

Fall 2021

Project 2
R Shiny App

Name: Ravleen Kaur

Class ID: 43

School ID: 1113138

Course: Statistics for Data Science

Course ID: DTSC-620-W01

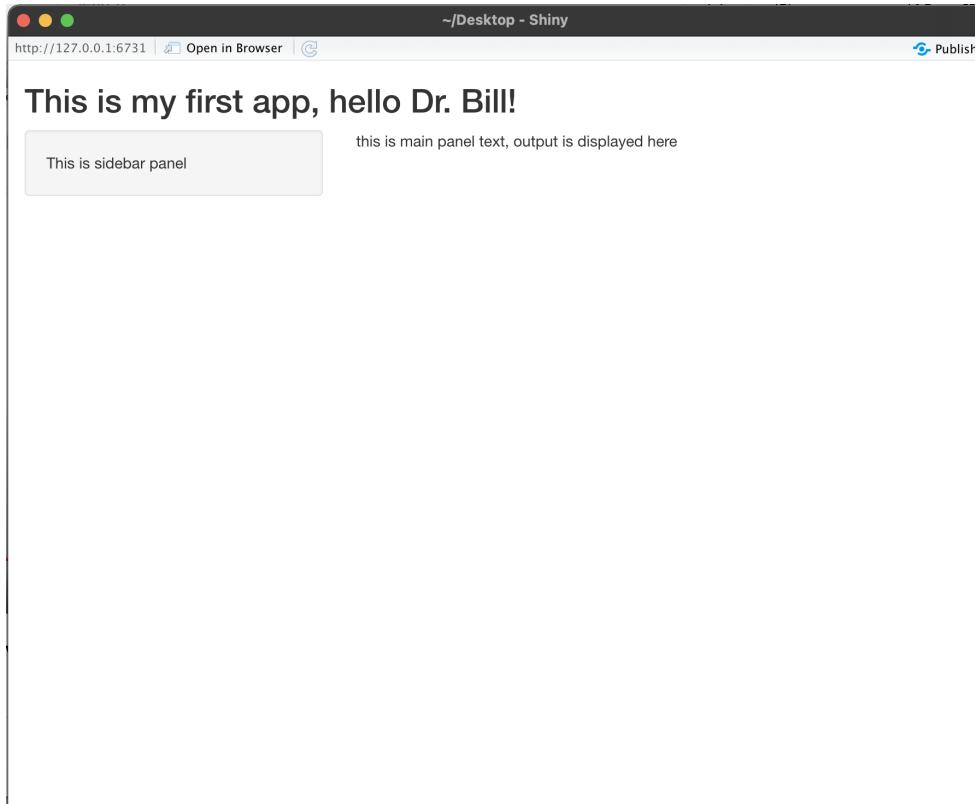
Downloading Shiny

```
> install.packages("shiny")
--- Please select a CRAN mirror for use in this session ---
also installing the dependencies 'fs', 'rappdirs', 'Rcpp', 'digest',
'sass', 'jquerylib', 'httpuv', 'mime', 'jsonlite', 'xtable', 'fontaw
'sourcetools', 'later', 'promises', 'crayon', 'rlang', 'fastmap', 'n
'glue', 'bslib', 'cachem', 'ellipsis', 'lifecycle'

trying URL 'https://cran.case.edu/bin/macosx/contrib/4.1/fs_1.5.2.tg
Content type 'application/x-gzip' length 545943 bytes (533 KB)
=====
downloaded 533 KB

trying URL 'https://cran.case.edu/bin/macosx/contrib/4.1/rappdirs_0.
Content type 'application/x-gzip' length 45542 bytes (44 KB)
=====
downloaded 44 KB
```

App Video #2



The image shows a screenshot of the RStudio IDE interface, displaying two code editors side-by-side.

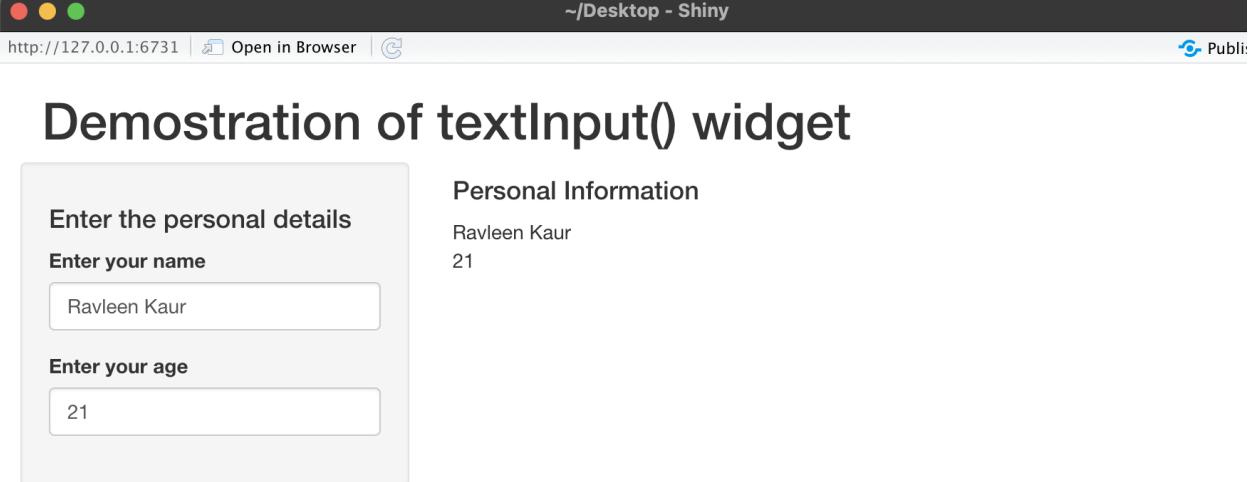
Top Editor (ui.r):

```
1 library(shiny)
2
3 shinyUI(fluidPage(
4
5   titlePanel(title= "This is my first app, hello Dr. Bill!"),
6   sidebarLayout(
7     sidebarPanel("This is sidebar panel"),
8     mainPanel("this is main panel text, output is displayed here")
9   )
10 )
11 )
12 )
13 )
14 )
15 )
16 )
```

Bottom Editor (server.r):

```
1 library(shiny)
2
3 shinyServer(
4
5   function(input, output) {
6
7   }
8 )|
```

Video 3



The screenshot shows a Shiny application interface. At the top, there are three colored dots (red, yellow, green) on the left, the text " ~/Desktop - Shiny" in the center, and a URL "http://127.0.0.1:6731" with a "Open in Browser" button and a refresh icon on the right. Below this is a "Public" link with a lock icon.

The main content area has a title "Demonstration of textInput() widget". On the left, under "Enter the personal details", there are two input fields: one for "Enter your name" containing "Ravleen Kaur" and another for "Enter your age" containing "21". On the right, under "Personal Information", the entered values are displayed: "Ravleen Kaur" and "21".

The screenshot shows the RStudio interface with the 'server.r' tab active. The code editor displays the following R script:

```
1 library(shiny)
2
3 shinyServer(
4
5   function(input, output) {
6
7     output$myname <- renderText({
8       # paste("My Name is :", input$name)
9       input$name
10    })
11
12   output$myage <- renderText({
13     # paste("My Age is : ", input$age)
14     input$age
15    })
16
17  }
18)
```

The screenshot shows the RStudio interface with the 'ui.r' tab active. The code in the editor is as follows:

```
1 # Load the shiny package
2 library(shiny)
3 # Define UI for the shiny application here
4 shinyUI(fluidPage(
5     # fluid pages scale their components in realtime to
6     # Header Panel : Create a header panel containing an
7     headerPanel(title = "Demostration of textInput() wid")
8     # SidebarLayout():It creates a layout with a sidebar
9     sidebarLayout(
10         # Sidebar panel
11         sidebarPanel(
12             h4("Enter the personal details"),
13
14            textInput("name", "Enter your name", ""),
15            textInput("age", "Enter your age", "")
16         ),
17         # Main Panel
18         mainPanel(
19             h4("Personal Information"),
20
21             textOutput("myname"),
22             textOutput("myage")
23
24
25
26
```

Video 4

~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser | C

Demostration of shiny widgets

Enter the personal details

Enter your name

Enter your age

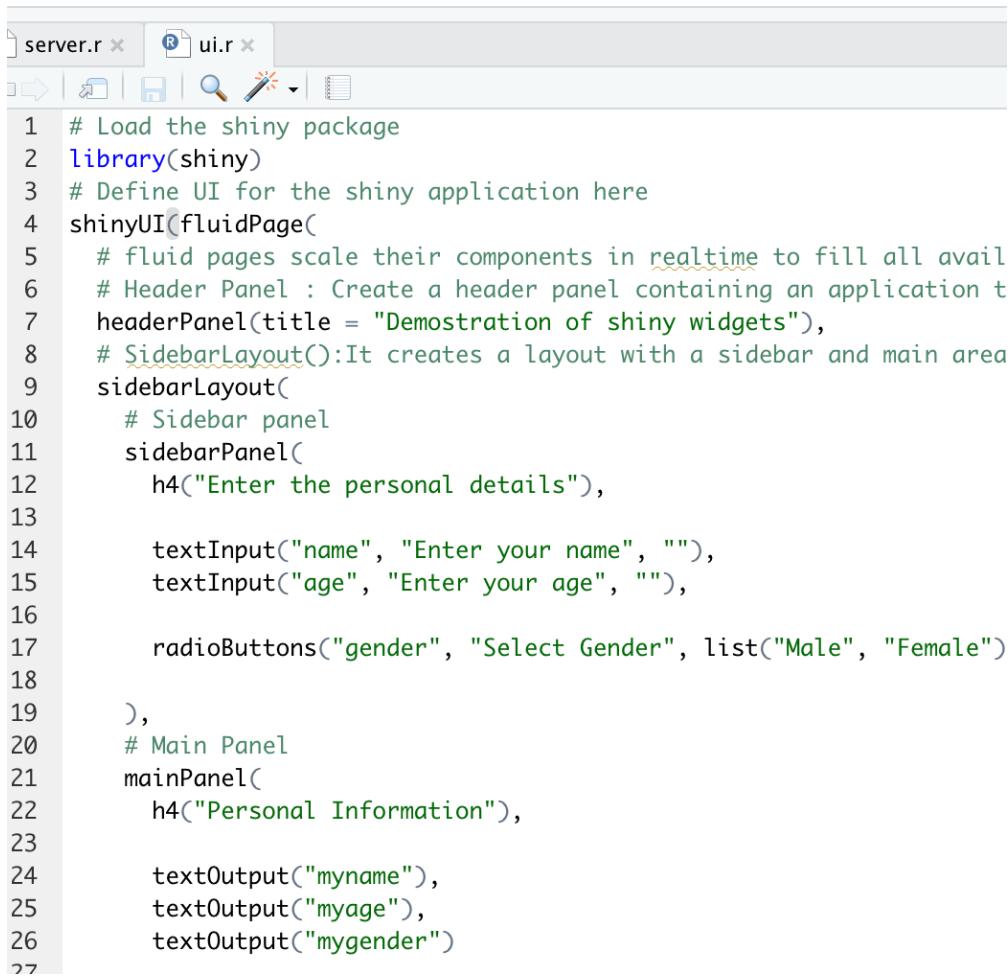
Select Gender

Male

Female

Personal Information

Ravleen Kaur
22
Female



The screenshot shows the RStudio interface with the 'ui.r' file open. The code defines a UI for a shiny application with a sidebar panel containing personal details input fields and a main panel displaying the entered information.

```
1 # Load the shiny package
2 library(shiny)
3 # Define UI for the shiny application here
4 shinyUI(fluidPage(
5   # fluid pages scale their components in realtime to fill all available space
6   # Header Panel : Create a header panel containing an application title
7   headerPanel(title = "Demonstration of shiny widgets"),
8   # SidebarLayout():It creates a layout with a sidebar and main area
9   sidebarLayout(
10     # Sidebar panel
11     sidebarPanel(
12       h4("Enter the personal details"),
13
14      textInput("name", "Enter your name", ""),
15      textInput("age", "Enter your age", ""),
16
17       radioButtons("gender", "Select Gender", list("Male", "Female"))
18     ),
19     # Main Panel
20     mainPanel(
21       h4("Personal Information"),
22
23       textOutput("myname"),
24       textOutput("myage"),
25       textOutput("mygender")
26     )
27   )
```

Video 5: Slide Input

~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser |

Demostration of sliderInput widget in shiny

Select the value from Slider

You selected the value: 3.4

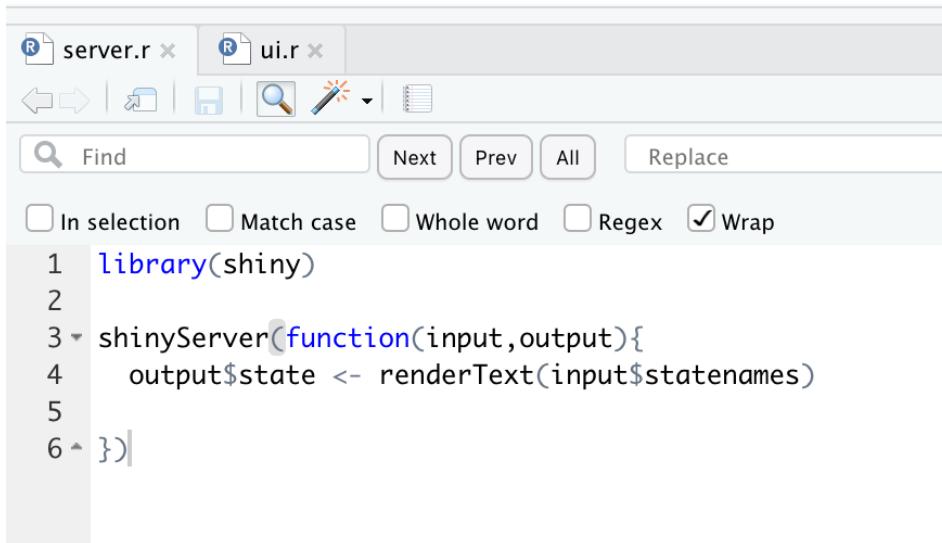
The screenshot shows the RStudio interface with the 'ui.r' file open. The code defines a shinyUI object with a fluidPage containing a titlePanel, sidebarLayout, and mainPanel. The sidebarPanel contains a sliderInput with the id 'slide' and a label 'Select the value from Slider'. The mainPanel contains a textOutput with the id 'out'.

```
1 library(shiny)
2
3 shinyUI(fluidPage(
4   titlePanel("Demonstration of sliderInput widget in shiny"),
5   sidebarLayout(
6     sidebarPanel(
7       sliderInput("slide", "Select the value from Slider", min = 0, max=5, value=2, step=0.2)
8     ),
9     mainPanel(
10    textOutput("out")
11  )
12 )
13 )
14 )
15 )
16 )
17 ))|
```

The screenshot shows the RStudio interface with the 'server.r' file open. The code defines a shinyServer function that takes input and output parameters. It uses renderText to output the selected value from the slider. The 'Find' and 'Replace' toolbars are visible at the top of the editor area.

```
1 library(shiny)
2 shinyServer(function(input, output){
3
4   output$out <- renderText(
5     paste("You selected the value: ", input$slide))
6
7 })|
```

Shiny Video 6



```
library(shiny)
shinyServer(function(input,output){
  output$state <- renderText(input$statenames)
})
```



~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser | Public

Demostration of the selectInput UI widget in shiny

Select the state

Texas

Texas

Select the quantitative Variable

Petal.Length

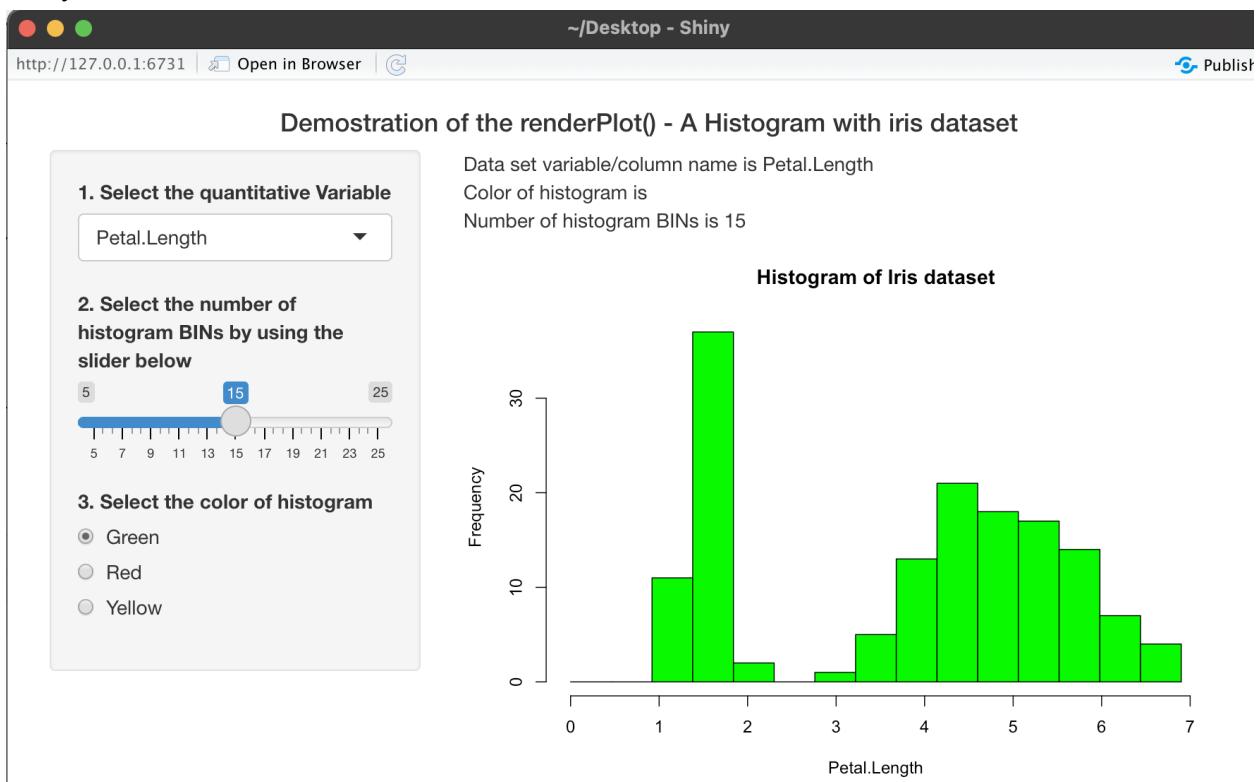
Sepal.Length

Sepal.Width

Petal.Length

Petal.Width

Shiny Video 7



The screenshot shows the RStudio interface with two code files open: `server.r` and `ui.r`. The `server.r` file contains R code for a shiny application, specifically for generating histograms. The `ui.r` file is currently closed or not visible.

server.r Content:

```
8 ~ function(input, output) {
9
10 ~   output$text1 <- renderText({
11     colm = as.numeric(input$var)
12     paste("Data set variable/column name is", names(iris[colm]))
13   })
14 ~
15
16 ~   output$text2 <- renderText({
17     paste("Color of histogram is", input$radio)
18   })
19
20 ~   output$text3 <- renderText({
21     paste("Number of histogram BINs is", input$bin)
22   })
23
24   output$myhist <- renderPlot(
25
26   {
27     colm = as.numeric(input$var)
28     hist(iris[,colm], col =input$colour, xlim = c(0, max(iris[,colm])), main = "Histogram of Iris dataset", breaks = seq(0, max(iris[,colm]), 1))
29   }
30 }
```

R Console Output:

```
R 4.1.1 · ~/ ◁
`citation()` on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
> shiny::runApp('Desktop')
Loading required package: shiny

Listening on http://127.0.0.1:6731
shinyUI(fluidPage(
# Define UI for the shiny application here
```

Shiny Video 8

```
server.r ui.r
library(shiny)
# Define UI for application
shinyUI(fluidPage(
  # Header or Title Panel
  titlePanel(title = h4("Iris Dataset", align="center")),
  sidebarLayout(
    # Sidebar panel
    sidebarPanel(
      selectInput("var", "1. Select the variable from the iris dataset", choices =c("Sepal.Length" = 1, "Sepal.Width" = 2, "Petal.Length" = 3, "Petal.Width" = 4),
      br(),
      sliderInput("bins", "2. Select the number of BINs for histogram", min=5, max = 25, value=15),
      br(),
      radioButtons("color", "3. Select the colour of histogram", choices=c("Green", "Red", "Yellow"), selected ="Green")
    ),
    # Main Panel
    mainPanel(
      tabsetPanel(type="tab",
        tabPanel("Summary", verbatimTextOutput("sum")),
        tabPanel("Structure", verbatimTextOutput("str")),
        tabPanel("Data", tableOutput("data")),
        tabPanel("Plot", plotOutput("myhist"))
      )
    )
  )
))

```

~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser | Publish

Iris Dataset

1. Select the variable from the iris dataset

Sepal.Length

2. Select the number of BINs for histogram

15

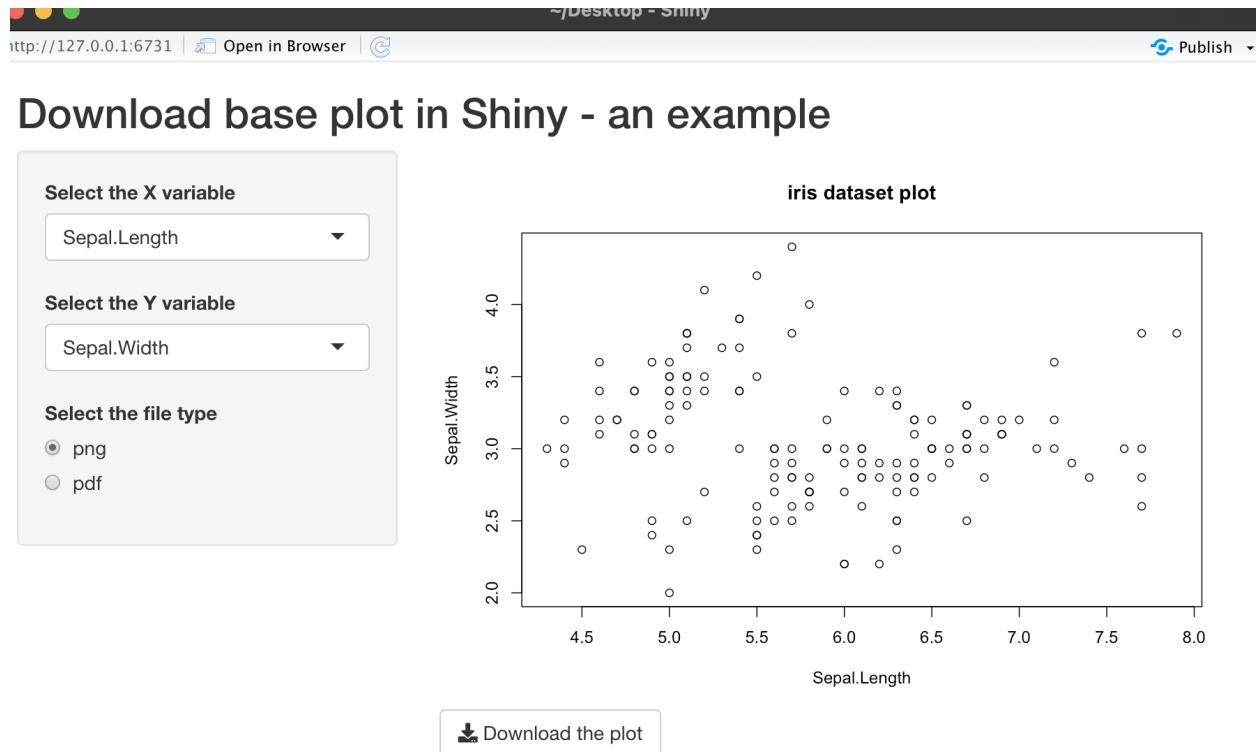
3. Select the colour of histogram

Green

Summary Structure Data Plot

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min.	:4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa
1st Qu.	:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor
Median	:5.800	Median :3.000	Median :4.350	Median :1.300	virginica
Mean	:5.843	Mean :3.057	Mean :3.758	Mean :1.199	
3rd Qu.	:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max.	:7.900	Max. :4.400	Max. :6.900	Max. :2.500	

How to download a base plot using the downloadButton() and downloadHandler() functions.



```
library(shiny)
shinyUI(fluidPage(
  titlePanel("Download base plot in Shiny - an example"),
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "var1", label = "Select the X variable", choices = c("Sepal.Length" = 1, "Sepal.Width" = 2, "Petal.Length" = 3, "Petal.Width" = 4)),
      selectInput(inputId = "var2", label = "Select the Y variable", choices = c("Sepal.Length" = 1, "Sepal.Width" = 2, "Petal.Length" = 3, "Petal.Width" = 4)),
      radioButtons(inputId = "var3", label = "Select the file type", choices = list("png", "pdf"))
    ),
    mainPanel(
      plotOutput("plot"),
      downloadButton(outputId = "down", label = "Download the plot")
    )
  )
))
```

```
1 library(shiny)
2 shinyServer(function(input,output){
3   # x contains all the observations of the x variable selected by the user. X is a reactive function
4   x <- reactive({
5     iris[,as.numeric(input$var1)]
6   })
7   # x contains all the observations of the y variable selected by the user. Y is a reactive function
8   y <- reactive({
9     iris[,as.numeric(input$var2)]
10  })
11  # xl contains the x variable or column name of the iris dataset selected by the user
12  xl <- reactive({
13    names(iris[,as.numeric(input$var1)])
14  })
15  # yl contains the y variable or column name of the iris dataset selected by the user
16  yl <- reactive({
17    names(iris[,as.numeric(input$var2)])
18  })
19  # render the plot so could be used to display the plot in the mainPanel
20  output$plot <- renderPlot({
21    plot(x=x(), y=y(), main = "iris dataset plot", xlab = xl(), ylab = yl())
22  })
```

The use of submitButton() in shiny:

The screenshot shows a Shiny application window titled "~/Desktop - Shiny". The URL is "http://127.0.0.1:6731". There are buttons for "Open in Browser" and "Publish".

The application title is "Demonstration of submitButton() in shiny".

The UI consists of two input fields and a button:

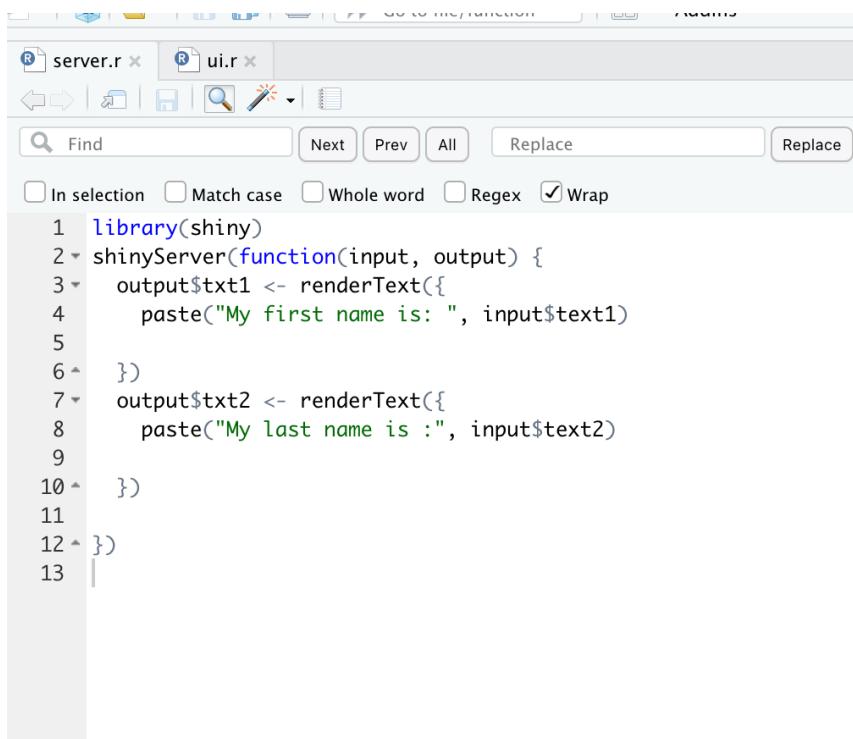
- Enter your first name:** An input field containing "Ravleen".
- Enter your last name:** An input field containing "Kaur".
- Update!**: A blue button.

To the right of the inputs, the output is displayed:

My first name is: Ravleen
My last name is : Kaur

A descriptive text below the button explains its function:

Click on the Update button to display the first and last name entered by the user. Here the reactivity of the input widget is controlled by submitButton



The screenshot shows a code editor interface with two tabs at the top: "server.r" and "ui.r". The "server.r" tab is active. Below the tabs is a toolbar with icons for back, forward, search, and other file operations. The main area is a search bar with fields for "Find", "Replace", and checkboxes for "In selection", "Match case", "Whole word", "Regex", and "Wrap". The "Wrap" checkbox is checked. Below the search bar is the R code for a shiny server:

```
1 library(shiny)
2 shinyServer(function(input, output) {
3   output$txt1 <- renderText({
4     paste("My first name is: ", input$text1)
5
6   })
7   output$txt2 <- renderText({
8     paste("My last name is : ", input$text2)
9
10  })
11
12})
13
```

Another use of submitButton() in shiny.

~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser | Publish

Demonstration of submitButton()

Choose a dataset:

Number of observations:

Update!

In this example, changing the user input (dataset or number of observation) will not reflect in the output until the Update button is clicked

submitButton is used to control the reactivity of the change in the user input

Structure of the dataset iris

```
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 ...'
```

First 6 observations of the dataset iris

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.10	3.50	1.40	0.20	setosa
4.90	3.00	1.40	0.20	setosa
4.70	3.20	1.30	0.20	setosa
4.60	3.10	1.50	0.20	setosa
5.00	3.60	1.40	0.20	setosa
5.40	3.90	1.70	0.40	setosa

server.r x ui.r x

Go to file/function Reload App

```
1 library(shiny)
2 # Define UI for dataset viewer application
3 shinyUI(fluidPage(
4
5   # Application title.
6   titlePanel("Demonstration of submitButton()"),
7
8   sidebarPanel(
9     # selectInput widget for the selection of dataset
10    selectInput("dataset", "Choose a dataset:",
11      choices = c("iris", "pressure", "mtcars")),
12
13    # numericInput for selection of the number of observation that user wants to view
14    numericInput("obs", "Number of observations:", 6),
15    # submitButton to create dependency of the reactivity on the event of pressing of the submitbutton.
16    submitButton("Update!"),
17    p("In this example, changing the user input (dataset or number of observation) will not reflect in the output until the Update button is pressed"),
18    p("submitButton is used to control the reactivity of the change in the user input")
19  ),
20  mainPanel(
21    # just a header for the heading
22    h4(textOutput("dataname")),
23    # display the structure of the selected dataset is dependent of the submit button
24    verbatimTextOutput("structure"),
25
26    # just a header for the heading
27    h4(textOutput("observation"))
28  )
29))
```

31:3 (Top Level) R Script

How to use renderUI() and uiOutput() functions to create the tabs dynamically based on the number of tabs selected by the user

~/Desktop - Shiny

<http://127.0.0.1:6731> | [Open in Browser](#) | [C](#)

[Publish](#)

Demonstration of renderUI in shiny - Dynamically creating the tabs based on user inputs

Enter the number of tabs needed

[tab no. 1](#)
[tab no. 2](#)
[tab no. 3](#)
[tab no. 4](#)

server.r x ui.r x

Find Next Prev All Replace All

In selection Match case Whole word Regex Wrap

```

1 library(shiny)
2 shinyServer(function(input,output){
3
4   output$tabs = renderUI({
5
6     # usual syntax of tabset -
7     ## tabsetPanel(tabPanel(title="")) ## for a single tab in the tabsetPanel,
8     ## tabsetPanel(tabPanel(title=""), tabPanel(title="")) ## for two tabs in the tabsetPanel..and so on ..
9     ## tabsetPanel(tabPanel(title=""), tabPanel(title=""),....., tabPanel(title="")) ## for n tabs in the tabsetPanel..and so on ..
10    #   imagine tab-1,tab no 1, tab-2, tab-3..tabn - these are the tab titles
11    #   using lapply(), we will apply the tabPanel function on each of tab title to get a list of tabPanel(s)
12    #   paste("tab no.", 1:input$n, sep="-")
13    Tabs <- lapply(paste("tab no.", 1:input$n, sep=" "), tabPanel)
14
15    # now we have the list of arguments and we can use the tabsetPanel function on this list of arguments and this could be achieved using
16    # do.call function allows you to call any R function, but instead of writing out the arguments one by one, you can use a list to hold t
17
18    do.call(tabsetPanel, Tabs)
19
20  })
21
22
23

```

Reload App

How to use the `fileInput()` function in RShiny to add a file upload option to a shiny app

```

1 library(shiny)
2 # use the below options code if you wish to increase the file input limit, in this example file input limit is increased from 5MB to 9*1024^2
3 # options(shiny.maxRequestSize = 9*1024^2)
4
5 shinyServer(function(input,output){
6
7   # This reactive function will take the inputs from UI.R and use them for read.table() to read the data from the file. It returns the
8   # file$datapath -> gives the path of the file
9   data <- reactive({
10     file1 <- input$file
11     if(is.null(file1)){return()}
12     read.table(file=file1$datapath, sep=input$sep, header = input$header, stringsAsFactors = input$stringAsFactors)
13   })
14 }
15
16 # this reactive output contains the summary of the dataset and display the summary in table format
17 output$filedf <- renderTable({
18   if(is.null(data)){return ()}
19   input$file
20 })
21
22 # this reactive output contains the summary of the dataset and display the summary in table format
23 output$sum <- renderTable({
24 })

```

(Top Level) R

```

1 library(shiny)
2 shinyUI(fluidPage(
3   titlePanel("File Input"),
4   sidebarLayout(
5     sidebarPanel(
6       fileInput("file","Upload the file"), # fileinput() function is used to get the file upload control option
7       helpText("Default max. file size is 5MB"),
8       tags$hr(),
9       h5(helpText("Select the read.table parameters below")),
10      checkboxInput(inputId = 'header', label = 'Header', value = FALSE),
11      checkboxInput(inputId = "stringAsFactors", "stringAsFactors", FALSE),
12      br(),
13      radioButtons(inputId = 'sep', label = 'Separator', choices = c(Comma=',',Semicolon=';',Tab='\t', Space=''), selected = ',')
14    ),
15    mainPanel(
16      uiOutput("tb")
17    )
18    # use below code if you want the tabset programming in the main panel. If so, then tabset will appear when the app loads for the first time
19    #   tabsetPanel(tabPanel("Summary", verbatimTextOutput("sum")),
20    #             tabPanel("Data", tableOutput("table")))
21  )
22
23 )
24 ))

```

24:3 (Top Level) R Script



File Input

Upload the file[Browse...](#)

No file selected

Powered by

Default max. file size is 5MB

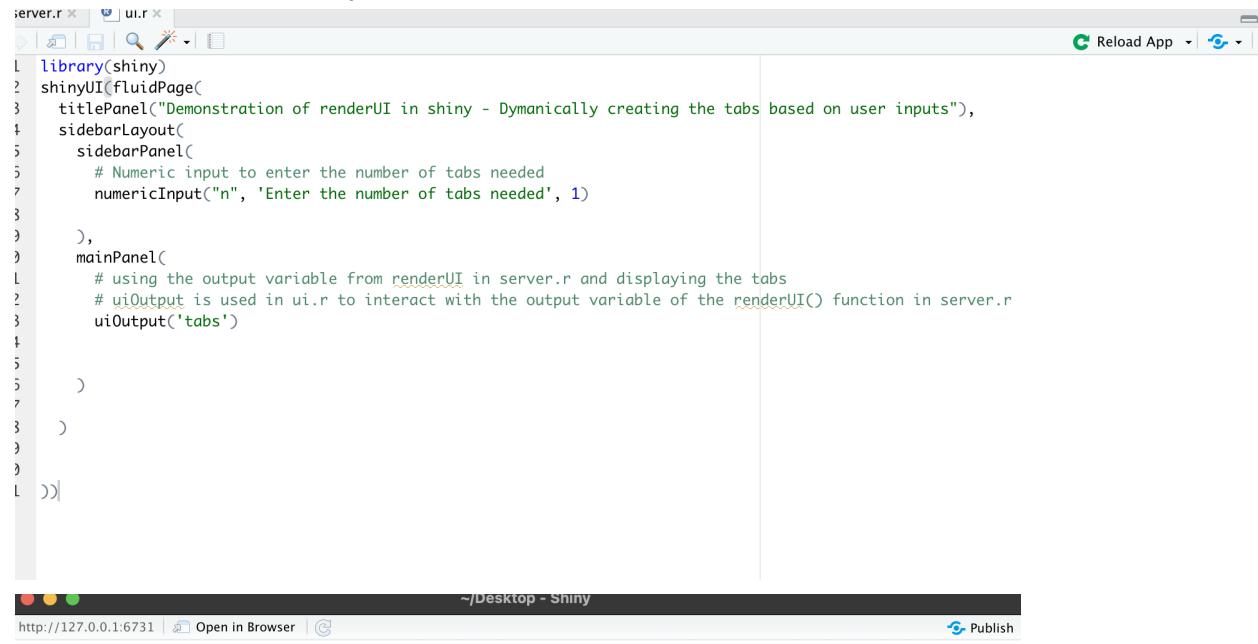
Select the read.table parameters below

- Header
- stringAsFactors

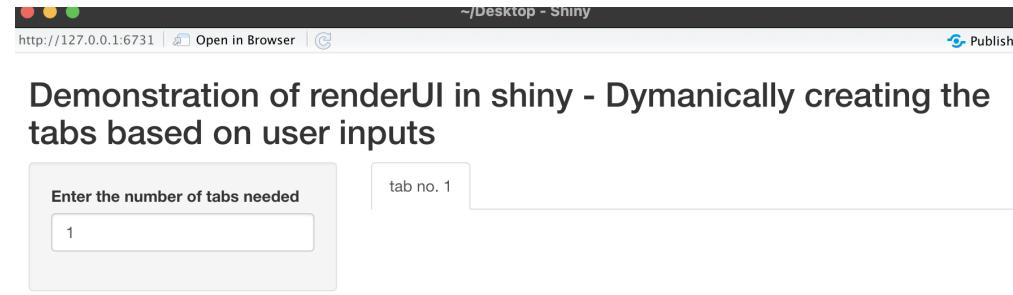
Separator

- Comma
- Semicolon
- Tab
- Space

how to use renderUI() and uiOutput() functions to create the tabs dynamically based on the number of tabs selected by the user.



```
server.r ui.r
library(shiny)
shinyUI(fluidPage(
  titlePanel("Demonstration of renderUI in shiny - Dynamically creating the tabs based on user inputs"),
  sidebarLayout(
    sidebarPanel(
      # Numeric input to enter the number of tabs needed
      numericInput("n", "Enter the number of tabs needed", 1)
    ),
    mainPanel(
      # using the output variable from renderUI in server.r and displaying the tabs
      # uiOutput is used in ui.r to interact with the output variable of the renderUI() function in server.r
      uiOutput('tabs')
    )
  )
))|
```



~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser | Publish

Demonstration of renderUI in shiny - Dynamically creating the tabs based on user inputs

Enter the number of tabs needed

tab no. 1

How to upload multiple files using fileinput() in R Shiny (1)

~/Desktop - Shiny

http://127.0.0.1:6731 | Open in Browser |

[Publish](#)

Demo - File Input - Upload multiple files

Upload the file

Browse... Screen Shot 2021-1
Upload complete

Default max. file size is 5MB

Select the read.table parameters below

Header

stringAsFactors

Separator

Comma

Semicolon

Tab

Space

Select the files for which you need to see data and summary stats

Select

Screen Shot 2021-12-21 at 2.35.13 PM.png ▾

name	size	type	datapath
Screen Shot 2021-12-21 at 2.35.13 PM.png	177909	image/png	/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//RtmpxLMpu9/77c9c1824395f9632a32c

Input File Object DF [Input File Object Structure](#) [Dataset](#) [Summary Stats](#)

data

```
/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//RtmpxLMpu9/77c9c1824395f9632a32c
```

In selection Match case Whole word Regex Wrap

```

1 library(shiny)
2 # use the below options code if you wish to increase the file input limit, in this example file input limit is increased from 5MB to 9MB
3 # options(shiny.maxRequestSize = 9*1024^2)
4
5 ~ shinyServer(function(input,output) {
6
7   ## input$file is a data frame and contains the details around the name, size and temp location of the files uploaded
8   # this reactive output display the content of the input$file dataframe
9 ~ output$filedf <- renderTable({
10   if(is.null(input$file)){return ()}
11   input$file # the file input data frame object that contains the file attributes
12 ~ })
13
14   # Extract the file path for file
15 ~ output$filedf2 <- renderTable({
16   if(is.null(input$file)){return ()}
17   input$file$datapath # the file input data frame object that contains the file attributes
18 ~ })
19
20   ## Below code to display the structure of the input file object
21 ~ output$fileob <- renderPrint({
22   if(is.null(input$file)){return ()}
23   str(input$file)
24 ~ })
25
26   ## Side bar select input widget coming through renderUI()
27   # Following code displays the select input widget with the list of file loaded by the user
28 ~ output$selectfile <- renderUI({
29   if(is.null(input$file)) {return ()}
30   list(hr(),
31     helpText("Select the files for which you need to see data and summary stats"),
32     selectInput("Select", "Select", choices=input$file$name)
33   )
34 ~ })
35 ~ })

```

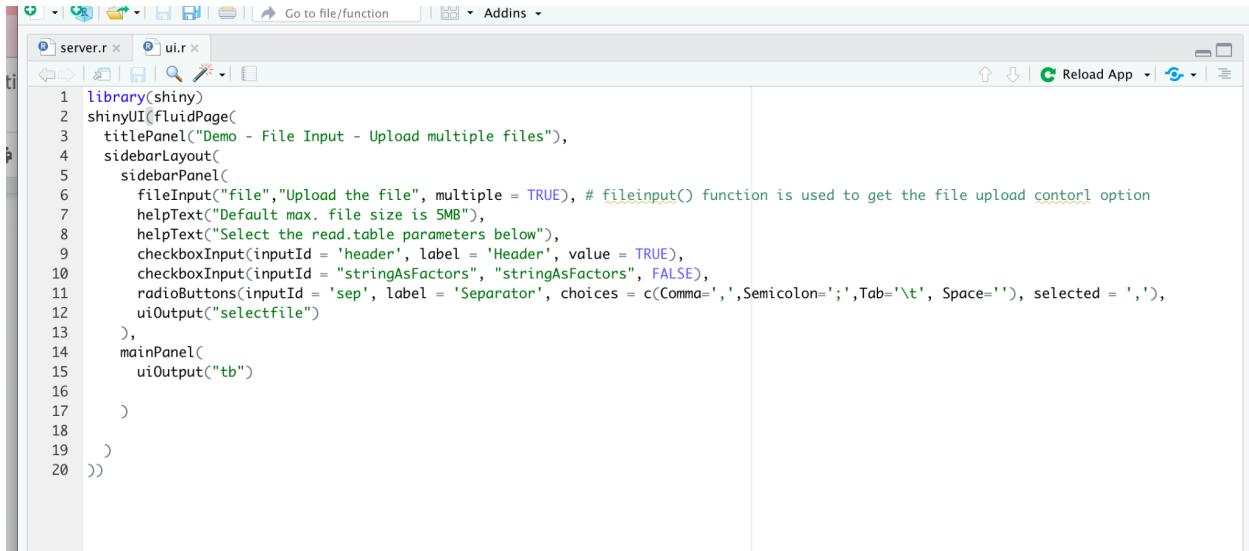
66:3 (Top Level) R Script

The screenshot shows a Shiny application interface with four tabs at the top: 'Input File Object DF', 'Input File Object Structure', 'Dataset', and 'Summary Stats'. The 'Input File Object DF' tab is currently selected. Below the tabs, there is a code editor window containing R code. The output of this code is displayed in a box below, showing the structure of the input file object.

```

'data.frame': 1 obs. of 4 variables:
 $ name      : chr "Screen Shot 2021-12-21 at 2.35.13 PM.png"
 $ size      : int 177909
 $ type      : chr "image/png"
 $ datapath: chr "/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//Rtmp"

```



The screenshot shows the RStudio interface with two tabs open: 'server.r' and 'ui.r'. The 'ui.r' tab contains the following R code:

```
1 library(shiny)
2 shinyUI(fluidPage(
3   titlePanel("Demo - File Input - Upload multiple files"),
4   sidebarLayout(
5     sidebarPanel(
6       fileInput("file","Upload the file", multiple = TRUE), # fileinput() function is used to get the file upload control option
7       helpText("Default max. file size is 5MB"),
8       helpText("Select the read.table parameters below"),
9       checkboxInput(inputId = 'header', label = 'Header', value = TRUE),
10      checkboxInput(inputId = "stringAsFactors", "stringAsFactors", FALSE),
11      radioButtons(inputId = 'sep', label = 'Separator', choices = c(Comma=',',Semicolon=';',Tab='\t', Space=''), selected = ','),
12      uiOutput("selectfile")
13    ),
14    mainPanel(
15      uiOutput("tb")
16    )
17  )
18 )
19 ))
20 ))
```

Demo rowbind data from multiple uploaded files

multiple files

Upload the file

Screen Shot 2021-1

Default max. file size is 5MB

Select the read.table parameters below

Header

stringAsFactors

Separator

Comma

Semicolon

Tab

Space

Select the files for which you need to see data and summary stats

Select

Screen Shot 2021-12-21 at
2.31.24 PM.png

[Input File Object DF](#)

[Input File Object Structure](#)

[Dataset](#)

[Summary Stats](#)

Merged Dataset

```
'data.frame': 1 obs. of 4 variables:  
 $ name    : chr "Screen Shot 2021-12-21 at 2.31.24 PM.png"  
 $ size    : int 159328  
 $ type    : chr "image/png"  
 $ datapath: chr "/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//Rtn"
```

```

98
99  ## Summary Stats code ##
100 # This reactive output contains the summary of the dataset and display the summary in table format
101 ~ output$summ <- renderPrint({
102   if(is.null(input$file)){return()}
103   summary(read.table(file=input$file$datapath[input$file$name==input$select],
104                     sep=input$sep,
105                     header = input$header,
106                     stringsAsFactors = input$stringAsFactors)})
107
108
109
110 ## Dataset code ##
111 # This reactive output contains the dataset and display the dataset in table format
112 ~ output$table <- renderTable({
113   if(is.null(input$file)){return()}
114   read.table(file=input$file$datapath[input$file$name==input$select], sep=input$sep, header = input$header, stringsAsFactors
115
116 })
117
118 ## MainPanel tabset renderUI code ##
119 # the following renderUI is used to dynamically generate the tabs when the file is loaded.
120 # Until the file is loaded, app will not show the tabset.
121 ~ output$tb <- renderUI({
122   if(is.null(input$file)) {return()}
123   else
124     tabsetPanel(
125       tabPanel("Input File Object DF ", tableOutput("filedf"), tableOutput("filedf2")),
126       tabPanel("Input File Object Structure", verbatimTextOutput("fileob")),
127       tabPanel("Dataset", tableOutput("table")),
128       tabPanel("Summary Stats", verbatimTextOutput("summ")),
129       tabPanel("Merged Dataset", downloadButton("download", "Download"), tableOutput("newdata")))
130     )
131 ~ })
132 ~ })
132:3 | (Top Level) ▾

```

[Input File Object DF](#)

[Input File Object Structure](#)

[Dataset](#)

[Summary Stats](#)

Merged Dataset

name	size	type	datapath
Screen Shot 2021-12-21 at 2.31.24 PM.png	159328	image/png	/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//RtmpxLMpu9/25dfcf186d69253f8cb07146/C

data

/var/folders/6r/5cl8vzcn2_l3qxg48sb0llqw0000gn/T//RtmpxLMpu9/25dfcf186d69253f8cb07146/0.png

upload a zip file and unzip it to local machine

Upload Zip file

[Browse...](#)

MyData.zip

Upload complete

[Unzip Files](#)

name	size	type	data
MyData.zip	394	application/x-zip-compressed	C:\U

```
1 library(shiny)
2 shinyUI(fluidPage(
3   titlePanel("File Input"),
4   sidebarLayout(
5     sidebarPanel(
6       fileInput("file","Upload the file"), # fileinput() function is used to get the file upload control option
7       helpText("Default max. file size is 5MB"),
8       tags$hr(),
9       h5(helpText("Select the read.table parameters below")),
10      checkboxInput(inputId = 'header', label = 'Header', value = FALSE),
11      checkboxInput(inputId = "stringAsFactors", "stringAsFactors", FALSE),
12      br(),
13      radioButtons(inputId = 'sep', label = 'Separator', choices = c(Comma=',',Semicolon=';',Tab='\t', Space=''), selected = ',')
14    ),
15    mainPanel(
16      uiOutput("tb")
17
18      # use below code if you want the tabset programming in the main panel. If so, then tabset will appear when the app loads for the fi
19      #         tabsetPanel(tabPanel("Summary", verbatimTextOutput("sum")),
20      #                     tabPanel("Data", tableOutput("table")))
21    )
22  )
23 ))
24 )
```

File Input

Upload the file

Browse...

No file selected

Powered by 

Default max. file size is 5MB

Select the read.table parameters below

Header

stringAsFactors

Separator

Comma

Semicolon

Tab

Space