labJournal5

Keiichiro Watanabe

2023-09-25

Part 0:

1. Shiny is an R package that allows you to create interactive web applications directly from your R code. This means you can turn your data analysis, visualization, and modeling scripts into interactive web apps that users can access through a web browser. It's a powerful way to share your work with others and make your data-driven applications accessible.
2. Shiny apps have a server-client architecture. The server contains the R code that processes user inputs, generates outputs, and updates the app's interface. The client is the web browser where users interact with the app's user interface. The two communicate to provide a dynamic and responsive user experience.
3. Reactivity is a fundamental concept in Shiny. It allows you to build apps where outputs (e.g., plots, tables) automatically update in response to changes in inputs (e.g., user selections, text inputs). You define reactive expressions and observe user inputs to trigger these updates, making your app responsive and interactive.

Part 1: R & Shiny

1

Purpose of ConditionalPanel: The main purpose of ConditionalPanel is to enable developers to create dynamic and context-aware user interfaces in Shiny apps. It helps tailor the display of UI elements or widgets based on user interactions, inputs, or other criteria. This can simplify the UI, making it more user-friendly, and also improve the app's performance by only rendering elements when necessary.

ConditionalPanel is useful in various scenarios:

1. When you have a form or input fields where certain fields should only be displayed or required when specific conditions are met.
2. When building data exploration or visualization tools, you can dynamically adjust the filtering criteria or sorting options based on user selections.
3. In interactive dashboards, you can use ConditionalPanel to display charts, tables, or summary statistics based on user-selected parameters, such as time periods or regions.

Example: I can use a conditional pane when I want to allow users to choose between different types of plots (e.g., scatter plot, bar chart, line chart) based on their data and analysis goals.

```r
library(shiny)

ui <- fluidPage(
  selectInput("plot_type", "Select Plot Type:",
              choices = c("Scatterplot", "Bar Chart", "Line Chart")),
  conditionalPanel(
    condition = "input.plot_type == 'Scatterplot'",
    plotOutput("scatterplot")
  ),

  conditionalPanel(
    condition = "input.plot_type == 'Bar Chart'",
    plotOutput("bar_chart")
  ),

  conditionalPanel(
    condition = "input.plot_type == 'Line Chart'",
    plotOutput("line_chart")
  )
)
```

2

The reading provides multiple tools and techniques for debugging Shiny applications. These are couple that stand out.

Breakpoints: You can set breakpoints in your Shiny code using the RStudio IDE. Simply click to the left of the line number to set a breakpoint. When you run your Shiny app, R will stop execution at the breakpoint, allowing you to step through your code, examine the environment, and check the callstack. Breakpoints are especially useful for pinpointing where a problem occurs in your code.

browser() Statements: The browser() statement is a debugging tool that acts like a breakpoint. When evaluated, it halts execution and enters the debugger. You can add it anywhere in your code where an R expression is valid. Unlike breakpoints, browser() works everywhere in your code, not just in Shiny server functions. You can use conditional browser() statements to create conditional breakpoints.

Showcase Mode: Showcase Mode is a debugging feature that displays your code alongside your Shiny application. It highlights your application's server code in yellow when it executes, making it easy to visualize which parts of your code are running as users interact with your app. You can enable Showcase Mode by running your app with the display.mode = "showcase" option.

The Reactive Log: The Shiny reactive log is a runtime tracing tool that helps you trace the execution of reactive in your app. It provides detailed information about which reactive are executing, their dependencies, and what's happening as Shiny evaluates your application. This tool is valuable for understanding how reactive expressions are behaving during app execution.

3

1. fluid-page: can have a sidebar and a main panel. The main panel typically occupies the larger portion of the screen, while the sidebar is used for additional controls, filters, or information.
2. full-dashboard: creating a multi-panel dashboard with various components such as charts, tables, filters, and text elements organized in a structured and visually appealing manner. This layout is commonly used for creating data-driven dashboards that provide users with an interactive and informative experience.

3. navbarPage: creates a tabbed layout where each tab represents a different section or page of an application. Users can switch between tabs to access different features or content.

The layout I might use in my project is the full-dashboard layout. I want to be able to convey multiple graphs and tables in my project, and a full-dashboard layout is perfect for my data-driven project.

4

In my project, I will use the renderText and textOutput functions in Shiny to create dynamic and interactive elements in the user interface that provide information and feedback to users based on their interactions with the application. Further, I plan to use two different input widgets in the Shiny user interface to enhance interactivity and allow users to customize their analysis. Here's how I intend to use these input widgets: Here's how I plan to use these functions:

1. In the context of my project, I may have a section of the user interface that displays real-time updates or statistics based on user input or data changes. For example, if users can select different parameters or filters that affect the number of shares an article got, I can use renderText to calculate and update summary statistics, and then use textOutput to display these statistics to the user.
2. I will incorporate a slider input widget labeled "Filter Article Age" in the user interface. This slider will allow users to filter articles based on their publication date. Users can specify a range of article ages (e.g., articles published within the last 30 days, 90 days, or a custom range).

5

The req() function is used to simplify handling missing inputs and other preconditions in a Shiny web application. Its primary purpose is to ensure that certain conditions or inputs are met before further code execution, helping you create more robust and error-resistant Shiny apps. You call the req() function with one or more arguments, typically representing conditions or inputs that you want to check.

Part 2: Reflection and questions on readings

1.

In 200 words describe your key takeaways from Chapter 11: Manipulate View (Munzner's book). Explain, How will you use it in your project?

Despite having the opportunity to make extremely complex data visualizations, that doesn't always mean that it should be taken advantage of. To encourage people to see the bigger picture you're trying to get across, you can use highlighting, which can make difficult to understand data visualizations make more sense to those with less knowledge in the topic you're educating on. Some of the representations are complex, and somewhat unreasonable for the data we're trying to represent. We should start with just static visual encodings and move towards dynamic encoding. When it comes to navigation, there should be constraints regarding the ability of users to interact with your data, as sometimes it can change the purpose of what you're trying to show.

2.

In 200 words describe your key takeaways from Chapter 12: Facet into Multiple Views (Munzner's book). Explain, How could you use it in your project? Consider alternative questions that can be answered with your data, if you think your current questions are not a good fit for multiple views.

Facets are extremely important because they can change the way that your data is interpreted by the viewer. When there are more than one data visualizations, there should be an easily identifiable discrepancy that can be found between the varying visualizations whether it be based on the color or the actual way the data is represented. The what-why-how table is very important in regards to deciphering the meaning of the data and the framework it's based upon, which is more for the creator of the visualization, as it's a way to keep yourself in check and make sure you're not overcomplicating. Layering is not only vital for three dimensional representations, but can also be used in two dimensional representations through color and texture.

Reference

Name(s) of all authors: Keiichiro Watanabe, Megan Bernacchi

Lab: Lab Journal 5

Due Monday, 2 October, 2023

Written/online sources used: (enter "none" if no sources other than the textbook and course website used) : none

Help obtained (Acknowledgments): none

"I/we confirm that the above list of sources is complete AND that I/we have not talked to anyone else about the solution to this problem."