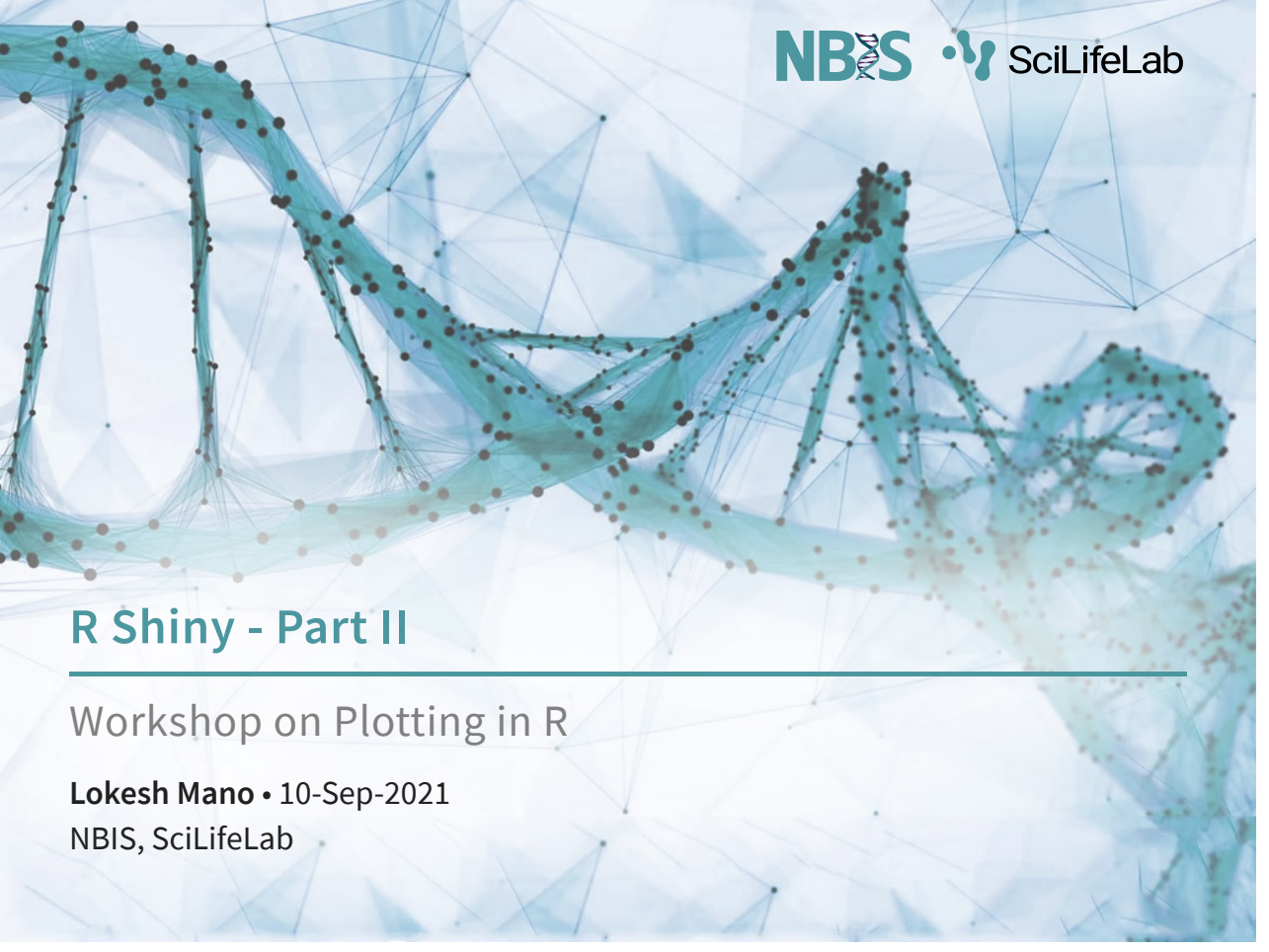


R Shiny - Part II

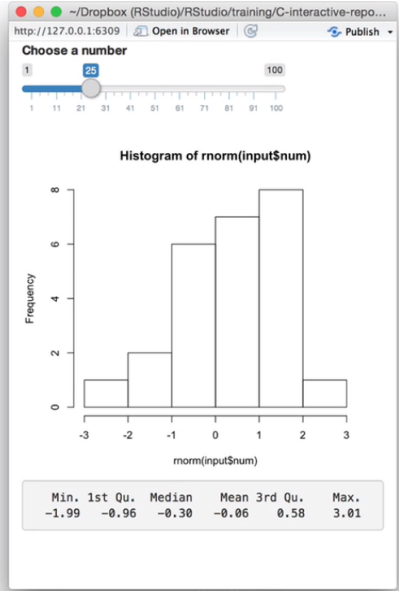
Workshop on Plotting in R

Lokesh Mano • 10-Sep-2021
NBIS, SciLifeLab

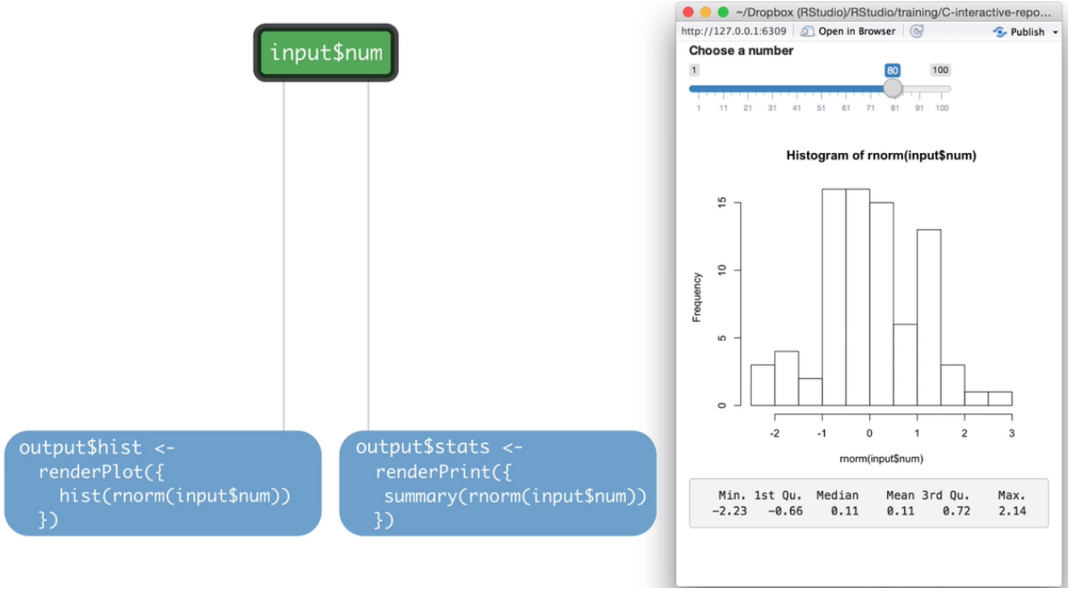
Contents

Reactivity

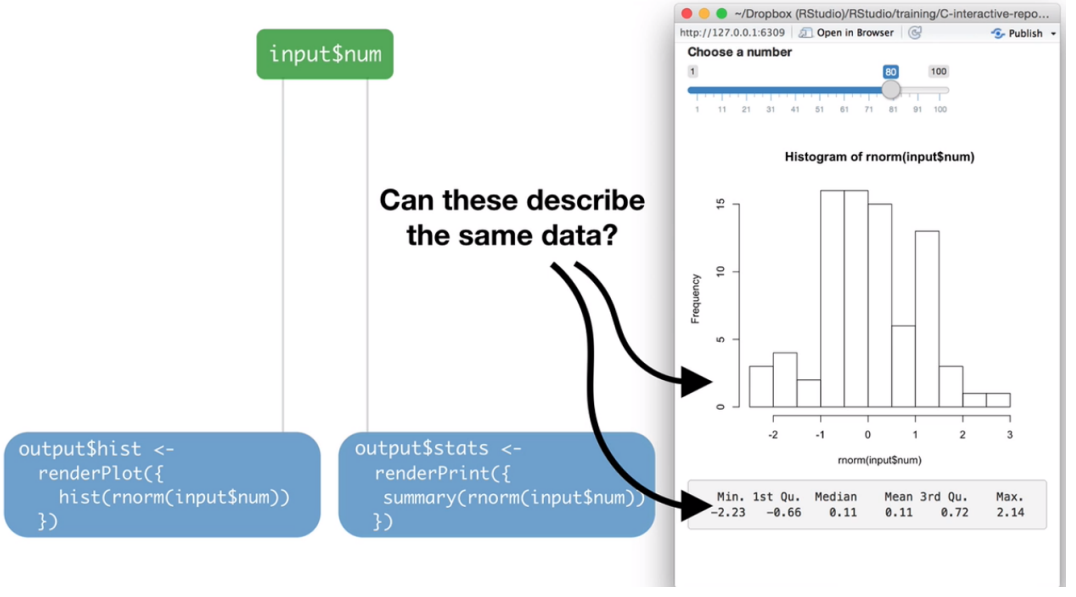
```
# 02-two-outputs

library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist"),
  verbatimTextOutput("stats")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
  output$stats <- renderPrint({
    summary(rnorm(input$num))
  })
}

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



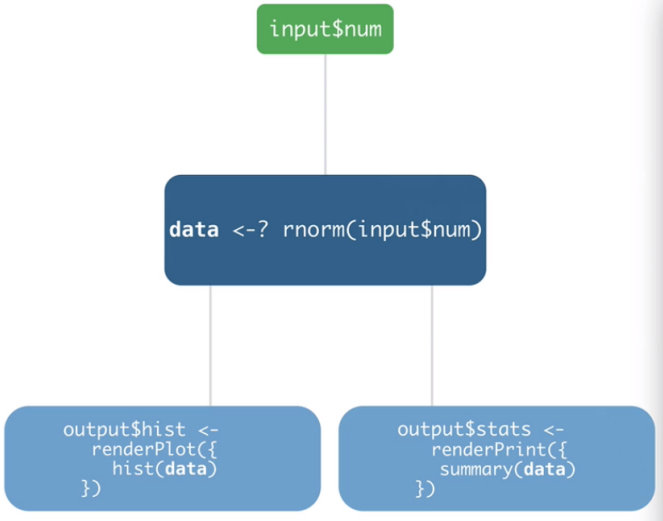
Reactivity

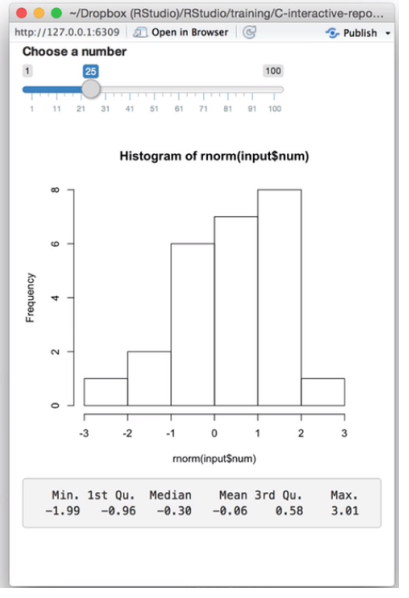

input$num
output$hist <-
renderPlot({
hist(rnorm(input$num))
})
output$stats <-
renderPrint({
summary(rnorm(input$num))
})
~/Dropbox (RStudio)/RStudio/training/C-interactive-repo...
http://127.0.0.1:6309
Open in Browser
Publish
Choose a number
1
80
100
Histogram of rnorm(input$num)
Frequency
rnorm(input$num)
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.23 -0.66 0.11 0.11 0.72 2.14

Reactivity


input$num
Can these describe the same data?
output$hist <-
renderPlot({
hist(rnorm(input$num))
})
output$stats <-
renderPrint({
summary(rnorm(input$num))
})
~/Dropbox (RStudio)/RStudio/training/C-interactive-repo...
http://127.0.0.1:6309
Open in Browser
Publish
Choose a number
Histogram of rnorm(input$num)
Frequency
rnorm(input$num)
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.23 -0.66 0.11 0.11 0.72 2.14

Reactivity


input$num
data <-? rnorm(input$num)
output$hist <-
renderPlot({
hist(data)
})
output$stats <-
renderPrint({
summary(data)
})


~/Dropbox (RStudio)/RStudio/training/C-interactive-repo...
http://127.0.0.1:6309
Open in Browser
Publish
Choose a number
1
25
100
Histogram of rnorm(input$num)
Frequency
rnorm(input$num)
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.99 -0.96 -0.30 -0.06 0.58 3.01

Reactivity

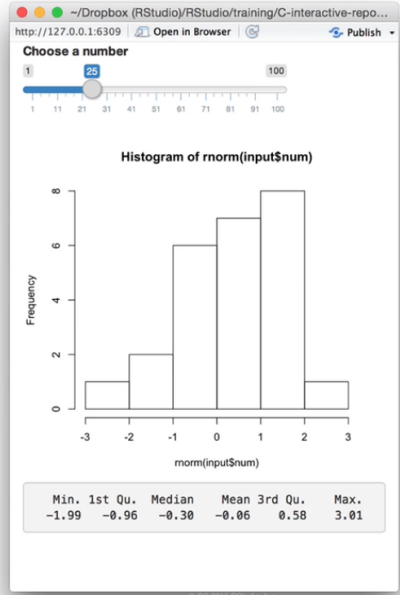
reactive()

Builds a reactive object (reactive expression)

```
data <- reactive(   { rnorm(input$num) })
```

Reactivity

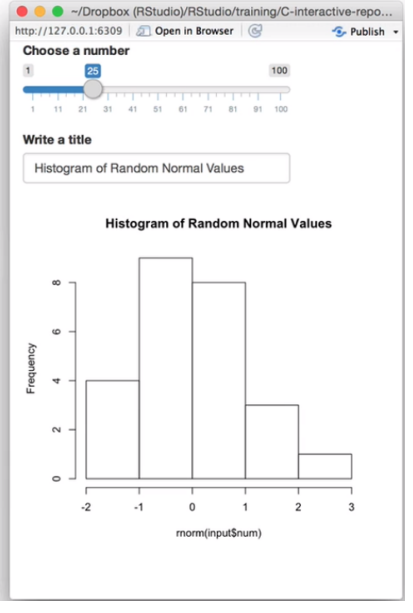
```
# 03-reactive

library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist"),
  verbatimTextOutput("stats")
)

server <- function(input, output) {
  data <- reactive({
    rnorm(input$num)
  })
  output$hist <- renderPlot({
    hist(data())
  })
  output$stats <- renderPrint({
    summary(data())
  })
}

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



Isolate reactivity

- Reactivity can be controlled.
- You will notice that as soon as you try to change the title, the histogram will update with new values

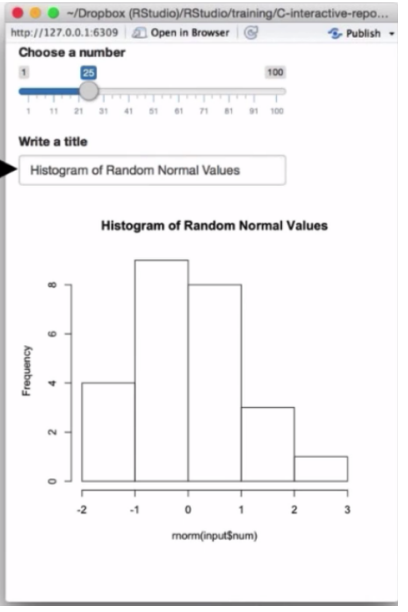
```
# 01-two-inputs

library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  textInput(inputId = "title",
    label = "Write a title",
    value = "Histogram of Random Normal Values"),
  plotOutput("hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num),
      main = input$title)
  })
}

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



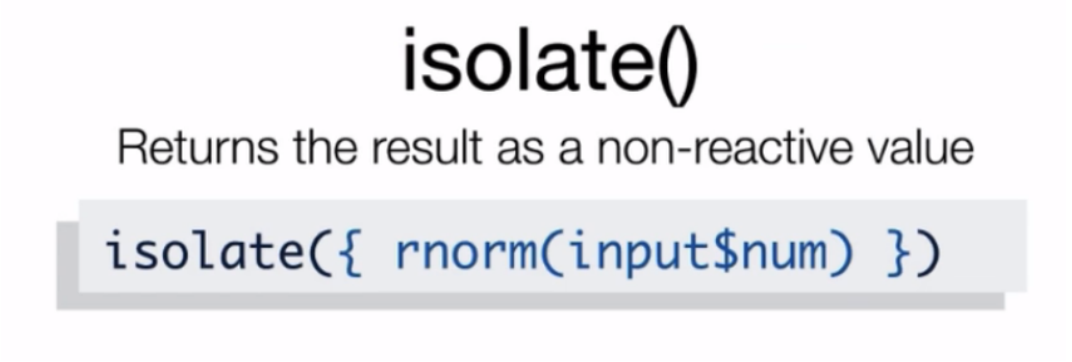
Isolate reactivity

```
# 01-two-inputs

library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  textInput(inputId = "title",
    label = "Write a title",
    value = "Histogram of Random Normal Values"),
  plotOutput("hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num),
      main = input$title)
  })
}

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

Can we prevent the title field from updating the plot?

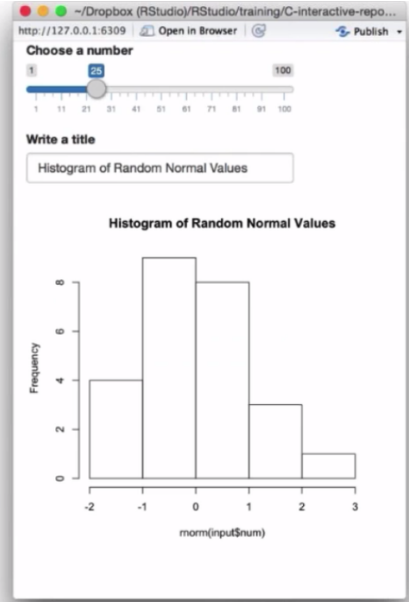


Isolate reactivity



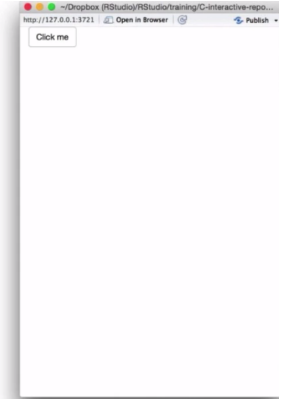
Isolate reactivity

```
# 04-isolate

library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  textInput(inputId = "title",
    label = "Write a title",
    value = "Histogram of Random Normal Values"),
  plotOutput("hist")
)

server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num),
      main = isolate({input$title}))
  })
}

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



observeEvent()

```
# 05-actionButton

library(shiny)

ui <- fluidPage(
  actionButton(inputId = "clicks",
    label = "Click me")
)

server <- function(input, output) {
  observeEvent(input$clicks, {
    print(as.numeric(input$clicks))
  })
}

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



Updating widgets

- Widgets can be updated once initialised.
- Add third argument **session** to server function

```
server=function(input,output,session) {}
```

- Example of a typical UI

```
ui=fluidPage(
  selectInput("select-input",label="selectInput",choices=c("A","B","C")),
  numericInput("numeric-input",label="numericInput",value=5,min=1,max=10),
  sliderInput("slider-input",label="sliderInput",value=5,min=1,max=10),
)
```

- Update functions can be used to update input widgets
- Reactive observer `observe({})` monitors for a conditional change

```
server=function(input,output,session) {
  observe({
    if(something) {
      updateSelectInput(session,"select-input",label="selectInput",choices=c("D","E","F")
      updateNumericInput(session,"numeric-input",label="numericInput",value=10,min=1,max=
      updateSliderInput(session,"slider-input",label="sliderInput",value=8,min=1,max=10)
    }
  })
}
```

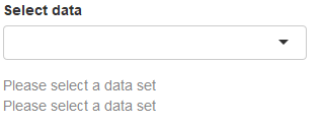
Error validation

- Shiny returns an error with missing or incorrect values

```
shinyApp(
ui=fluidPage(
  selectInput("data_input",label="Select data",
              choices=c("","mtcars","faithful","iris")),
```

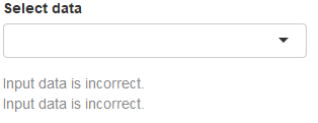
- Errors can be handled in a controlled manner
- `validate()` can be used to check input
- `validate()` using `need()`

```
shinyApp(
ui=fluidPage(
  selectInput("data_input",label="Select data",
              choices=c("","mtcars","faithful","iris")),
```



- `validate()` using custom function

```
valfn <- function(x) if(is.null(x) | is.na(x) | x=="") retu
shinyApp(
ui=fluidPage(
  selectInput("data_input",label="Select data",
```



- `shiny::req()` checks input variable and silently stops execution

Download • Data

- Add button and `downloadHandler()` function

```
shinyApp(
ui=fluidPage(
  selectInput("data_input",label="Select data",
              choices=c("mtcars","faithful","iris")),
  textOutput("text_output"),
  downloadButton("button_download","Download")
),
server=function(input, output) {
  getdata <- reactive({ get(input$data_input, 'package:datasets') })
  output$text_output <- renderText(paste0("Selected dataset: ",input$data_input))

  output$button_download <- downloadHandler(
  filename = function() {
     paste0(input$data_input,".csv")
   },
  content = function(file) {
     write.csv(getdata(),file,row.names=FALSE,quote=F)
   })
})
```

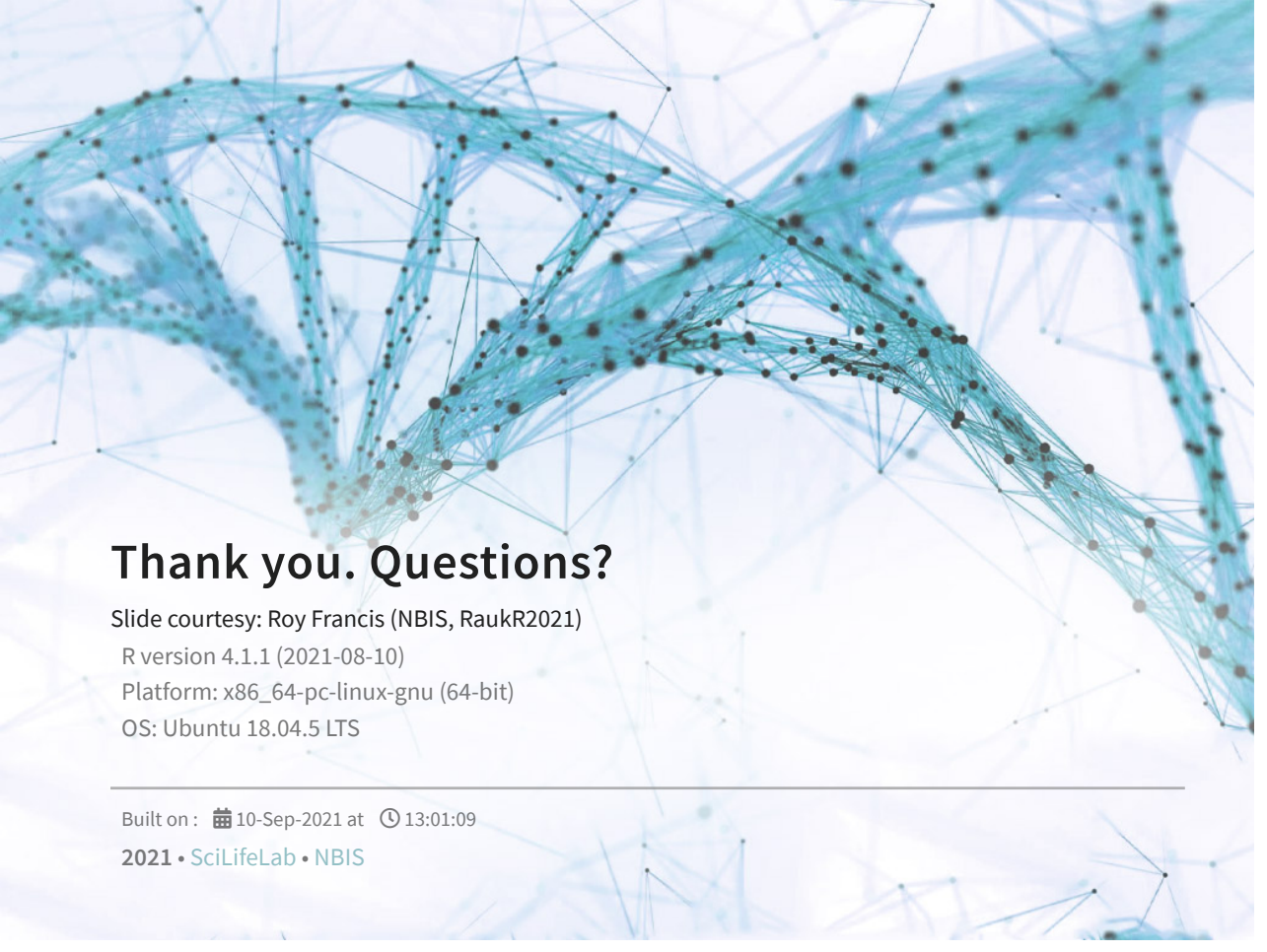
- Run in system browser if Rstudio browser doesn't work
- See usage of download buttons

Download • Plots

NBIS SciLifeLab

```
shinyApp(
ui=fluidPage(
  selectInput("data_input",label="Select data",
              choices=c("mtcars","faithful","iris")),
  textOutput("text_output"),
  plotOutput("plot_output",width="400px"),
  downloadButton("button_download", "Download")
),
server=function(input, output) {
  getdata <- reactive({ get(input$data_input, 'package:datasets') })
  output$text_output <- renderText(paste0("Selected dataset: ",input$data_input))
  output$plot_output <- renderPlot({hist(getdata()[, 1])})

output$button_download <- downloadHandler(
  filename = function() {
    paste0(input$data_input,".png")
  },
  content = function(file) {
    png(file)
    hist(getdata()[, 1])
    dev.off()
  })
})
```

- Run in system browser if Rstudio browser doesn't work
- See usage of download buttons



Thank you. Questions?

Slide courtesy: Roy Francis (NBIS, RaukR2021)

R version 4.1.1 (2021-08-10)

Platform: x86_64-pc-linux-gnu (64-bit)

OS: Ubuntu 18.04.5 LTS

Built on : 10-Sep-2021 at 13:01:09

2021 • SciLifeLab • NBIS