# Building interactive visualizations with R

Pau Palop-García and Felix Haaß

GIGA German Institute of Global and Area Studies

June 2018

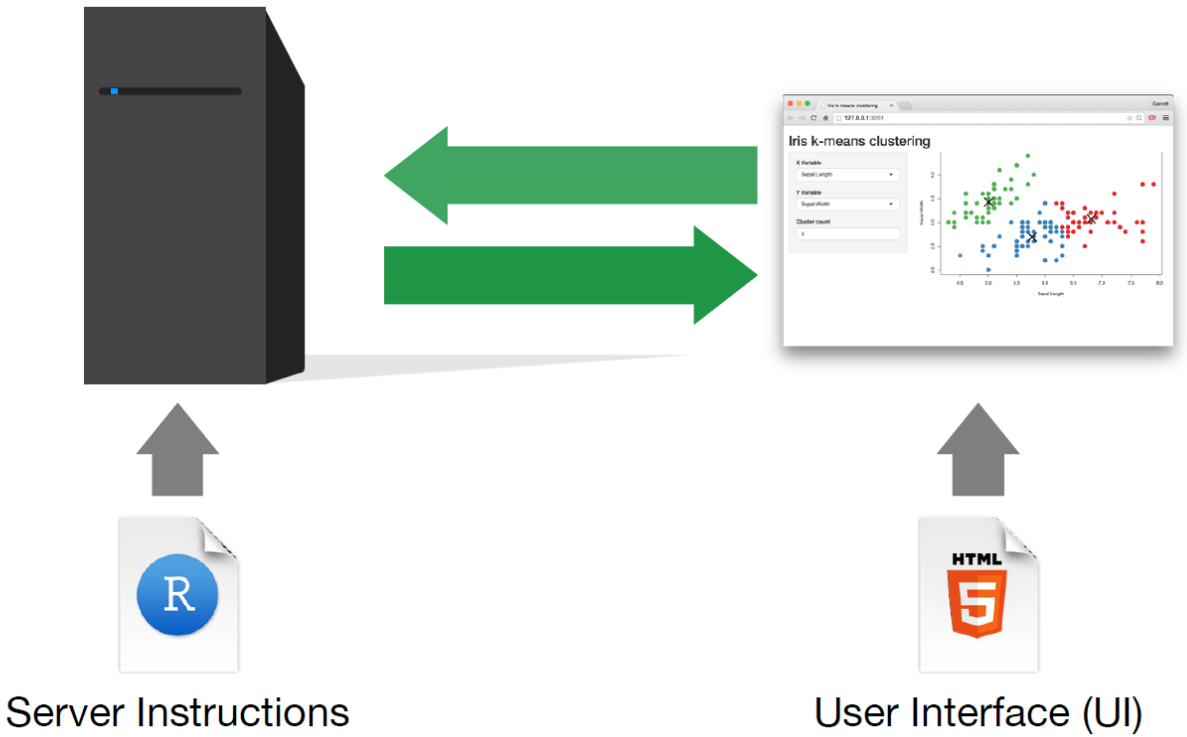
# Introduction

- What is an interactive visualization?
- Course outline

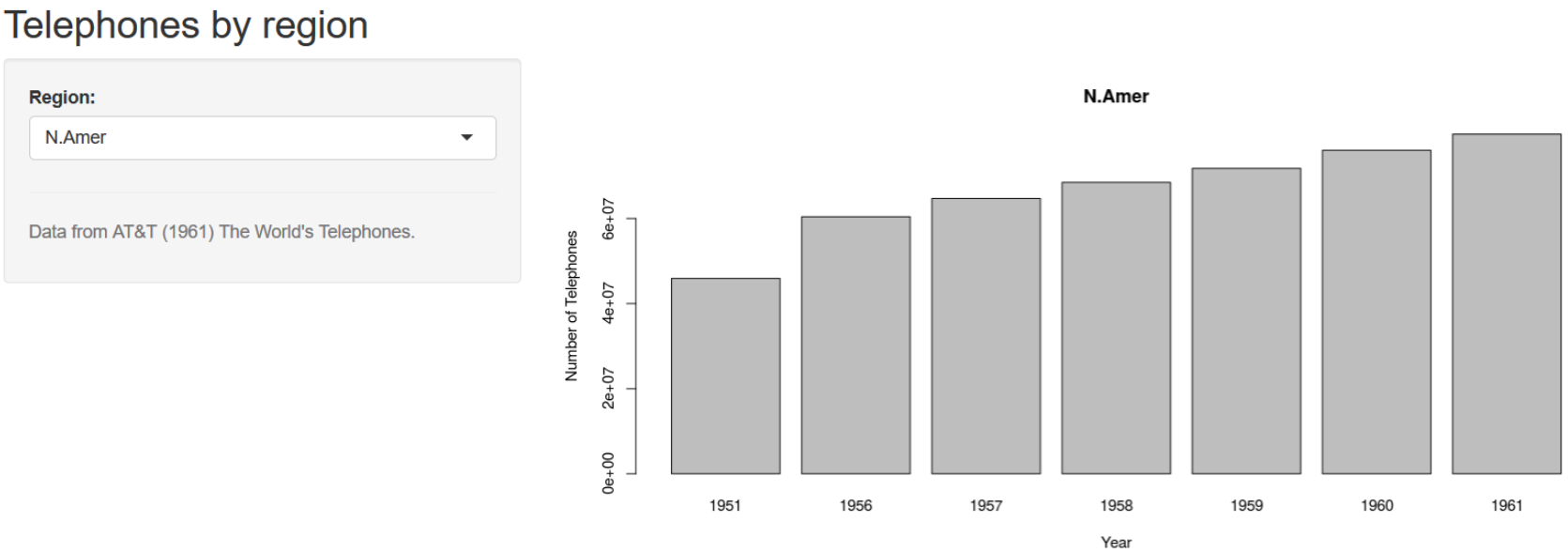
# Main steps for building an app

- Set main goal of the app
- Install R Studio and necessary packages (e.g. "shiny")
- Select visualization
- Prepare the data accordingly
- Code the app
- Deploy and share the app

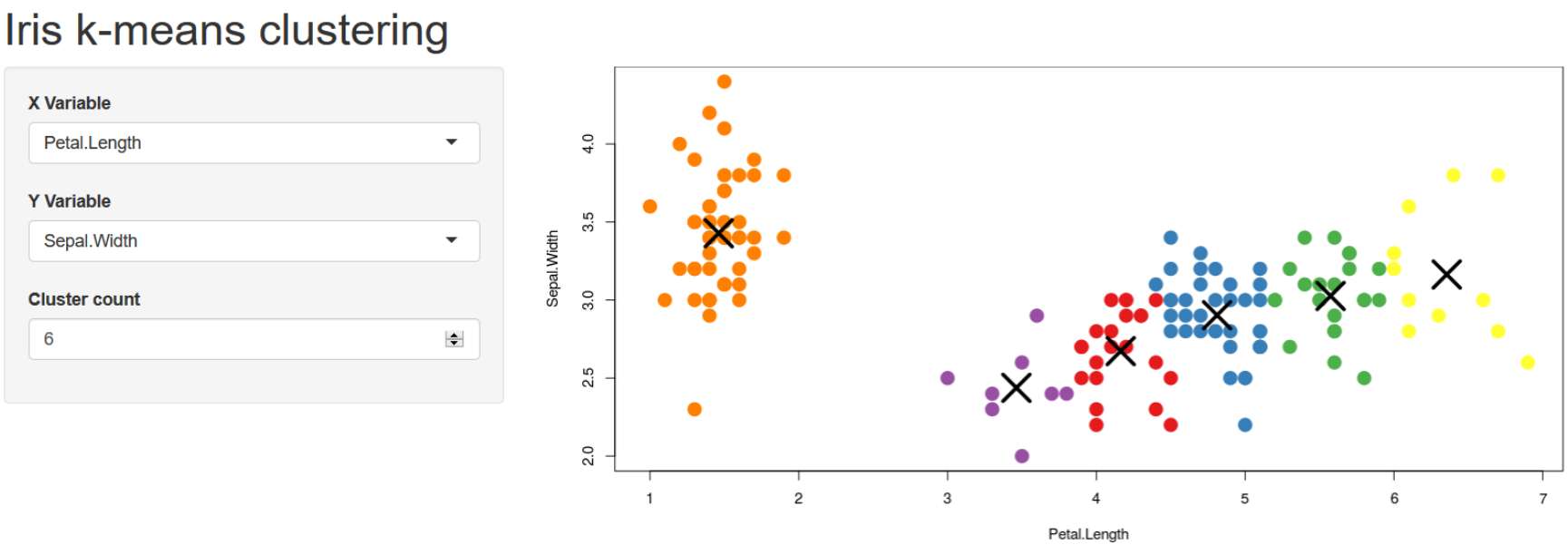
# An app...



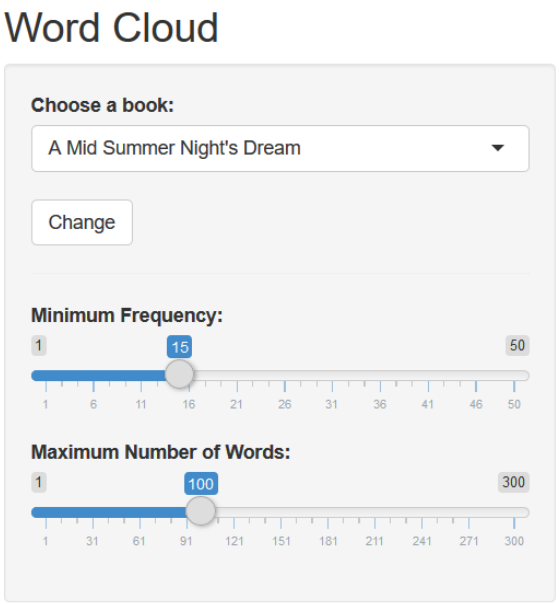
# Selecting the appropriate visualization

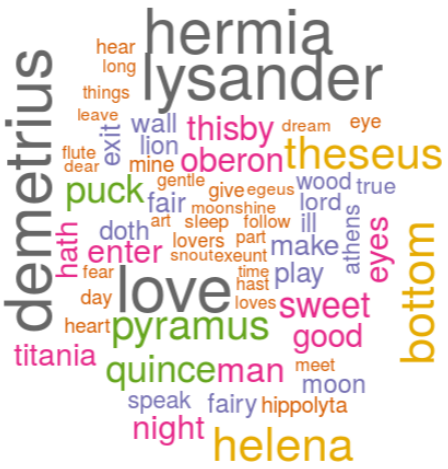


# Selecting the appropriate visualization

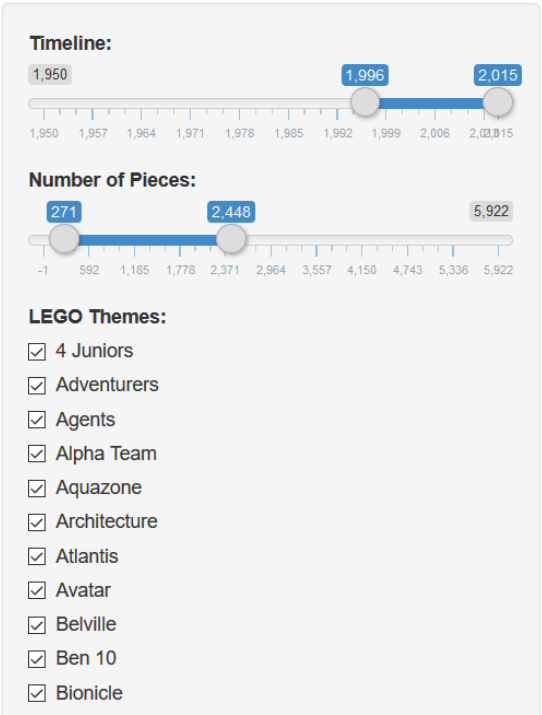


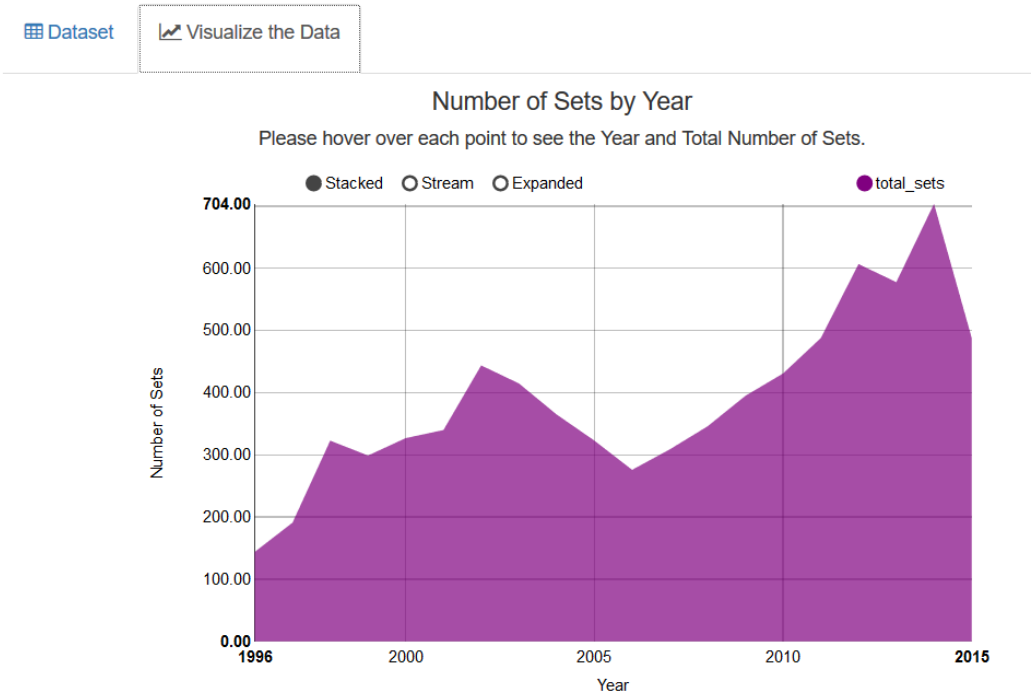
# Selecting the appropriate visualization



# Selecting the appropriate visualization



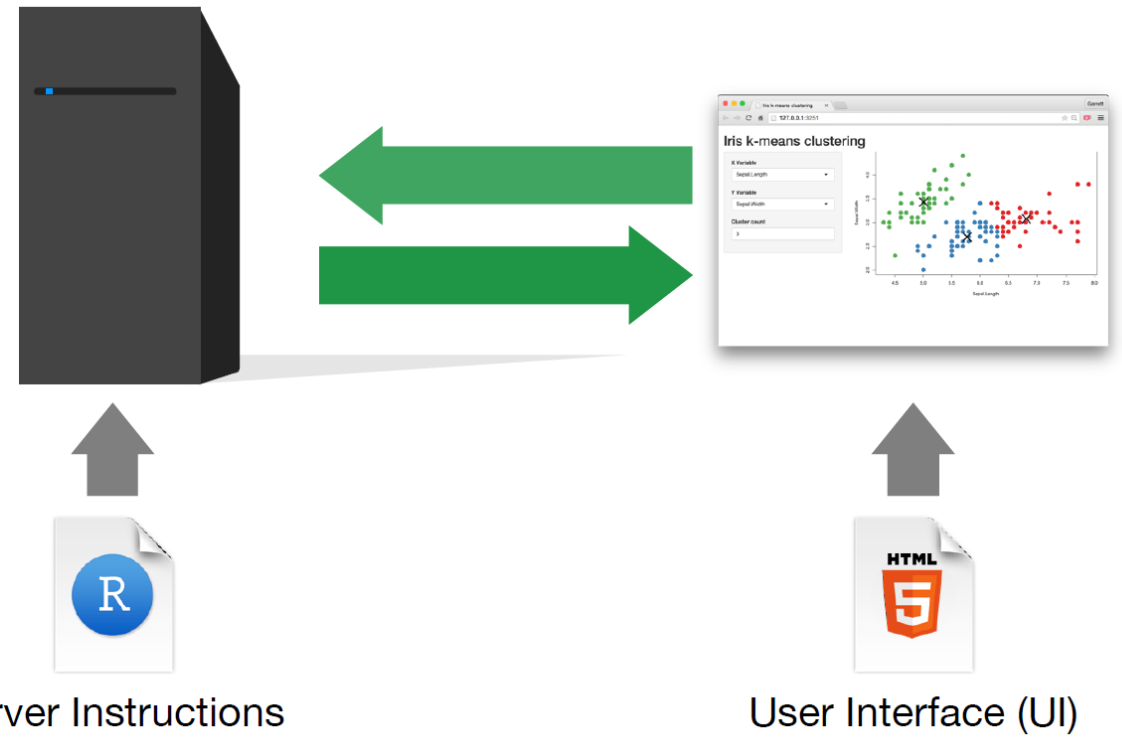
# Preparing the data

- Depends on the app
- General rule: keep it simple

# Structure of a ShinyApp

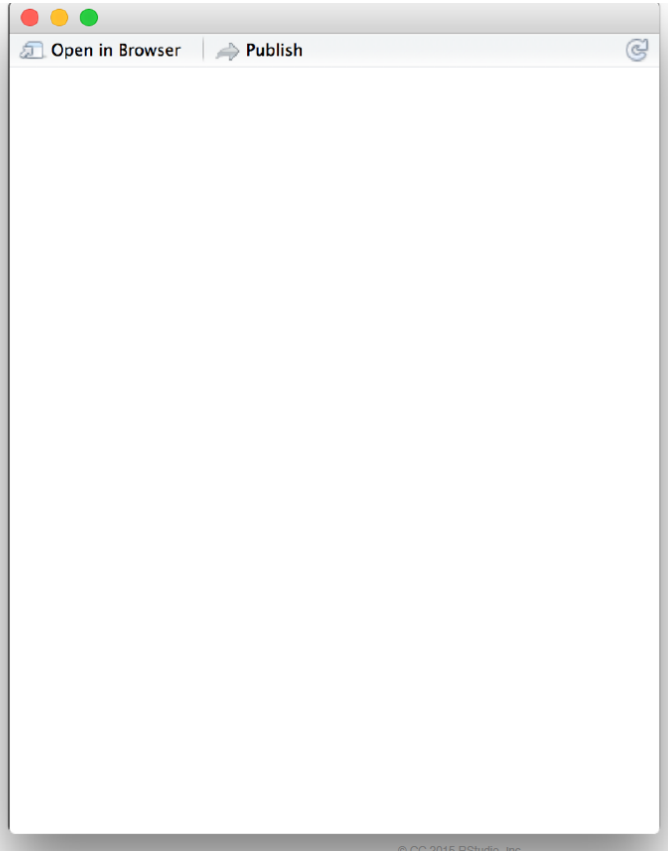
- **A user interface (ui):** controls the app layout
- **A server function (server):** contains all the functions needed to build the app
- **A call to the shinyApp function:** creates a shiny app that pairs ui and server

# Building the simplest app



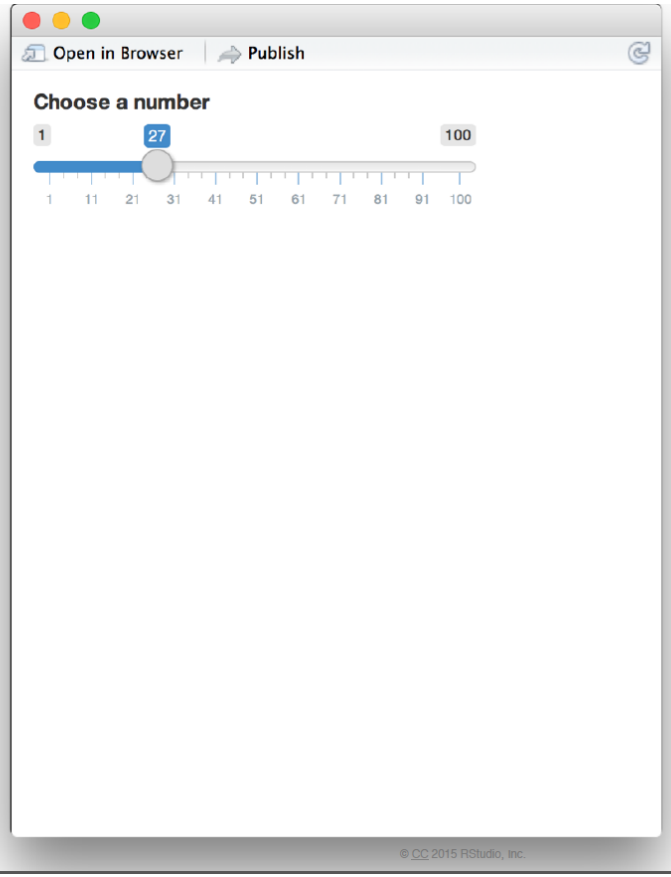
# Building the simplest app

```
library(shiny)

ui <- fluidPage(




)

server <- function(input, output) {



}

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



# Building the simplest app

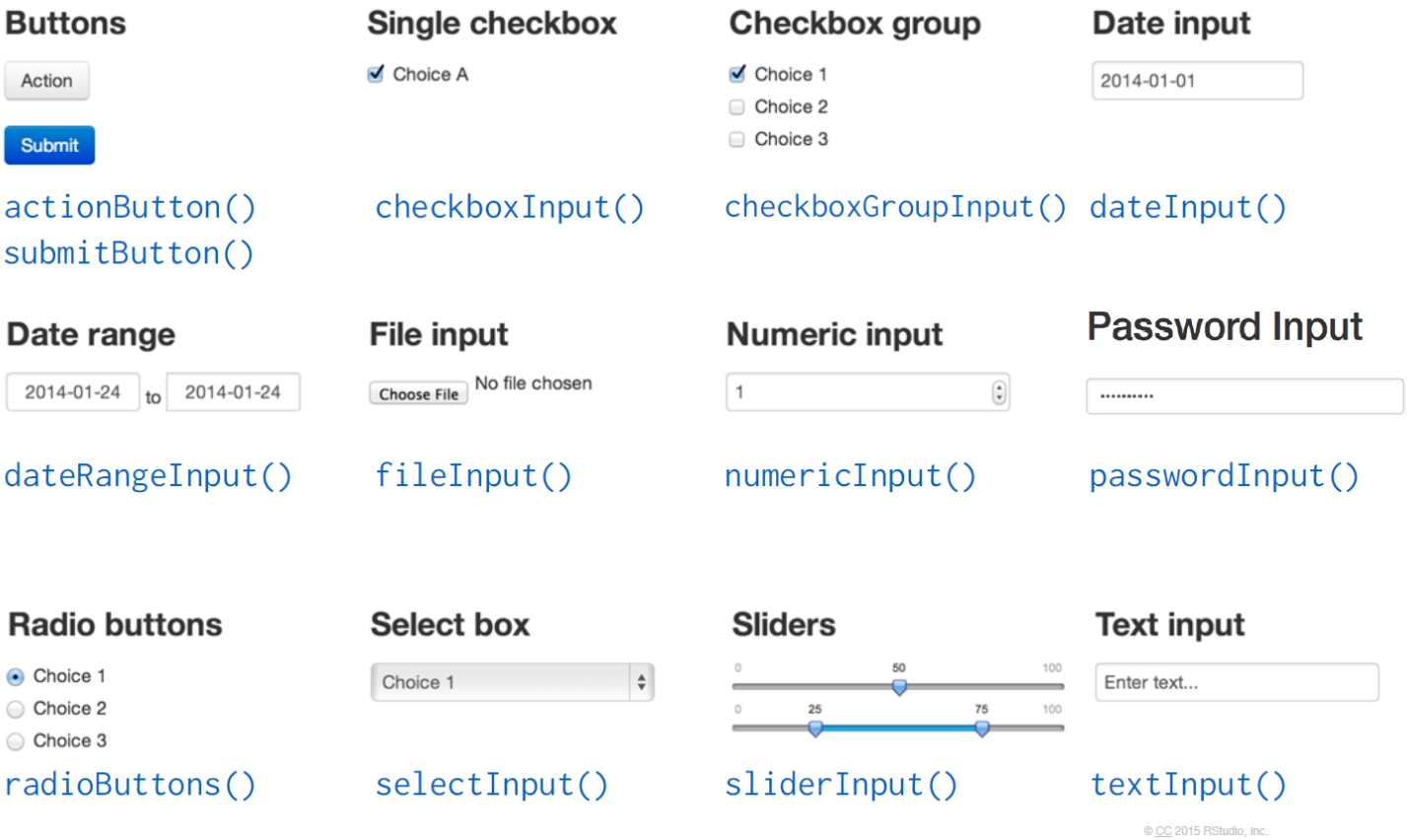
```
library(shiny)

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

)

server <- function(input, output) {



}

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



# Widgets

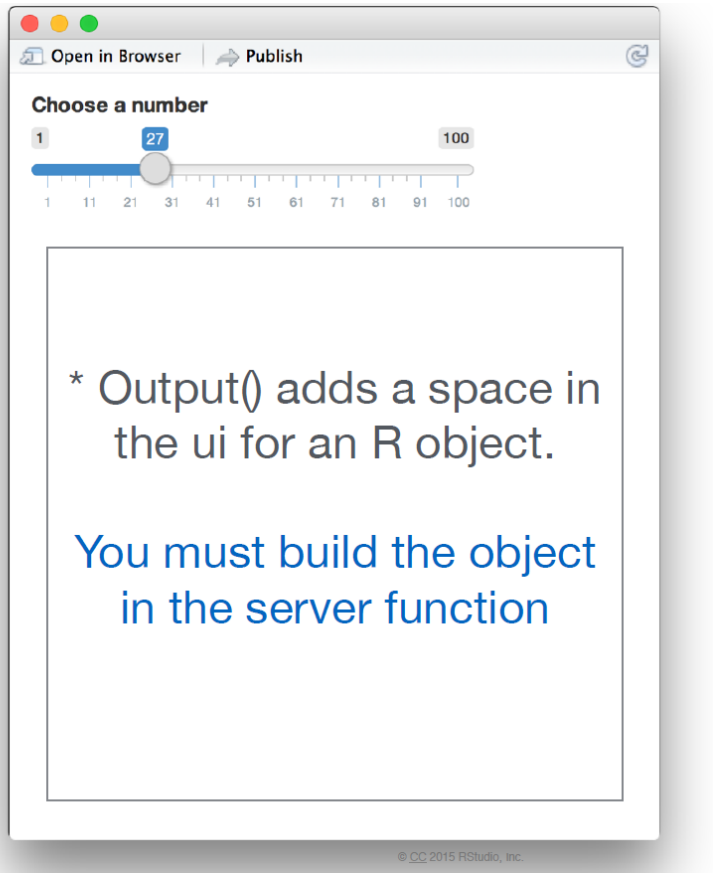
- **Definition:** a web element that users can interact with
- A way for users to send messages to the Shiny app
- Widgets are R functions: they required at least two elements to work: a name and a label

# Widgets



# Building the simplest app

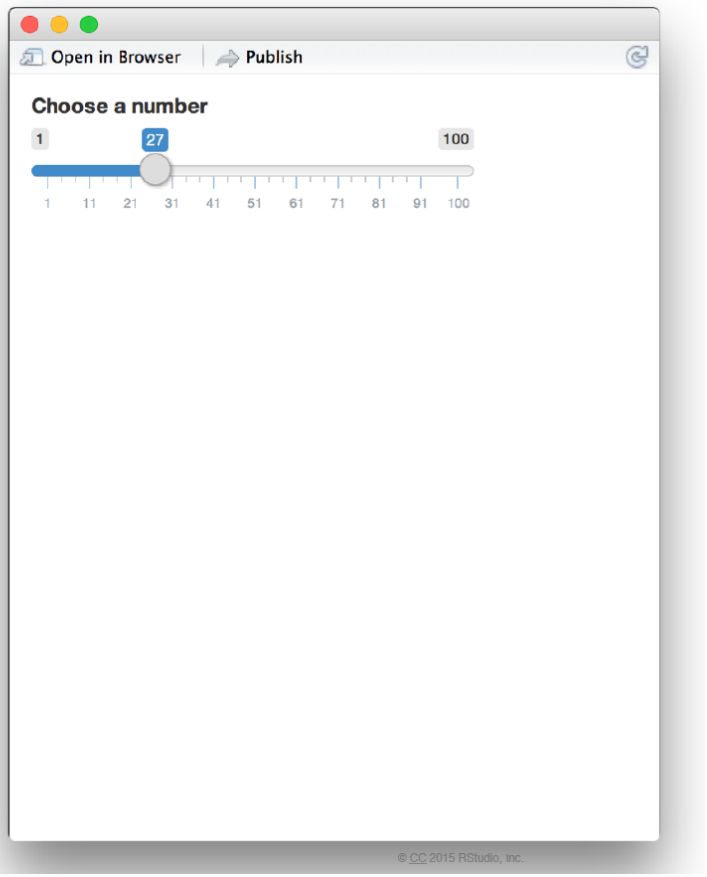
```
library(shiny)

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

server <- function(input, output) {



}

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



# Building the simplest app

```
library(shiny)

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

server <- function(input, output) {
  output$hist <-


}

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

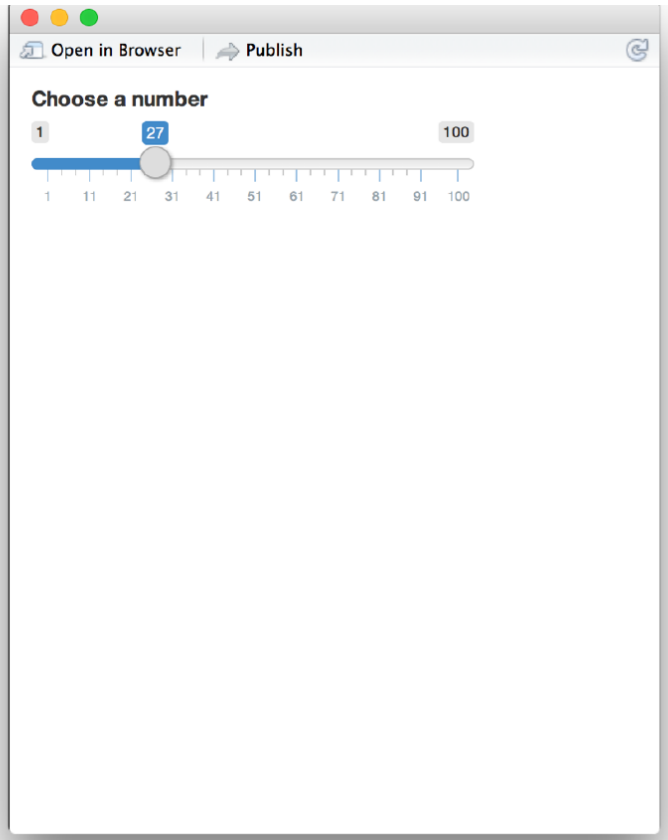
1



# Building the simplest app

```
library(shiny)

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

server <- function(input, output) {
  output$hist <- renderPlot({

  })
}

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

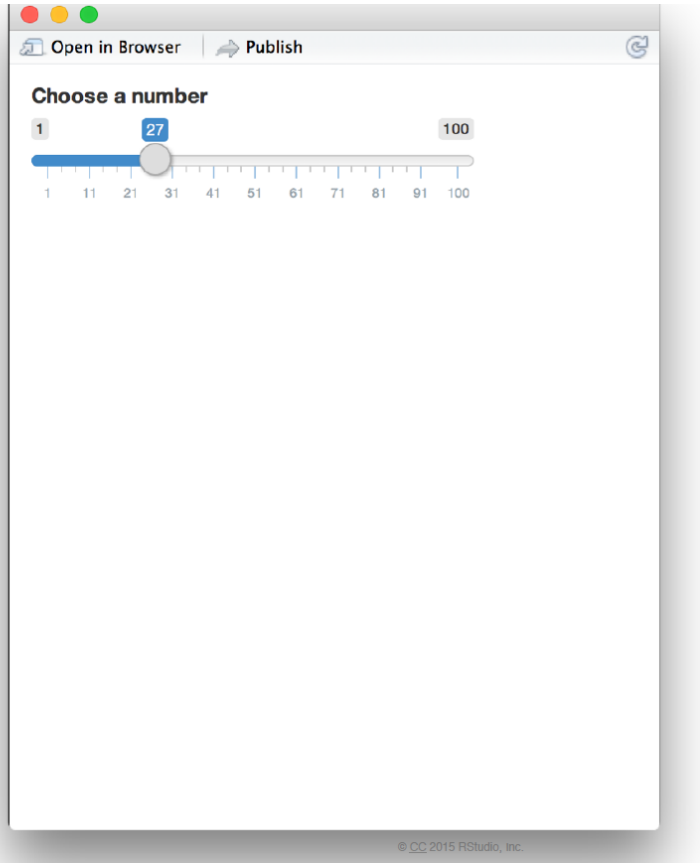
2



# Building the simplest app

```
library(shiny)

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

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

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

3



# Reactive outputs

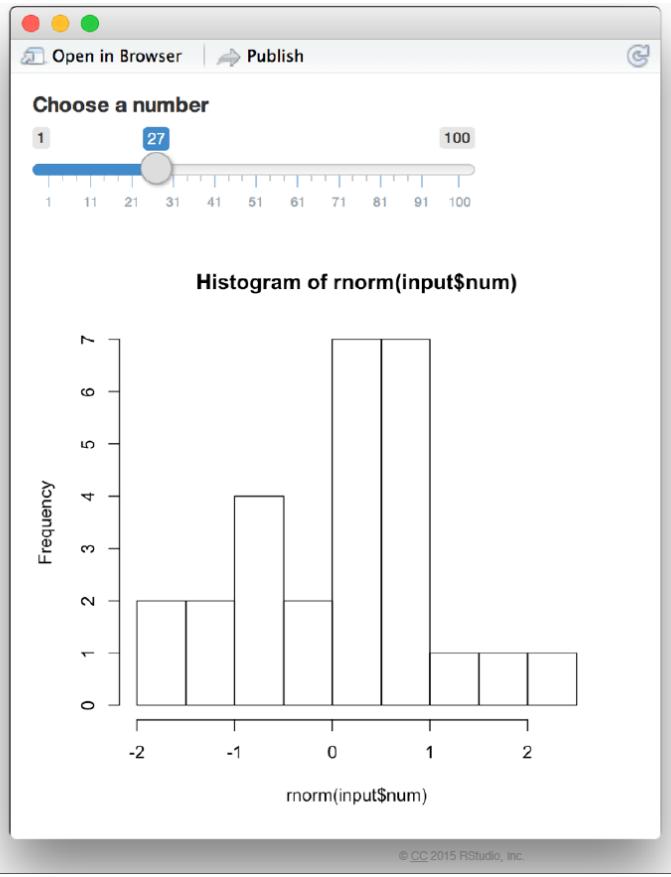
**Reactive outputs respond when users toggles a widget**

Two main steps:

- Add an R object to the user interface
- Tell shiny how to build the object in the server function

# Building the simplest app

```
library(shiny)

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

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

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



# Deploying and sharing the app

Every shiny app is maintained by a computer running R How to save it:

- app.R
- datasets, images, css, helper scripts, etc.

# Deploying and sharing the app

Using Shinyapps.io

- Free
- Secure
- Scalable

# Sharing in GIGA website via iframe

# More info:

- https://shiny.rstudio.com/tutorial/
- https://www.shinyapps.io/
- https://www.rstudio.com/