



*R Shiny*

## Introduction



- Web Applications with [R Shiny](#)
- Requires the [Shiny Package](#) in R
- Check Out R Shiny [Cheat Sheet](#)
- [Gallery](#) of Shiny Applications
- Deployable by [shinyapps.io](http://shinyapps.io)



# Introduction



- Web Programming Basics
  - HTML: Controls the Organization of the Web Page and Gives Markup Instructions
  - CSS: Controls the Style
  - Javascript: Interactivity



# Introduction



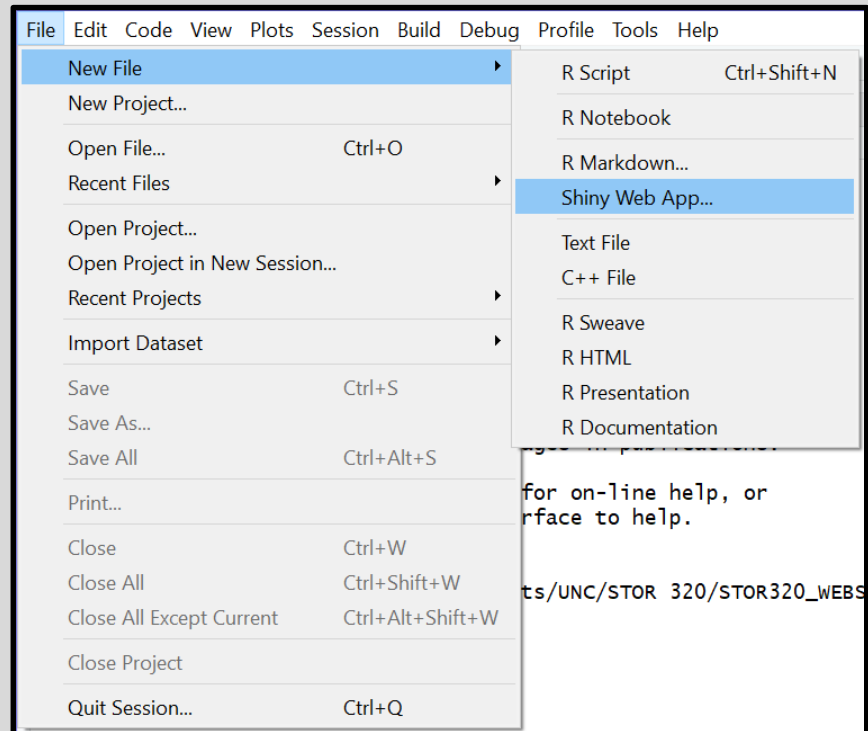
- Planning What You Want to Do
  - User Controls \_\_\_\_\_
  - Output Given is \_\_\_\_\_
  - R Code I Need is \_\_\_\_\_



# Getting Started



- Step 1: Install Shiny Package  
> `install.packages("shiny")`
- Step 2: Load the Library  
> `library(shiny)`
- Step 3: Create a New Shiny App




# Getting Started



- Step 4: Initiate Your Shiny App

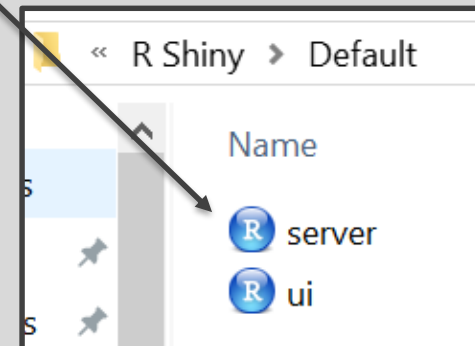
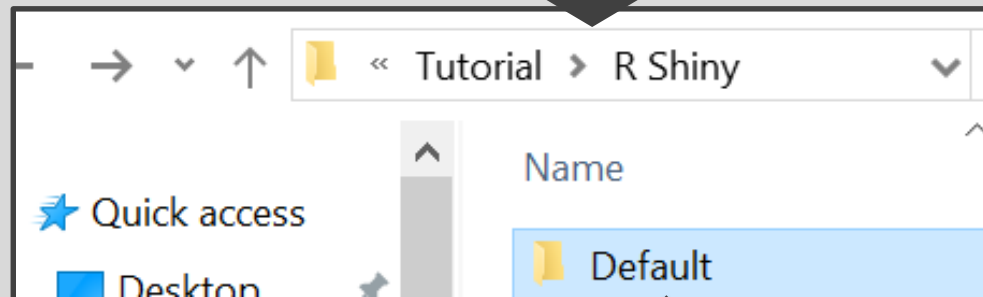
New Shiny Web Application



Application name:

Application type:  Single File (app.R)  
 Multiple File (ui.R/server.R)

Create within directory:



# Getting Started

- Step 5: Run the Shiny App

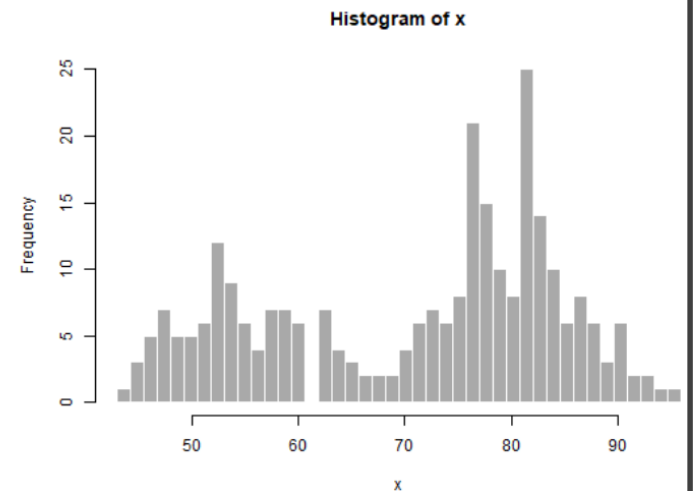
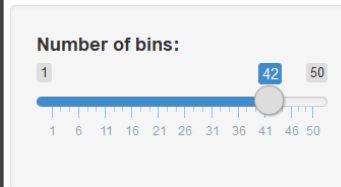
```
ui.R x server.R x
1 #
2 # This is the user-interface definition of a Shiny web application. You can
3 # run the application by clicking 'Run App' above.
4 #
5 # Find out more about building applications with Shiny here:
6 #
7 # http://shiny.rstudio.com/
8 #
```

Look in Top Right Corner

▶ Run App

- Step 6: Play With the App

Old Faithful Geyser Data



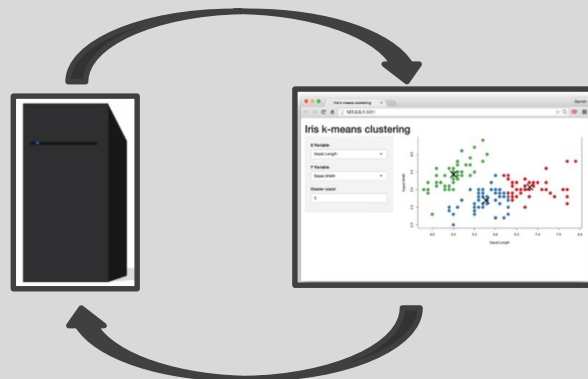
# Shiny Fundamentals



- How it Works?
  - Communication Between Your Computer and Your App



- Sharing Through the Cloud From a Web Based Server





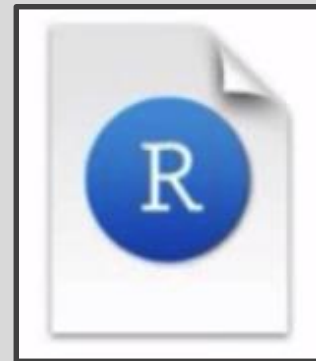
# Shiny Fundamentals



- Two Components in R
  - ui.R
    - User Interface
    - Written in HTML
    - Specify Inputs
    - Displays Outputs



- server.R
  - Instructions in R
  - Uses Inputs
  - Computations
  - Graphics
  - Models
  - Generates Outputs



# Shiny Fundamentals



- Types of Inputs (UI)
  - What User Can Control
  - List of Possible Widgets

input function	widget
actionButton	Action Button
checkboxGroupInput	A group of check boxes
checkboxInput	A single check box
dateInput	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

- [Shiny Widget Gallery](#)

# Shiny Fundamentals



- Types of Inputs (UI)
  - First Two Arguments
    - inputId = Unique Variable Name So Server Knows When to Use It
    - label = Text That is Seen in Widget to Guide User
  - Other Arguments Depend on the Type of Input Function

```
sliderInput(inputId = "bins",  
            label = "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)
```

# Shiny Fundamentals



- Types of Outputs (UI)
  - What User Can See
  - List of Possible Output Types

<b>output function</b>	<b>creates</b>
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	raw HTML
verbatimTextOutput	text

# Shiny Fundamentals



- Types of Outputs (UI)
  - `outputId` = Connected to Output Created on the Server Side

```
plotOutput(outputId = "distPlot")
```

- Render Functions Make These Outputs on the Server Side

render function	creates
<code>renderImage</code>	images
<code>renderPlot</code>	plots
<code>renderPrint</code>	any printed output
<code>renderTable</code>	data frame, matrix, other table like structures
<code>renderText</code>	character strings
<code>renderUI</code>	a Shiny tag object or HTML

## Gapminder Shiny App



- Instructions
  - Step 1: Download Tutorial
  - Step 2: Unzip Folder
  - Step 3: Open Both R Files in Gapminder\_Start
  - Step 4: Install Gapminder
    - > `install.packages("gapminder")`
  - Step 5: Run the App
- [Gapminder](#) Data
  - Used in Chapter 20 (R4DS)
  - Non-Profit Project Promoting a Fact-based World
  - Popularized by Hans Rosling
  - > `library(gapminder)`

# Gapminder Shiny App



- Data Content in Gapminder

```
> head(gapminder)
# A tibble: 6 x 6
  country      continent  year  lifeExp      pop  gdpPercap
  <fct>        <fct>    <int>  <dbl>    <int>    <dbl>
1 Afghanistan Asia      1952   28.8  8425333    779.
2 Afghanistan Asia      1957   30.3  9240934    821.
3 Afghanistan Asia      1962   32.0 10267083    853.
4 Afghanistan Asia      1967   34.0 11537966    836.
5 Afghanistan Asia      1972   36.1 13079460    740.
6 Afghanistan Asia      1977   38.4 14880372    786.
```

- Begin Using the App
  - Enter Name, Select Countries, Select Variable, and Submit
  - Observe the Use of CSS Code
  - Observe the tabsetPanel Style
  - Observe the Use of renderUI with uiOutput

# Gapminder Shiny App



- Part 1: Data Selected
  - Modification for server.R

```
#Part 1: Create a Table Previewing Data
output$OUTpreview<-renderTable({
  gapminder2 %>%
    select(Country,Continent,Year,input$INvariable)%>%
    filter(Country %in% input$INcountry) %>%
    arrange(Year)
})|
```

- Modification for ui.R

```
h2("Data Selected"),
br(),
#1: Print Data for Desired
# | Countries and Variable
tableOutput("OUTpreview"),
br(),
```

- Run the Shiny App
- Close the Shiny App



# Gapminder Shiny App



- Part 2: Data Summary
  - Modification for server.R

```
#Part 2: Create a Table Summarizing Data by Country
output$OUTsummary<-renderTable({
  gapminder2 %>%
    select(Country,Continent,Year,input$INvariable)%>%
    filter(Country %in% input$INcountry) %>%
    arrange(Year)%>%
    group_by(Country) %>%
    summarize(N=n(),
              MIN=min(get(input$INvariable)),
              Q1=quantile(get(input$INvariable),0.25),
              Q2=quantile(get(input$INvariable),0.5),
              Q3=quantile(get(input$INvariable),0.75),
              MAX=max(get(input$INvariable)),|
              CHANGE=MAX-MIN,
              MEAN=mean(get(input$INvariable)),
              SD=sd(get(input$INvariable))
    )
})
```

## Gapminder Shiny App



- Part 2: Data Summary (Continued)
  - Modification for ui.R

```
h2("Country Comparison"),  
br(),  
#2: Print Summary  
tableOutput("OUTsummary|"),  
br()
```
  - Run the Shiny App
  - Notice the get() Function
    - Input is a Character String
    - Need to Drop the Quotes
    - Evaluates Functions on The Variable Object
  - Close the Shiny App

## Gapminder Shiny App



- Part 3: Trend Plots
  - Modification for server.R

```
#Part 3: Create a Graphic Showing Trends
output$OUTtrendvar<-renderText({
  expr=paste("Trend Comparison for",input$INvariable)
})

output$OUTtrendplot<-renderPlot({
  gapminder2 %>%
    select(Country,Continent,Year,input$INvariable)%>%
    filter(Country %in% input$INcountry) %>%
    arrange(Year)%>%
    ggplot(aes(x=Year,y=get(input$INvariable))) +
    geom_line(aes(color=Country),size=input$width)+
    ylab(input$INvariable) +
    theme_minimal()
})
```

- Notice: We are Going to Control the Width of the Lines

## Gapminder Shiny App



- Part 3: Trend Plots (Continued)
  - Modification for ui.R
  - Creation of Slider Input

```
uiOutput(outputId="OUTvariable"),  
  
#Part 3:Line width for Trend Graphic  
sliderInput(inputId="width",  
            label="width of Trend Lines",  
            min=1,max=3,value=1,step=1),  
  
#Submit Button For Updates
```

- Displaying Graphic Output

```
tabPanel("Graphics",  
  
        #3:Print Trend Graphic  
        h2(textOutput("OUTtrendvar")),  
        br(),  
        plotOutput("OUTtrendplot"),  
        br())
```

- Run the Shiny App

## R Shiny Tutorials



- Official 3 Part [Video Tutorial](#)
- Official Shiny [Cheat Sheet](#)
- Video Tutorials by [Abhinav Agrawal](#)
- [Video](#) Combining Shiny with Rmd
- [Video Tutorial](#) on Shiny Dashboard
- [Video Tutorials](#) by Johns Hopkins Data Science Lab Produced by [Brian Caffo](#)

Closing



Disperse  
and Make  
Reasonable  
Decisions