Shiny app for beginners: Activities

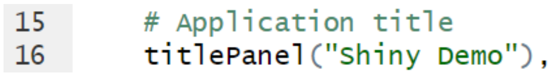
Dr. Xijin “Steven” Ge https://www.linkedin.com/in/steven-ge-ab016947/

Activity #1: Start the demo app

1. Start the demo app. If you are using RStudio from your own laptop, go to **New project → New Directory → Shiny app.** If you are on Posit.cloud, go to **New Project → New RStudio project → File → New File → Shiny app**.
2. Name the app **Shiny_demo**. This will be come a name of a folder.
3. Run the app.

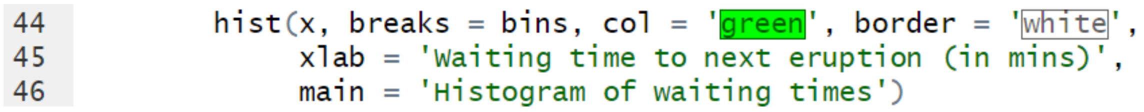
Activity #2: Making small changes

1. Change the application title to “Shiny demo”. Run the app to see the effect.



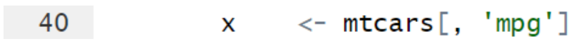
```
    # Application title
    titlePanel("Shiny Demo"),
```

2. Change the color from “darkgray” to “green”. Run the app.



```
        hist(x, breaks = bins, col = 'green', border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
             main = 'Histogram of waiting times')
```

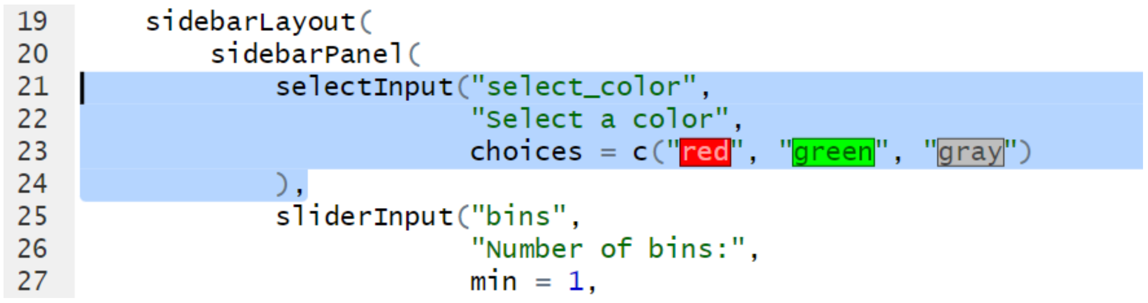
3. Swap data to mtcars



```
        x    <- mtcars[, 'mpg']
```

Activity #3: Adding and using control widgets

1. Bring the Shiny Cheat sheet: Help → Cheat Sheets → Web application with Shiny
2. Add a color select to the UI. Add these 5 lines. Run the app. Make sure there is no errors and it looks good.



```
    sidebarLayout(
        sidebarPanel(
            selectInput("select_color",
                        "Select a color",
                        choices = c("red", "green", "gray")
            ),
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
```

3. Change the color of the plot. Change this line and run the app.

```
        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = input$select_color, border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
```

Activity #4: Change data columns

1. Add a selectInput to select the columns. Run the app to make sure the control widget works.

```
    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            selectInput("select_column",
                        "Select a column",
                        choices = colnames(mtcars)
            ),
            selectInput("select_color",
                        "Select a color",
                        choices = c("red", "green", "gray")
```

2. Use the select column to plot. Run the app.

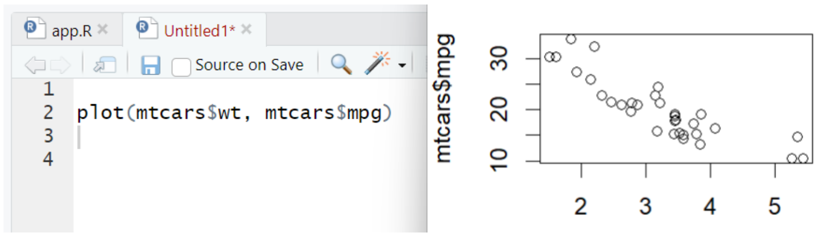
```
        # generate bins based on input$bins from ui.R
        x    <- mtcars[, input$select_column]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)
```

Activity 5. Add a plot.

1. First add the plotOutput in UI. Run the app.

```
        mainPanel(
           plotOutput("distPlot"),
           plotOutput("sPlot")
        )
    )
)
```

2. Generate the plot in a separate R script.

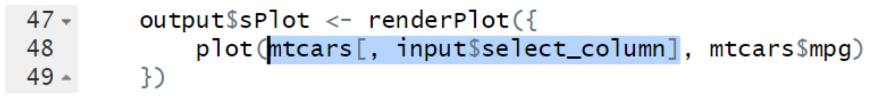


```
plot(mtcars$wt, mtcars$mpg)
```

3. Copy the code into the Shiny app inside an output function.

```
# Define server logic required to draw a histogram
server <- function(input, output) {

    output$sPlot <- renderPlot({
        plot(mtcars$wt, mtcars$mpg)
    })

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
```

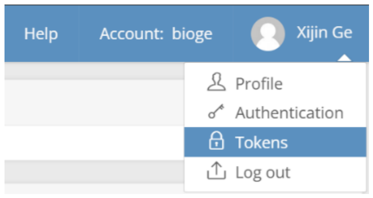
4. Change the code to use selected columns.



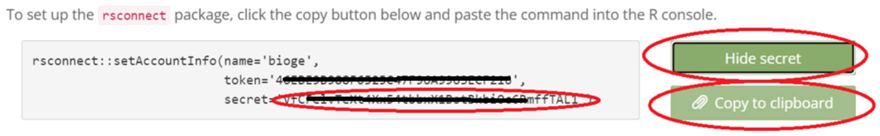
```
output$sPlot <- renderPlot({
    plot(mtcars[, input$select_column], mtcars$mpg)
})
```

Activity 6: Publishing the app on Shinyapp.io

1. Log into Shinyapps.io
2. Click on the top right corner (your name). Then select Tokens.



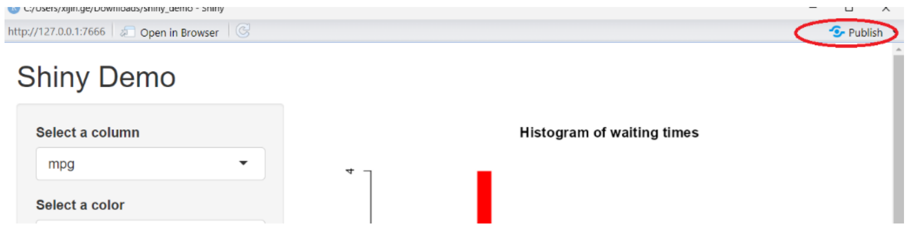
3. Click on Add Token
4. Click on Show Token and the **Show Secret**. Copy to clipboard.



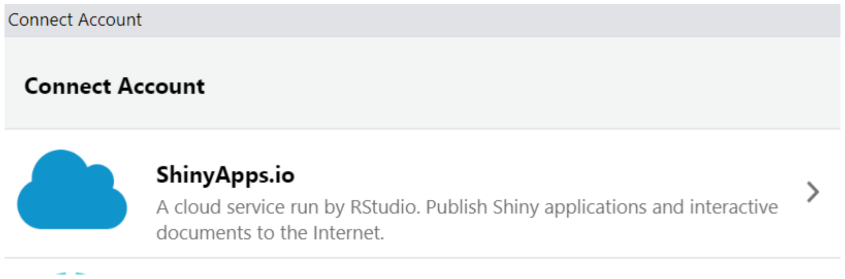
To set up the rsconnect package, click the copy button below and paste the command into the R console.

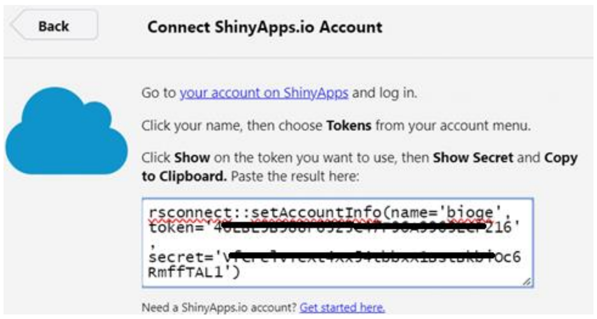
```
rsconnect::setAccountInfo(name='bioge',
                          token='4[redacted]',
                          secret='[redacted]mffTAL1')
```

5. Go back to RStudio. Run the app. Click the Publish button on the top right.



6. Connect to Shinyapps.io by pasting the secret.


Connect Account
Connect Account
ShinyApps.io
A cloud service run by RStudio. Publish Shiny applications and interactive documents to the Internet.



Go to your account on ShinyApps and log in.

Click your name, then choose **Tokens** from your account menu.

Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard.** Paste the result here:

```
rsconnect::setAccountInfo(name='bioge',
token='[redacted]216'
,
secret='[redacted]oc6
RmffTAL1')
```

Need a ShinyApps.io account? Get started here.

7. Select relevant data files before click publish. After a few minutes, your app is deployed to the cloud. You will get a URL that points to your app. Check that URL from a browser.

Activity 7. Interacting with ChatGPT through API from Shiny

1. Start a new R script file. Save as ChatGPT.R. Install the "openai" R package from GitHub. See https://github.com/irudnyts/openai

```
install.packages("remotes")
remotes::install_github("irudnyts/openai", ref = "r6")
```

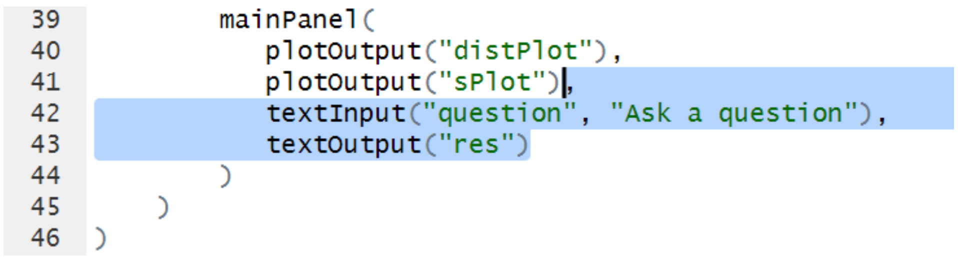
2. Go to OpenAI website, obtain an API key.
3. Set up the API key as an environment variable named "OPENAI_API_KEY". This can be done in Windows using the Edit Environment Variable menu. Alternatively, you can set this in R using this command.

```
Sys.setenv(
  OPENAI_API_KEY = 'sk-proj-XEFD_g86'
)
```

4. Test if the API call works.

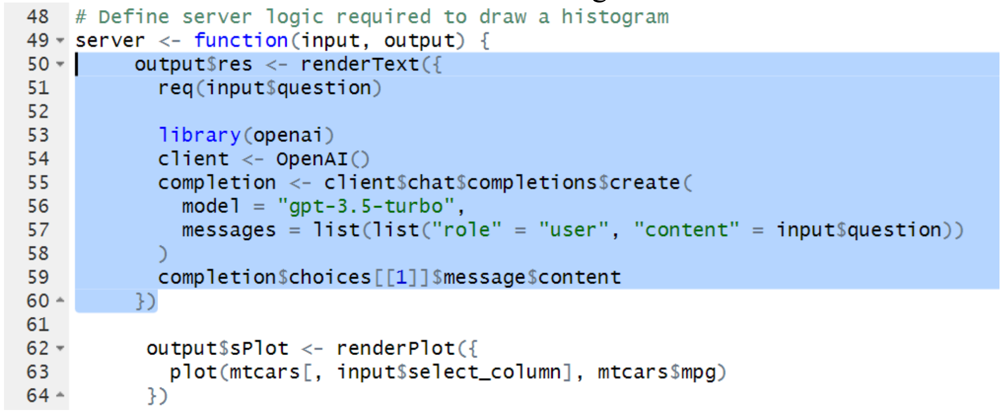
```
library(openai)
client <- OpenAI()
completion <- client$chat$completions$create(
  model = "gpt-3.5-turbo",
  messages = list(list("role" = "user", "content" = "Tell me a joke"
)

completion$choices[[1]]$message$content
```

5. Add ui elements to enable Chatbot.



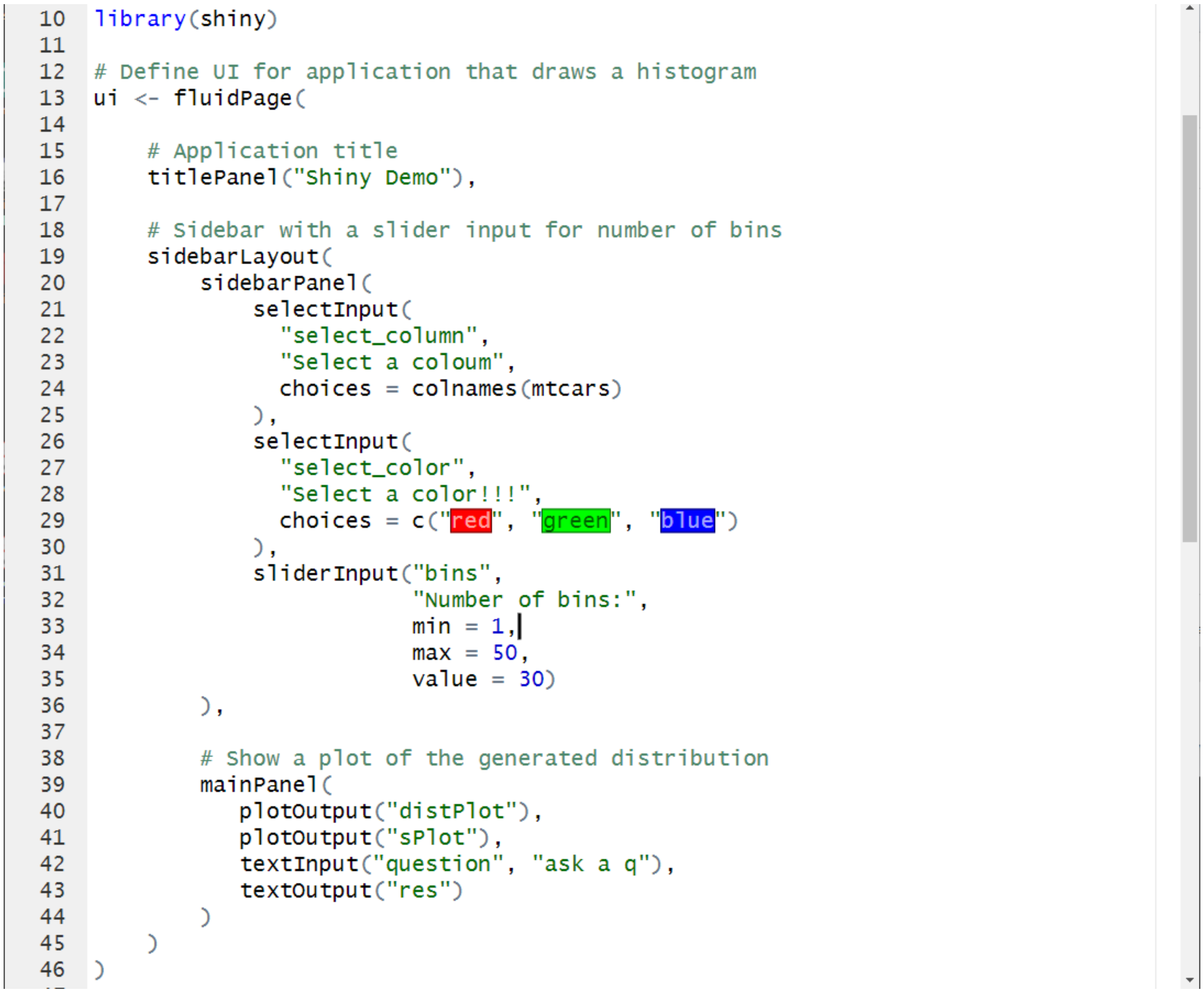
```
        mainPanel(
           plotOutput("distPlot"),
           plotOutput("sPlot"),
           textInput("question", "Ask a question"),
           textOutput("res")
        )
    )
)
```

6. Add server function to handle the chat using ChatGPT.



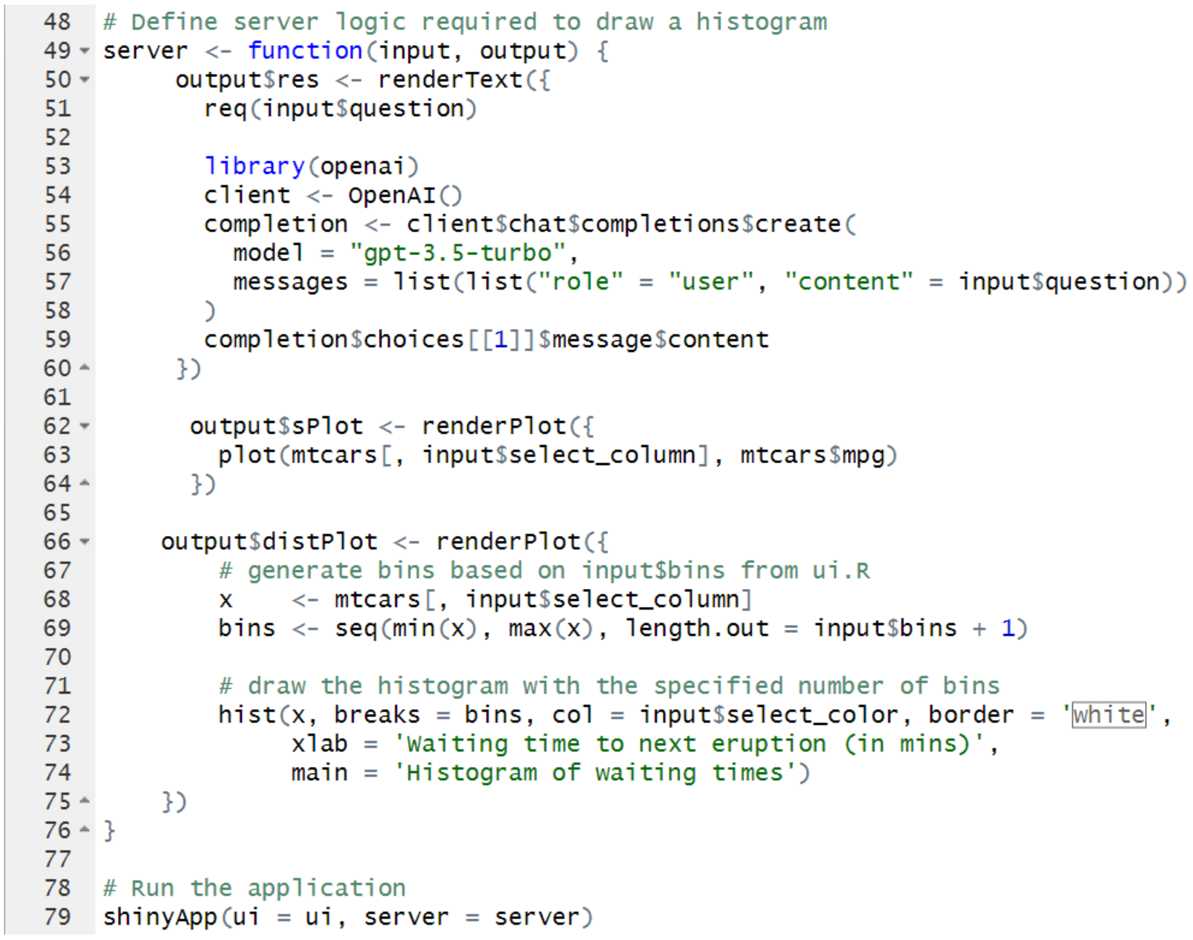
```r
# Define server logic required to draw a histogram
server <- function(input, output) {
     output$res <- renderText({
       req(input$question)

       library(openai)
       client <- OpenAI()
       completion <- client$chat$completions$create(
         model = "gpt-3.5-turbo",
         messages = list(list("role" = "user", "content" = input$question))
       )
       completion$choices[[1]]$message$content
     })

      output$sPlot <- renderPlot({
        plot(mtcars[, input$select_column], mtcars$mpg)
      })
```

The final UI code after all these activities:



```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(

    # Application title
    titlePanel("Shiny Demo"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            selectInput(
              "select_column",
              "Select a coloum",
              choices = colnames(mtcars)
            ),
            selectInput(
              "select_color",
              "Select a color!!!",
              choices = c("red", "green", "blue")
            ),
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("distPlot"),
           plotOutput("sPlot"),
           textInput("question", "ask a q"),
           textOutput("res")
        )
    )
)
```

The final Server function:



```r
# Define server logic required to draw a histogram
server <- function(input, output) {
     output$res <- renderText({
       req(input$question)

       library(openai)
       client <- OpenAI()
       completion <- client$chat$completions$create(
         model = "gpt-3.5-turbo",
         messages = list(list("role" = "user", "content" = input$question))
       )
       completion$choices[[1]]$message$content
     })

      output$sPlot <- renderPlot({
        plot(mtcars[, input$select_column], mtcars$mpg)
      })

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- mtcars[, input$select_column]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)

        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = input$select_color, border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
             main = 'Histogram of waiting times')
    })
}

# Run the application
shinyApp(ui = ui, server = server)
```