Building web applications with `shiny` and RStudio Connect

Shiv Sundar

June 18, 2018

Introduction

In this demo, we will cover how to use `shiny` and the differences between RStudio Connect and the shinyapps.io (https://www.shinyapps.io) service.

What is `shinyapps.io`?

`shinyapps.io` is a service that aims to get your interactive web application published as soon as possible. It only offers support for applications and has limited authentication features.

What is RStudio Connect?

RStudio Connect is a service provided by RStudio. It is focused on publishing multiple file types, including Shiny applications and R Markdown reports enhanced with Shiny.

What are the differences between the two?

`shinyapps.io`	**RStudio Connect**
Support for Shiny apps only	Support for Shiny apps, RMarkdowns, and more
Focused on simple development	Provides a robust framework for publishing
Used by smaller applications	Used by academic institutions and companies
Free tier exists	Basic service costs $15,000

`shiny`

`shiny` is a package used to build web applications in R.

Accessing data in `shiny`

`shiny` stores all objects in a vector called `input`. For example, if one creates an input object with the object id `button2`, then that object's value can be accessed by calling `input$button2`.

Types of inputs available in Shiny

In the `shiny` package, there are 14 total UI input components that one can use. These are:

- `actionButton()`
- `actionLink()`
- `checkboxGroupInput()`
- `checkboxInput()`
- `dateInput()`
- `dateRangeInput()`
- `fileInput()`
- `numericInput()`
- `passwordInput()`
- `radioInput()`
- `selectInput()`
- `sliderInput()`
- `submitButton()`
- `textInput()`

`shiny` also has some output components. These are:

- `dataTableOutput()`
- `plotOutput()`
- `verbatimTextOutput()`
- `tableOutput()`
- `textOutput()`
- `uiOutput()`
- `htmlOutput()`

For example, here we can see the implementation of a application that displays the selected date. Selecting a date updates the text to display what was selected.

```
dateInput("calendar", label = "Choose date to display")
renderText({paste("Selected Date: ", as.character(input$calendar), sep = "")})
```

Choose date to display

2018-07-19

Selected Date: 2018-07-19

Building a Shiny app

We will build an app that can take a .txt file, be provided the word delimiters, and then display the average length of the words. We will also add a graph that shows the frequency of word lengths. First, we need to create a new Shiny app and we can do this by clicking on File -> New File -> Shiny Web App.

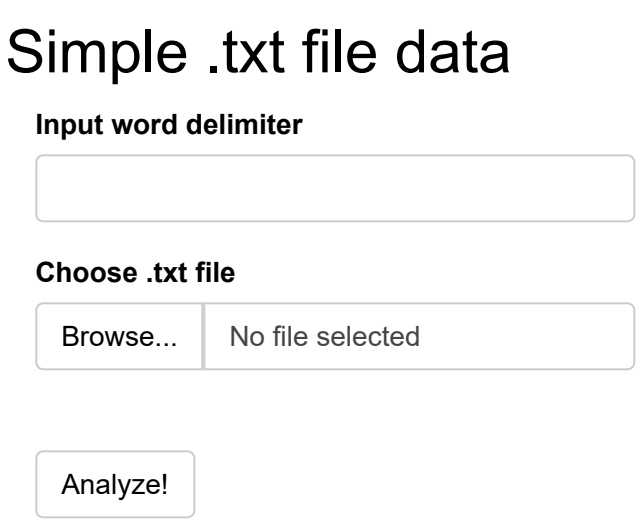
Let’s get all the code and then explain it.

```
1  ui <- fluidPage(
2   titlePanel("Simple .txt file data"),
3   mainPanel(
4     textInput("delimiter", placeholder = "Input word delimiter"),
5     fileInput("fileInput", "Choose .txt file to analyze", accept = c("text/plain")),
6     actionButton("fileChosen", "Analyze!"),
7     textOutput("avgLen"),
8     plotOutput("wordLCount")
9   )
10 )
11
12 server <- function(input, output) {
13   data <- reactive({
14     textFile <- scan(as.character(input$fileInput$datapath),
15                      what = "",
16                      sep = input$delimiter)
17
18     return(textFile)
19   })
20
21   observeEvent(input$fileChosen, {
22     output$avgLen <- renderText({
23       paste("Average length of words in file:", sum(nchar(data(), "chars"))/length(data()))})
24
25     output$wordLCount <- renderPlot({
26       vals <- as.data.frame(table(nchar(data(), "chars")))
27       ggplot(vals, aes(Var1, Freq)) +
28         labs(y = "Number of Occurrences", x = "Length of Word") +
29         geom_bar(stat = "identity")
30     })
31   })
32 }
33
34 shinyApp(ui = ui, server = server)
```

- Line `1` creates the UI, which is what the user will see.
- Line `2` is the title of the app.
- Line `3` creates the main panel of the app.
- Line `4` creates a `textInput` object with the `outputId` being `"delimiter"`. The help text in this object asks the user to enter the delimiter for the file being analyzed.
- Line `5` creates a `fileInput` object with an `outputId` of `"fileInput"`. The title of this `fileInput` is `"Choose .txt file"`. Lastly, this `fileInput` will accept only `.txt` files. It is worth noting that the restriction of file types will only work once the app is published.
- Line `6` creates an `actionButton`. The `outputId` is `"fileChosen"` and the text on this button is `"Analyze!"`. This is the trigger that will be used to build the analysis.
- Line `7` creates a placeholder for text pulled from the `output$avgLen` object.
- Line `8` creates a placeholder for a plot taken from the `output$wordLCount` object.
- Line `12` starts the server function for any computation that the user doesn't need to see. This is where the reactivity of the Shiny app happens.
- Line `13` creates a reactive object called `data`. In the rest of the app, this object can be referred to by calling `data()`.

- Line `14` creates an object called `textFile` and uses the `scan` method. `input$fileInput$datapath` refers to the `fileInput` object created in the UI. Once the user picks a file and clicks on the `fileChosen actionButton`, this object will be created.
- Line `16` uses the 1 byte size character (default is whitespace) provided by the `delimiter textInput` object as the delimiter for the `scan` method.
- Line `18` returns the `textFile` object and sets it to be `data()`. * Line `21` runs the `observeEvent` method and uses the `fileChosen actionButton` as the trigger. `observeEvent` is a method that doesn't run its code until it is triggered.
- Line `22` creates a `renderText` object named `avgLen`. This is the object that line `7` pulls from.
- Line `23` sets the text for `avgLen`.
- Line `25` creates a `renderPlot` object named `wordLCount` referred to by the `plotOutput` in line `8`.
- Line `26` creates a data frame named `vals`. The data stored in this data frame is the number of occurrences of a specific length of word.
- Lines `27` to `29` plot the data frame using the `ggplot2` library.
- Line `34` runs the app.

Below, you can see the implementation of this app.



Publishing the app on `shinyapps.io`

To publish our newly created app, we first need to create an account. You can do this by visiting this link (https://www.shinyapps.io/admin/#/signup) to sign up.

Now, we need to add our `shinyapps.io` account to our R session. To do this, follow the steps below.

Make sure `rsconnect` is installed and the library is added to your session. These commands should be run in the console.

```
install.packages('rsconnect')
library(rsconnect)
```

Now we connect our account to our current R session. Place the relevant information in each field marked by “”. Alternatively, you can copy the text from your `shinyapps.io` account dashboard. This command should be run in the console.

```
rsconnect::setAccountInfo(name="username",
                          token="public token",
                          secret="secret token")
```

After doing this, run the following command to publish the currently open app to `shinyapps.io` .

```
deployApp()
```