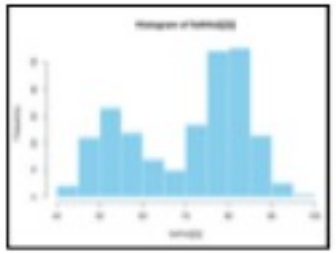

Shiny: July 31, 2012–present
10 years!
rstudio:conf(2022)

SHINY

Render data online

og det kan shiny

Vi har vores grafer og tabeller



```
'data.frame':   3 obs. of  2 variables:
 $ Sepal.Length: num  5.1 4.9 4.7
 $ Sepal.Width : num  3.5 3 3.2
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
4	4.60	3.10	1.50	0.20	setosa
5	5.00	3.60	1.40	0.20	setosa
6	5.40	3.90	1.70	0.40	setosa

Som skal ud på en skærm
”ud på” = ”renderes”

```
ui <- fluidPage(
  titlePanel(title=h4("Races", align="ce
  sidebarPanel(
    sliderInput("num", "Number:",min=0,m
  mainPanel(plotOutput("plot2")))
```

Source on Save

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

  output$plot2<-renderPlot({
    ggplot(dat(),aes(x=date,y=num))+
      geom_point(colour='red')})}

shinyApp(ui, server)
```

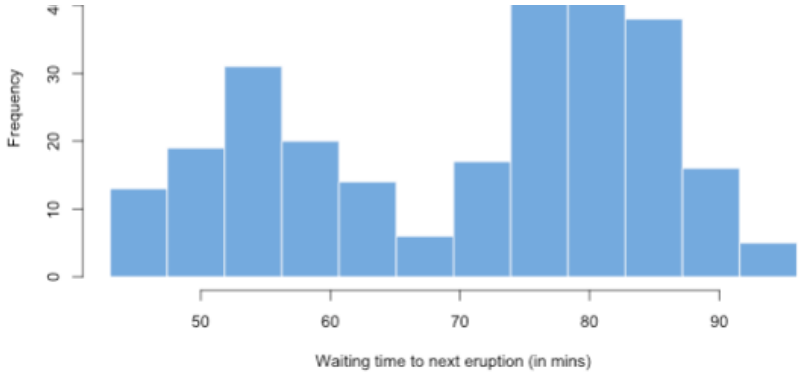
SHINY

Render data online

Som *defaultes* ud på en skærm

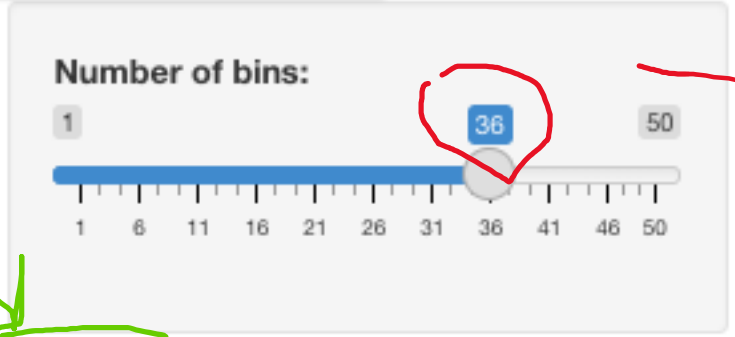
Vi starter med data

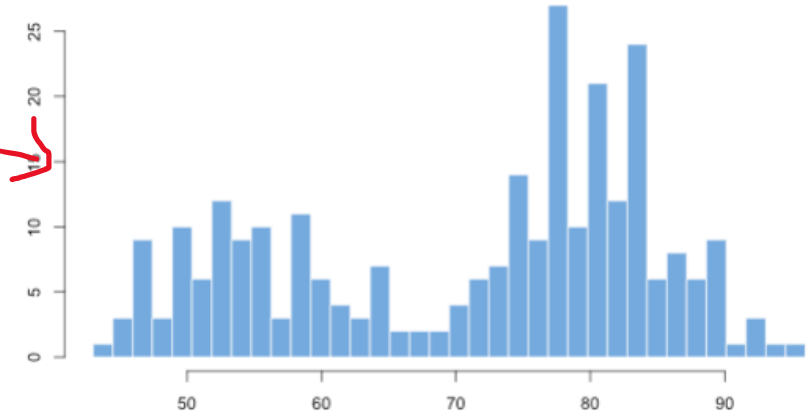
	name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle
1	Cheetah	Acinonyx	carni	Carnivora	lc	12.1	NA	NA
2	Owl monkey	Aotus	omni	Primates	NA	17.0	1.8	NA
3	Mountain beaver	Aplodontia	herbi	Rodentia	nt	14.4	2.4	NA
4	Greater short-tailed shrew	Blarina	omni	Soricomorpha	lc	14.9	2.3	0.1333333
5	Cow	Bos	herbi	Artiodactyla	domesticated	4.0	0.7	0.6666667
6	Three-toed sloth	Bradypus	herbi	Pilosa	NA	14.4	2.2	0.7666667
7	Northern fur seal	Callorhinus	carni	Carnivora	vu	8.7	1.4	0.3833333
8	Vesper mouse	Calomys	NA	Rodentia	NA	7.0	NA	NA
9	Dog	Canis	carni	Carnivora	domesticated	10.1	2.9	0.3333333
10	Roe deer	Capreolus	herbi	Artiodactyla	lc	3.0	NA	NA
11	Goat	Capri	herbi	Artiodactyla	lc	5.3	0.6	NA
12	Guinea pig	Cavis	herbi	Rodentia	domesticated	9.4	0.8	0.2166667
13	Grivet	Cercopithecus	omni	Primates	lc	10.0	0.7	NA
14	Chinchilla	Chinchilla	herbi	Rodentia	domesticated	12.5	1.5	0.1166667
15	Star-nosed mole	Condylura	omni	Soricomorpha	lc	10.3	2.2	NA
16	African giant pouched rat	Cricetomys	omni	Rodentia	NA	8.3	2.0	NA
17	Lesser short-tailed shrew	Cryptotis	omni	Soricomorpha	lc	9.1	1.4	0.1500000
18	Long-nosed armadillo	Dasypus	carni	Cingulata	lc	17.4	3.1	0.3833333
19	Tree hyrax	Dendrohyrax	herbi	Hyracoidea	lc	5.3	0.5	NA
20	North American Opossum	Didelphis	omni	Didelphimorphia	lc	18.0	4.9	0.3333333
21	Asian elephant	Elephas	herbi	Proboscidea	en	3.9	NA	NA
22	Big brown bat	Eptesicus	insecti	Chiroptera	lc	19.7	3.9	0.1166667




```
titlePanel(title=h4("Races", align="ce
sidebarPanel(
  sliderInput('num',  "Num
mainPanel(plotOutput("plo
```

Ændres af brugeren i UI





Source on Save

```
server <- function(input,o

  output$plot2<-renderPlot
    ggplot(dat(),aes(x=date,y=num))+
```

SHINY

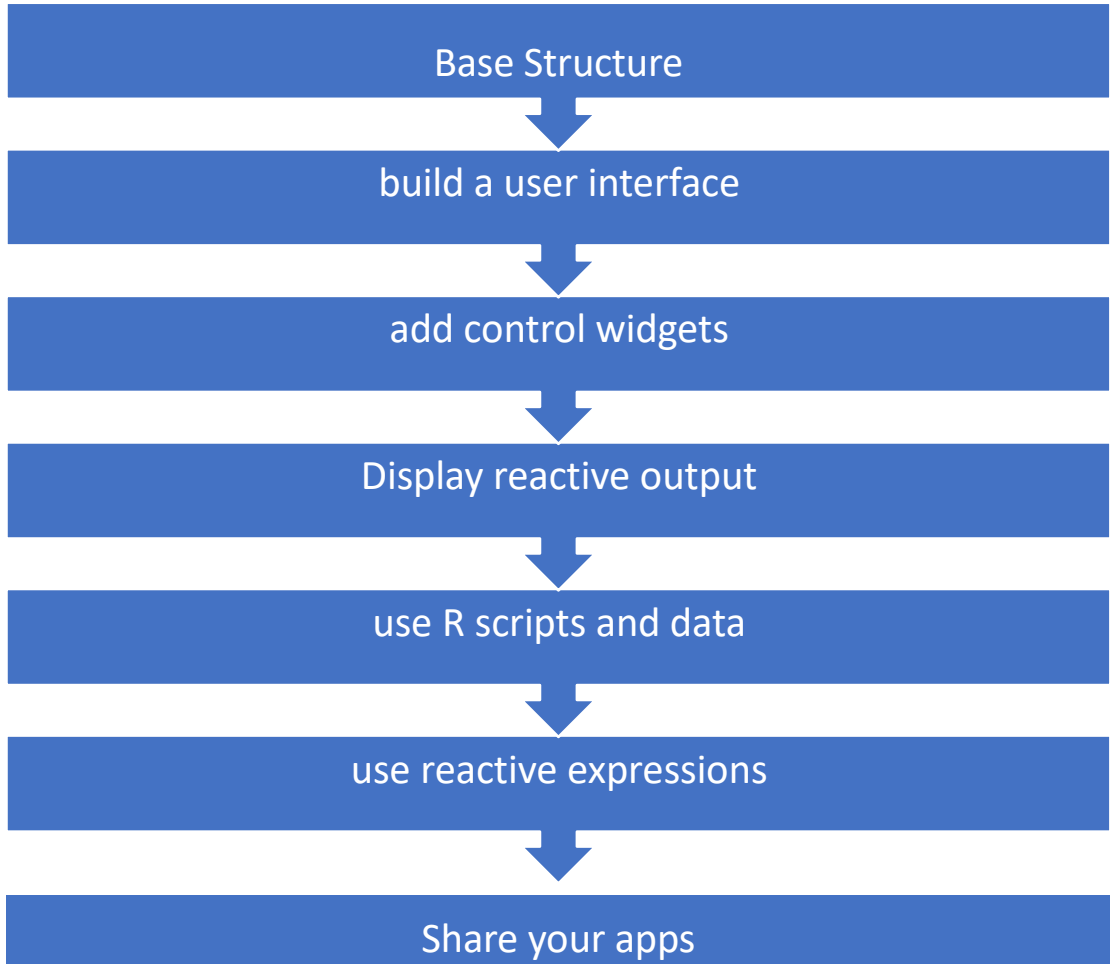
App template

Begin writing a new app with this template. Preview the app by running the code at the R command line.

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

- **ui** - nested R functions that assemble an HTML user interface for your app
- **server** - a function with instructions on how to build and rebuild the R objects displayed in the UI
- **shinyApp** - combines `ui` and `server` into a functioning app. Wrap with **runApp()** if calling from a sourced script or inside a function.

SHINY


Base Structure
build a user interface
add control widgets
Display reactive output
use R scripts and data
use reactive expressions
Share your apps

SHINY UI

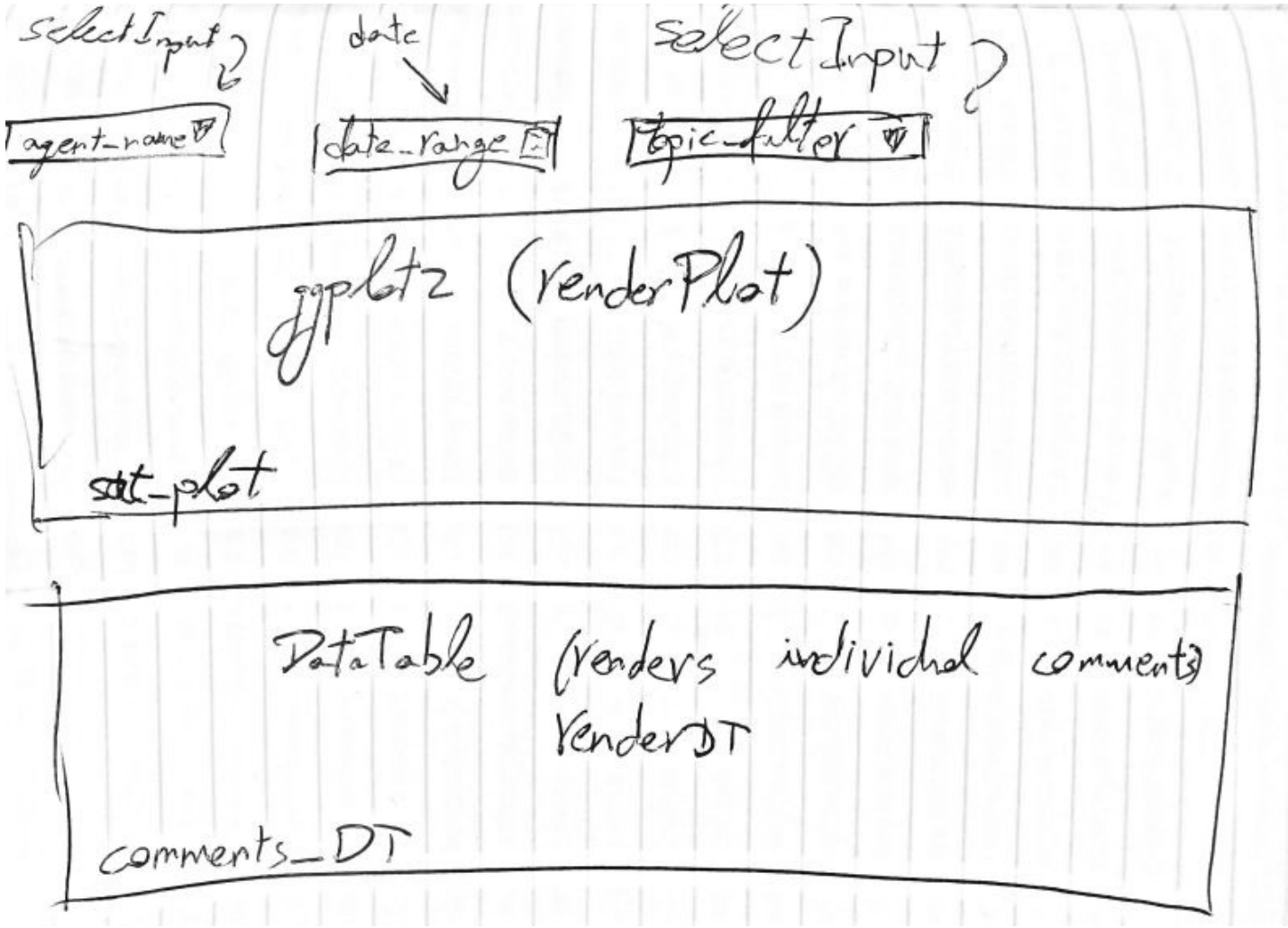
Base Structure

```r
library(shiny)

# Define UI ----
ui <- fluidPage(

)

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

}

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

SHINY UI

build a user interface

Do a mockup even if it's just a piece of paper which took you 5 minutes to draw. It will be worth it



SHINY UI

build a user interface

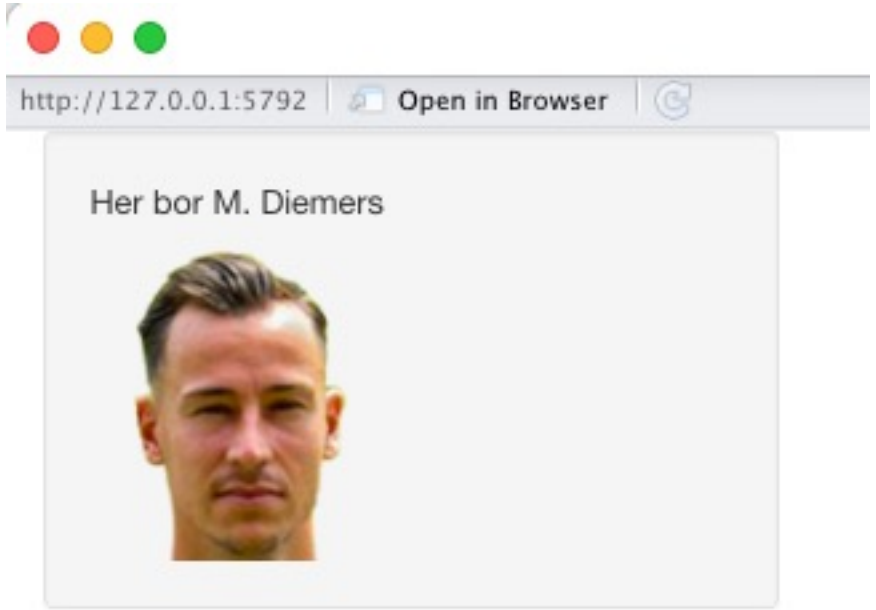
```
ui <- fluidPage(
  titlePanel("title panel"),

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

*You can add HTML content to your Shiny ap by placing it inside a *Panel function*

```
ui <- fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      img(src = "rstudio.png", height = 140, width = 400)
      h6("Episode IV", align = "center"),
      h6("A NEW HOPE", align = "center"),
      p("p creates a paragraph of text."),
      p("A new p() command starts a new paragraph. Supply a style attribute to change the format of
the entire paragraph.", style = "font-family: 'times'; font-si16pt"),
      div("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", style = "color:blue"),
```

SHINY øvelse

build a user interface



SHINY UI

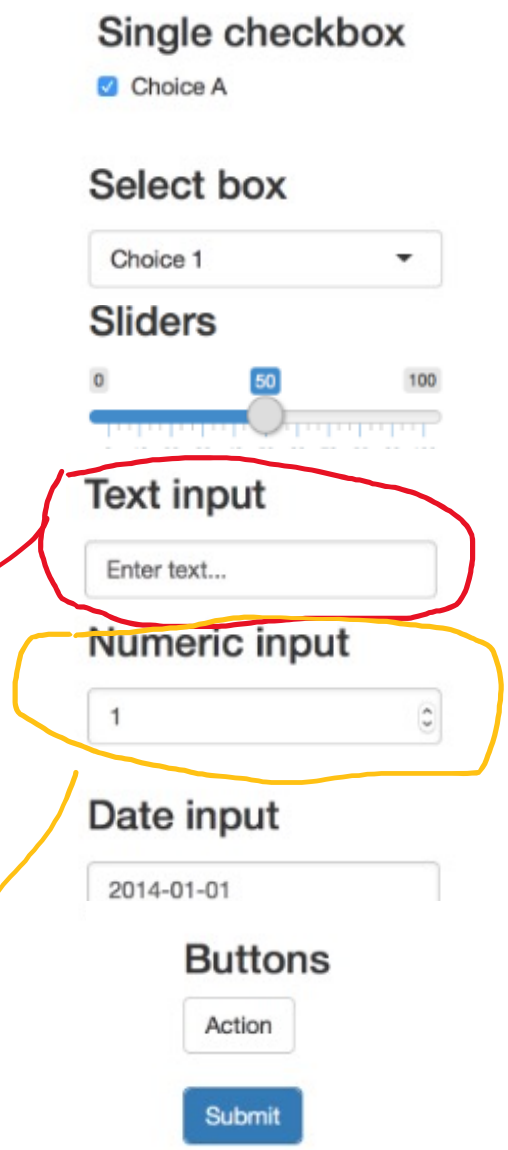
A **web element** that your users can **interact** with. Widgets provide a way for your users to **send messages** to the Shiny app.

Shiny widgets **collect a value** from your user. When a user **changes** the widget, the value will **change** as well

ad control widgets

```
fluidPage(

  # Copy the line below to make a text input box
  textInput("text", label = h3("Text input"), value = "Enter text..."),

  hr(),
  fluidRow(column(3, verbatimTextOutput("value")))

)
```

```
fluidPage(

  # Copy the line below to make a number input box into the UI.
  numericInput("num", label = h3("Numeric input"), value = 1),

  hr(),
  fluidRow(column(3, verbatimTextOutput("value")))

)
```



Tjek: [widget-gallery]

SHINY UI

Reactive output **automatically responds** when your user toggles a **widget**.

Step 1: Add an R object to the UI

Step 2: Provide R code to build the object.

Make the text reactive by asking Shiny to **call a widget value** when it builds the text.

display reactive output

```
ui <- fluidPage(
  titlePanel("censusVis"),
```

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

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

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

}
```

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

  }
```

Output function	Creates
dataTableOutput	DataTable
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	raw HTML
verbatimTextOutput	text

SHINY UI

display reactive output

```
ui <- fluidPage(
  titlePanel("censusVis"),
    selectInput("var",
                label = "Choose a variable to display",
                choices = c("Percent White",
```

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

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

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

}
```

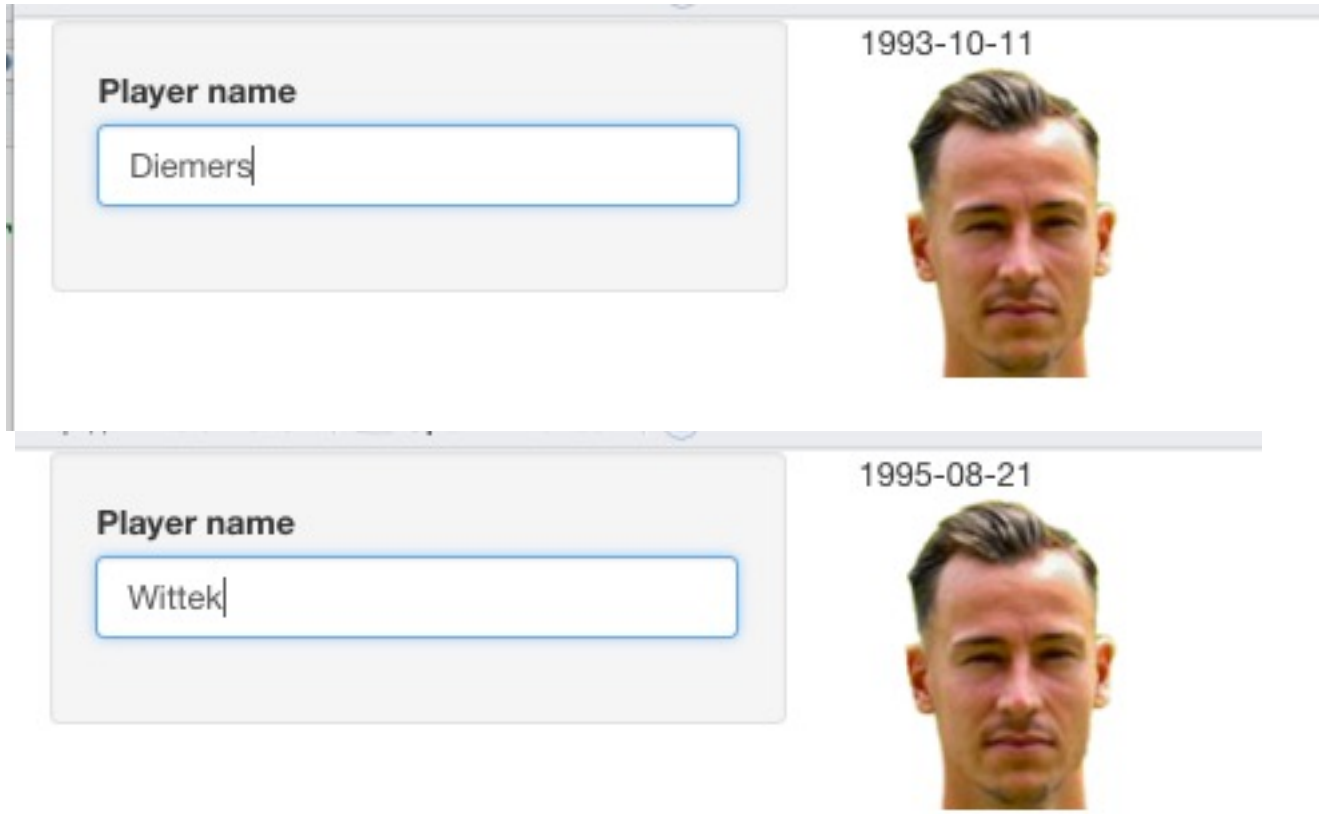
Make the text reactive by asking Shiny to

call a widget value

when it builds the text.

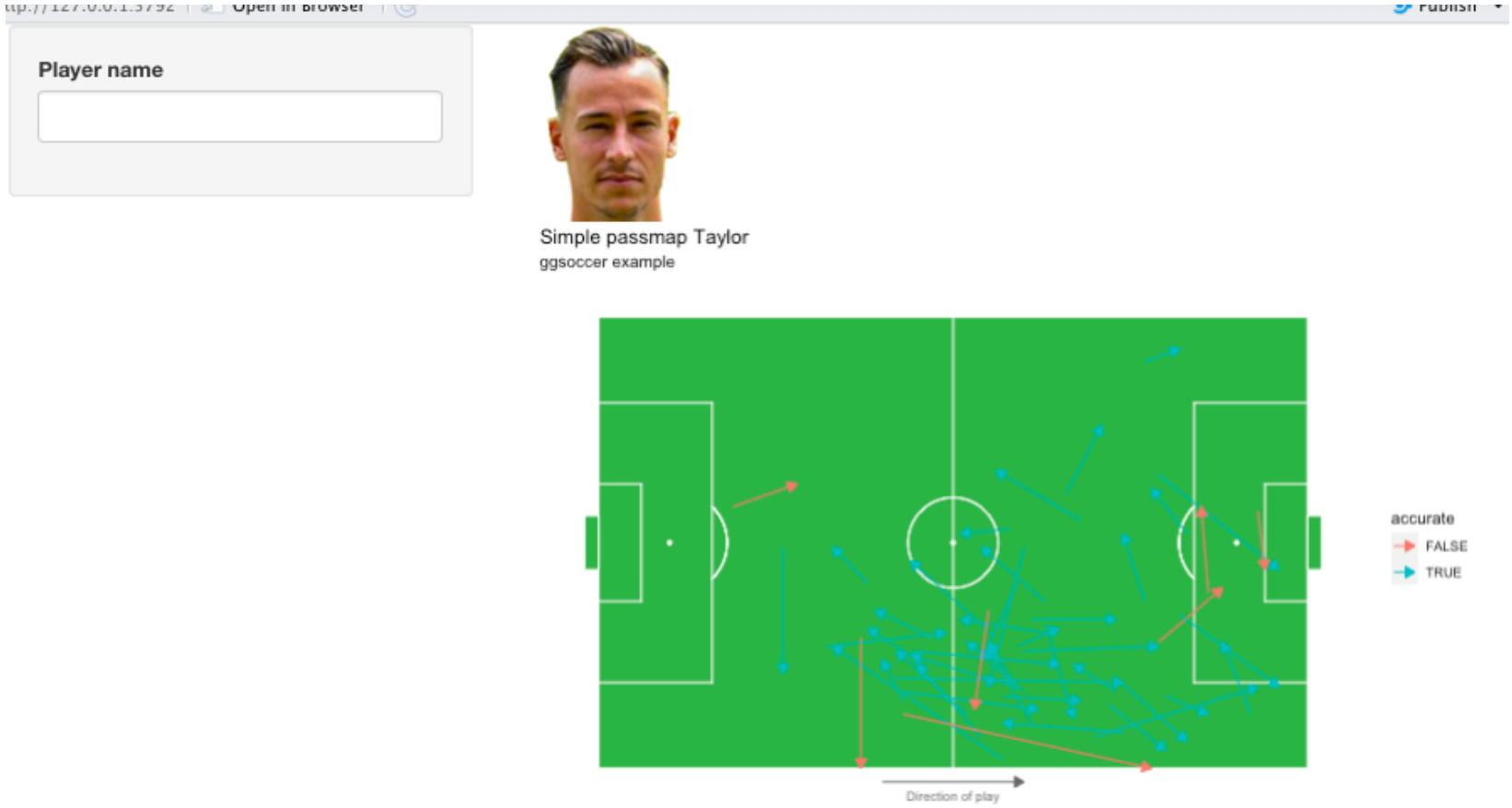
SHINY øvelse

build a user interface



SHINY øvelse

build a user interface



SHINY SERVER

use R scripts and data

Shiny will run the whole script the **first time** you call runApp.

Each time a new user visits your app, Shiny runs the server function again, one time. The function helps Shiny build a **distinct set of reactive objects** for each user.

As users interact with the widgets and change their values, Shiny will re-run the R expressions assigned to each reactive object that depend on a widget whose value was changed.

```
# Server logic ----
server <- function(input, output) {
  output$map <- renderPlot({
    percent_map( # some arguments )
  })
}
```

Run once when app is launched

Run once each time a user visits the app

Run once each time a user changes a widget that output$map depends on

SHINY SERVER

use R scripts and data

CREATE **OUTPUT** *to* UI <- renderfunctions

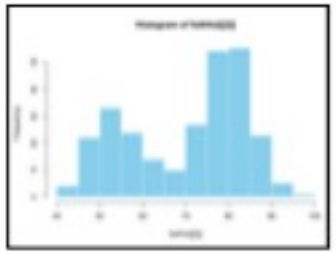
ACESSING **INPUT VALUES** *from* UI <- input$inputID

```
server <- function(input,output) {
  # handle output
  output$uiidofplot <- renderPlot(
   range <- input$slideID
   ...
  )
```

SHINY SERVER

use R scripts and data

Vi har vores data



```
'data.frame':   3 obs. of  2 variables:
 $ Sepal.Length: num  5.1 4.9 4.7
 $ Sepal.Width : num  3.5 3 3.2
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
4	4.60	3.10	1.50	0.20	setosa
5	5.00	3.60	1.40	0.20	setosa
6	5.40	3.90	1.70	0.40	setosa

Og R code (ggplot)

Source on Save

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

  output$plot2<-renderPlot({
    ggplot(dat(),aes(x=date,y=num))+
      geom_point(colour='red')})}

shinyApp(ui, server)
```

Og Shiny's funktioner til at sende vore data til brugeren

DT::**renderDataTable**(expr, options, callback, escape, env, quoted)	works with	**dataTableOutput**(outputId, icon
renderImage(expr, env, quoted, deleteFile)		**imageOutput**(outputId, width, he dblclick, hover, hoverDelay, hoverI brush, clickId, hoverId, inline)
renderPlot(expr, width, height, res, …, env, quoted, func)		**plotOutput**(outputId, width, heig dblclick, hover, hoverDelay, hoverI brush, clickId, hoverId, inline)
renderPrint(expr, env, quoted, func, width)		**verbatimTextOutput**(outputId)
renderTable(expr,…, env, quoted, func)		**tableOutput**(outputId)
renderText(expr, env, quoted, func)		**textOutput**(outputId, container, i

```r
ui <- fluidPage(
  titlePanel(title=h4("Races", align="ce
  sidebarPanel(
    sliderInput("num", "Number:",min=0,m
  mainPanel(plotOutput("plot2")))
```

SHINY SERVER

use R scripts and data

DT::**renderDataTable**(expr, options, callback, escape, env, quoted)	works with	**dataTableOutput**(outputId, icon
renderImage(expr, env, quoted, deleteFile)		**imageOutput**(outputId, width, he dblclick, hover, hoverDelay, hoverI brush, clickId, hoverId, inline)
renderPlot(expr, width, height, res, …, env, quoted, func)		**plotOutput**(outputId, width, heig dblclick, hover, hoverDelay, hoverI brush, clickId, hoverId, inline)
renderPrint(expr, env, quoted, func, width)		**verbatimTextOutput**(outputId)
renderTable(expr,…, env, quoted, func)		**tableOutput**(outputId)
renderText(expr, env, quoted, func)		**textOutput**(outputId, container, i

SHINY SERVER

use R scripts and data

renderText(expr, env, quoted, func**)**

textOutput(outputId, container, inline**)**

```
output$text1 <- renderText({paste("You have selected", input$var)
```

```
textOutput(outputId = "showslider")
```

SHINY SERVER

Reactive expressions let you control which parts of your app update when

A reactive expression is an R expression that **uses widget input** and returns a value.

If accessed **outside** a render-context you will get an error

use reactive expressions

Reactivity

- `server.r` can run any R code, but can't access inputs unless put into a reactive context
- All `render*` functions are reactive contexts

```
49    inputval=34
50    print(paste("Du endte på ", input$bins))
51    #print(paste("Du endte på ", inputval))
52
```

Console | Terminal | Background Jobs

R 4.1.1 · ~/Git/ShinyPlay/

```
Advarsel: Error in $: Can't access reactive value 'bins' outside of
i Do you need to wrap inside reactive() or observe()?
  55: <Anonymous>
Error in input$bins :
  Can't access reactive value 'bins' outside of reactive consumer.
```

SHINY SERVER

use reactive expressions

Reactivity

- `server.r` can run any R code, but can't access inputs unless put into a reactive context
- All `render*` functions are reactive contexts

Other Reactive Contexts

- `reactive({})` function allows for reactivity and creation of a new variable
- `observe({})` function allows for reactivity

```
newvar <- reactive({
  val <- paste("Du endte på ", input$bins)
})
output$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 = paste("Histogram of waiting times",newvar()))
})
```

SHINY SERVER

use reactive expressions

Reactive expressions let you control which parts of your app update when, which **prevents unnecessary computation**

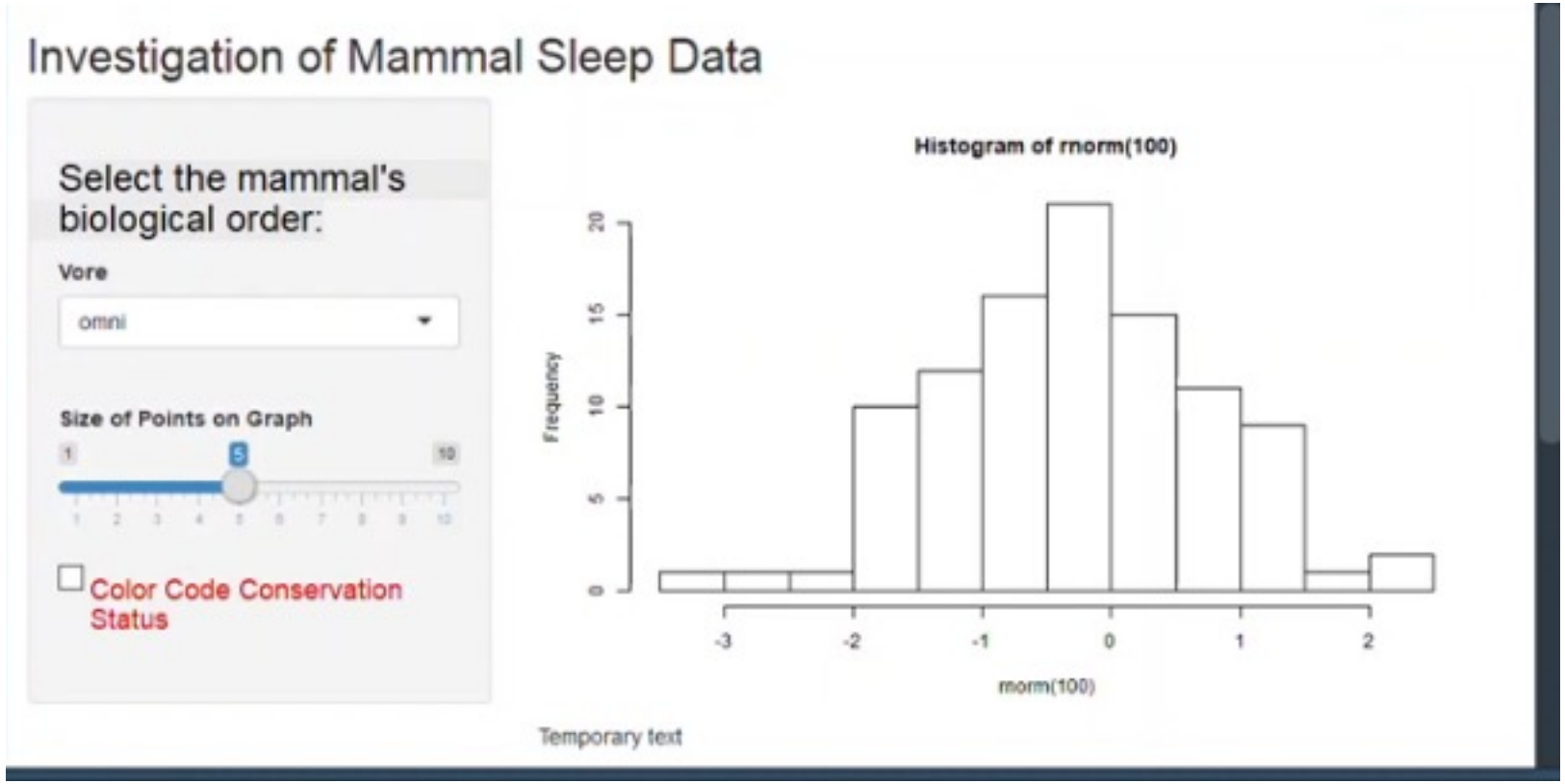
A reactive expression is an R expression that **uses widget input** and returns a value. The reactive expression will update this value **whenever the original widget changes**

```
output$plot <- renderPlot({
  data <- getSymbols(input$symb, src = "yahoo",
                     from = input$dates[1],
                     to = input$dates[2],
                     auto.assign = FALSE)

  chartSeries(data, theme = chartTheme("white"),
              type = "line", log.scale = input$log, TA = NULL)
})
```

```
dataInput <- reactive({
  getSymbols(input$symb, src = "yahoo",
    from = input$dates[1],
    to = input$dates[2],
    auto.assign = FALSE)
})
```

```
output$plot <- renderPlot({
  chartSeries(dataInput(), theme = chartTheme("white"),
    type = "line", log.scale = input$log, TA = NULL)
})
```

SHINY Øvelse

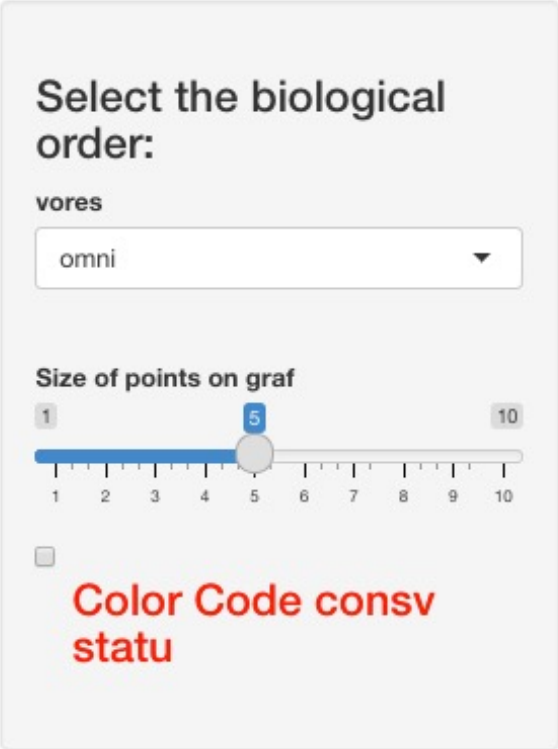


SHINY Øvelse

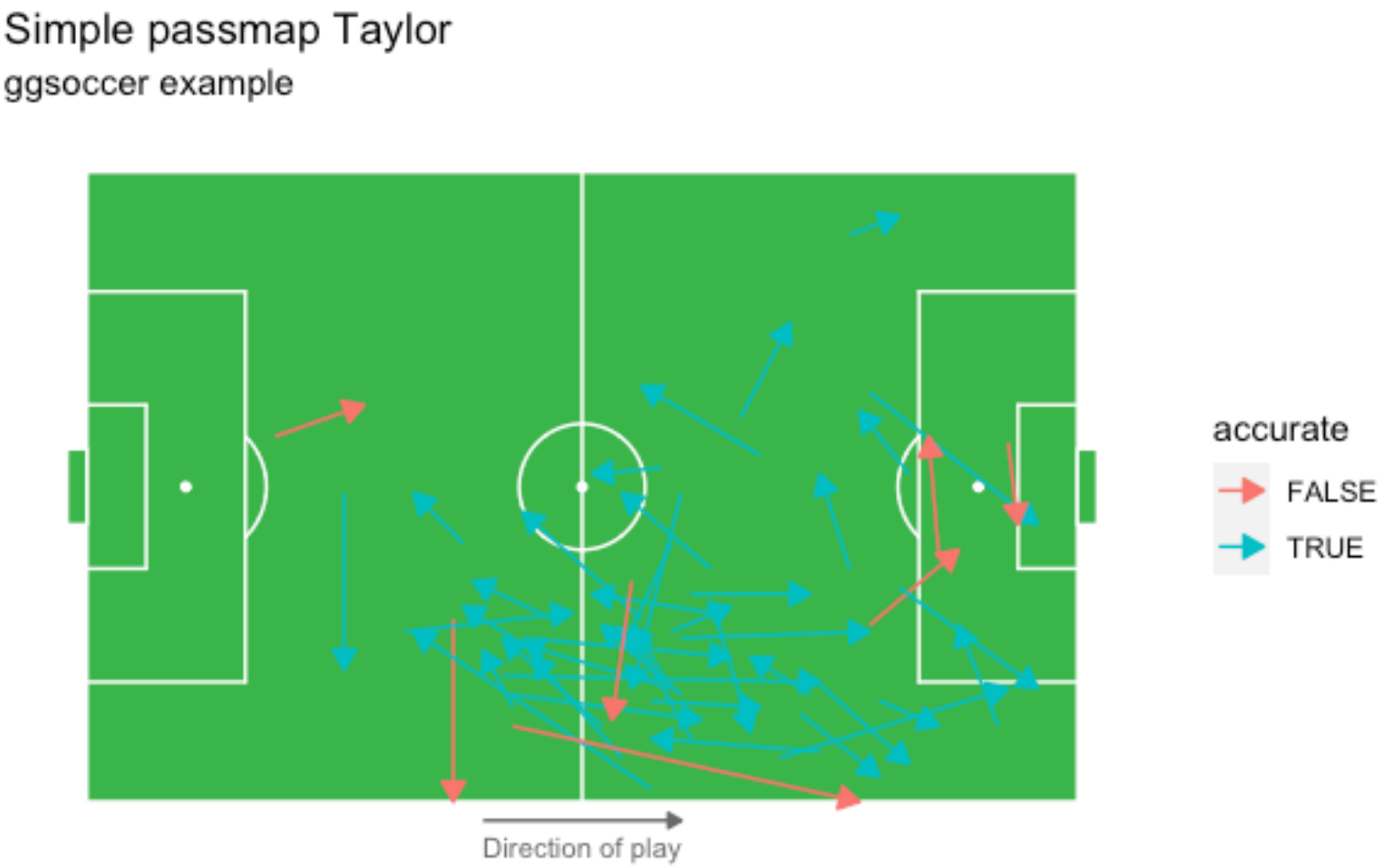
```
> str(msleep)
tibble [83 × 11] (S3: tbl_df/tbl/data.frame)
 $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-
 $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
 $ vore        : chr [1:83] "carni" "omni" "herbi" "omni" ...
 $ order       : chr [1:83] "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
 $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
 $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
 $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
 $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
 $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
 $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
 $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

vore: organiseret efter føde
genus: slægt (Panthera del af Rovdy, Vulpes er en slægt af ræve)
name: engelsk navn for dens genus
order: art (Carnivora=Rovdyr)

SHINY Øvelse

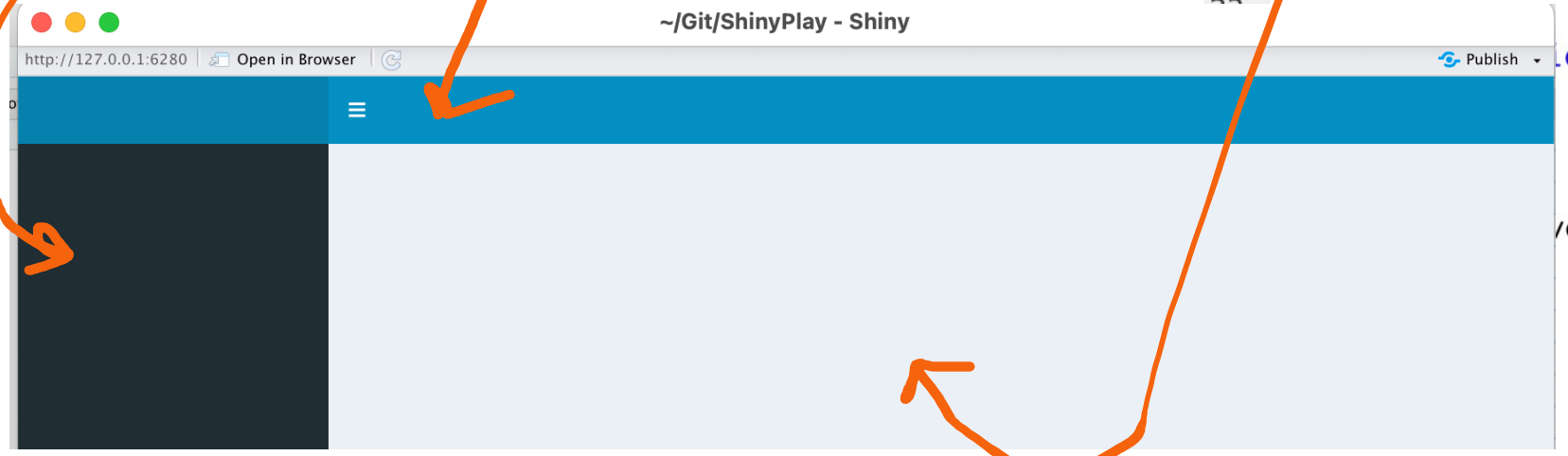


SHINY Øvelse



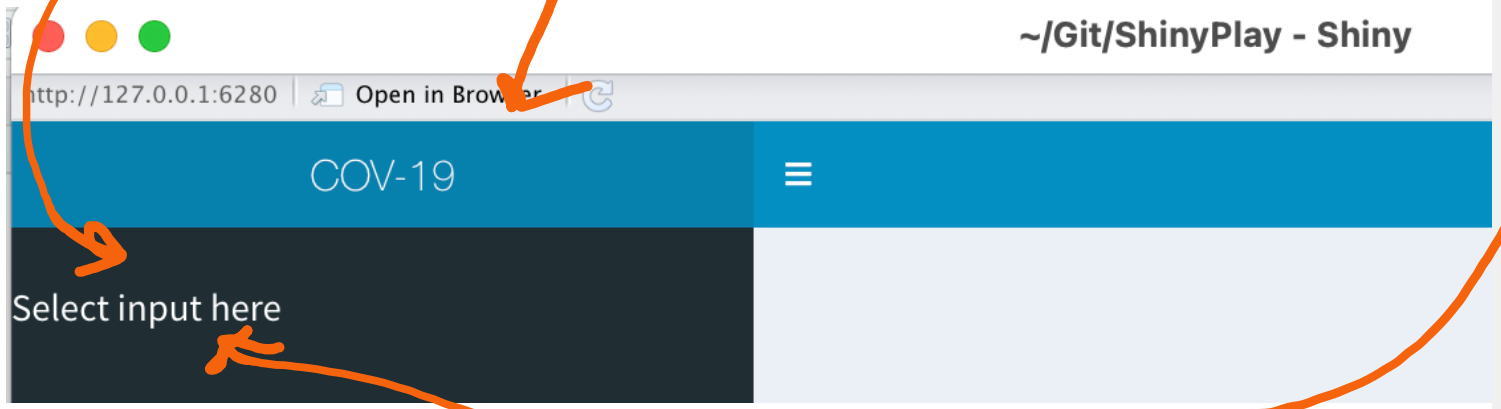
SHINY II

```
library(shinydashboard)


#ui <- fluidPage(
ui <- dashboardPage(
  dashboardHeader(

  ),

  dashboardSidebar(

  ),
  dashboardBody(

  )
)
```




```
on(input,output,session) {


ver)
```

SHINY II

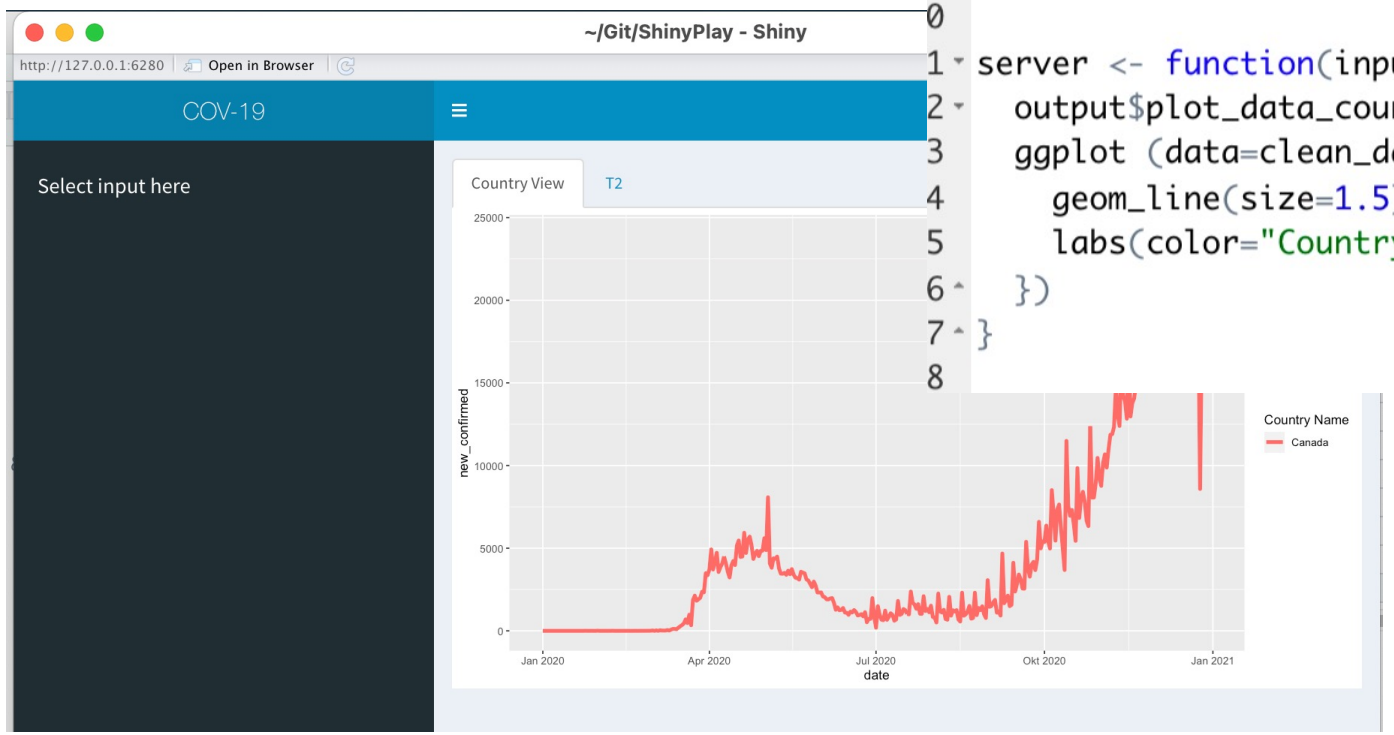



```r
library(shinydashboard)


#ui <- fluidPage(
ui <- dashboardPage(

  dashboardHeader(
    title = "COV-19",
    titleWidth=350
  ),

  dashboardSidebar(
    width=350,
    br(),
    h4("Select inpulllt here", style = "padding-l
  ),

  dashboardBody(

  )
)
```

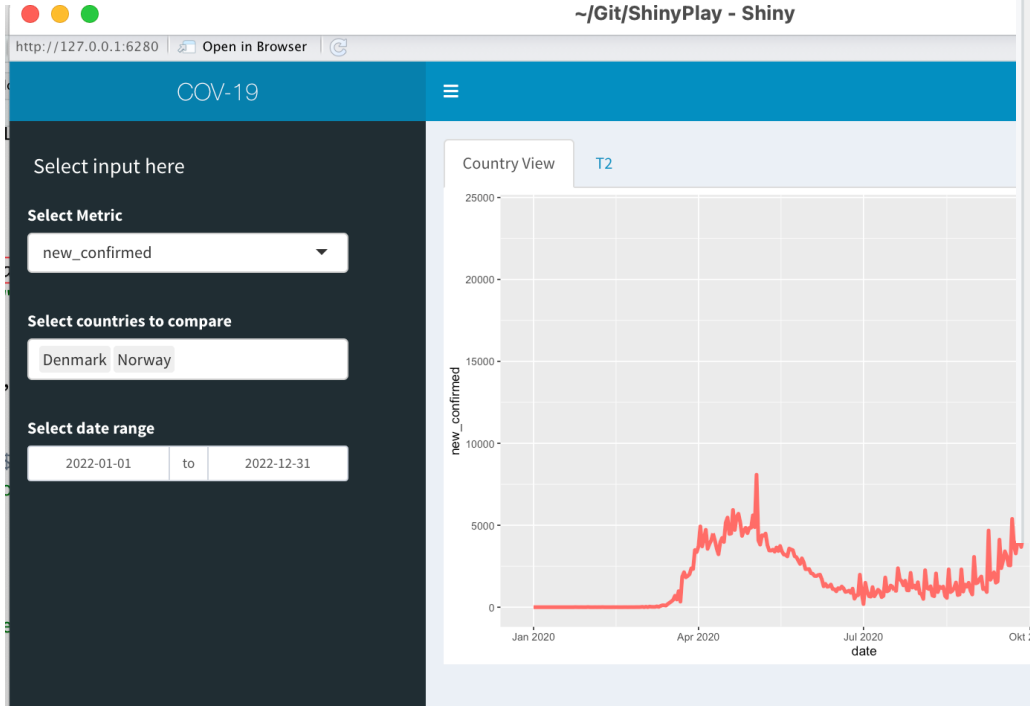
SHINY II

```
  dashboardSidebar(
    width=350,
    br(),
    h4("Select input here", style = "padding-left:20px")
  ),
```

```
)

server <- function(input,output,session) {
  output$plot_data_country <- renderPlot({
  ggplot (data=clean_dat, aes(y=new_confirmed,x=date, color=country_name))+
    geom_line(size=1.5) +
    labs(color="Country Name")
  })
}
```

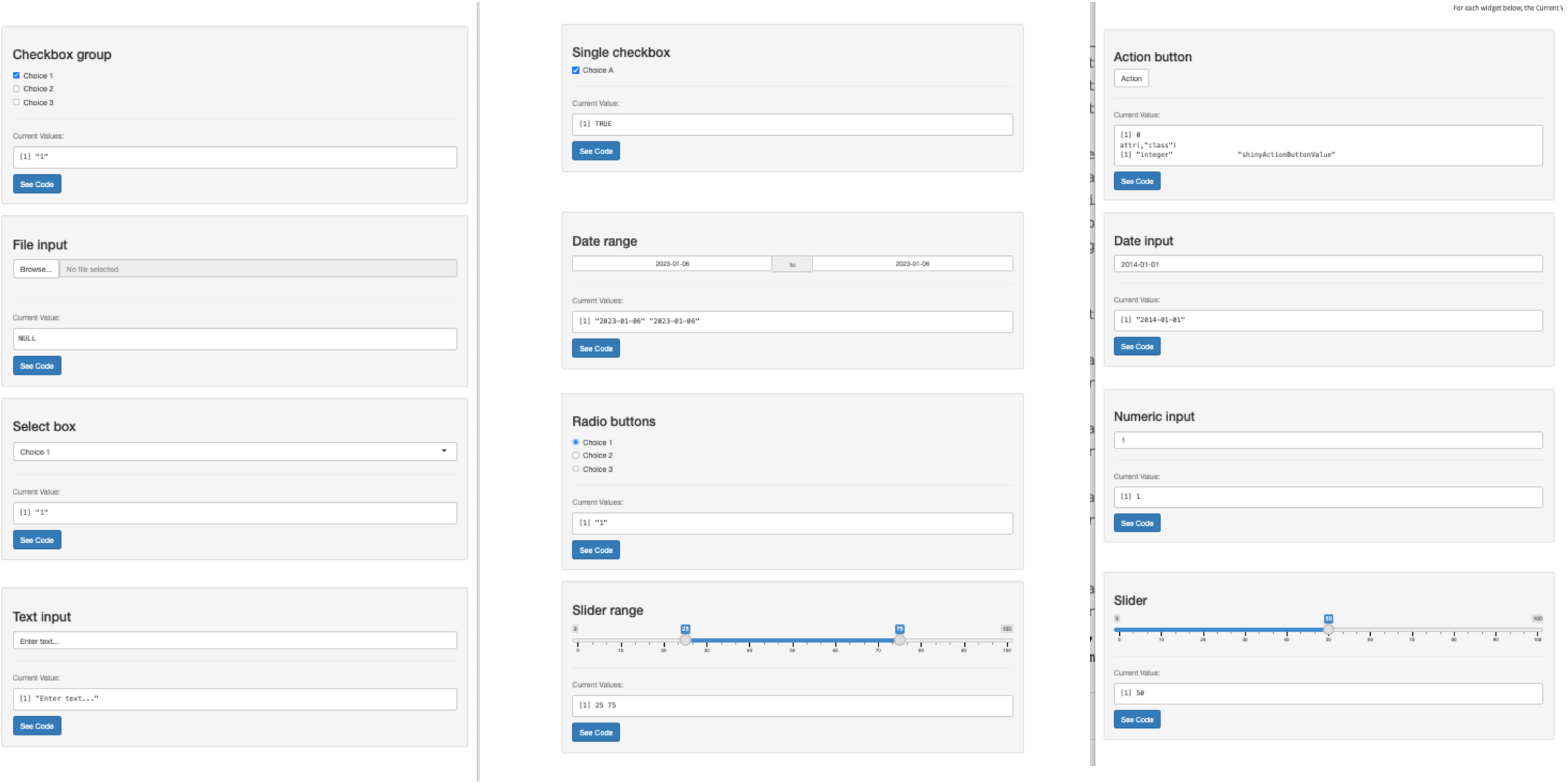


SHINY II




```r
    selectInput(
      inputId = "mtr",
      label = "Select Metric",
      choices = sort(colnames(dat)[4:ncol(dat)]),
      selected = "new_confirmed"
    ),

   selectInput( inputId = "ctr",
      multiple = T,
      label = "Select countries to compare",
      choices = sort(unique(dat$country_name)),
      selected = c("Denmark","Norway","Sweeden")
    ),

    dateRangeInput(
      inputId = "dR",
      label = "Select date range",
      start = "2022-01-01",
      end = "2022-12-31"
    )
  ),
```

SHINY III


Checkbox group
Choice 1
Choice 2
Choice 3
Current Values:
[1] "1"
See Code
File input
Browse...
No file selected
Current Value:
NULL
See Code
Select box
Choice 1
Current Value:
[1] "1"
See Code
Text input
Enter text...
Current Value:
[1] "Enter text..."
See Code
Single checkbox
Choice A
Current Value:
[1] TRUE
See Code
Date range
2023-01-06
to
2023-01-06
Current Values:
[1] "2023-01-06" "2023-01-06"
See Code
Radio buttons
Choice 1
Choice 2
Choice 3
Current Values:
[1] "1"
See Code
Slider range
25
75
Current Values:
[1] 25 75
See Code
For each widget below, the Current V
Action button
Action
Current Value:
[1] 0
attr(,"class")
[1] "integer" "shinyActionButtonValue"
See Code
Date input
2014-01-01
Current Value:
[1] "2014-01-01"
See Code
Numeric input
1
Current Value:
[1] 1
See Code
Slider
50
Current Value:
[1] 50
See Code

SHINY III