Package: Modules

Title: Modular Shiny Functions for Building Simulation Apps

Version: 0.0.0.9000

Authors@R:

Ginger Anderson, ginger.anderson@pfizer.com

Description: The Package provides a user previously made R code scripts to efficiently develop R Shiny app's

License: GPL-2

Encoding: UTF-8

Table of Contents

1. Objectives

The Shiny package for R programming language has been utilized to produce applications (apps) and dynamic visuals to provide insight into the pharmacokinetic (PK) profile of drugs and biomarkers. This insight leads to better PK model development and effective communication with the clinical team. Despite this advantage, the development and validation of Shiny apps is very time-intensive and makes app production challenging for many members of the GPD-Clinical Pharmacology (CP) community. The objectives of producing multiple ready-to-use R Shiny Modules or modular code are to:

- increased efficiency of PK app development in the CP community
- greatly improve efficiency for standardized tasks

2. User Interface (ui.R)

2.1 Objective

The objective of the user interface (UI) is to organize the information for the point of human user input. The UI defines how your app looks. This objective includes:

- labels the inputs from the human
- displays the calculated outputs

2.2 Functions

The main function of the ui.R module is the "app_ui" function which utilizes the ui shiny functions navbarPage, sidebarLayout, sidebarPanel, and mainPanel to organize the display to the human user. The mainPanel and sidebarPanel can also call the following functions in other modules which are all module UI's:

- "mod_DoseInput_ui"
- "mod_multiDoseInput_ui"
- "mod_mrgSimulate_ui"
- "mod_table_ui"
- "mod_plot_CT_ui"
- "mod_plot_his_ui"

Each module UI's first line is the id string which creates a namespace function for the module. The same "id" needs to be used between the UI and server due to them sharing the same environment. Servers are further explained in the following section. All input/output ID need to be wrapped in ns(). Often a "taglist" is used to bundle together components without giving direction on the layout of the components. To learn more about UI's in module code, see the following:

- Shiny - Modularizing Shiny app code (rstudio-staging.com)

2.3 Flow

The ui.R module inputs the following from the human user:

- Simulated Population Size
- Typical Baseline Body Weight (kg)
- Sex
- Dosing Regimen

This information is then put in the global environment as an input to be used by the other modules. The module also takes the final plot or table outputted from other modules and displays it.

2.4 Customizable Options

The sidebarPanel() input in the ur.R module can be changed to display other covariate input from the human user. The initial or selected value can also be customized.

3. Server (server.R)

3.1 Objective

The server function defines how your app works. In the server.R module the objective of the module is to call the other modules and bring them together. It also defines:

- initial values
- units
- axis titles
- color schemes

3.2 Functions

The main function of the server.R module is the "app_server" function which utilizes the R shiny module feature to call other functions in other modules and bring the information together. The server.R module can also call the following functions in other modules which are all module servers:

- "mod_DoseInput_server"
- "mod_multiDoseInput_server"
- "mod_mrgSimulate_server"
- "mod_plot_CT_server"
- "mod_plot_his_server"

To learn more about servers in module code, see the following:

- Shiny - Modularizing Shiny app code (rstudio-staging.com)

3.3 Flow

The server.R module uses the Shiny callModule function to call other functions that are smaller parts of other modules. The "app_server" function first calls the Dose or Multi-Dose Module to use the mod_DoseInput_server and input the initial values and dose units for the dose. The "app_server" function then calls the Simulation Module to use the mod_mrgSimulate_server function which inputs the population size, the model for the drug in question, dose, weight, and sex. The "app_server" function then calls the Plot Modules to use the mod_plot_CT_server or

mod_plot_his_server function to input the axis titles and input color schemes for an outputted plot or histogram.

3.4 Customizable Options

The dose regimen initial values for the application can be customized when callModule is used the call the mod_DoseInput_server or mod_multiDoseInput_server. The units for the values of the dose regimen can also be customized. The covariates can be customized when callModule is used the call the mod_mrgSimulate_server. For the plot modules' sever, the line color, fill, plot title, axis titles quantiles for the concentration-time curve and y axis upper limit for the histogram can also be customized.

4. Envir (envir.R)

4.1 Objective

The objective of envir. R module is to:

- call all the needed libraries for all of the modules to use
- clear the environment
- "source in" all the functions in other modules or bring them all into the global environment to use
- bring in the ggplot2 theme functions
- upload the model for the drug
- Lastly, actually run the app

The envir.R ends in "shinyApp(ui = app_ui, server = app_server)" which actually runs the app. The envir.R is the only module that needs to be run to bring all the other modules together.

4.2 Functions

The envir.R module does not use any custom made functions. This module only uses functions or packages that are already available in R such as mrgsolve.

4.3 Flow

The module first clears all objects from the environment. The module then brings in all the needed libraries in other modules. After this, the module sources in all functions and modules that are needed to run the code. The module then brings in the functions for ggplot2. Then the module loads the module for the drug in question. The module ends with the line of code "shinyApp(ui = app_ui, server = app_server)" which actually runs the app.

4.4 Customizable Options

The PK model can be uploaded and customized. The path and files for the server and user interface module can be modified.

5. Simulation Module (mod_mrgSimulate.R)

5.1 Objective

The objective of the simulation module is to simulate the pharmacokinetic model with the input from the human app user.

5.2 Functions

The simulation module contains the mod_mrgSimulate_ui, mod_mrgSimulate_server and mod_mrgSimulate functions. The mod_mrgSimulate_ui function is used by the ui.R module to input the updated population and dosing regimen inputted by the human user. The mod_mrgSimulate_server function is used by the server.R module to summarize the simulation input data. The mod_mrgSimulate function is the function which actually uses mrgsolve to simulate.

5.3 Flow

The Simulation Module first defines the mod_mrgSimulate_ui function and then it defines the mod_mrgSimulate_server function. In the mrgSimulate_server function the model, dose regimen, number of patients in the simulation, human user input and covariate information is formatted for the mrgsolve simulation input. In the mod_mrgSimulate function, the inputted data is then modeled by mrgsolve and then packaged to be outputted to be used in the plot or table modules for graphing or exploration.

6. Dose and Multi-Dose Module (mod_DoseInput.R/mod_multiDoseInput.R)

6.1 Objective

The objective of the dose module is to input and organize the inputted dose regimen from the human user through the ui.R module. The multi-dose module can be used to display dose loading while using the mod_plot_CT.R module.

6.2 Functions

The dose modules contains a ui and server function. The mod_DoseInput_ui or mod_multiDoseInput_ui function sets up the taglist for the collection of the dose regimen to be used in the ui.R module. The mod_DoseInput_server or mod_multiDoseInput_server function defines "rv" which are the reactive objects for the dose, interval and duration. The mod_DoseInput_server or mod_multiDoseInput_server function also defines the "Rui" which is the reactive function to organize the inputted dose. "moduleOutput" is the reactive function that process the dosing information.

6.3 Flow

The dose module inputs the dose regimen from the human user through ui.R module.
The server functions records and formats the selected dose regimen for use by the simModule and is called in the server module.

7. Plot Module - CT (mod_plot_CT.R)

7.1 Objective

The objective of the plot module "mod_plot_CT.R" is to provide visualizations of the outputted simulations concentration-time curve.

7.2 Functions

The plot module contains the mod_plot_CT_ui and the mod_plot_CT_server functions. The mod_plot_CT_ui function sets up the taglist for the plot to be used in the ui.R module. The mod_plot_CT_server function takes the simulation information and uses ggplot2 to create a plot to be used in ui.R.

7.3 Flow

The mod_plot_CT_server function takes the simulation input and uses ggplot2 to create visualizations. The mod_plot_CT_server function is called in the server.R module. The outputted plot is displayed in ui.R.

8. Plot Module - Histogram (mod_plot_his.R)

8.1 Objective

The objective of the plot module "mod_plot_his.R" is to provide a histogram visualization for volume of distribution (L), absorption rate constant (1/hr) and clearance (L/hr).

8.2 Functions

The plot module contains the mod_plot_his_ui and the mod_plot_server functions. The mod_plot_his_ui function sets up the taglist for the plot to be used in the ui.R module. The mod_plot_his_server function takes the simulation information and uses the generic function hist() to create a histogram to be used in ui.R.

8.3 Flow

The mod_plot_his_server function takes the simulation input and uses hist() to create a histogram visualization. The mod_plot_his_server function is called in the server.R module. The outputted histogram is displayed in ui.R.

9. Table Module (mod_table.R)

9.1 Objective

The objective of the table module is to provide a table summary of the simulation. The table displays single dose and steady-state summaries of median, mean, standard deviation, maximum, minimum, the lower 90% CI bound, and the upper 90% CI bound.

9.2 Functions

The plot module contains the mod_table_ui and the mod_table_server functions. The mod_table_ui function sets up the taglist for the tables to be used in the ui.R module. The mod_table_server function takes the simulation information and creates a table to be used in ui.R.

9.3 Flow

The mod_table_server function takes the simulation input and to create a table visualization. The mod_table_server function is called in the server.R module. The outputted table is displayed in ui.R.

10. QC Plan

The software will be tested using the excel “Module_TestingWorkSheet”. The “Module_TestingWorkSheet” utilizes datasets found on teams and a clinical pharmacology tester to confirm the output of the application is correct.

11. “Module” Package Plan

The “Module” package consists of 13 functions:

- mod_DoseInput_ui
- mod_DoseInput_server
- mod_multiDoseInput_ui
- mod_multiDoseInput_sever
- fct_mrgSimulate
- mod_mrgSimulate_server
- mod_mrgSimulate_ui
- mod_plot_CT_ui
- mod_plot_CT_server
- mod_plot_his_ui
- mod_plot_his_server
- mod_table_ui
- mod_table_server

The ui.R, server.R, and envir.R modules are not part of the “Module” package due to them being able to be modified/customized by the user.