**User Guide for Updating the GEMMA Toolbox**

The **GEMMA toolbox** main script is organized into two key sections: the **User Interface (UI)** and the **Server**. These two sections collaborate to build an interactive application. External scripts should be added in the script sub-folder.

**1. Overview of the GEMMA Toolbox Structure**

**UI Function:**  
The UI function determines the application's layout and appearance. It defines input widgets (e.g., sliders, text boxes, dropdown menus) and output components (e.g., plots, tables, text). Using layout functions such as *fluidPage()*, *sidebarLayout()*, or *navbarPage()*, you can structure the interface effectively.

**Server Function:**  
The server function handles the application’s logic and computations. It processes user inputs and dynamically updates outputs using reactive expressions and render functions like *renderPlot()* or *renderTable()*.

Together, the UI and server functions create a seamless interaction loop: the UI gathers inputs from users, while the server computes result and displays them back via the UI.

**2. Adding a New Section to the GEMMA Toolbox**

This guide provides instructions for adding a new section to the GEMMA toolbox. While advanced structural implementations are recommended for experienced programmers, minor customizations (e.g., styling, adding statistical tests) can be performed by any R user.

**Step 1: Update the Library List**

If your new section requires additional R libraries, ensure they are added to the GEMMA toolbox library list. Refer to the figure below for guidance.

Immagine che contiene testo, schermata

Descrizione generata automaticamente

**Step 2: Update the UI Function**

In this example, we’ll add a new panel called **“DCA”** with two subpanels: **“MAIN”** and **“BIPLOTS”**.

* Panels and subpanels can be created using the *tabPanel()* function.
* Use columns and rows to organize the graphical user interface (GUI).
* Incorporate dynamic elements using the *uiOutput()* function, which allows for GUI updates during the workflow.

The figures below illustrate the structure of the new subpanels.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma

Descrizione generata automaticamente

**Step 3: Modify the Server Function**

The server function is the most complex part of the application, handling:

* Iterative processes
* Data analysis
* Script calls
* Plot updates
* Data export

**Important:** When modifying the server function:

* Avoid conflicts between global and local variables.
* Ensure the software remains functional under iterative, multi-analysis scenarios.

**Server Function Features:**

1. **Dynamic Variable Updates:**  
   Use the *observe()* function to monitor variables in **MATRIX A** and update the checkbox list for DCA analysis using the *renderUI()* function.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

1. **Triggering Events:**  
   The observeEvent() function detects when the **"start\_dca"** button is clicked. Once triggered, the DCA process begins.
   * Scripts (e.g., RDA.R) can be executed using the local() function for a modular design.
   * Outputs, such as scree plots, can be saved using the ggsave() function.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

1. **Monte Carlo Test Integration:**
   * A dedicated section runs the **DCA\_MC** script, updates the dca\_mc\_pvals plot area, and allows PDF export.
   * The **"BIPLOTS"** subpanel dynamically updates with retained DCA axes and provides a button for running the biplot script.

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

1. **Biplot Creation:**
   * After completing the Monte Carlo test, users can access the biplot subsection.
   * DCA axes can be selected via a checkbox list.
   * The DCA\_BIPLOT.R script generates biplots, and areas like dca\_biplot\_1, dca\_biplot\_2, and dca\_biplot\_3 are updated accordingly.

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

**Step 4: Create the App**

The final step is to build the app using the shinyApp() function. See the figure below for an example.

Immagine che contiene schermata, linea, viola

Descrizione generata automaticamente

**Additional Notes**

* For advanced customizations, maintain code modularity and perform extensive testing to ensure stability.
* Minor updates (e.g., styling or additional statistical features) can be made without altering the core structure.

This guide should help you extend the GEMMA toolbox while preserving its functionality and integrity.