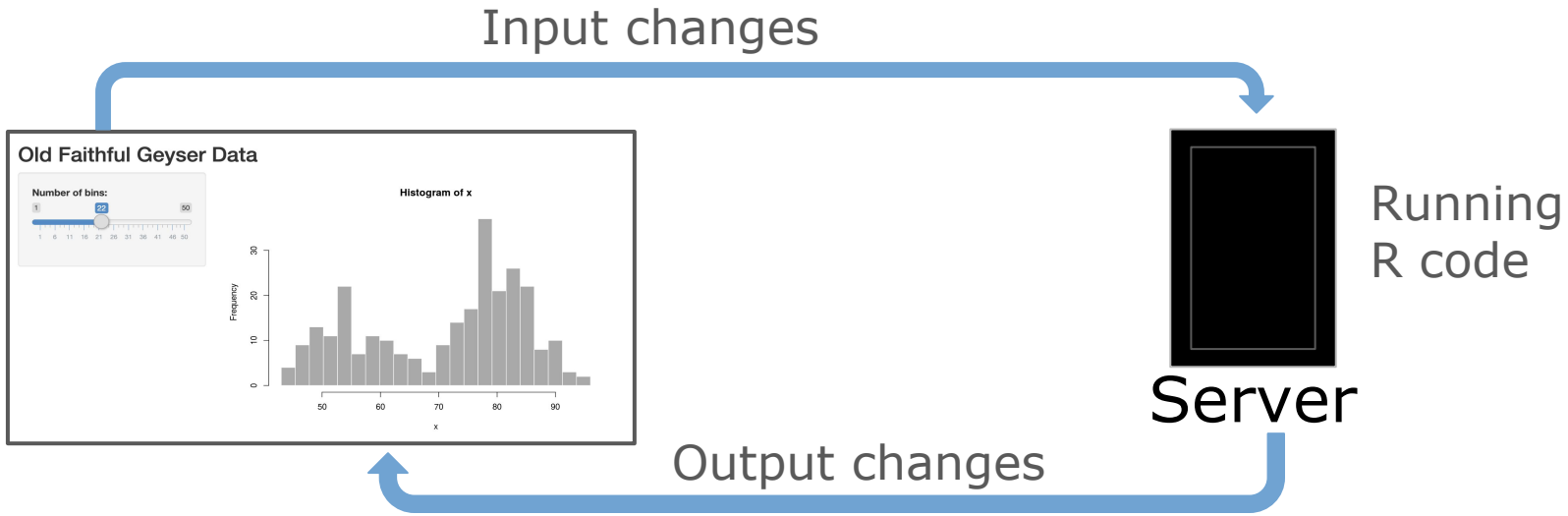
2.

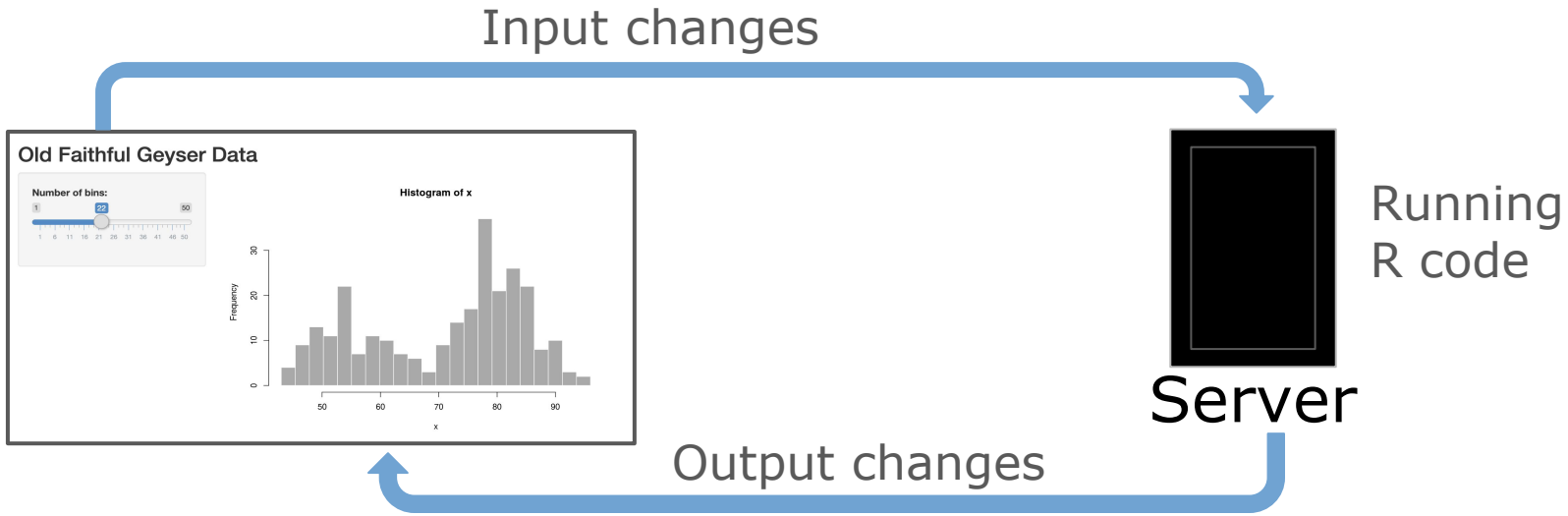


3.



RStudio


Input changes
Old Faithful Geyser Data
Number of bins:
1
22
50
1 6 11 16 21 26 31 36 41 46 50
Histogram of x
Frequency
0
10
20
30
50
60
70
80
90
x
Running
R code
Server
Output changes



UI

What do I see / touch?

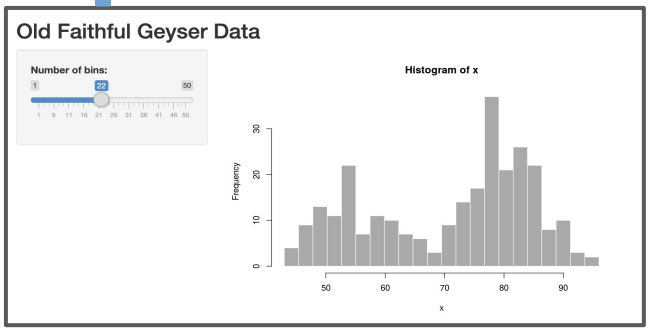
Server

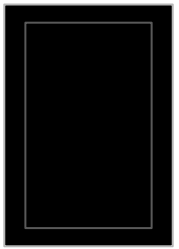
What does the app DO?

Shiny Template

```
library(shiny)

ui <- fluidPage(

)

server <- function(input, output) {

}

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

Input changes





Server

Running
R code

Output changes

UI

- Title
- Slider
- graph

Server

- Receive the slider input
- Create a ggplot

The Shiny UI

UI

- Title
- Slider
- graph

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

The Shiny UI

UI

- **Title**
- Slider
- graph

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

The Shiny UI

UI

- Title
- **Slider**
- graph

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

The Shiny UI

UI

- Title
- Slider
- **graph**

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

The Shiny UI

UI
- Title
- Slider
- Graph
- **Layout**

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

The Shiny UI: Input Controls

```
sliderInput("bins","Number of bins:",
                        min =  1,
                        max =  50,
                        value =  30)
```

The Shiny UI: Input Controls

```
sliderInput("bins","Number of bins:",
                        min =  1,
                        max =  50,
                        value =  30)
```

The Shiny UI: Input Controls

```
sliderInput(<inputID>,"Number of bins:",
                        min =  1,
                        max =  50,
                        value =  30)
```

The Shiny UI: Input Controls

```
sliderInput(<inputID>,<Readable Text>,
                          min =  1,
                          max =  50,
                          value =  30)
```

The Shiny UI: Input Controls

```
sliderInput(<inputID>,<Readable Text>,
                          min =  1,
                          max =  50,
                         <value>)
```

The Shiny UI: Input Controls

```
*Input(<inputID>,<Readable Text>,
                              <value>, <other>)
```

There are a variety of input control options, that may have additional argument options. Check out the options in the shiny widgets gallery.

Let’s add another input control to the app

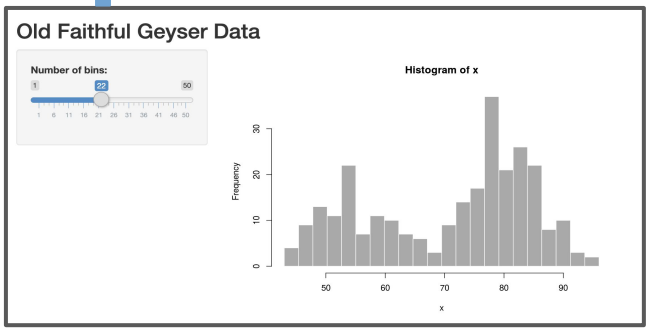
1. Go to the shiny widgets gallery. Select your desired widget and hit the See Code button.

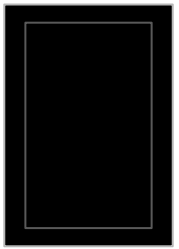
2. Copy the function in the UI section of the code and paste it into your geyser app UI code.

3. Run the app and check out your new widget!

RStudio

Input changes





Server

Running
R code

Output changes

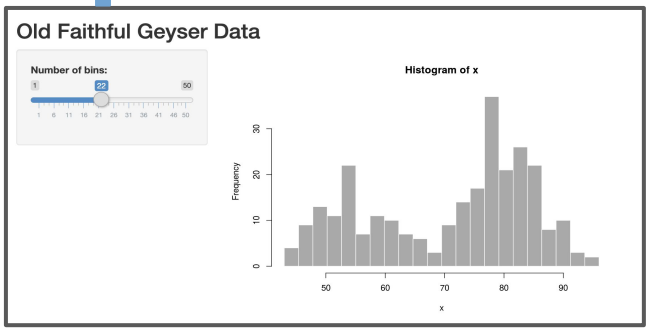
UI

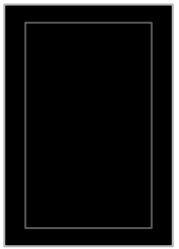
```
sliderInput("bins", …)
```

input$bins

Server

Input changes





Server

Running
R code

Output changes

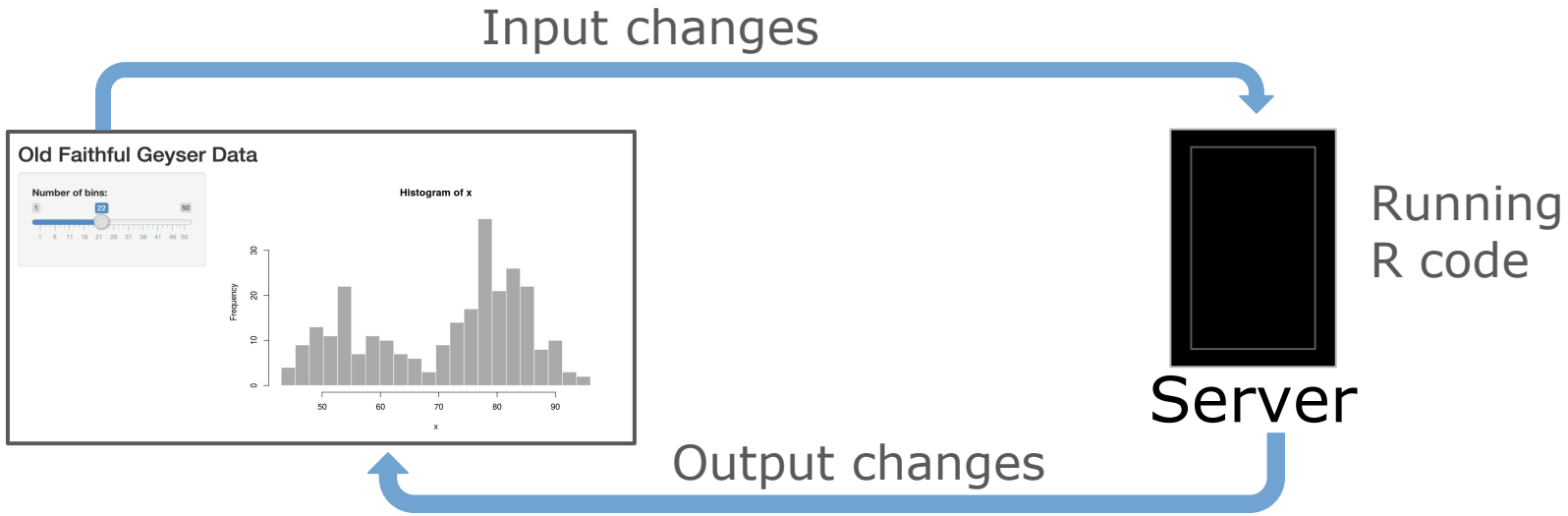
UI

```
sliderInput("bins", …)
```

```
plotOutput("distPlot")
```

```
input$bins
```

Server



UI

```
sliderInput("bins", …)
```

```
plotOutput("distPlot")
```

```
input$bins
```

Server

?

The Shiny UI: Output controls

Outputs - render*() and *Output() functions work together to add R output to the UI

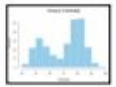
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

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

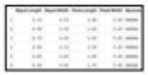


renderPlot(expr, width, height, res, …, env, quoted, func)

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)

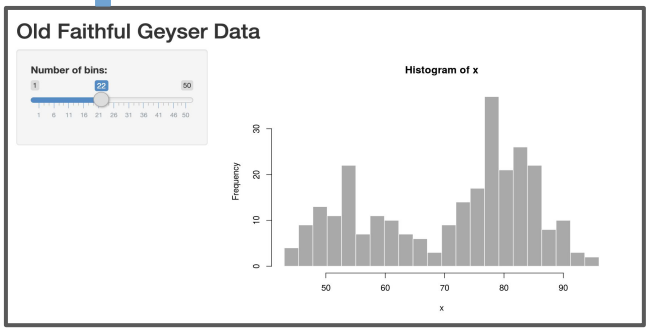
renderUI(expr, env, quoted, func)

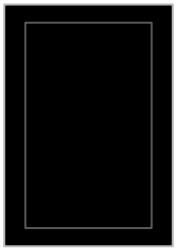
uiOutput(outputId, inline, container, …)

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

Output controls have a render*() pair

Input changes





Server

Running
R code

Output changes

UI

```
sliderInput("bins", …)
```

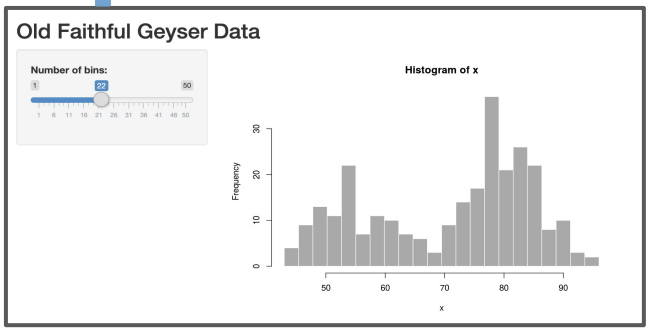
```
plotOutput("distPlot")
```

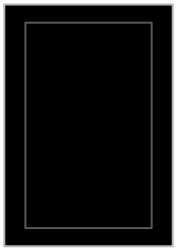
```
input$bins
```

Server

```
renderPlot({...})
```

Input changes





Running R code

Server

Output changes

UI

```
sliderInput("bins", …)
```

```
plotOutput("distPlot")
```

input$bins

output$distPlot

Server

```
output$distPlot<-renderPlot(
{...
})
```

3 Rules for the Server Function

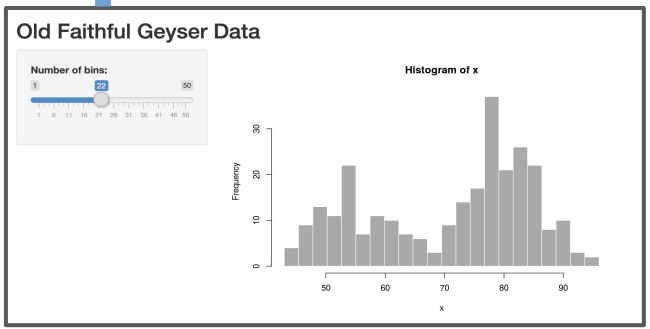
1. Save objects you want to display as **output$**
2. Build object with a **render*()**

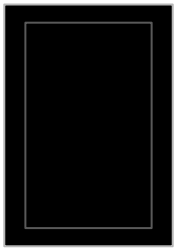
```
output$distPlot<-renderPlot({...})
```

3. Access input values with **input$**

```
output$distPlot<-renderPlot({...
        bins <- seq(min(x), max(x), length.out = input$bins + 1)
....})
```

Input changes





Running R code

Server

Output changes

UI

```
sliderInput("bins", …)
```

```
plotOutput("distPlot")
```

input$bins

output$distPlot

Server

```
output$distPlot<-renderPlot(
{...
})
```

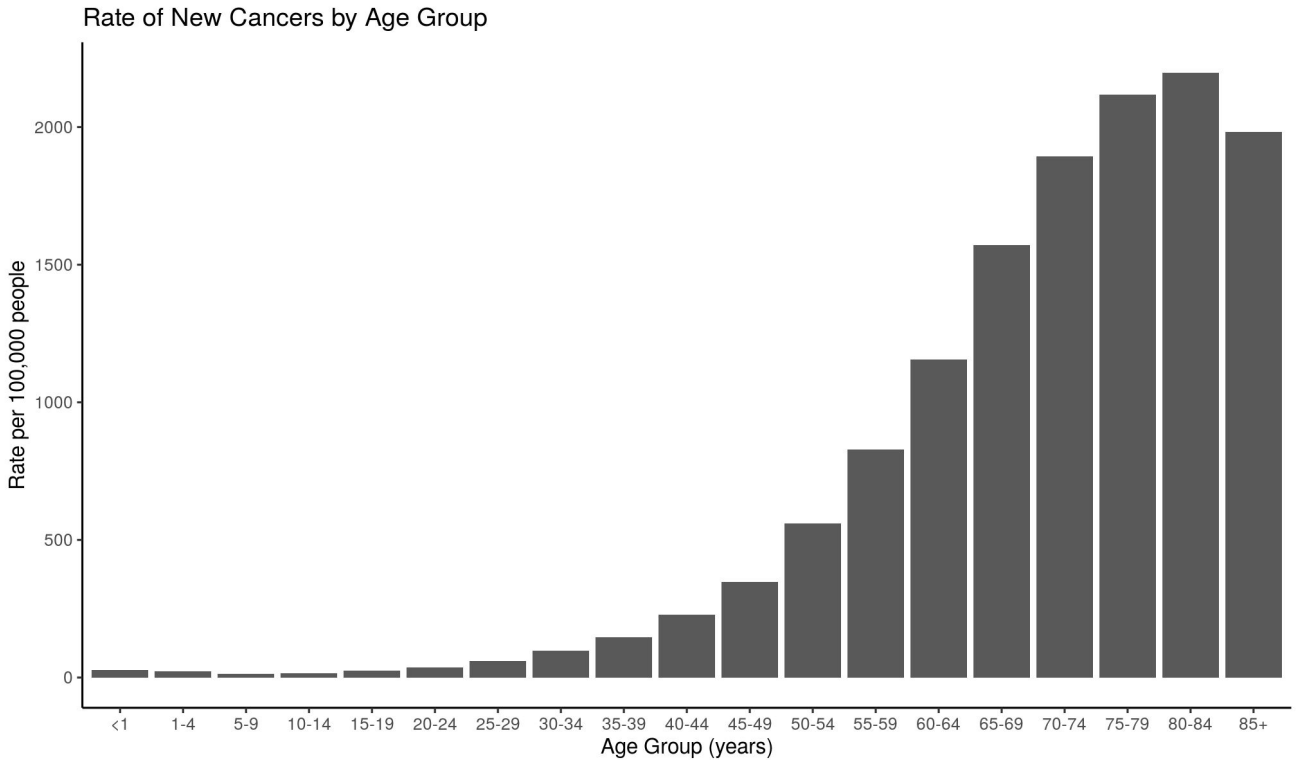
Exercise: Unscramble code for the CDC Cancer Incidence Rate App

1. Open the folder scrambled_app and open the app.R file. Unscramble the code to create your shiny app.

RStudio

Cancer Type

All Cancer Sites Combined



Bonus Exercise: Take unscrambled code and add input widget to select year

Your server function will need to be changed to look like this:

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

    output$plot<-renderPlot({
        ggplot(filter(data, SITE == input$SITE, YEAR== input$YEAR))+
            geom_bar(aes(x=AGE, y=as.numeric(RATE)), stat="identity")+
            ylab("Rate per 100,000 people")+
            xlab("Age Group (years)")+
            ggtitle("Rate of New Cancers by Age Group")+theme_classic()
    })

}
```

RStudio

Bonus Exercise: Take unscrambled code and add input widget to select year

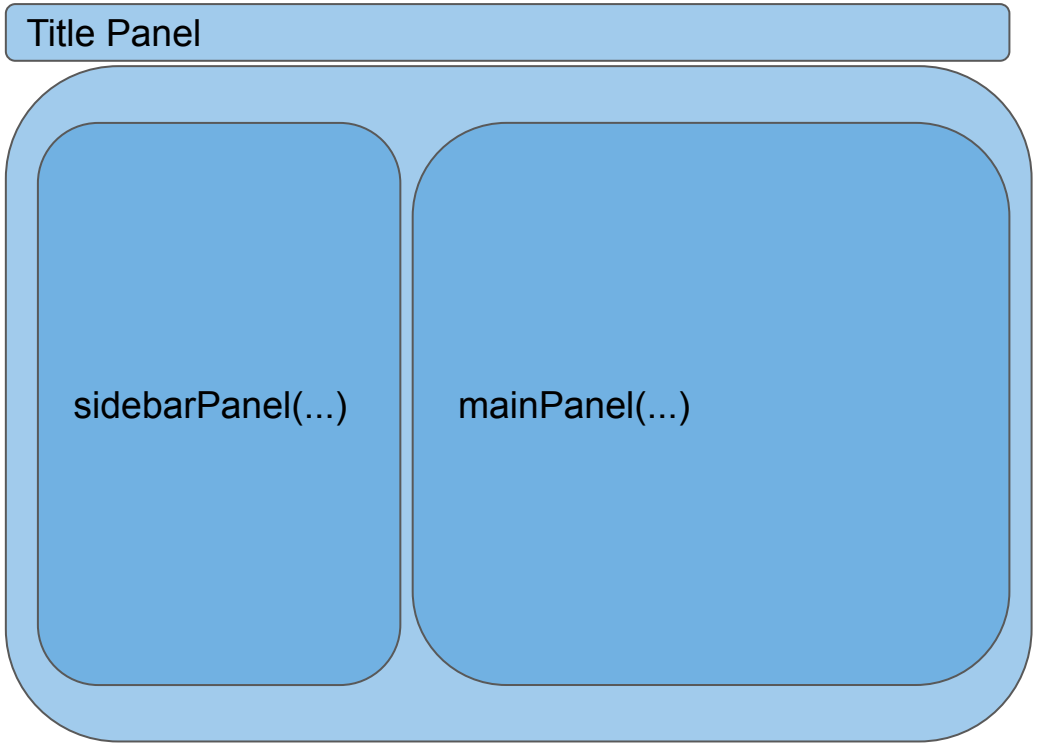
Your server function will need to be changed to look like this:

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

    output$plot<-renderPlot({
        ggplot(filter(data, SITE == input$SITE, YEAR== '2016'))+
            geom_bar(aes(x=AGE, y=as.numeric(RATE)), stat="identity")+
            ylab("Rate per 100,000 people")+
            xlab("Age Group (years)")+
            ggtitle("Rate of New Cancers by Age Group")+theme_classic()
    })

}
```

RStudio

```
ui <- fluidPage(

selectInput("SITE", "Cancer Type", choices = unique(data$SITE)),

selectInput("YEAR", "Year", choices = unique(data$YEAR)),

plotOutput("plot")

)


server <- function(input, output) {

    output$plot<-renderPlot({
        ggplot(filter(data, SITE == input$SITE, YEAR=input$YEAR))+
            geom_bar(aes(x=AGE, y=as.numeric(RATE)), stat="identity")+
            ylab("Rate per 100,000 people")+
            xlab("Age Group (years)")+
            ggtitle("Rate of New Cancers by Age Group")+theme_classic()
    })

}

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

Shiny Layouts




```
ui <- fluidPage(
    titlePanel("Old Faithful
Geyser Data"),
    sidebarLayout(

        sidebarPanel(
            ...
        ),

        mainPanel(
         ...
        )
    )
)
```

Exercise: Improve the CDC Cancer Incidence Rate App

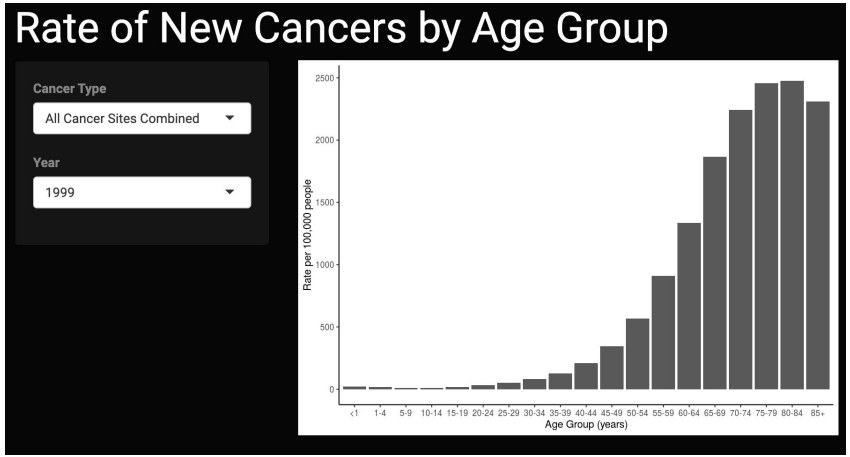
1. Convert the app into a sidebar layout

```
fluidPage(
    sidebarLayout(

        sidebarPanel(
            ...
        ),

        mainPanel(
         ...
        )
    )
)
```

RStudio

Changing the aesthetic with shinythemes

```
ui <- fluidPage(theme=
shinytheme("cyborg"),
```



Check out the gallery of shiny themes here.

Saving your shiny app

Save your template as **app.R**. Alternatively, split your template into two files named **ui.R** and **server.R**.

```
library(shiny)
ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
shinyApp(ui = ui, server = server)
```

```
# ui.R
fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
```

```
# server.R
function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
```

ui.R contains everything you would save to ui.

server.R ends with the function you would save to server.

No need to call **shinyApp()**.

Save each app as a directory that contains an **app.R** file (or a **server.R** file and a **ui.R** file) plus optional extra files.

The directory name is the name of the app

(optional) defines objects available to both ui.R and server.R

(optional) used in showcase mode

(optional) data, scripts, etc.

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

Launch apps with `runApp(<path to directory>)`

Sharing your shiny app

Deploy to the cloud

Shinyapps.io

Host your Shiny apps on the web in minutes with Shinyapps.io. It is easy to use, secure, and scalable. No hardware, installation, or annual purchase contract required. Free and paid options available.

Learn more | FAQ

Deploy on-premises (open source)

Shiny Server

Deploy your Shiny apps and interactive documents on-premises with open source Shiny Server, which offers features such as multiple apps on a single server and deployment of apps behind firewalls.

Learn more

Deploy on-premises (commercial)

RStudio Connect

RStudio Connect is our flagship publishing platform for the work your teams create in R. With RStudio Connect, you can share Shiny applications, R Markdown reports, dashboards, plots, and more in one convenient place with push-button publishing from the RStudio IDE. Features include scheduled execution of reports and flexible security policies to bring the power of data science to your entire enterprise.

Learn more | FAQ

Recap what you learned

- Basic Structure of a Shiny App
- Customizing the shiny UI
- Shiny Server Rules
- Saving & Publishing Your App

Keep Learning about Shiny!

- Shiny Gallery
- Shiny tutorials
- Shiny Cheatsheet
- Mastering Shiny (WIP)