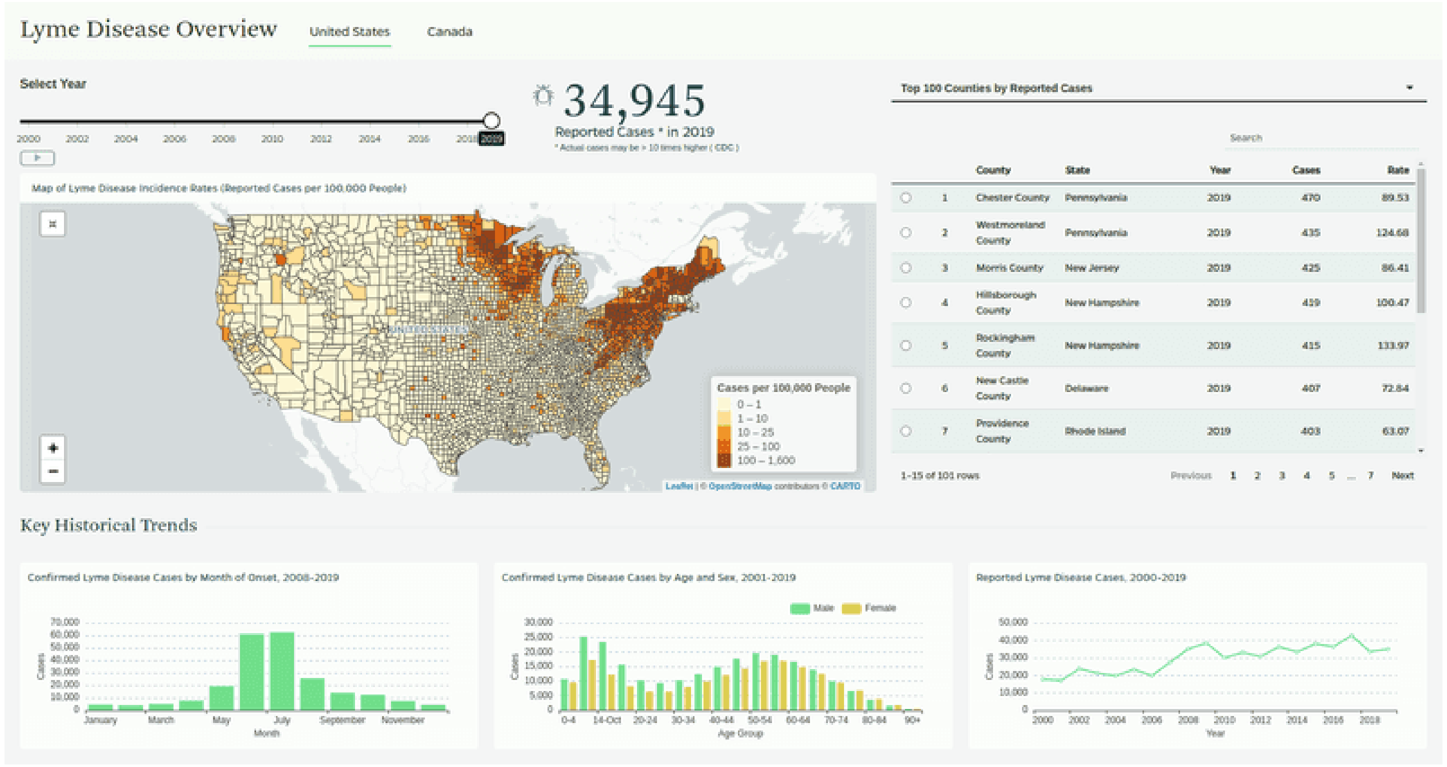
Shiny Apps

Von Sebastian Fay


Lyme Disease Overview

34,945

Reported Cases * in 2019

* Actual cases may be > 10 times higher (CDC)

Map of Lyme Disease Incidence Rates (Reported Cases per 100,000 People)

Top 100 Counties by Reported Cases

		County	State	Year	Cases	Rate
○	1	Chester County	Pennsylvania	2019	470	89.53
○	2	Westmoreland County	Pennsylvania	2019	435	124.68
○	3	Morris County	New Jersey	2019	425	86.41
○	4	Hillsborough County	New Hampshire	2019	419	100.47
○	5	Rockingham County	New Hampshire	2019	415	133.97
○	6	New Castle County	Delaware	2019	407	72.84
○	7	Providence County	Rhode Island	2019	403	63.07

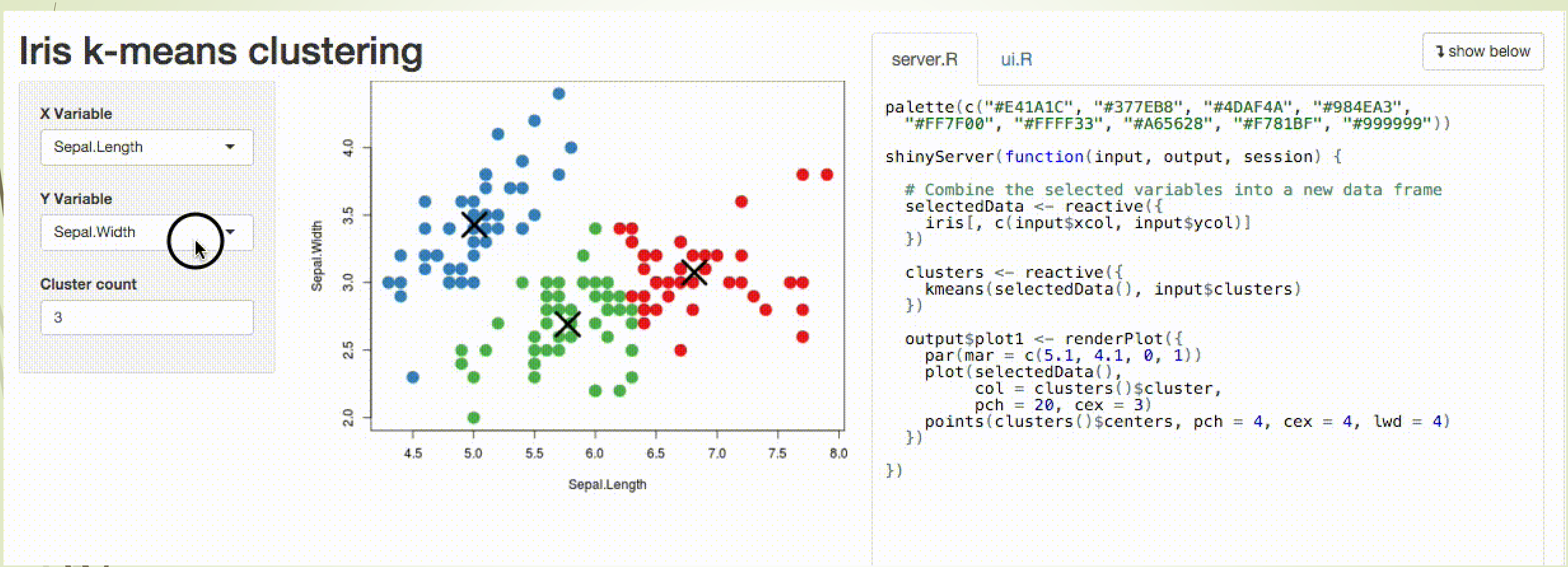
1-15 of 101 rows

Key Historical Trends

Confirmed Lyme Disease Cases by Month of Onset, 2008-2019

Confirmed Lyme Disease Cases by Age and Sex, 2001-2019

Reported Lyme Disease Cases, 2000-2019



Iris k-means clustering

X Variable

Sepal.Length

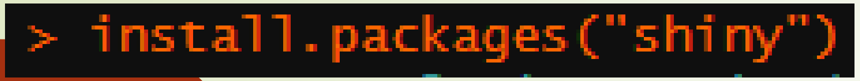
Y Variable

Sepal.Width

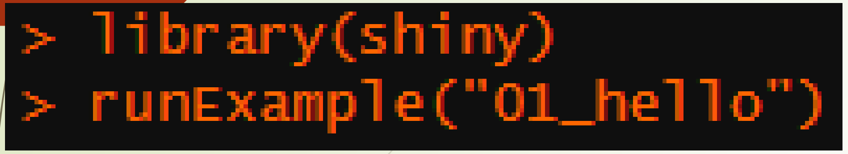
Cluster count

3

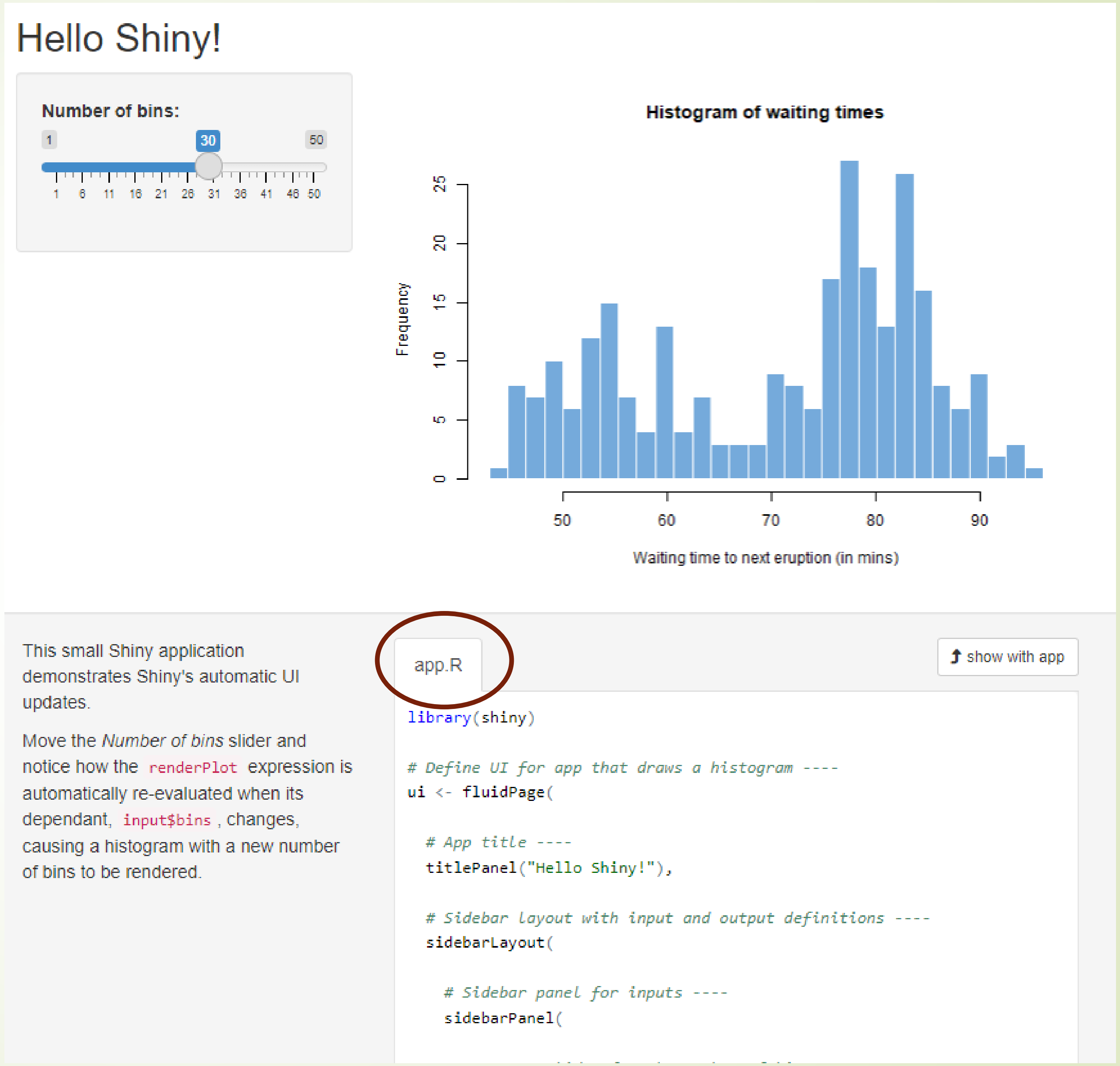
```
palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
  "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

shinyServer(function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

})
```



```
> install.packages("shiny")
```



```
> library(shiny)
> runExample("01_hello")
```



Hello Shiny!

Number of bins: 30

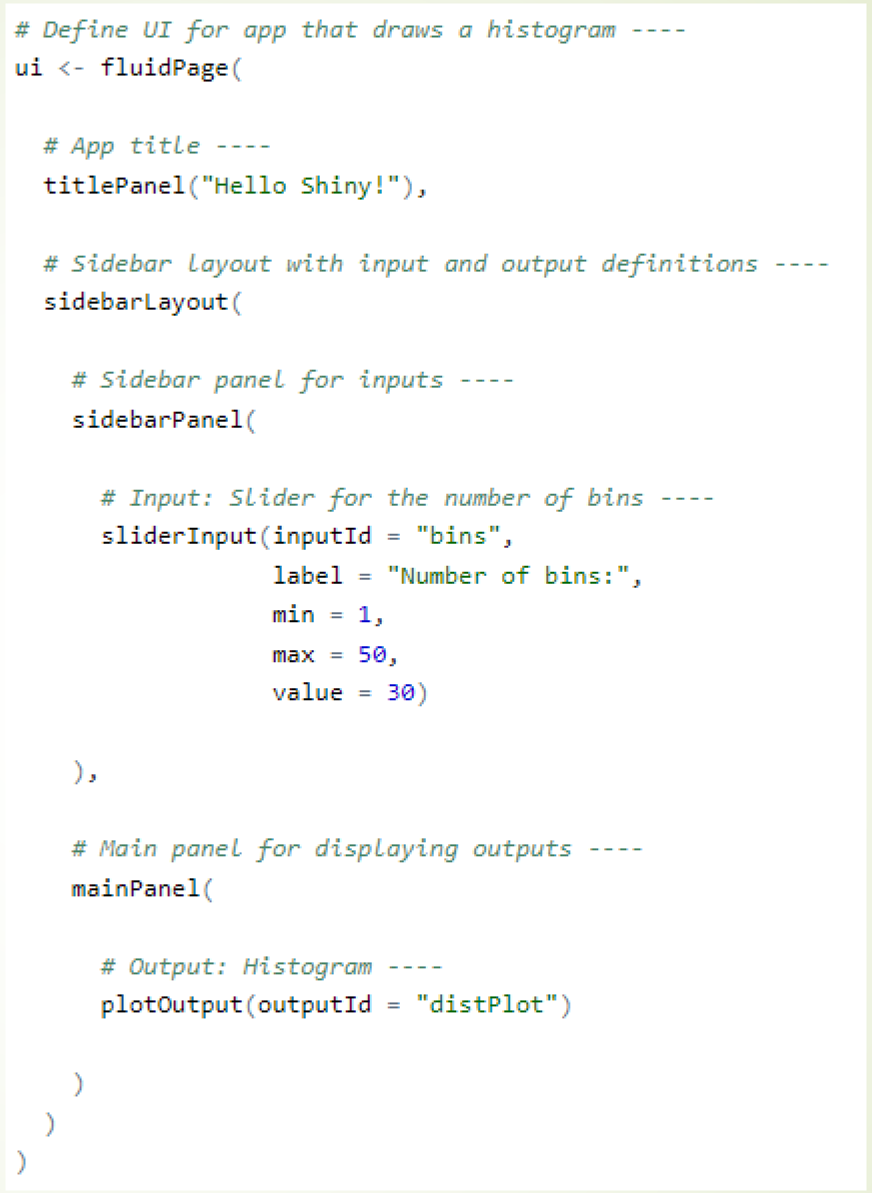
This small Shiny application demonstrates Shiny's automatic UI updates.

Move the *Number of bins* slider and notice how the `renderPlot` expression is automatically re-evaluated when its dependant, `input$bins`, changes, causing a histogram with a new number of bins to be rendered.

```
library(shiny)

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

  # App title ----
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(
```

Aufbau

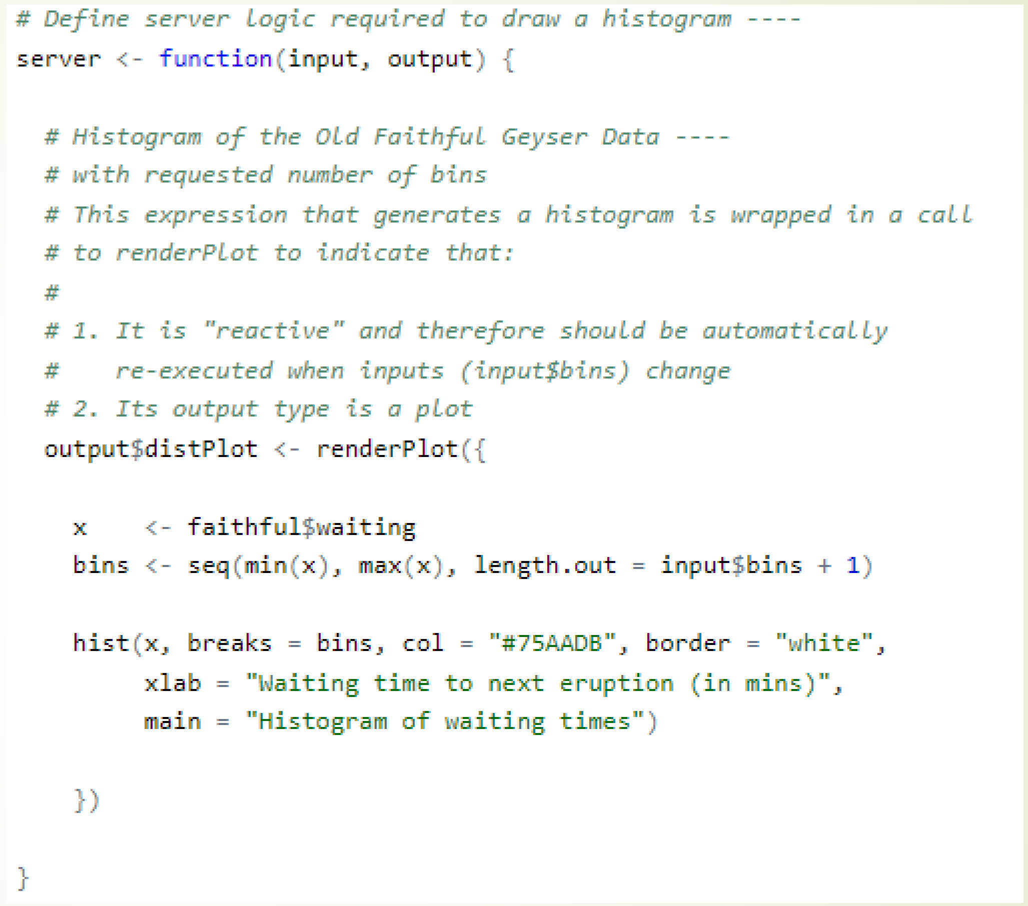
- app.R
  - Ui interface object



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

  # App title ----
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(

      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),

    # Main panel for displaying outputs ----
    mainPanel(

      # Output: Histogram ----
      plotOutput(outputId = "distPlot")

    )
  )
)
```

Aufbau

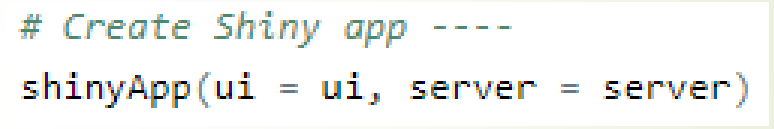
- app.R
  - Ui interface object
  - Server function



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

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

    })

}
```

Aufbau

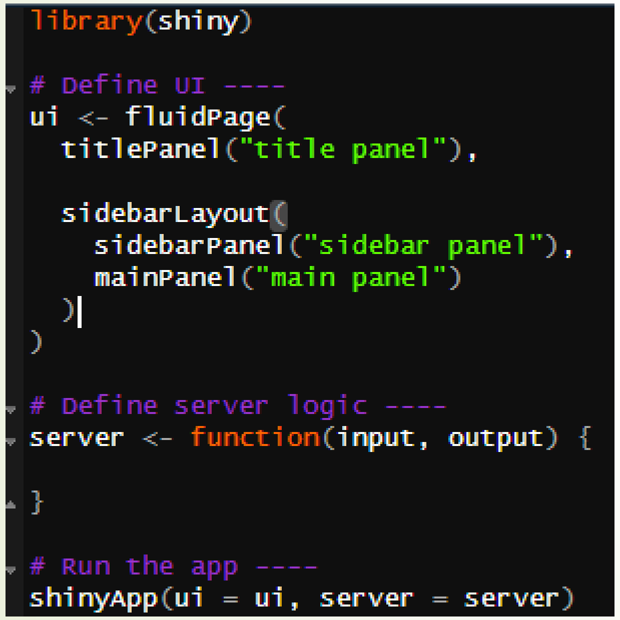
- app.R
  - Ui interface object
  - Server function
  - Call shinyApp function



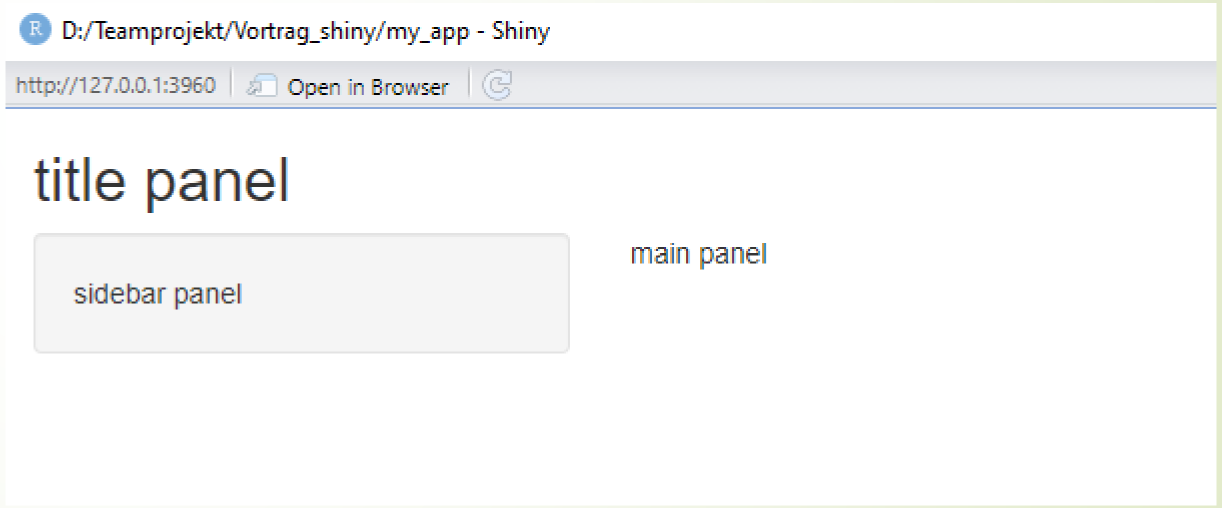
```
# Create Shiny app ----
shinyApp(ui = ui, server = server)
```

UI Design

- Fluidpage passt sich automatisch dem browser window an



```
library(shiny)

# Define UI ----
ui <- fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel("sidebar panel"),
    mainPanel("main panel")
  )
)

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

}

# Run the app ----
shinyApp(ui = ui, server = server)
```



D:/Teamprojekt/Vortrag_shiny/my_app - Shiny

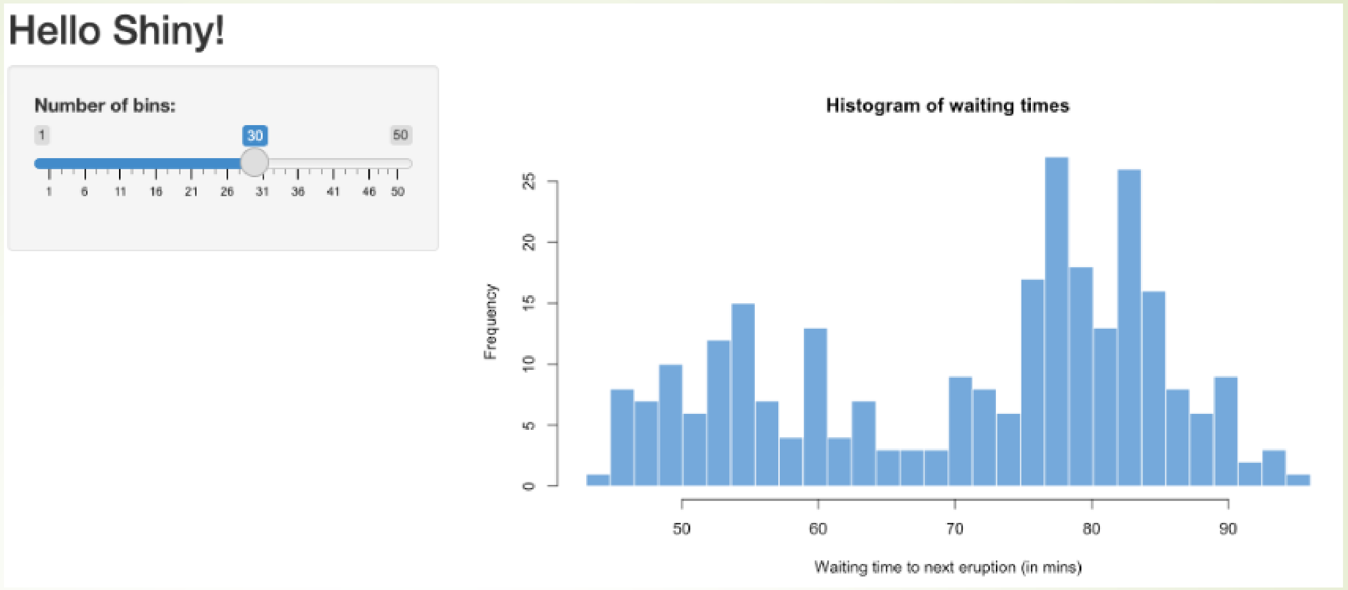
http://127.0.0.1:3960 Open in Browser

title panel

sidebar panel

main panel

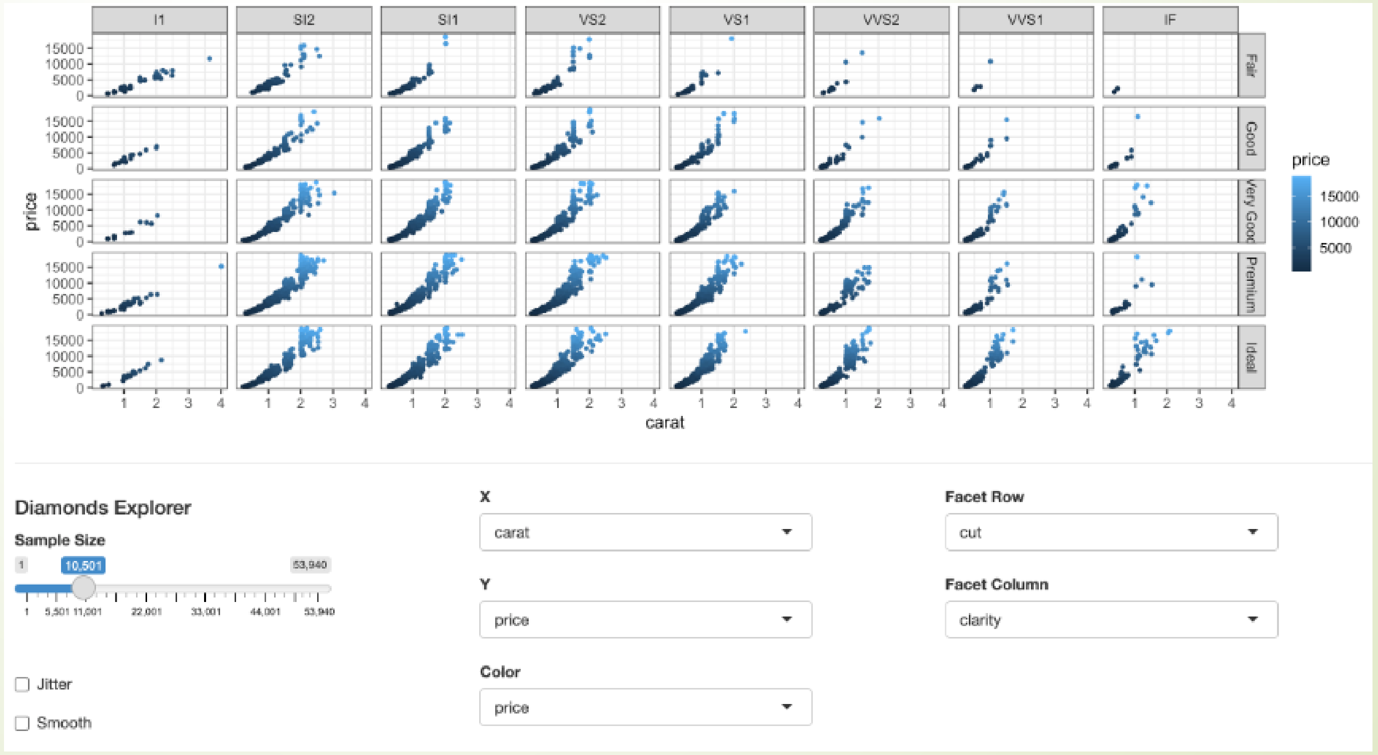
```
  sidebarLayout(position = "right",
                sidebarPanel("sidebar panel"),
                mainPanel("main panel")
  )
)
```



Hello Shiny!

Number of bins: 30

```
fluidRow(

  column(4,
         wellPanel(
           sliderInput(
             "bins", label = "Number of bins:",
             min = 1, value = 30, max = 50
           )
         )
  ),

  column(8,
         plotOutput("distPlot")
  )
)
```



Diamonds Explorer

Sample Size: 10,501

Jitter

Smooth

X: carat

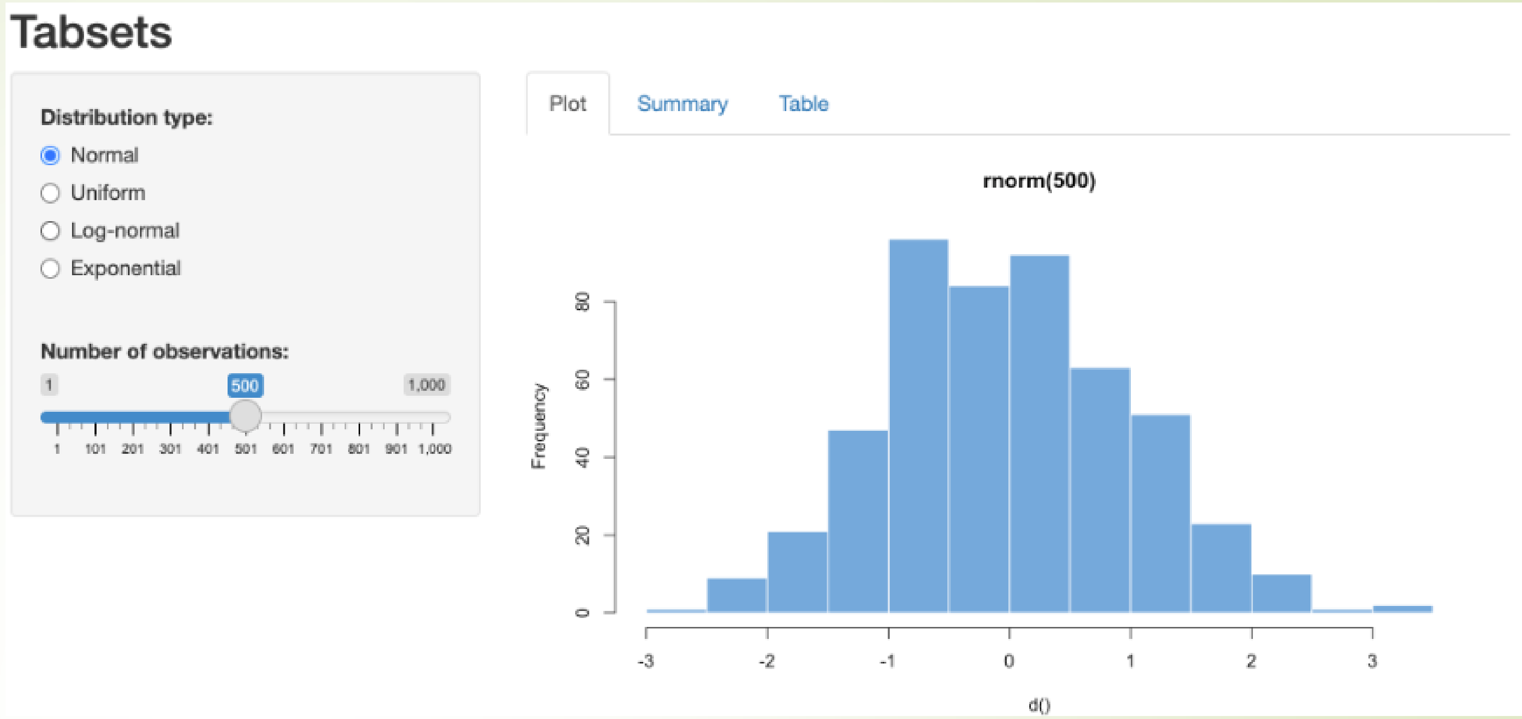
Y: price

Color: price

Facet Row: cut

Facet Column: clarity

```
  sidebarPanel(
    # Inputs excluded for brevity
  ),

  mainPanel(
    tabsetPanel(
      tabPanel("Plot", plotOutput("plot")),
      tabPanel("Summary", verbatimTextOutput("summary")),
      tabPanel("Table", tableOutput("table"))
    )
  )
)
```

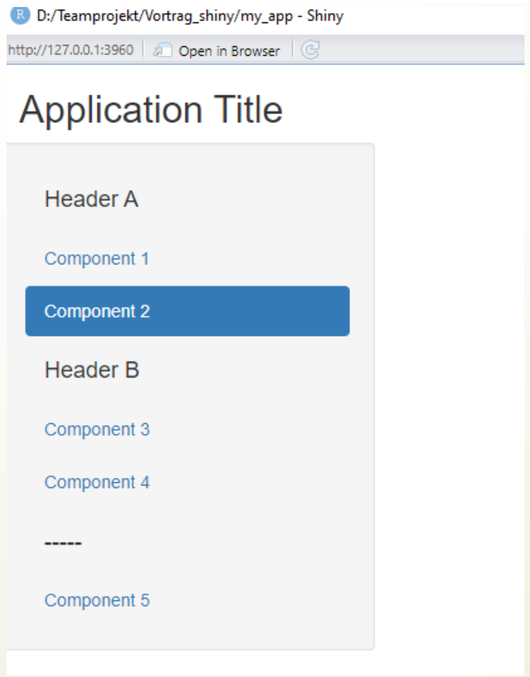


Tabsets

Distribution type:
- [✓ Normal]
- Uniform
- Log-normal
- Exponential

Number of observations: 500

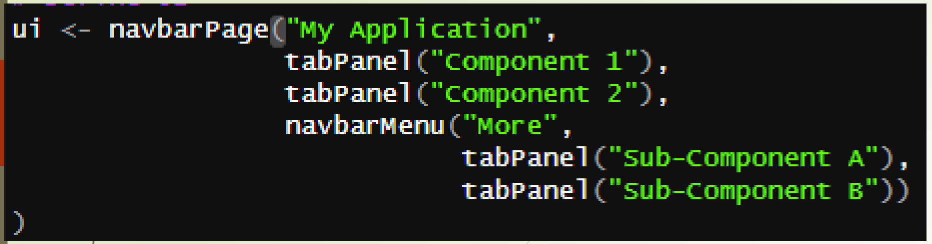
```
navlistPanel(
  "Header A",
  tabPanel("Component 1"),
  tabPanel("Component 2"),
  "Header B",
  tabPanel("Component 3"),
  tabPanel("Component 4"),
  "-----",
  tabPanel("Component 5")
)
```



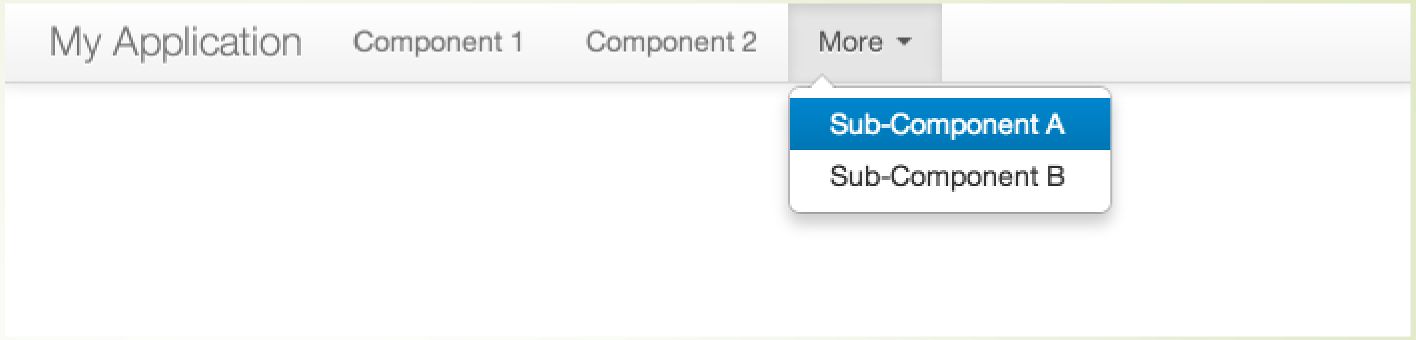
D:/Teamprojekt/Vortrag_shiny/my_app - Shiny

http://127.0.0.1:3960

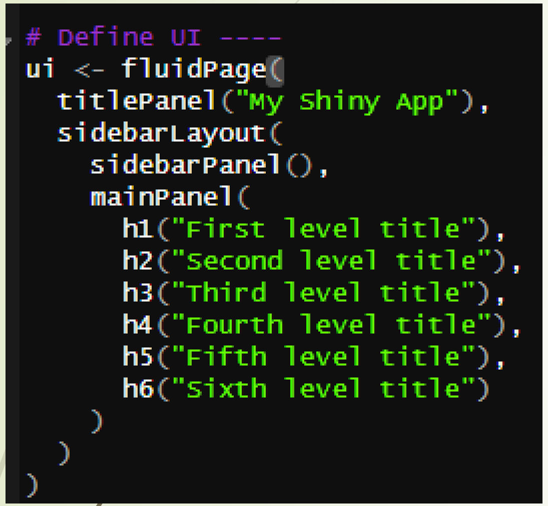
Application Title



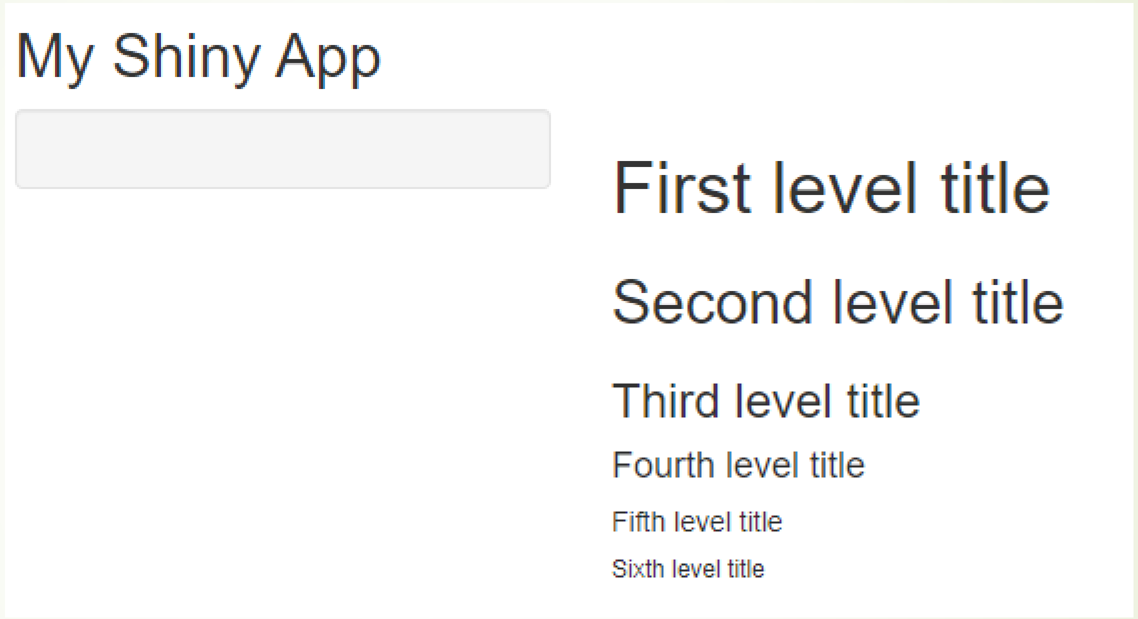
```r
ui <- navbarPage("My Application",
                 tabPanel("Component 1"),
                 tabPanel("Component 2"),
                 navbarMenu("More",
                            tabPanel("Sub-Component A"),
                            tabPanel("Sub-Component B"))
)
```



HTML Code



```
# Define UI ----
ui <- fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      h1("First level title"),
      h2("Second level title"),
      h3("Third level title"),
      h4("Fourth level title"),
      h5("Fifth level title"),
      h6("Sixth level title")
    )
  )
)
```



My Shiny App

First level title

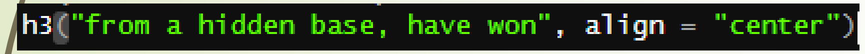
Second level title

Third level title

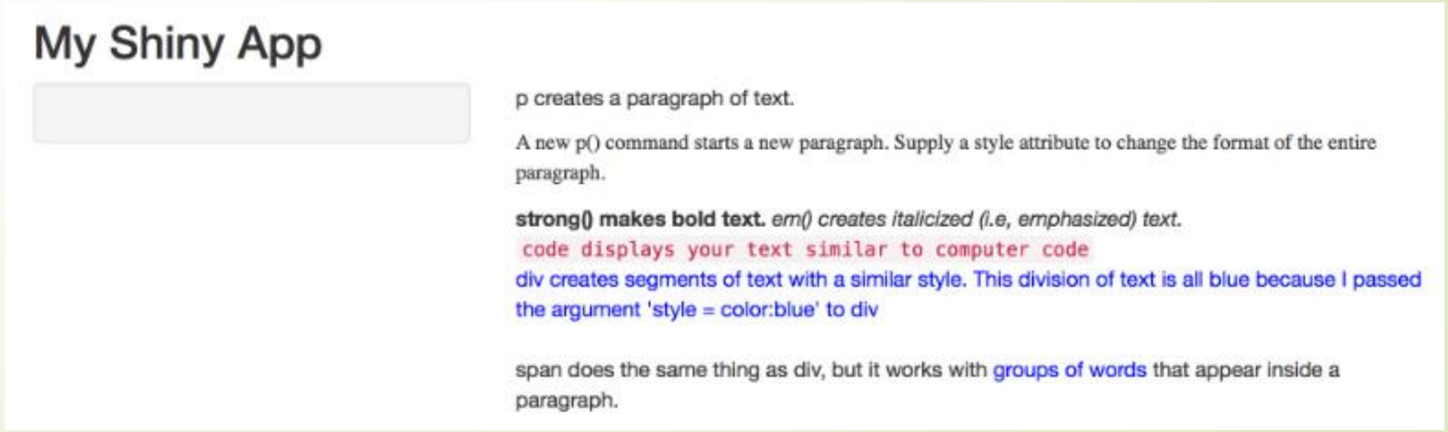
Fourth level title

Fifth level title

Sixth level title



```
h3("from a hidden base, have won", align = "center")
```



My Shiny App

p creates a paragraph of text.

A new p() command starts a new paragraph. Supply a style attribute to change the format of the entire paragraph.

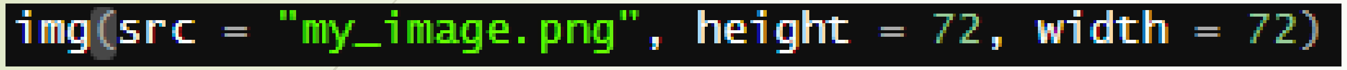
strong() makes bold text. *em() creates italicized (i.e, emphasized) text.*

`code displays your text similar to computer code`

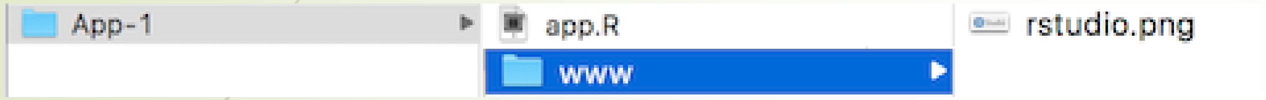
div creates segments of text with a similar style. This division of text is all blue because I passed the argument 'style = color:blue' to div

span does the same thing as div, but it works with groups of words that appear inside a paragraph.

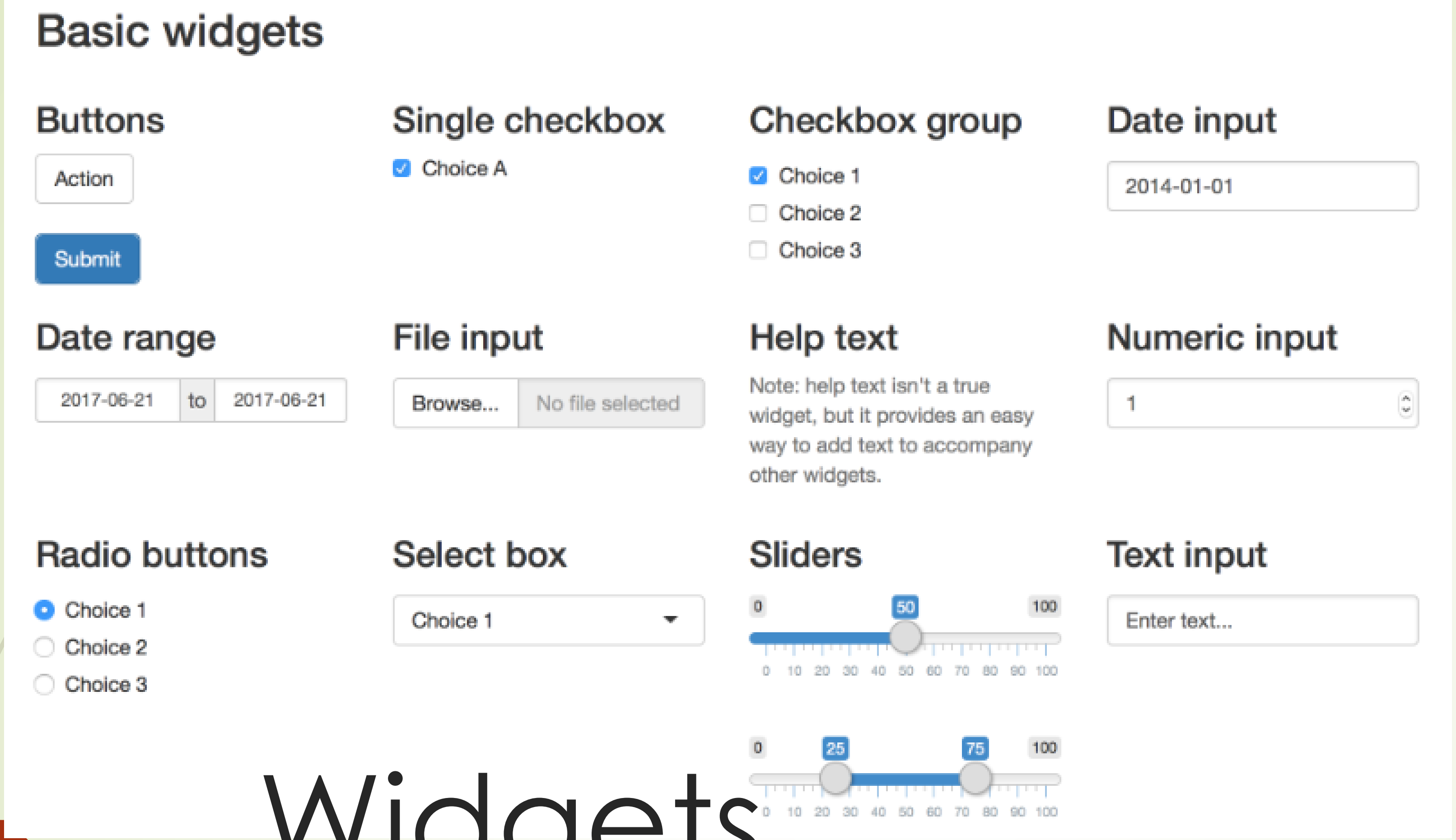
Images



```
img(src = "my_image.png", height = 72, width = 72)
```



Muss www Ordner sein im app Ordner



Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

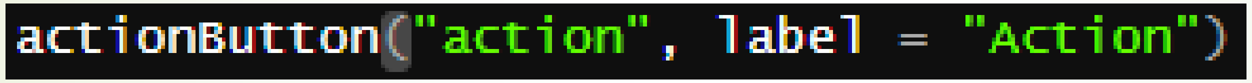
Widgets

Arten von Widgets

actionButton	Action Button
checkboxGroupInput	A group of check boxes
checkboxInput	A single check box
dateInput	A calendar to aid date selection
dateRangeInput	A pair of calendars for selecting a date range
fileInput	A file upload control wizard
helpText	Help text that can be added to an input form
numericInput	A field to enter numbers
radioButtons	A set of radio buttons
selectInput	A box with choices to select from
sliderInput	A slider bar
submitButton	A submit button
textInput	A field to enter text

Aufbau Widget

- Name (access)
- Label (in app)

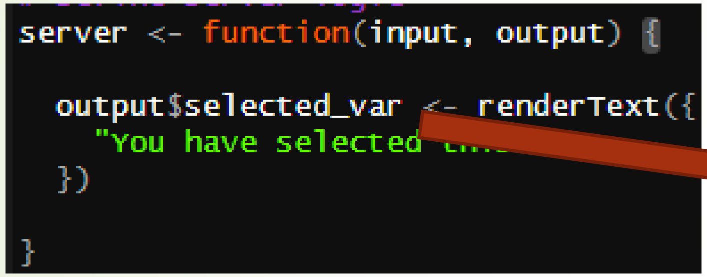


```
actionButton("action", label = "Action")
```

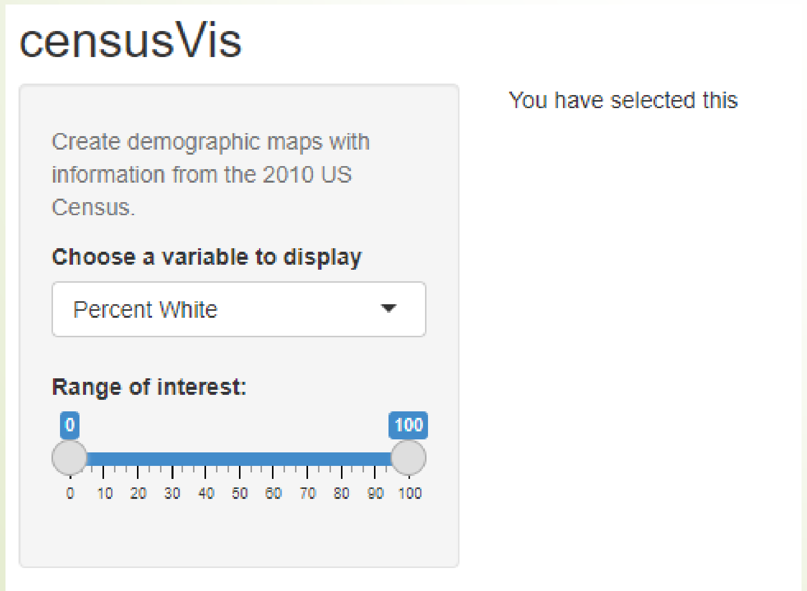
Reactive Output

Server

- List like object output



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

  output$selected_var <- renderText({
    "You have selected
  })

}
```



censusVis

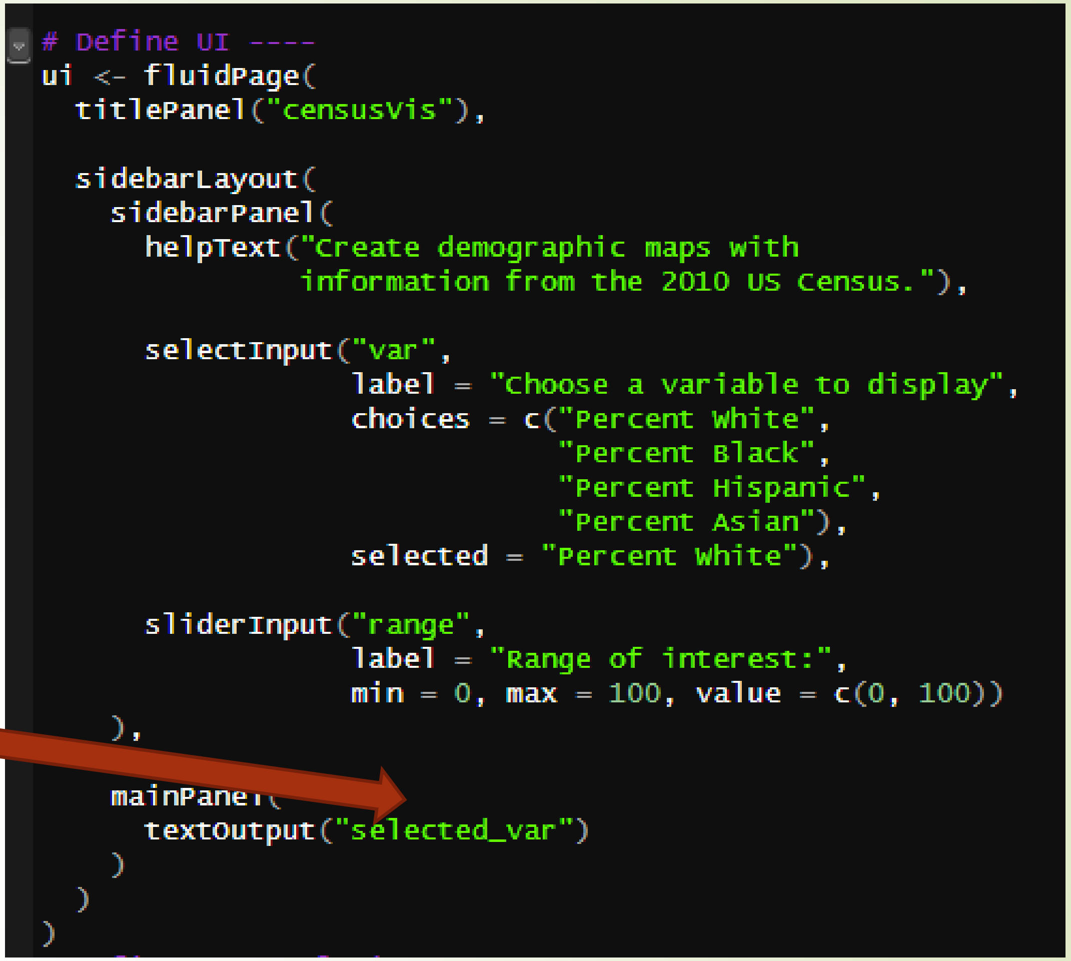
Create demographic maps with information from the 2010 US Census.

Choose a variable to display

Percent White

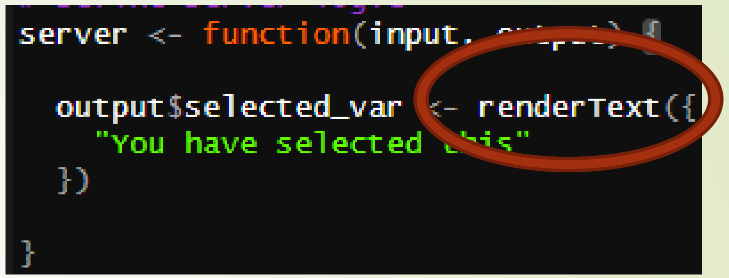
Range of interest:

You have selected this



```
# Define UI ----
ui <- fluidPage(
  titlePanel("censusVis"),

  sidebarLayout(
    sidebarPanel(
      helpText("Create demographic maps with
               information from the 2010 US Census."),

      selectInput("var",
                  label = "Choose a variable to display",
                  choices = c("Percent White",
                              "Percent Black",
                              "Percent Hispanic",
                              "Percent Asian"),
                  selected = "Percent White"),

      sliderInput("range",
                  label = "Range of interest:",
                  min = 0, max = 100, value = c(0, 100))
    ),

    mainPanel(
      textOutput("selected_var")
    )
  )
)
```

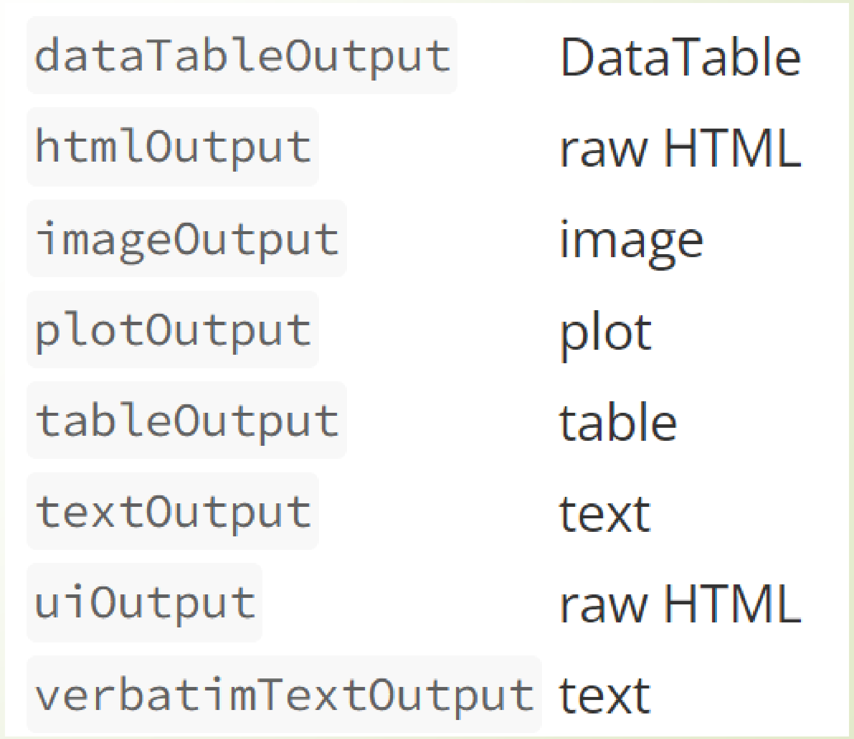
Server



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

  output$selected_var <- renderText({
    "You have selected this"
  })

}
```

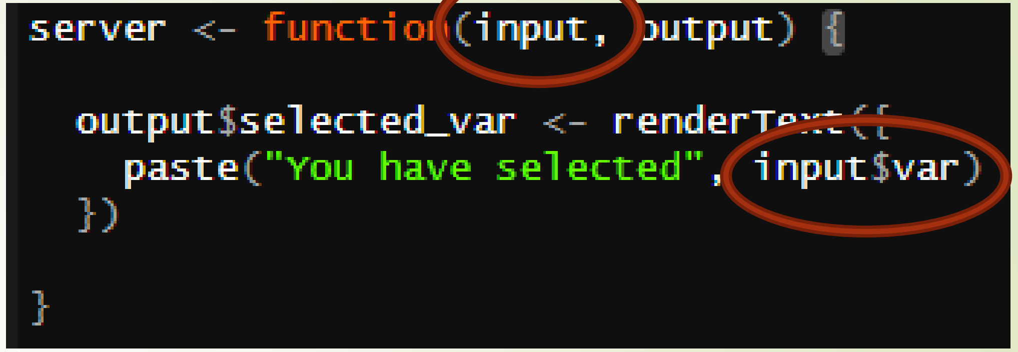
- Sollte renderfunction benutzen
- Für Output auch Funktionen (R Objekt in Output)

renderDataTable	DataTable
renderImage	images (saved as a link to a source file)
renderPlot	plots
renderPrint	any printed output
renderTable	data frame, matrix, other table like structures
renderText	character strings
renderUI	a Shiny tag object or HTML



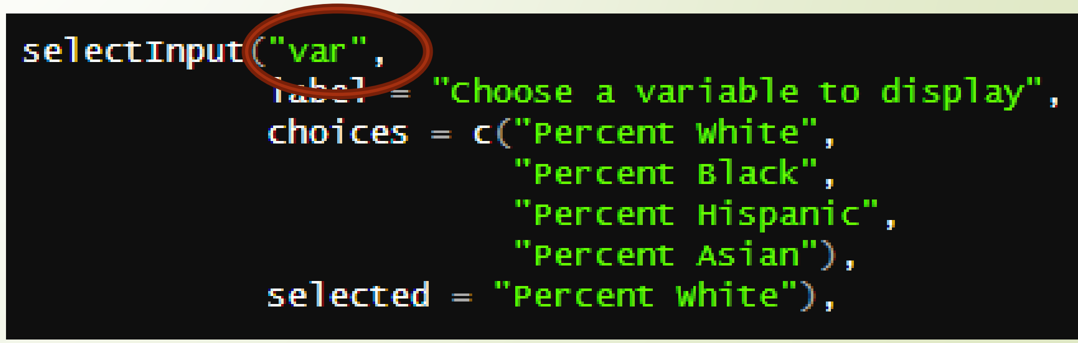
`dataTableOutput`	DataTable
`htmlOutput`	raw HTML
`imageOutput`	image
`plotOutput`	plot
`tableOutput`	table
`textOutput`	text
`uiOutput`	raw HTML
`verbatimTextOutput`	text

Reactive

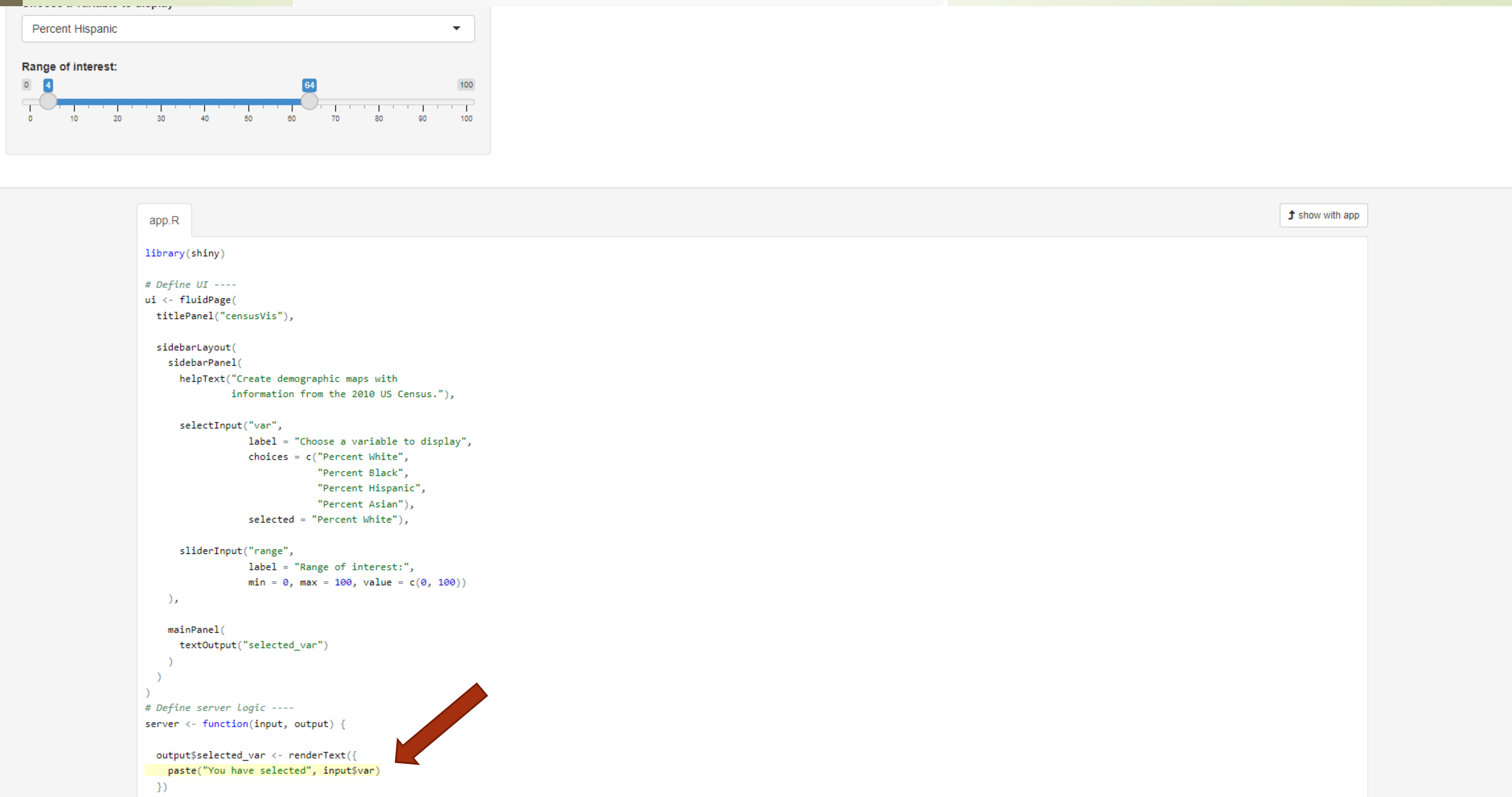


```
server <- function(input, output) {

  output$selected_var <- renderText({
    paste("You have selected", input$var)
  })

}
```

- Input list like object (to read: reactive env z.B. render)
- Speichert widget values unter name
- Shiny trackt welche outputs von welchen widgets abhängen und ändert diese sobald sich das widget ändert



```
selectInput("var",
            label = "Choose a variable to display",
            choices = c("Percent White",
                        "Percent Black",
                        "Percent Hispanic",
                        "Percent Asian"),
            selected = "Percent White"),
runApp("census-app", display.mode = "showcase")
```

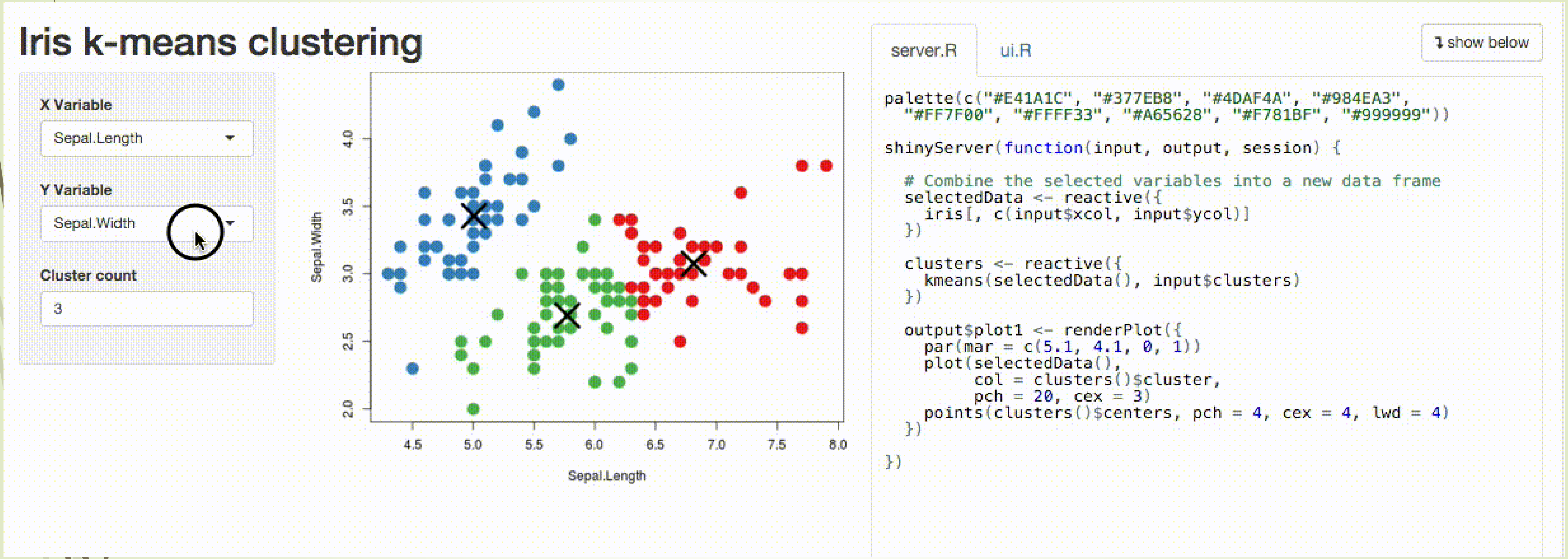


Percent Hispanic

Range of interest:

0 4 64 100

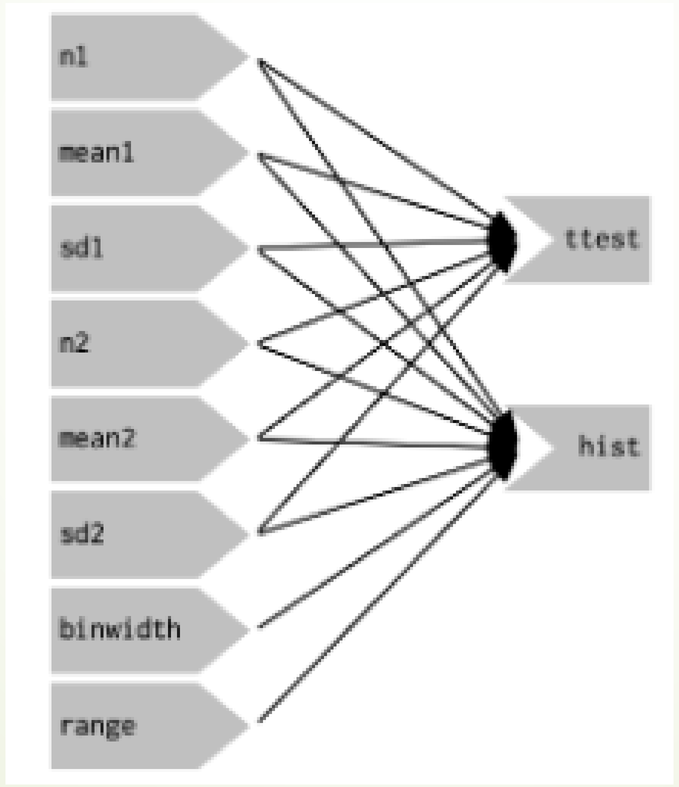
```
library(shiny)

# Define UI ----
ui <- fluidPage(
  titlePanel("censusVis"),

  sidebarLayout(
    sidebarPanel(
      helpText("Create demographic maps with
               information from the 2010 US Census."),

      selectInput("var",
                  label = "Choose a variable to display",
                  choices = c("Percent White",
                              "Percent Black",
                              "Percent Hispanic",
                              "Percent Asian"),
                  selected = "Percent White"),

      sliderInput("range",
                  label = "Range of interest:",
                  min = 0, max = 100, value = c(0, 100))
    ),

    mainPanel(
      textOutput("selected_var")
    )
  )
)
# Define server logic ----
server <- function(input, output) {

  output$selected_var <- renderText({
    paste("You have selected", input$var)
  })
```



Iris k-means clustering

```
palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
  "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

shinyServer(function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

})
```

Für Performance aufpassen was geupdated wird!

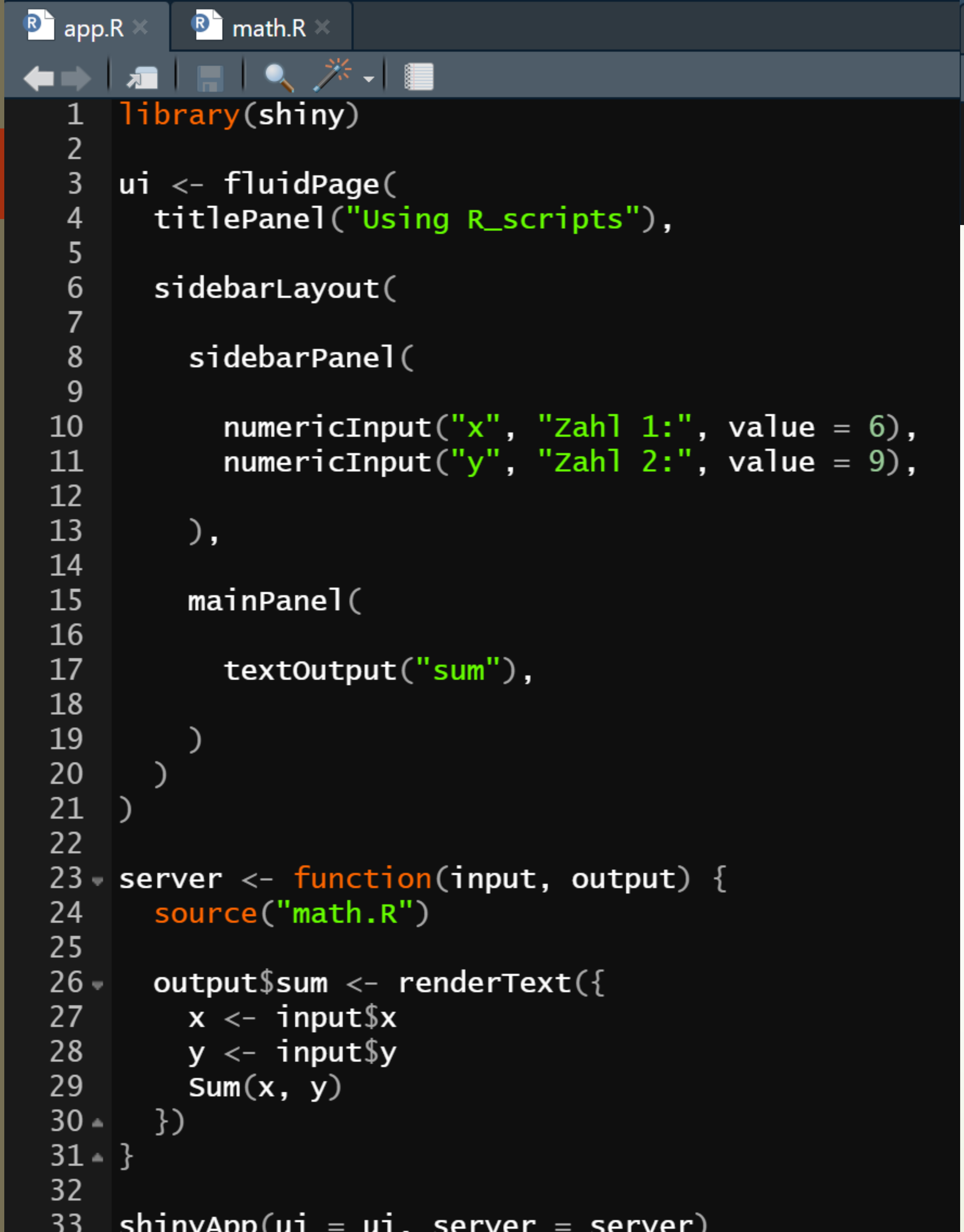


Best practices in shiny

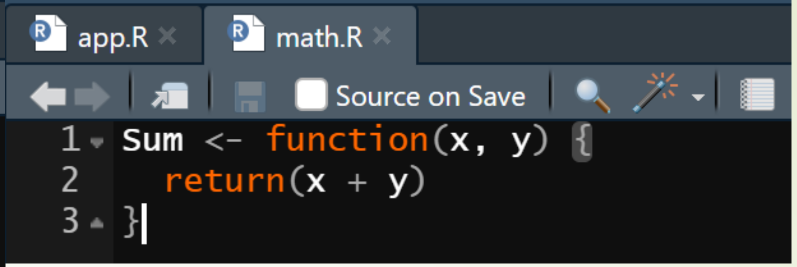
Für eine gute Zusammenarbeit ☺

functions

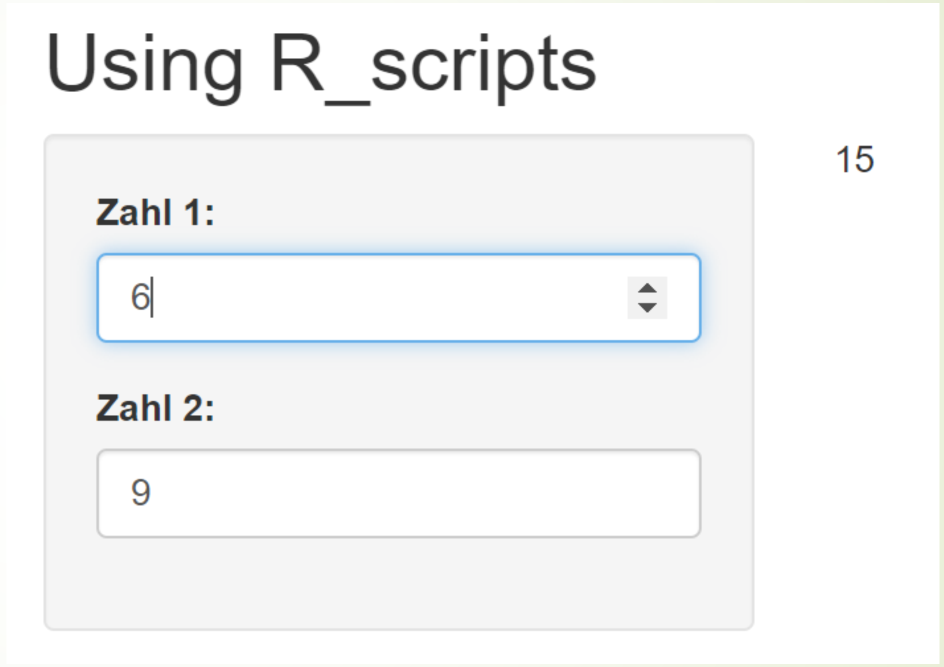
- Können außerhalb von app.R existieren
  - für große eigenes file: R/{function-name}.R
  - Für kleine Bündel file: R/utils.R oder R/ui.R



```r
library(shiny)

ui <- fluidPage(
  titlePanel("Using R_scripts"),

  sidebarLayout(

    sidebarPanel(

      numericInput("x", "Zahl 1:", value = 6),
      numericInput("y", "Zahl 2:", value = 9),

    ),

    mainPanel(

      textOutput("sum"),

    )
  )
)

server <- function(input, output) {
  source("math.R")

  output$sum <- renderText({
    x <- input$x
    y <- input$y
    Sum(x, y)
  })
}

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



```r
Sum <- function(x, y) {
  return(x + y)
}
```



Using R_scripts

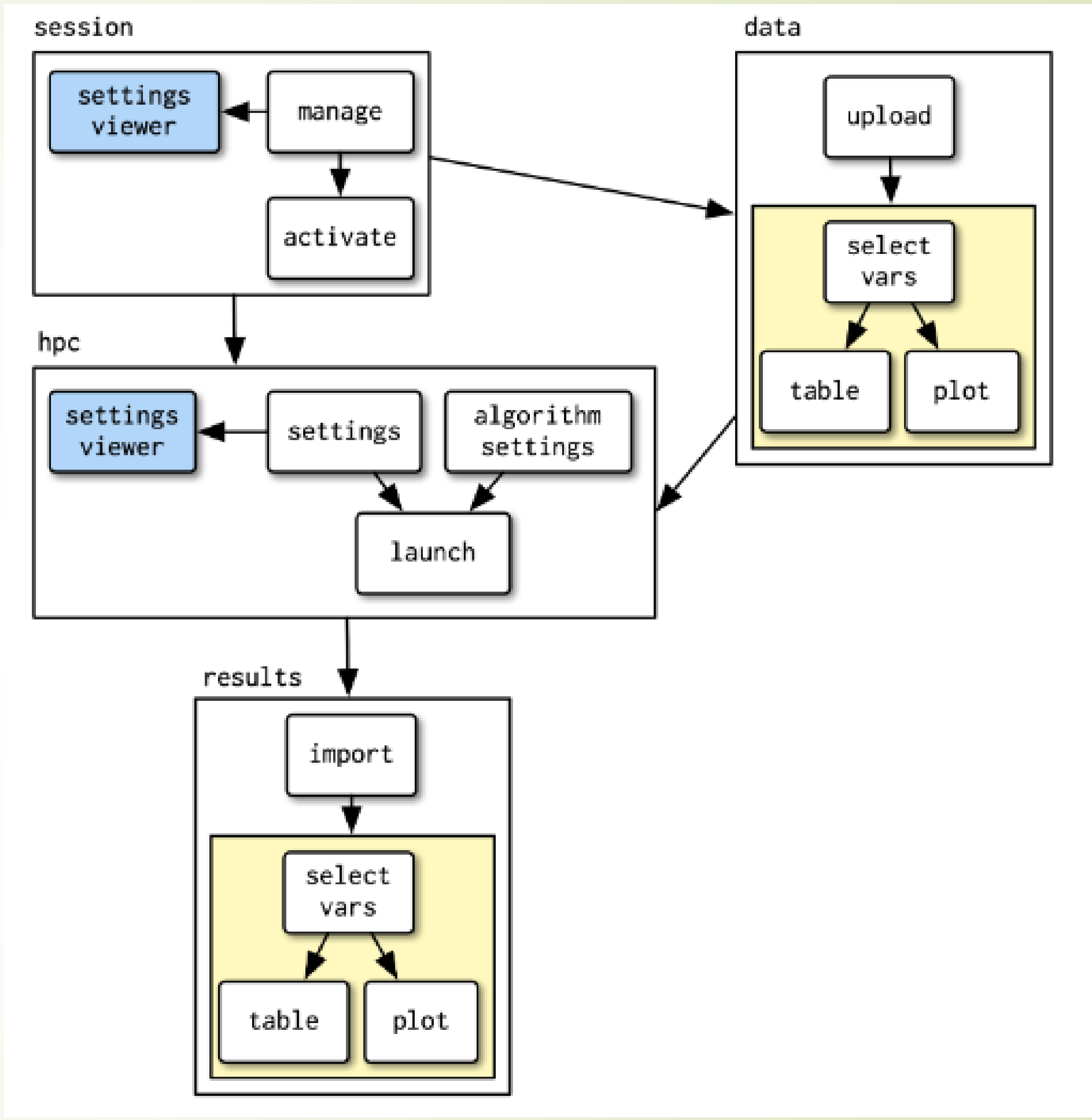
Zahl 1: 6

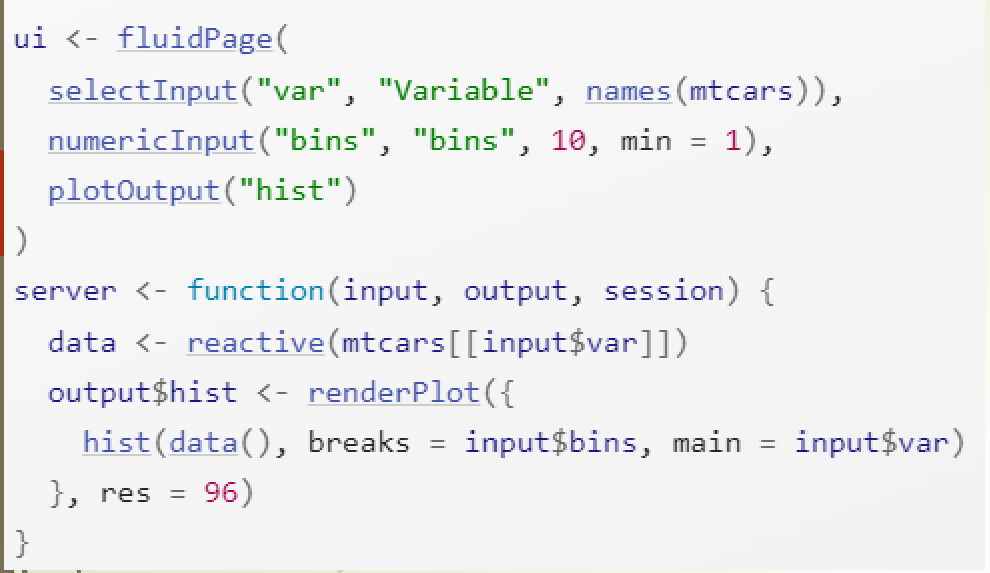
Zahl 2: 9

15

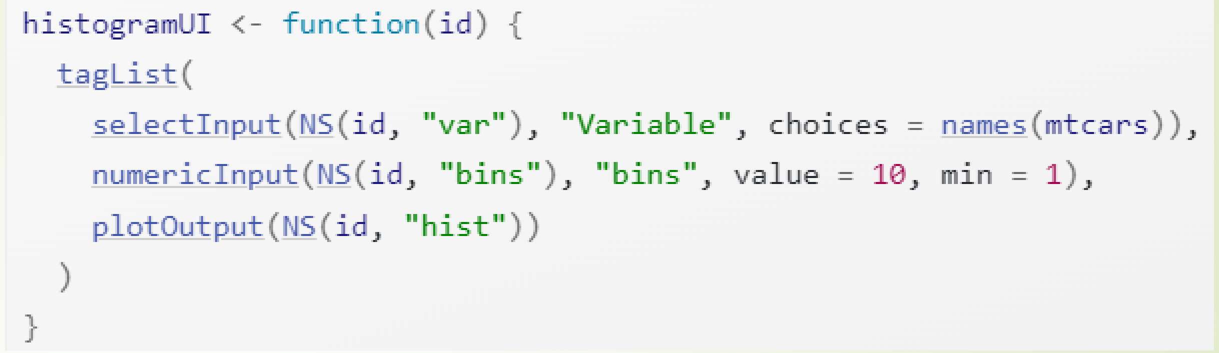
Modules

- Gleicher namespace
- Davor: Server sieht alle ids in UI

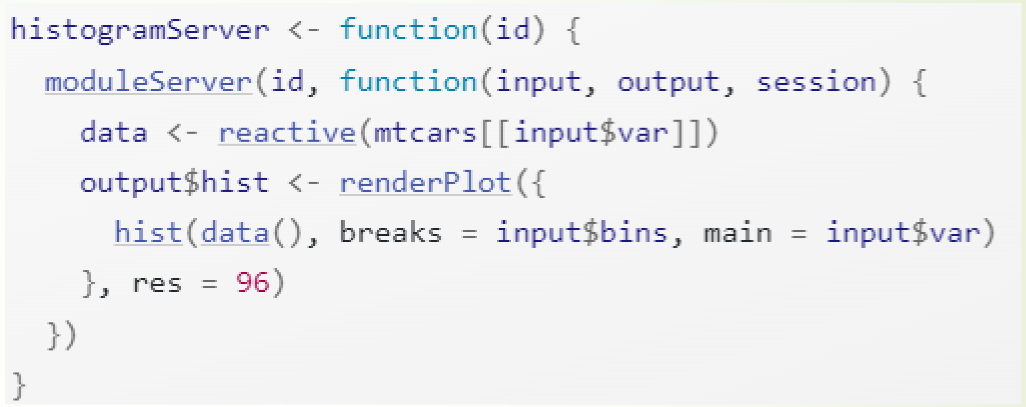




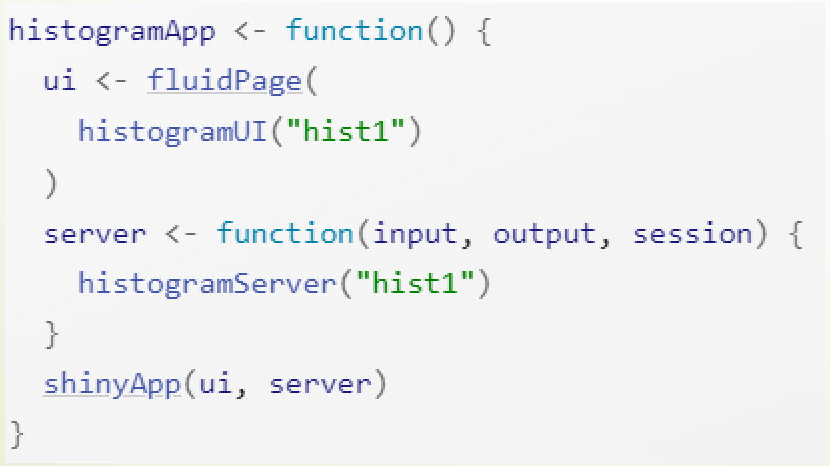
```r
ui <- fluidPage(
  selectInput("var", "Variable", names(mtcars)),
  numericInput("bins", "bins", 10, min = 1),
  plotOutput("hist")
)
server <- function(input, output, session) {
  data <- reactive(mtcars[[input$var]])
  output$hist <- renderPlot({
    hist(data(), breaks = input$bins, main = input$var)
  }, res = 96)
}
```



```r
histogramUI <- function(id) {
  tagList(
    selectInput(NS(id, "var"), "Variable", choices = names(mtcars)),
    numericInput(NS(id, "bins"), "bins", value = 10, min = 1),
    plotOutput(NS(id, "hist"))
  )
}
```



```r
histogramServer <- function(id) {
  moduleServer(id, function(input, output, session) {
    data <- reactive(mtcars[[input$var]])
    output$hist <- renderPlot({
      hist(data(), breaks = input$bins, main = input$var)
    }, res = 96)
  })
}
```



```r
histogramApp <- function() {
  ui <- fluidPage(
    histogramUI("hist1")
  )
  server <- function(input, output, session) {
    histogramServer("hist1")
  }
  shinyApp(ui, server)
}
```

Homework

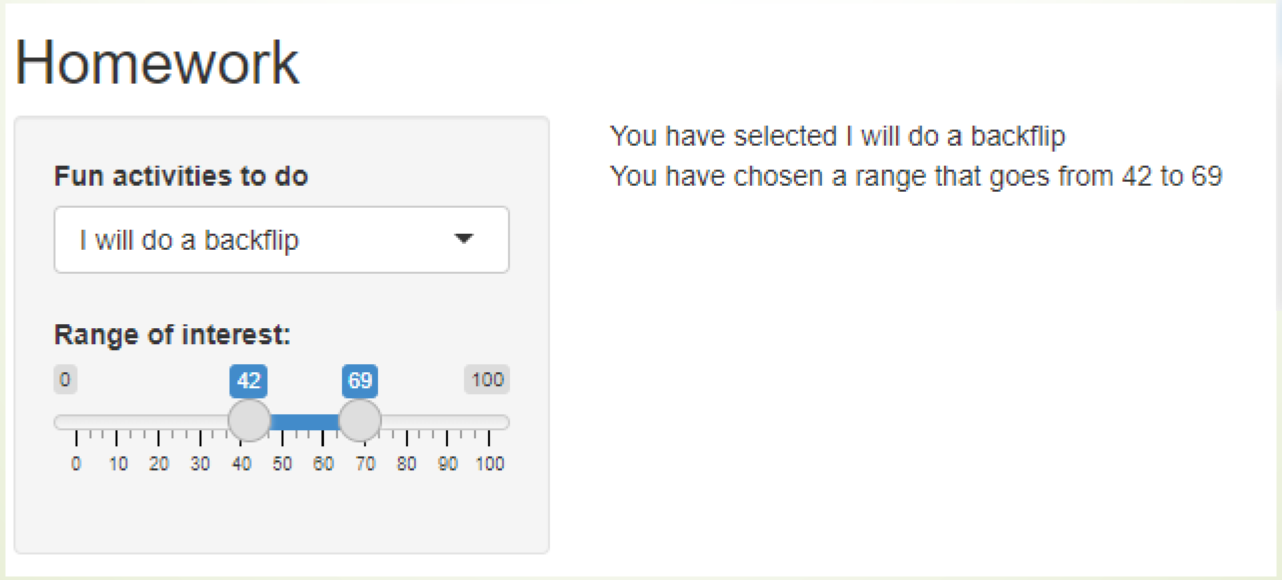
Hausaufgabe (kleines Anwendungsbeispiel)

- Programming statistical illusions in R using shiny Apps
- Zeit bis Juli
- Benutzt Git

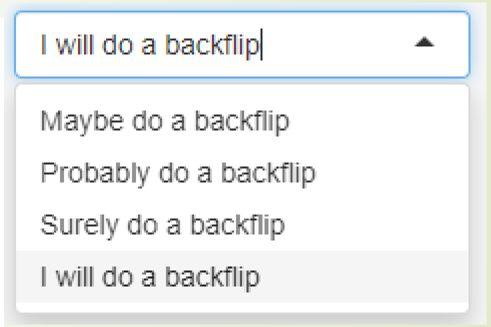
Viel Spaß!

Actual Homework

- Optionen wichtig!
- 2 textOutput im mainpanel
  - Oberer zeigt an was ausgewählt wurde
  - Unterer zeigt an welche range im slider gewählt wurde



You have selected I will do a backflip
You have chosen a range that goes from 42 to 69



Quellen

- https://shiny.rstudio.com/tutorial/written-tutorial
- https://shiny.rstudio.com/articles/layout-guide.html
- https://mastering-shiny.org/

- https://shiny.rstudio.com/articles/debugging.html