

MZES SSDL - Shiny Apps: Development and Deployment

Konstantin Gavras

MZES, University of Mannheim

15 October 2019

Shiny

Shiny from RStudio



- ▶ Shiny is a package by RStudio to build interactive web pages. . .
 - ▶ without having any knowledge of web development (HTML/CSS/JavaScript)
- ▶ Shiny Apps interact with R
 - ▶ Allows for calculations, display of R objects, presentation of results . . .

Shiny App Components

1. Front end

- ▶ the web page actually shown to the user
- ▶ the HTML page written by Shiny
- ▶ includes layout, appearance, design features
- ▶ in Shiny terminology: `ui` (user interface)

2. Back end

- ▶ code running the app, including all functions, data import, etc.
- ▶ involves the logic of the app
- ▶ responsible for creating objects on the front end
- ▶ in Shiny terminology: `server`

Setting up a Shiny App

Shiny Apps can be set up in two different ways:

1. Single file App

- ▶ `ui` and `server` are stored in one script
- ▶ used when developing very simple Shiny Apps
- ▶ name of the file has to be `app.R!!!`

2. Two file App

- ▶ `ui` and `server` are stored in separate scripts
- ▶ clear separation between front end and back end
- ▶ highly preferable when developing more advanced Shiny Apps
- ▶ names of the files have to be `ui.R` and `server.R!!!`

→ We are going to develop Shiny Apps using the Two File method

Developing Shiny Apps - Step by Step

Let's get started!

Workshop materials:

<https://github.com/KostaGav/shiny-development-deployment>

Features covered in the workshop:

Development:

1. Building a Shiny App from scratch
2. Building the plain UI
3. Getting output objects and control widgets into the UI
4. Implementing the server logic
5. Output/Input Reaction
6. Rendering objects
7. Reactivity

Deployment:

1. Deploy your app using `shinyapps.io`
2. Deploy on your own VM using Shiny Server

Building a Shiny App from scratch

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

```
library(shiny)
```

```
runExample("01_hello")
```

#To show alternative Apps, please type runExample(NA)

#and choose another example

Building a Shiny App from scratch

- ▶ Create a new folder with two R scripts:

ui.R:

```
library(shiny)
ui <- fluidPage()
```

server.R:

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

- ▶ Launch the Shiny App by pressing the 'Run App' button in the top right corner

Building the plain UI

- ▶ When building a Shiny App, one should have, in general, in mind how the app should 'look' like
- ▶ Thus, we build the UI first
- ▶ In simple Shiny App, the whole UI fits in the `fluidpage`
 - ▶ every new object is passed comma-separated
 - ▶ text can be passed to the UI by entering strings
- ▶ In order to format text, Shiny uses HTML wrappers:
 - ▶ these wrappers are functions taking one object as argument (+ further style options)
 - ▶ `h1()`: Top-level header
 - ▶ `h2()`: secondary header
 - ▶ `strong()`: make text bold
 - ▶ `em()`: make text italicized
 - ▶ `br()`: add line break
- ▶ We can add an official header using `titlePanel()`

Q: Can you see any particular differences between using `h1()` and the `titlePanel()` when using them as title?

Building the plain UI

- ▶ Until now, we only have a plain white page
- ▶ We need a proper layout to make it appear nicer:
 - ▶ sidebarlayout is the simplest layout format
 - ▶ Input and control widgets on the left side, results and plots on the right hand side

```
ui <- fluidPage(  
  titlePanel("Title of my Shiny App"),  
  sidebarLayout(  
    sidebarPanel("My input goes here"),  
    mainPanel("The results go here")  
  )  
)
```

Adding Input and Control Widgets

- ▶ In order to interact with Shiny Apps, we need control widgets
- ▶ User can specify inputs, enter text or select specific dates to create a certain results
- ▶ all input function have two arguments: `inputId` and `label`
 - ▶ `inputId`: name Shiny uses to refer to this input, when retrieving values for the back end
 - ▶ has to be unique! (WARNING: if you provide two ids with the same name, there won't be an error message!)
 - ▶ `label`: Text displaying the label of the control widget

Adding Input and Control Widgets

Button

Action

`actionButton()`

Single checkbox

☒ Choice A

`checkboxInput()`

Checkbox group

☒ Choice 1
☐ Choice 2
☐ Choice 3

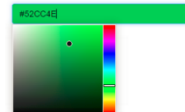
`checkboxGroupInput()`

Date input

2014-01-01

`dateInput()`

Colour input



`colourpicker::colourInput()`

Date range

2014-01-24 to 2014-01-24

`dateRangeInput()`

File input

Choose File No file chosen

`fileInput()`

Numeric input

1

`numericInput()`

Password Input

.....

`passwordInput()`

Radio buttons

☒ Choice 1
☐ Choice 2
☐ Choice 3

`radioButtons()`

Select box

Choice 1

`selectInput()`

Sliders



`sliderInput()`

Text input

Enter text...

`textInput()`

Text area

Multiple lines
of text

`textAreaInput()`