

Web Apps using Package 'shiny'

Contents

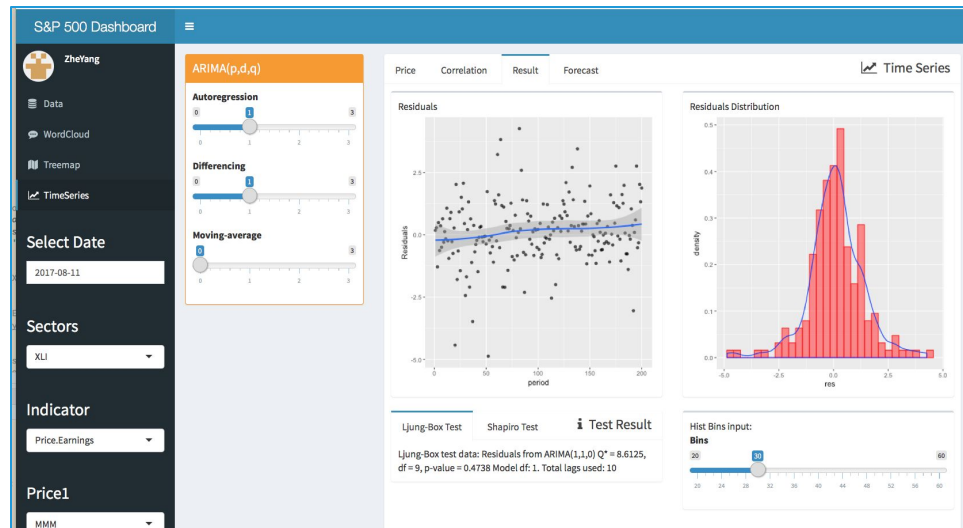
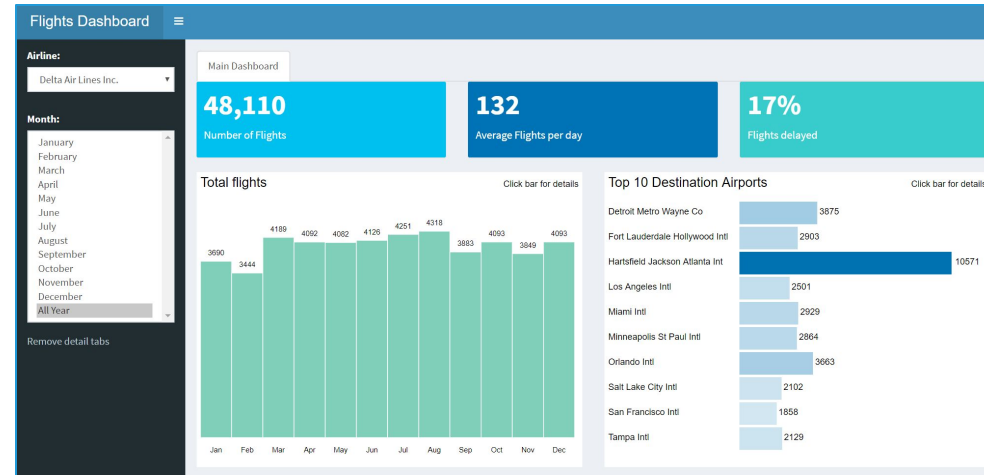
1. Introduction to Shiny
2. Understanding the Structure of Shiny App!
3. Layout of the App
4. How UI and SERVER Interact?

Introduction to Shiny

- Shiny is an R application from RStudio that makes it incredibly easy to build interactive web applications (apps) in R.
- It provides elegant and powerful web framework for building interactive web applications using R.
- Shiny apps are easy to write. No web development skills are required.
- It helps you turn your analyses into interactive web applications without requiring HTML, CSS, or JavaScript knowledge, but if you know them you can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.
- **Shiny is an extremely powerful tool.** Unlike the more traditional workflow of creating presentations, you can now create an app that allows your readers to change the assumptions underlying your analysis and see the results immediately. It **reduces the effective time by more than 80%** and makes documentation fun and interactive!

Introduction to Shiny

Here's how a Shiny dashboard looks.



Let us now see the basic steps for making your own app

New to Shiny!

First you'll have to install the shiny package from CRAN on RStudio.

Install and load the package “shiny”

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

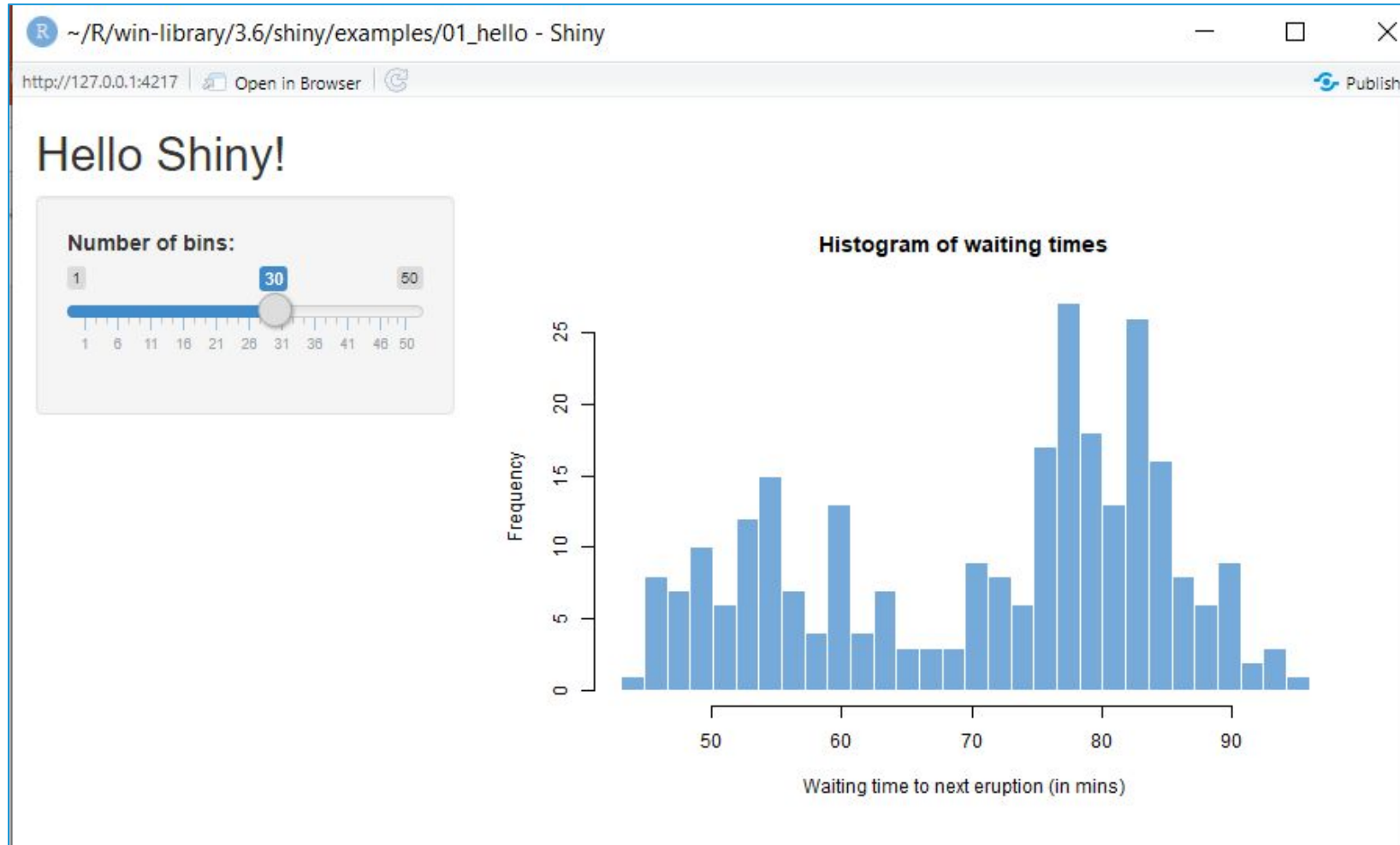
The **shiny** package has some inbuilt examples that each demonstrates how shiny works. Run the **Hello Shiny** example with the code given below :

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

This example displays a histogram of R's faithful dataset with a configurable number of bins. Number of bins can be changed with the slider bar, and the app will immediately respond to the input given by user.

New to Shiny!

Hello Shiny Example:



Understand the Structure of Shiny App!

A Shiny app consists of 2 main components :

- User Interface script (ui.R)
- Server script (server.R)


- The **user - interface script** controls the layout and the appearance of your app. It is defined and stored with the name **ui.R**.
- The **server script** contains the instructions that your computer needs, to build the app. It is defined and stored with the name **server.R**
- The user interface and the server interact with each other through input and output objects.
- The instructions for creating input objects are in the UI.

Understand the Structure of Shiny App!

- Create a new directory named **Myshinyapp** in your working directory. Then copy - paste your **ui.R** and **server.R** scripts within **Myshinyapp**.
- You can check your working directory by **getwd()** function and set it where you'd like by **setwd()** or manually on R studio by clicking on Session.

Alternate way to create a Shiny app is by defining UI and SERVER in a single file named app.R. This file must return an object created by the shinyApp() function like this :

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



This method is more appropriate for smaller applications.



Note: Both the ui and server files work in conjunction and should be in the same working directory along with any input datasets (if any) used in your app.

Layout of the app

The user interface consists of 2 basic Panels:

- Title Panel
- Sidebar Layout
 - Sidebar panel
 - Main panel

The basic structure of the user-interface and the server script is:

ui.R

```
library(shiny)
fluidPage(
  titlePanel("This is the title panel"),
  sidebarLayout(
    sidebarPanel("This is the sidebar panel"),
    mainPanel("This is the main panel")
  )
)
```

server.R

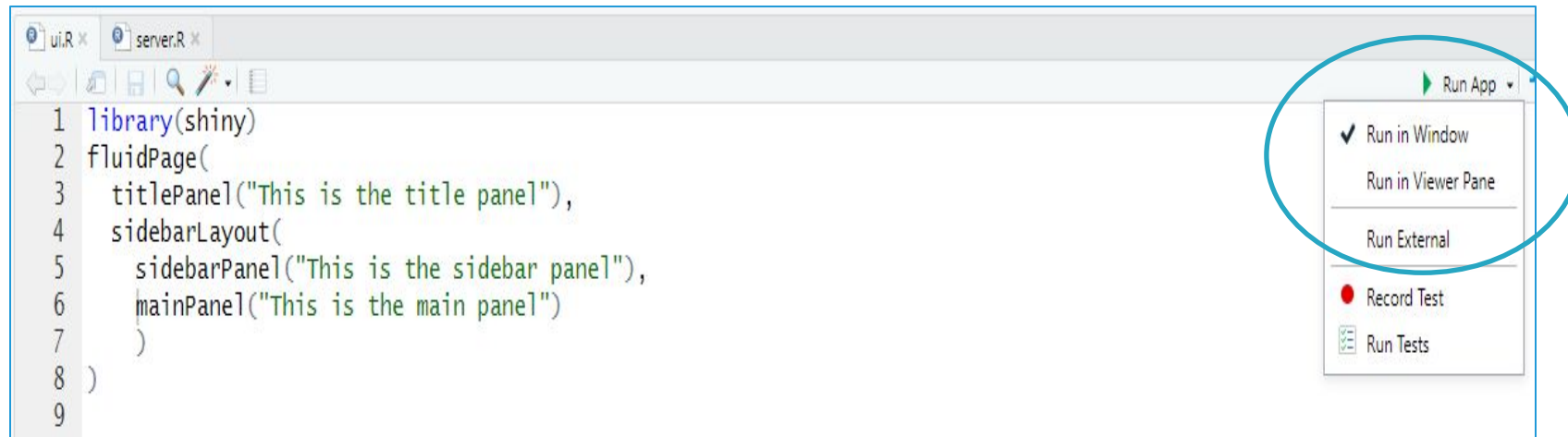
```
function(input,output){
}
```

- ❑ The **ui.R** script uses the **fluidPage()** function to make the app page responsive.
- ❑ **fluidPage()** can enable the app to work on any platforms without affecting the layout of the app. It automatically adjusts to the dimensions of the browser window.

We define server by creating a function and arguments input and output are supplied to refer to inputs and outputs defined in **ui.R**

Layout of the app

Notice the green **Run App** button appear when the file is saved. This button also allows you to control whether the app runs in a browser window, in the RStudio Viewer pane, or in an external RStudio window.



Click on 'Run App' to run the app.

Layout of the app



Most of the inputs of the shiny app are presented in the sidebar panel with the output in the main panel. However this is subject to change according to the requirement of the app.

More Features that can be added to Layout

– `fluidRow()` & `column()`

- **Sidebar Layout:**

Our **Myshinyapp** app has a layout which provides a sidebar for inputs and a large main area for output.

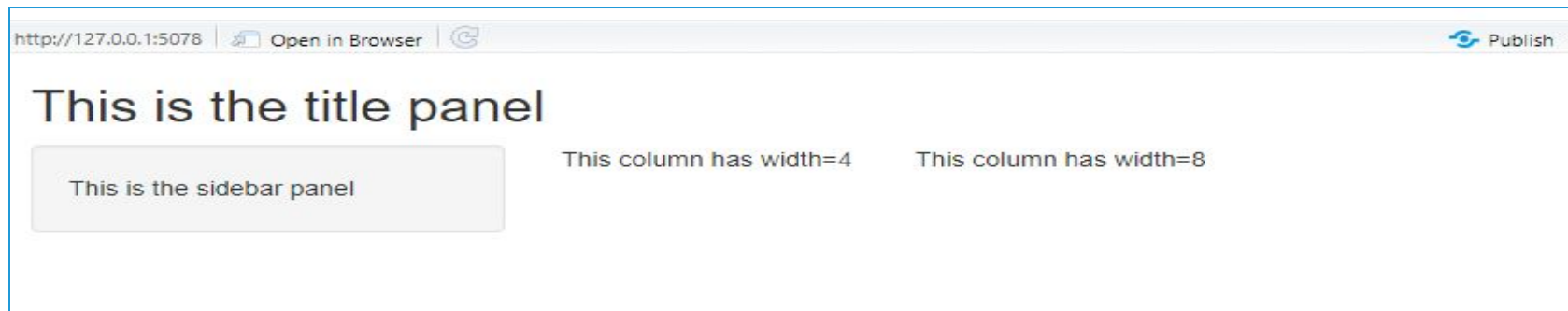
- **Fluid Grid System:**

To create a layout based on the fluid system you use the `fluidPage()` function which we have already seen in our **Myshinyapp** example. This grid system uses 12 column grid layout which flexibly be subdivided into rows and columns. To create rows use `fluidRow()` function and include columns defined by the `column()` function. Grid layouts can be used anywhere within a `fluidPage()` and can even be nested within each other.

More Features that can be added to Layout

– fluidRow() & column()

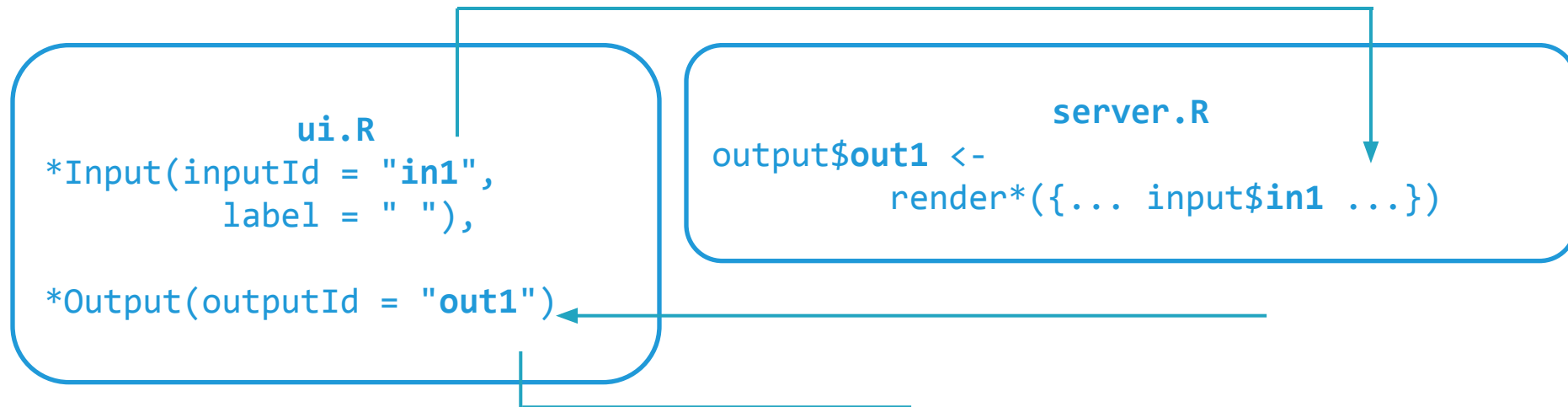
```
fluidPage(  
  titlePanel("This is the title panel"),  
  sidebarLayout(  
    sidebarPanel("This is the sidebar panel"),  
    mainPanel(fluidRow(  
      column(4,"This column has width=4"),  
      column(8,"This column has width=8")  
    )  
  ))  
)
```



You can move the columns to the right by specifying the number of columns to offset to **offset=** argument. Each unit of offset increases the left-margin of a column by a whole column.

How UI and SERVER Interact?

The user interface and the server interact with each other through input and output objects.



- **Input objects** are a way for users to interact with Shiny App. Inputs in UI are given by adding widgets with functions like **`selectInput()`** or **`radioButtons()`** and are used within **`render*()`** functions in the server object to create output objects.
- **Output objects** are placed in the UI using output functions like **`plotOutput()`** or **`textOutput()`**.