Decision Modeling for Public Health

R Shiny interface

DARTH

11/09/2020

This worksheet highlights some key settings where `R` can help make your research easier to reproduce and present to key stakeholder through `R`'s Shiny interface and web applications.

`R` Shiny Applications

`R` Shiny is a powerful and elegant package that that allows you to build web-based applications using `R`. It lets you turn your analyses into interactive web applications without requiring JavaScript, CSS or HTML coding.

To create a new Shiny App, click

File -> New File -> Shiny Web App...

and choose a name for your App and the working directory it will be saved in.

EXERCISE 1 Create a new Shiny App file and name it "demo".

After you create this app, you will see `R` has opened up a file named `app.R` –> this is the script for your App. The name you picked is the name of the app folder `R` has now created.

By default, `R` provides an example of a Shiny App which you can use by clicking the **Run App** button on the top right corner of the Script Editor. This App plots a histogram of the distribution of waiting time to eruption of the geyser from a the Old Faithful Geyser dataset, which is provided by standard in `R`.

This App interactively generates and displays a histogram with the binwidth the user selects. Changing the bindwidth parameter allows the user to change the width of the bins of the histogram. The user of the App does not have to write or run any `R` code as everything is done through clicks. This is what makes Shiny Apps cool and powerful!

To build and run Shiny apps, you would need to load the `R` package `shiny` first.

The following section briefly introduces how to build a shiny app for your analysis. To begin, call the `shiny` library.

```r
library(shiny)
```

All shiny apps have two components: **UI** and **Server**.

- **UI** stands for "User Interface", this is where you code what the user can see and click on.
- **Server** contains functions that produce the output that is displayed in or used by the UI.

It is easier to start with the Server since it feeds things into the UI.

In the demo App, the **Server**:

- Contains a function that first stores the 2nd column (eruption wait time) of the Old Faithful Geyser dataset into a variable called `x`.
- Then, it takes different number of bins defined in the UI (called "bins") to create the `bins` variable.
- Finally, it plots the histogram using the appropriate arguments (`x`, `bins`, etc).
- The entire operation is wrapped by Shiny function `renderPlot()` (a function that takes plots produced in `R`) and saved into a variable called `output$distPlot`.
- `output` tells `R` that this is an output to be sent to the UI and `distPlot` is the name given to this operation.

Now we switch to the **UI**,

- This UI has two parts: the side panel and the main panel.
- The side panel contains a side bar with slider input that contains different number of bins (input name is "bins") ranging from 1 to 50, with the default being 30.
- The main panel contains a function called `plotOutput()` that takes our histogram and plots it on the main panel of the App.

To finalize the App, the command `shinyApp(ui = ui, server = server)` is inserted at the very end.

EXERCISE 2 Follow the steps below to build a Shiny App that makes stratified boxplots for the Framingham data:

- Open up `ShinyApp.R`.
- `sliderInput()` creates a slider with minimum and maximum default values.
- Create one such slider input for the parameter `c_S` (cost of being in the sick state) with minimum value = 1000, maximum value = 10000 and initial value = 5000.
- Run the App and test it.

4. Further Reading

There are loads of great resources available to learn more about R:

1. Christopher Gandrud, Reproducible Research with R and RStudio
2. Yihui Xie, Dynamic Documents with R and knitr
3. The tidyverse website, https://www.tidyverse.org/
4. The R Markdown website, https://rmarkdown.rstudio.com/
5. Hadley Wickham, Advanced R.