**APP STRUCTURE**

**User interface**

This code defines the user interface (UI) for a Shiny app, including input controls (checkboxes, radio buttons), a download button, and output displays (histogram plot and table). The **sidebarLayout** function arranges the UI into a sidebar and main panel. The input controls have unique IDs (**select\_parentarea**, **select\_indicators**, **dtype**) that can be referenced in the server-side processing code to retrieve user selections.

**Function to plot curve**

This code defines a function called **plot\_epicurve** that creates a bar plot with error bars based on the input data, selected parentarea, and selected indicators. The function first checks if either the **parentarea** or **indicators** input is NULL, and if so, returns an empty plot with no data. If both **parentarea** and **indicators** are selected, the function filters the data based on these selections, calculates summary statistics (mean, standard deviation, sample size, standard error, lower and upper confidence interval), and then creates a bar plot with error bars using **ggplot2**. The plot is also faceted by parent\_area using **facet\_wrap**. The function returns the plot object.

**Server**

The **server** function defines the behavior of the Shiny app on the server side. Here are the detailed inline comments for each part of the code:

* **dat** is a reactive data object that filters the optdata4 dataset based on user input.
* The reactive function creates a reactive expression that re-evaluates whenever its dependencies (in this case, the input values) change. When dat is called, it returns the filtered data based on the selected parent area and indicator.
* **data <- optdata4:** This line assigns the "optdata4" dataset to the variable "data".
* **if(!is.null(input$select\_parentarea)&&!is.null(input$select\_indicators)):** This line checks if both the "select\_parentarea" and "select\_indicators" inputs are not null.
* **data <- data %>% filter(parent\_area %in% input$select\_parentarea) %>% filter(indicator %in% input$select\_indicators):** If the previous condition is true, this line filters the "data" variable based on the selected "parent\_area" and "indicator". The %>% operator is used for piping, which applies the filter functions to the "data" object one after the other.
* **data**: This line returns the filtered dataset as the output of the reactive object.
* **output$measures\_histogram** is a plot output that renders the epicurve plot using the filtered data. The **renderPlot** function takes the reactive dat object as its first argument, and the selected parent area and indicator as the other two arguments. This means that the plot will update whenever the selected parent area or indicator changes.

**output$table\_data** is a table output that renders a table showing the filtered data. The **renderTable** function takes a function as its argument that returns the filtered data if both the parent area and indicator are selected. If either one is not selected, it returns NULL. This means that the table will only update when both the parent area and indicator are selected.

**output$dwd\_data** is a download handler that defines the behavior of the download button. The **downloadHandler** function takes two arguments: filename and content. filename is a function that sets the name of the downloaded file based on the selected parent area, indicator, and data type. content is a function that writes the filtered data to a CSV file. When the download button is clicked, the file will be downloaded to the user's computer.

Top of Form

Bottom of Form