Feature 1: “Download CSV” Button

Motivation:

A “Download CSV” button in the Shiny app allows users to export the data they are viewing or manipulating within the Shiny app. This is crucial because it provides the flexibility to use the data outside of the app for various purposes such as detailed analysis, reporting, or record-keeping.

Description:

When a user interacts with the Shiny app and perhaps filters or manipulates data, they can click the "Download CSV" button. This action triggers a download of the current dataset in a .csv format to the user's local machine. Users might want to use this feature because of data portability and the convenience of having a local copy of the data for further use. The feature is intuitive to use in the App: it is clearly visible and labeled.

Use of the Features:

Function Call:

Assume ‘data_to_download’ is the data filtered by the user. We can call the function directly via:

```
write.csv(data_to_download, file, row.names = FALSE)
```

Through the App:

In the Shiny App, the "Download CSV" button allows users to download the data they are currently viewing or have filtered in the app interface.

Here’s how it works in the app:

1. The user applies various filters such as brief title keywords, sponsor type, outcome type, study type, and date interval using the provided input fields in the sidebar panel.
2. After applying the desired filters, the user clicks on the "Download CSV" button located in the sidebar panel.
3. The click event triggers the server to capture the current filtered dataset.
4. The data is then written to a CSV file, which is automatically downloaded to the user's computer.

Feature 2: Data Summary Table

Motivation:

A Data Summary Table in the Shiny app serves as a powerful tool to synthesize and present key statistics from a dataset, allowing users to quickly grasp essential aspects and metrics of the data under investigation. This could be vital for researchers and analysts who are interested in understanding the distribution and characteristics of clinical trials without having to dig through raw data.

Description:

The Data Summary Table feature provides a concise and informative snapshot of the dataset pertaining to clinical trials. It aggregates data to present important metrics such as the total number of trials, average trial duration, and the distribution of trials across different phases. This summary can be generated dynamically based on the current dataset within the app, reflecting any filters or search criteria applied by the user. Users can quickly derive meaningful insights from the summarized data, which can inform further analysis, decision-making processes, or provide a quick check on data distribution. The feature is intuitive to use in the App as the summary table is clearly labeled and presented in a way that is easy to understand.

Use of the Features:

Function Call:

Assume ‘studies_df’ contains the desired data selected by the user.
We first calculate metrics:

```
total_num_trials <- nrow(studies_df)
average_duration <- mean(studies_df$completion_date - studies_df$start_date, na.rm =
 TRUE)
num_trials_per_phase <- table(factor(studies_df$phase, levels = all_phases))
```

Then we create the summary text:

```
summary_text <- paste(
  "Total number of trials: ", total_num_trials, "\n",
  "Average trial duration: ", round(average_duration), " days\n",
  "Number of trials per phase:\n",
  paste(names(num_trials_per_phase), num_trials_per_phase, sep = ": ", collapse = "\n"),
  sep = ""
)
```

Assume this function is named ‘data_summary,’ we can call this function via:
`data_summary <- data_summary(studies_df)`

Through the App:

Within the Shiny App, the "Data Summary Table" outputs the calculated summary statistics as text in the UI.
When the application is running:

1. Users do not need to perform any additional actions specifically to view the data summary.

2. The data summary is automatically updated and displayed in the main dashboard area whenever the underlying data changes due to user interactions with other inputs.
3. Then the summary table provides a quick overview of key metrics, such as the total number of trials, average trial duration, and number of trials for each phase.

Feature 3: Date Interval

Motivation:

Clinical trials have specific durations and are often scheduled with precise start and completion dates. When analyzing the data, researchers and stakeholders may be interested in trials conducted within a particular timeframe. This is crucial for understanding trends, outcomes, or the density of clinical trials during certain periods. The date interval selection feature allows users to filter the clinical trial dataset based on a start and an end date, enabling focused analysis on trials active within the chosen date range.

Description:

The Date Interval Selection feature in the Shiny application provides an interface for users to specify a start date and an end date. This feature will filter the displayed data to only include clinical trials that were active between these dates. The filtering is applied to both the tabular data presentation and the visualizations that represent the data. It allows for a granular approach to data analysis, focusing on specific timelines which could be critical for longitudinal studies, seasonal analysis, or impact assessments.

Use of the Features:

Function Call:

Direct Function Call:
This feature can be utilized programmatically via a function called filter_by_date_interval(studies, start_date, end_date).
Example usage in R:

```
# Assuming 'studies' is a dataframe with clinical trials data
filtered_studies <- filter_by_date_interval(studies, "2021-01-01", "2021-12-31")
```

This would return a subset of the studies dataframe where the trials' start and completion dates fall within the specified range from January 1, 2021, to December 31, 2021.

Through the App:

Within the Shiny application, users interact with this feature through two date input fields:
Start Date (start_date): A date input allowing users to pick or enter the start date for the filtering.
End Date (end_date): A date input allowing users to pick or enter the end date for the filtering.
When the application is running:

1. Navigate to the sidebar panel on the left.
2. Locate the "Start Date" and "End Date" fields.
3. Select or input the desired start date.
4. Select or input the desired end date.
5. The action triggers a reactivity event in the application that automatically updates the displayed data according to the selected date interval.

The server-side logic ensures that all the reactive outputs that depend on the studies dataset are updated accordingly. This includes data tables, summary statistics, and any plots or graphs that represent the timeframe-based data.
Notes for Users:

1. Make sure the dates entered are within the range of the dataset.

2. The end date should not precede the start date; the application will not return any data if this is the case.
3. In the tabular data output, only trials that have a start date on or after the start_date and a completion date on or before the end_date will be displayed.
4. For visualizations, the filter will affect the data points represented within the specified interval, potentially changing the insights that can be drawn from the graphed information.

Feature 4: Interactive Plotly

Motivation:

Data visualization is a critical aspect of data analysis and interpretation, particularly in the context of clinical trials where the data can be complex and multi-dimensional. Static plots provide a snapshot of data, but interactive plots created with Plotly allow users to engage with the visualization directly. They can zoom in on areas of interest, hover to get more details, or even select specific data points for a more in-depth look. This interactivity enhances the user's ability to understand and derive insights from the data.

Description:

The Interactive Plot feature leverages the Plotly library to enhance the graphical representation of clinical trials data within the Shiny application. Plotly's interactive capabilities include tooltips, zooming, panning, and dynamic updating. In the context of this Shiny app, it is used to visualize the distribution of clinical trials across different phases and to show concurrent trials over time.

Use of the Features:

Function Call:

The Plotly interactive plots can be generated using Plotly's functions in R.
Example usage in R for a simple scatter plot:

```
library(plotly)
# Assuming 'data' is a dataframe with 'x' and 'y' columns for plotting
interactive_plot <- plot_ly(data, x = ~x, y = ~y, type = 'scatter', mode = 'lines+markers')
# To display the plot in an R environment
interactive_plot
```

Through the App:

Within the Shiny application, the Plotly plots are embedded within the main panel and are automatically generated based on the dataset filtered by other input controls.
To interact with the Plotly plot within the application:

1. Run the Shiny app and navigate to the "Phase" or "Concurrent" tab in the main panel.
2. The plots will be displayed based on the data provided or filtered.
3. Users can:

- Hover over any data point to see detailed information.
- Click and drag to zoom in on a specific area of the plot.
- Double-click on the plot to zoom back out.
- Use the toolbar that appears when hovering over the plot to interact with the plot further (e.g., save as an image, reset axes, etc.).

The interactive plots update reactively based on the user's interactions with other UI elements, such as date filters, keyword search, or sponsor type selections.
Notes for Users:

1. Interactive plots require JavaScript to be enabled in the web browser.
2. While hovering over data points, the tooltip will display specific data related to the point, such as trial IDs, phase information, or dates.

3. The interactivity of the plots is designed to enhance the user experience without overwhelming information. Users should utilize the provided toolbar for optimal interaction.
4. The responsiveness of the plot can be affected by the volume of data. With very large datasets, consider filtering the data further for a more responsive experience.

Feature 5: Outcome Dropdown Menu

Motivation:

Clinical trials can have different types of outcomes, such as primary, secondary, or other pre-specified outcomes. Each type of outcome has a different level of importance and relevance to the goals of the study. Users interested in analyzing or reviewing the results of clinical trials may want to focus on specific types of outcomes depending on their research needs or questions. Therefore, including a feature that allows users to filter trials by outcome type is very useful.

Description:

In the Shiny app, the outcomes dropdown menu is implemented using selectInput(). This creates a user interface element that lets users select from a list of predefined options. The choices provided are "Primary," "Secondary," "Other Pre-specified," and "Post-Hoc," which represent common categories of clinical trial outcomes. When a user selects one or more of these categories, the app will filter the dataset to include only those trials that report the selected type of outcome.

Use of the Features:

Function Call:

Assume we have a dataset outcomes with a column outcome_type, and we want to filter based on a selected outcome type:

```
library(dplyr)
filtered_studies_by_outcome <- outcomes %>%
  filter(outcome_type == selected_outcome_type)
```

Through the App:

1. In the sidebar panel, alongside other input elements, there is a drop-down menu (implemented using selectInput in Shiny) labeled "Select Outcome Type."
2. The drop-down contains various outcomes types such as "Primary", "Secondary", "Other Pre-specified", and "Post-Hoc". These reflect different categorizations of study outcomes.
3. The user can click on the drop-down menu and select one of these outcome types. Unlike the checkboxes, this is typically a single selection, meaning only one outcome type can be chosen at a time.
4. Upon selection, the Shiny server logic is triggered, particularly the reactive block where the dataset is joined with another dataset (outcomes) based on the "nct_id" and then filtered by the selected outcome type.
5. The main panel reacts to this filtering by updating its contents to display only the studies that report the chosen type of outcome. This could potentially refresh the content across different UI components such as data tables, summary text, or plots.

Feature 6: Study Type Checkbox

Motivation:

Users may be interested in a specific type of clinical study. For example, some researchers might only be interested in interventional studies where participants are assigned to receive one or more interventions (or no intervention) so that researchers can measure their effects on health-related biomedical or behavioral outcomes. Filtering out irrelevant data makes the dataset more manageable and the analysis more relevant to the user's specific needs.Providing checkboxes enhances the user experience by allowing for quick, easy, and multiple selections, which can be more user-friendly than dropdowns or text inputs for multi-select scenarios.

Description:

The feature is a group of checkboxes where each box corresponds to a type of clinical trial, such as "Interventional", "Observational", "Observational [Patient Registry]", "Expanded Access", or "NA" (Not Available or Not Applicable).Users can select any number of checkboxes, which allows for the inclusion of multiple study types in the query simultaneously.The selection made by the user through these checkboxes is used to filter the dataset for the types of studies selected.

Use of the Features:

Function Call:

Assuming we have a dataset studies with a column study_type, and we want to filter this dataset based on a vector of study types:

```
library(dplyr)
# Filter using dplyr
filtered_studies_by_type <- studies %>%
  filter(study_type %in% selected_study_types)
```

Through the App:

1. On the sidebar panel of the Shiny app, users will see multiple checkboxes each labeled with a type of study — for example, "Interventional", "Observational", etc.
2. Users can select or deselect any number of these checkboxes according to their interest. For instance, if a user is only interested in "Interventional" studies, they can select just that option
3. As a user checks or unchecks these boxes, the Shiny app reacts to these interactions. The reactive expression in the server part of the Shiny app captures these inputs.
4. The reactive expression uses these inputs to filter the dataset accordingly. If "Interventional" is selected, the dataset is filtered to include only interventional studies.
5. The main panel's output — such as tables, summaries, or plots — is updated to reflect the filtered data set based on the study types selected.