Shiny

What is Shiny?

- Shiny is a R package that makes it easy to build interactive web applications (apps) straight from R.
- Usually such presentations are shown as a Research and Analysis finding.
- Presenting the findings with an interactive view has much greater influence than any other type presentation.

Sane's STATS Academy of Statistics

Structure of Shiny App

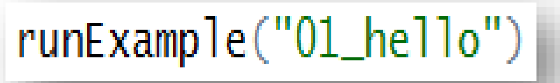
- Shiny apps have two components:
  - a user-interface script
  - a server script
- The user-interface (ui) script controls the layout and appearance of your app. It is defined in a source script named ui.R.
- The server.R script contains the instructions that your computer needs to build your app.
- As of version 0.10.2, Shiny supports single-file applications. You no longer need to build separate server.R and ui.R files for your app; you can just create a file called app.R (or any name to the file) that contains both the server and UI components.

Running an App of Shiny

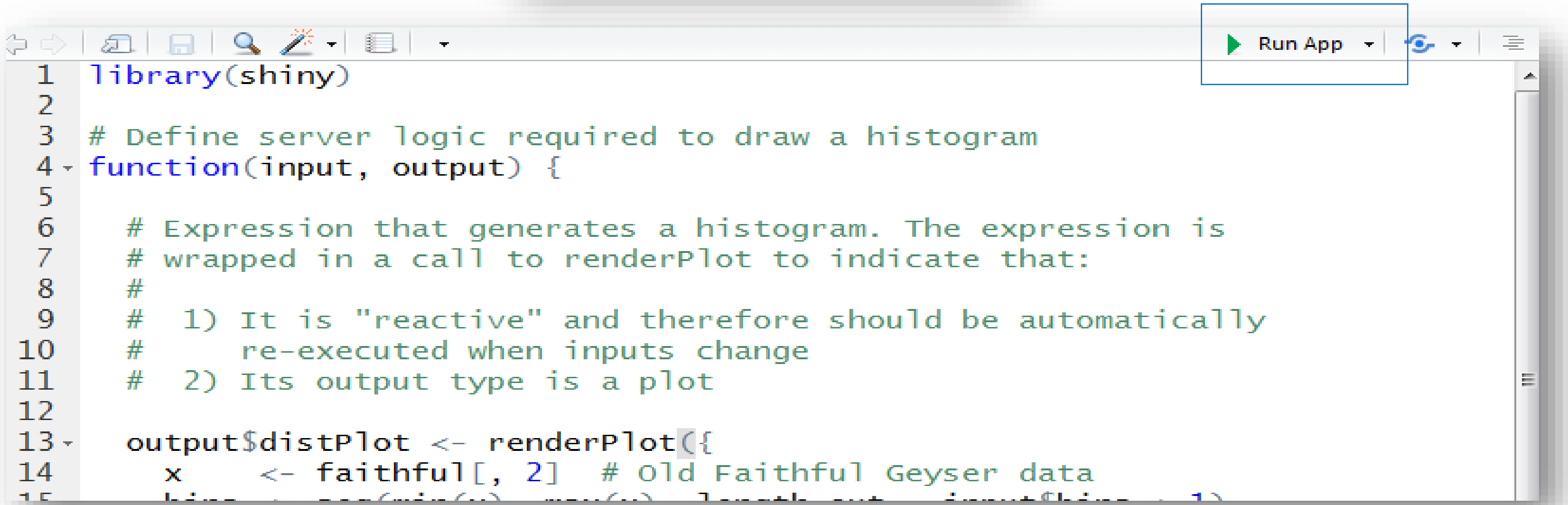
There are two ways by which a Shiny App can be run:

1. You can run a Shiny app by giving the name of its directory to the function runApp like

2. Click on the “RunApp” button provided on the smart editor window



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



```
library(shiny)

# Define server logic required to draw a histogram
function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should be automatically
  #     re-executed when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
```

Layout

- Shiny ui.R scripts use the functions like fluidPage and pageWithSidebar to create a display that automatically adjusts to the dimensions of your user’s browser window.
- You lay out your app by placing elements in these functions

fluidPage Elements

- titlePanel and sidebarLayout are the two most popular elements to add to fluidPage. They create a basic Shiny app with a sidebar.
- sidebarLayout always takes two arguments:
    - sidebarPanel function output
    - mainPanel function output

titlePanel & sidebarLayout functions

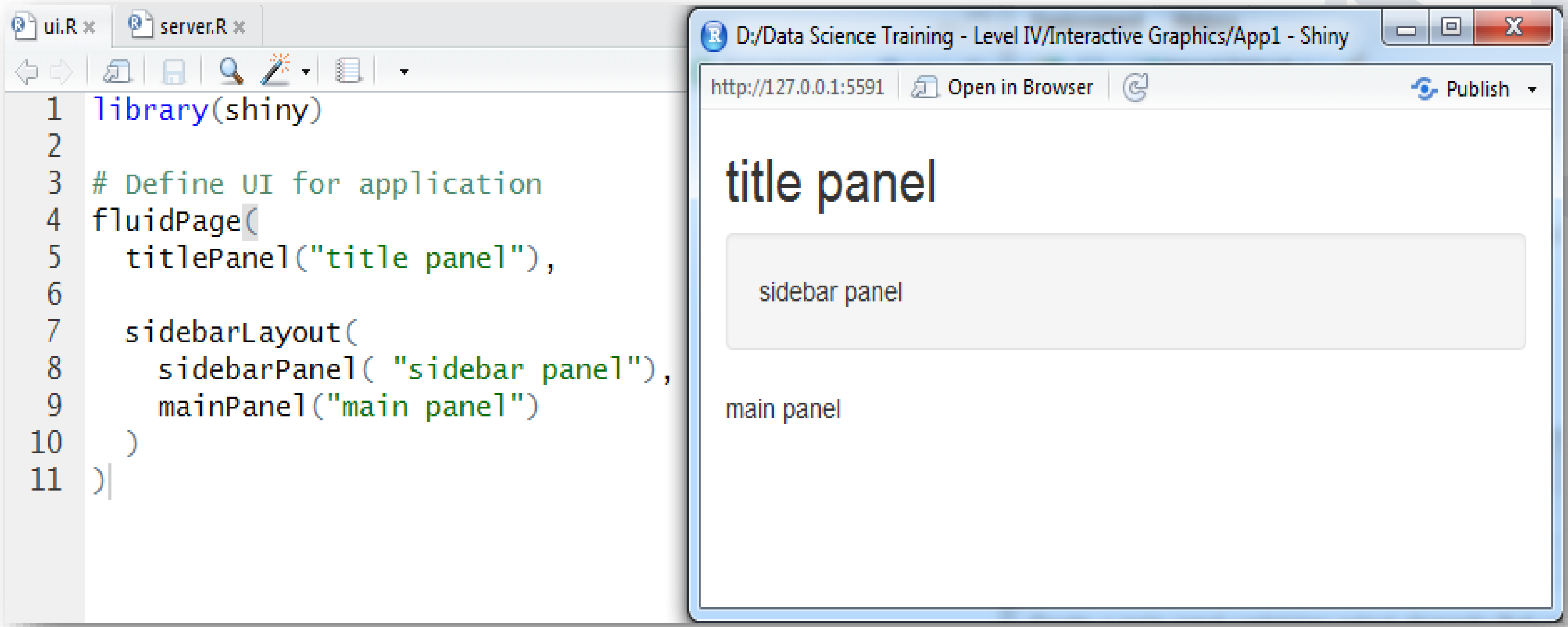
- titlePanel(title, windowTitle = title)
  - title : title to be displayed
  - windowTitle : The title that should be displayed by the browser window
- sidebarLayout(sidebarPanel, mainPanel, position = c("left", "right"), fluid = TRUE)
  - sidebarPanel : sidebarPanel function call containing input controls
  - mainPanel : mainPanel function call containing outputs
  - position : The position of the sidebar relative to the main area ("left" or "right")
  - fluid : TRUE to use fluid layout; FALSE to use fixed layout

sidebarPanel & mainPanel functions

- sidebarPanel(..., width = 4)
  - UI elements to include on the sidebar
  - The width of the sidebar. For fluid layouts this is out of 12 total units; for fixed layouts it is out of whatever the width of the sidebar's parent column is.
- mainPanel(..., width = 8)
  - Output elements to include in the main panel
  - The width of the main panel. For fluid layouts this is out of 12 total units; for fixed layouts it is out of whatever the width of the main panel's parent column is.

Sane's STATS Academy of Statistics

Example



```r
library(shiny)

# Define UI for application
fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel( "sidebar panel"),
    mainPanel("main panel")
  )
)
```

D:/Data Science Training - Level IV/Interactive Graphics/App1 - Shiny
http://127.0.0.1:5591
Open in Browser
Publish

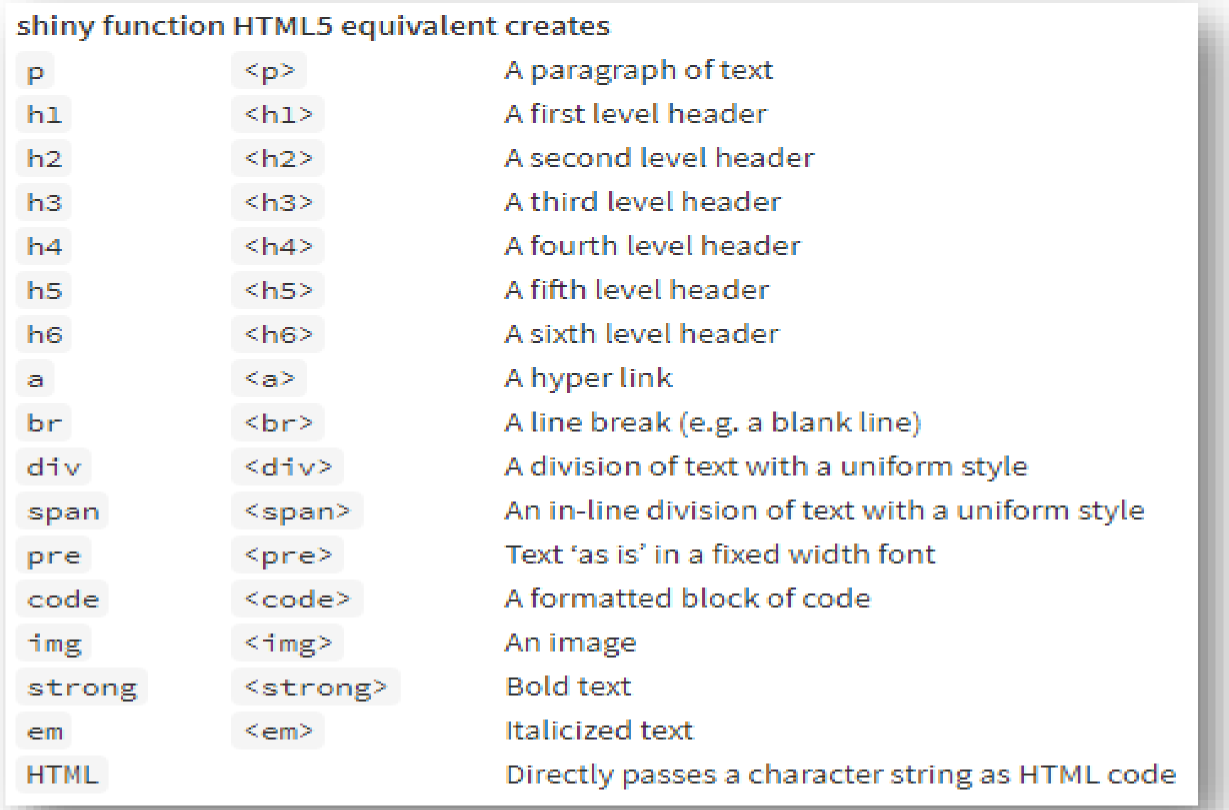
title panel

sidebar panel

main panel

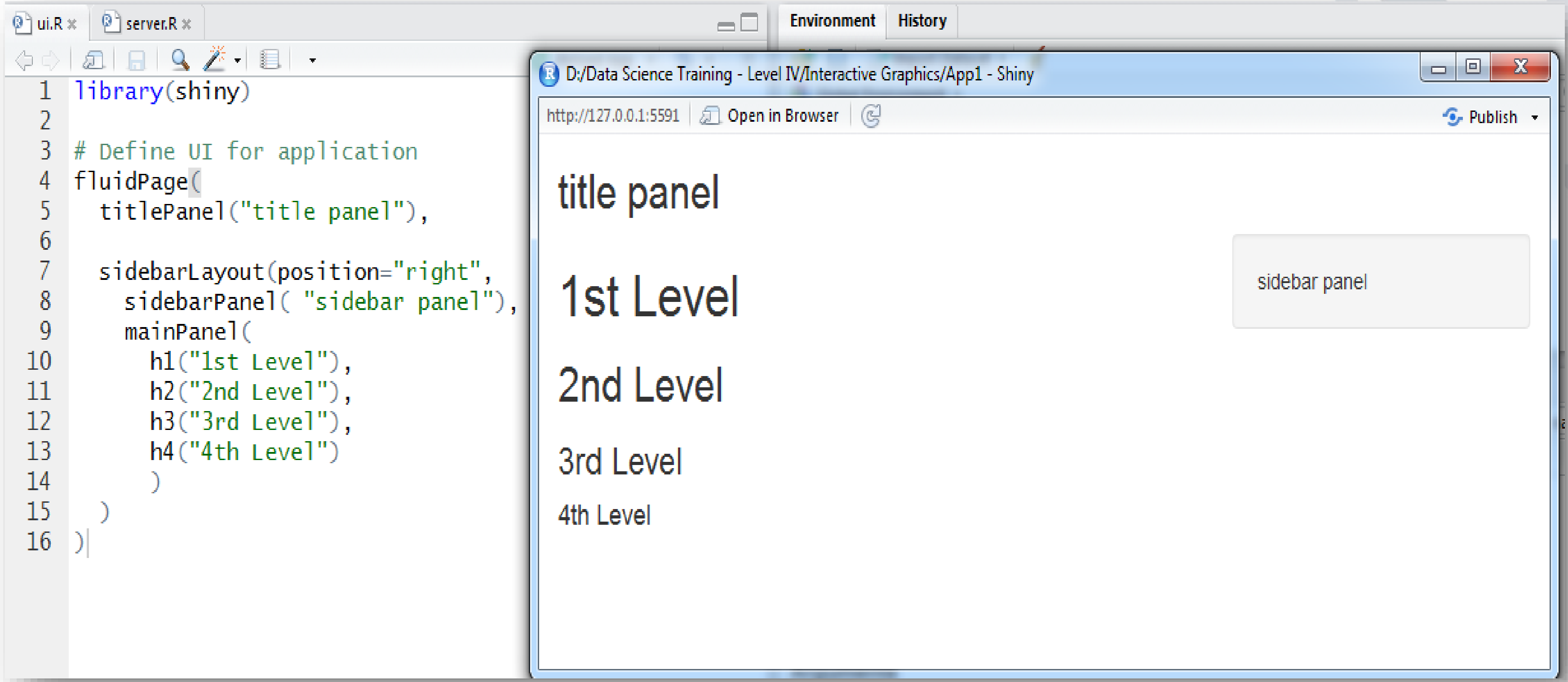
Enhancing text with HTML

- You can add content to your Shiny app by placing it inside any of the panel functions



shiny function	HTML5 equivalent	creates
p	`<p>`	A paragraph of text
h1	`<h1>`	A first level header
h2	`<h2>`	A second level header
h3	`<h3>`	A third level header
h4	`<h4>`	A fourth level header
h5	`<h5>`	A fifth level header
h6	`<h6>`	A sixth level header
a	`<a>`	A hyper link
br	`<br>`	A line break (e.g. a blank line)
div	`<div>`	A division of text with a uniform style
span	`<span>`	An in-line division of text with a uniform style
pre	`<pre>`	Text 'as is' in a fixed width font
code	`<code>`	A formatted block of code
img	`<img>`	An image
strong	`<strong>`	Bold text
em	`<em>`	Italicized text
HTML		Directly passes a character string as HTML code

Example



```r
library(shiny)

# Define UI for application
fluidPage(
  titlePanel("title panel"),

  sidebarLayout(position="right",
    sidebarPanel( "sidebar panel"),
    mainPanel(
      h1("1st Level"),
      h2("2nd Level"),
      h3("3rd Level"),
      h4("4th Level")
      )
  )
)
```

D:/Data Science Training - Level IV/Interactive Graphics/App1 - Shiny

http://127.0.0.1:5591

title panel

1st Level

2nd Level

3rd Level

4th Level

sidebar panel

Control Widgets

- Shiny comes with a family of pre-built widgets, each created with a transparently named R function

function	**widget**
actionButton	Action Button
checkboxGroupInput	A group of check boxes
checkboxInput	A single check box
dateInput	A calendar to aid date selection
dateRangeInput	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

Widget Function Usage

- You can add widgets to your web page in the same way as you add other types of HTML content
- To add a widget to your app, place a widget function in sidebarPanel or mainPanel in your ui.R file or UI related function call
- First two arguments of any widget functions are
    - inputId: The user will not see this name, but can use it to access the widget's value. The name should be a character string
    - label: This label will appear with the widget in your app. It should be a character string, but it can be an empty string ""
- The remaining arguments may vary from widget to widget

About server.R

- This file should contain a anonymous function with two arguments input and output
- The argument input is a list-like object. It stores the current values of all of the widgets in your app. These values will be saved under the names that you gave the widgets in ui.R.
- The argument output is also a list-like object that stores instructions for building the R objects in your app.

Render Functions

renderImage	images (saved as a link to a source file)
renderPlot	plots
renderPrint	any printed output
renderTable	data frame, matrix, other table like structures
renderText	character strings
renderUI	a Shiny tag object or HTML

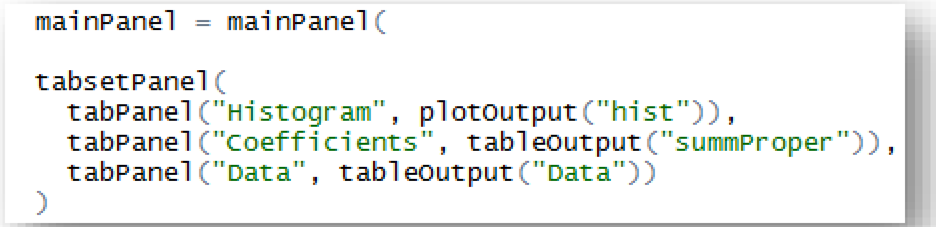
- Each render* function takes a single argument: an R expression surrounded by braces, {}.
- The expression can be one simple line of text, or it can involve many lines of code, as if it were a complicated function call.
- Shiny will run the instructions when you first launch your app, and then Shiny will re-run the instructions every time it needs to update your object.
- For this to work, your expression should return the object you have in mind (a piece of text, a plot, a data frame, etc). You will get an error if the expression does not return an object, or if it returns the wrong type of object.

Sane's STATS Academy of Statistics

Tabsets

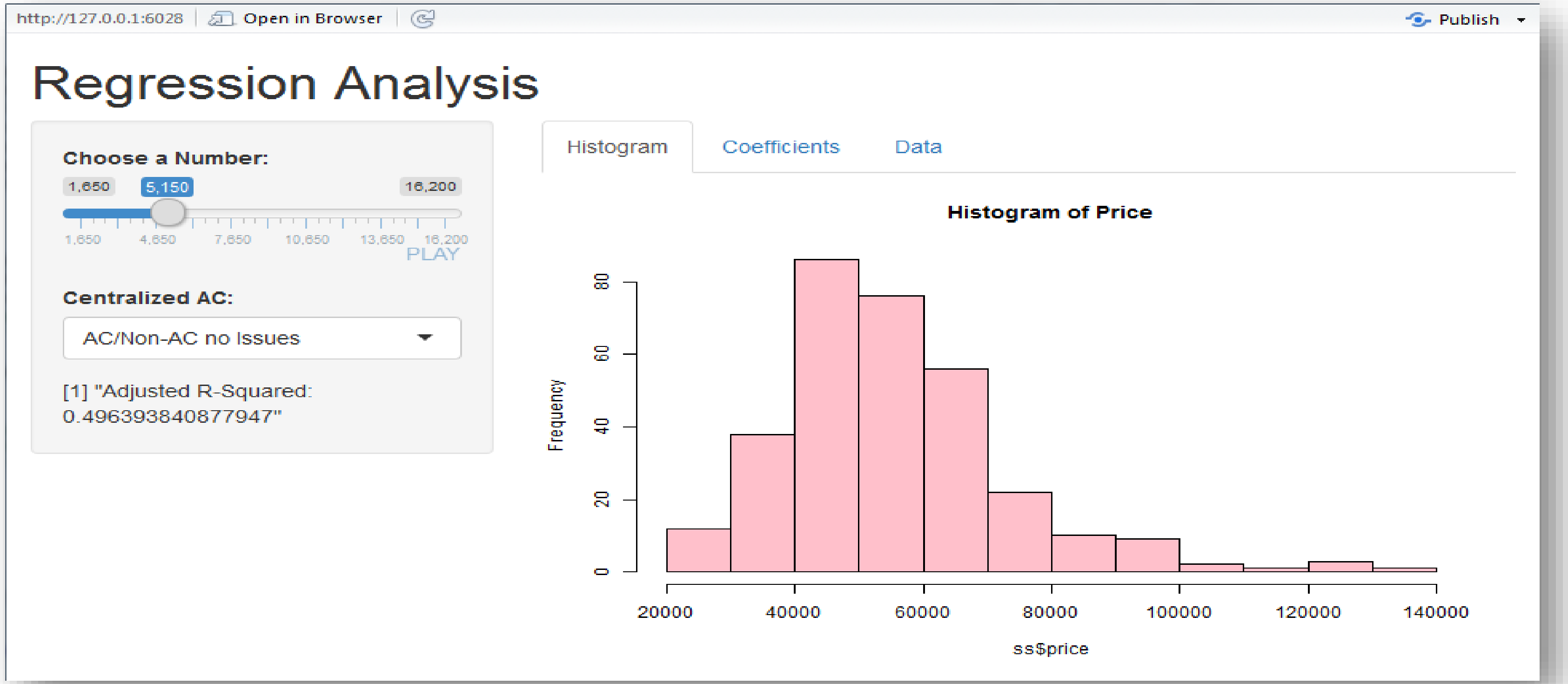
- In order to display multiple outputs in different tabs simultaneously, tabsets widget can be used
- Tabsets are created by calling the `tabsetPanel` function with a list of tabs created by the `tabPanel` function.
- Each tab panel is provided a list of output elements which are rendered vertically within the tab.

Tabsets Example



```
mainPanel = mainPanel(

tabsetPanel(
  tabPanel("Histogram", plotOutput("hist")),
  tabPanel("Coefficients", tableOutput("summProper")),
  tabPanel("Data", tableOutput("Data"))
)
```



```
[1] "Adjusted R-Squared: 0.496393840877947"
```