Package ‘mmints’

February 12, 2025

Title Workflows for Building Web Applications

Version 0.2.0

Description Sharing statistical methods or simulation frameworks through 'shiny' applications often requires workflows for handling data. To help save and display simulation results, the postgresUI() and postgresServer() functions in 'mmints' help with persistent data storage using a 'PostgreSQL' database. The 'mmints' package also offers data upload functionality through the csvUploadUI() and csvUploadServer() functions which allow users to upload data, view variables and their types, and edit variable types before fitting statistical models within the 'shiny' application. These tools aim to enhance efficiency and user interaction in 'shiny' based statistical and simulation applications.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/mightymetrika/mmints

BugReports https://github.com/mightymetrika/mmints/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports DT, pool, RPostgres, shiny, shinyauthr, sodium

NeedsCompilation no

Author Mackson Ncube [aut, cre],
mightymetrika, LLC [cph, fnd]

Maintainer Mackson Ncube <macksonncube.stats@gmail.com>

Repository CRAN

Date/Publication 2025-02-12 17:50:09 UTC

Contents

`authServer` *Server function for Authentication Shiny Module*

Description

This function sets up the server-side logic for the Authentication module, handling user authentication, signup, and guest access.

Usage

```
authServer(id, postgres_module, user_table = "users")
```

Arguments

`id`	A character string that matches the ID used in `authUI()`
`postgres_module`	A postgresModule instance to handle database operations
`user_table`	A character string specifying the name of the users table

Value

A list containing authentication status and user information

Examples

```
server <- function(input, output, session) {
  postgres <- postgresServer("postgres_module", ...)
  auth <- authServer("auth_module", postgres, "users")
}
```

authUI *Create UI elements for Authentication Shiny Module*

Description

This function generates the UI components for the Authentication module, including login, signup, and guest access options.

Usage

```
authUI(id)
```

Arguments

id	A character string that uniquely identifies this module instance

Value

A list containing UI elements for authentication

Examples

```
shiny::fluidPage(
  authUI("auth_module")
)
```

citationServer *Server Function for Citation Module*

Description

This function defines the server logic for the citation module.

Usage

```
citationServer(id, citations)
```

Arguments

id	A character string that matches the ID used in citationUI.
citations	A named list of citations. Each element can be: • A character string containing a formatted citation. • A function that returns a formatted citation string. • A citation object that can be passed to format_citation.

Value

A Shiny module server function.

Examples

```
citations <- list(
  "Example Citation" = "Author, A. (Year). Title. Journal, Vol(Issue), pages.",
  "R Citation" = function() format_citation(utils::citation())
)
server <- function(input, output, session) {
  citationServer("my_citations", citations)
}
```

---

citationUI *UI Function for Citation Module*

Description

This function creates the UI elements for the citation module.

Usage

```
citationUI(id)
```

Arguments

id	A character string that defines the namespace for the module.

Value

A list containing two elements:

- button: An action button to show citations.
- output: A tag list containing the citation header and output.

Examples

```
citationUI("my_citations")
```

csvUploadServer *Server Function for CSV Upload Module*

Description

This function defines the server logic for the CSV upload module.

Usage

```
csvUploadServer(id, vars_title = "Available Variables")
```

Arguments

id	A character string that matches the ID used in csvUploadUI.
vars_title	A character string for the title of the variables table.

Value

A reactive expression containing the uploaded data.

Examples

```
server <- function(input, output, session) {
  csvUploadServer("my_data", "My Variables")
}
```

csvUploadUI *UI Function for CSV Upload Module*

Description

This function creates the UI elements for the CSV upload module.

Usage

```
csvUploadUI(id)
```

Arguments

id	A character string that defines the namespace for the module.

Value

A list containing two elements:

- input: The file input UI for uploading a CSV file.
- output: The UI for displaying the variables table.

Examples

```
csvUploadUI("my_data")
```

---

format_citation *Format Citation*

Description

This function formats a citation object into a format ready for use in 'shiny' applications

Usage

```
format_citation(cit)
```

Arguments

cit	A citation object obtained from `utils::citation()`.

Value

A character string containing the formatted citation.

Examples

```
format_citation(utils::citation("base"))
```

---

generateRunCode *Generate a Unique Run Code*

Description

This function generates a unique run code for use in Shiny applications, particularly those running simulations. The code combines a timestamp with a random string to ensure uniqueness for each row or run.

Usage

```
generateRunCode(time_format = "%Y%m%d%H%M%S", string_length = 5)
```

Arguments

time_format	A string specifying the format for the timestamp. Default is "%Y%m%d%H%M%S" (year, month, day, hour, minute, second).
string_length	An integer specifying the length of the random string. Default is 5.

Value

A character string containing the unique run code, composed of a timestamp and a random alphanumeric string, separated by an underscore.

Note

This function uses the current system time and a random string to generate the run code. While collisions are extremely unlikely, they are theoretically possible, especially if the function is called multiple times within the same second and with a short string_length.

Examples

```
generateRunCode()
generateRunCode(time_format = "%Y%m%d", string_length = 8)
```

---

list_null *Handle Null Values for Text to List Conversions*

Description

Handle Null Values for Text to List Conversions

Usage

```
list_null(par_input = "", alt_na = NULL)
```

Arguments

par_input	A string input, default is "".
alt_na	If alt_na is not set to NULL (the default), then it is an alternative string used to represent NA. Usually, this is a string such as "NA", "NaN", etc

Value

NULL if input is NA, if input is empty, or if input is alt_na (and alt_na is not NULL). Otherwise return a parsed list.

Examples

```
# Convert missing value to null
list_null()
list_null(NA)
list_null("na", alt_na="na")

# Convert non-missing value to list
list_null("'one' = 1, 'two' = 2, 'three' = 3")
```

```
# Convert to null when a single vector is missing
list_null("'one' = 1, 'two' = NA, 'three' = 3", alt_na = "NA")
list_null("'one' = 1, NA, 'three' = 3", alt_na = "NA")
```

`postgresServer` *Server function for Postgres Shiny Module*

Description

This function sets up the server-side logic for the Postgres Shiny module, handling database connections, data submission, retrieval, and download.

Usage

```
postgresServer(id, dbname, datatable, host, port, user, password, data)
```

Arguments

`id`	A character string that matches the ID used in `postgresUI()`
`dbname`	A character string specifying the name of the database
`datatable`	A character string specifying the name of the table in the database
`host`	A character string specifying the host of the database
`port`	A numeric value specifying the port number for the database connection
`user`	A character string specifying the username for database access
`password`	A character string specifying the password for database access
`data`	A reactive expression that provides the data to be submitted

Value

A list of functions and reactive values:

`executeQuery`	A function to run arbitrary SQL
`saveData`	A function to save data to the database
`loadData`	A function to load data from the database
`current_data`	A reactive value containing the current data in the table
`data_to_submit`	A reactive value containing the data to be submitted

Examples

```
server <- function(input, output, session) {
  postgres_module <- postgresServer("postgres_module", "my_db", "my_table",
                                    "localhost", 5432, "user", "password",
                                    reactive({ input$data }))
 }
```

postgresUI *Create UI elements for Postgres Shiny Module*

Description

This function generates the UI components for the Postgres Shiny module, including a submit button, a data table, and a download button.

Usage

```
postgresUI(id)
```

Arguments

id	A character string that uniquely identifies this module instance

Value

A list containing three UI elements:

submit	An action button for submitting data to database
table	A DT output for displaying the database data
download	A download button for exporting database data to csv

Examples

```
shiny::fluidPage(
  postgresUI("postgres_module")$submit,
  postgresUI("postgres_module")$table,
  postgresUI("postgres_module")$download
  )
```

text_to_list *Convert Text Input to List*

Description

Convert Text Input to List

Usage

```
text_to_list(text_input)
```

Arguments

text_input	A text representation of a list.

Value

A list parsed from the input string.

Examples

```
# Create a named list
text_to_list("'one' = 1, 'two' = 2, 'three' = 3")

# Create a list of vectors
text_to_list("c('x1', 'x2'), c('x3', 'x4')")
```

---

text_to_vector *Convert Text Input to Vector*

Description

The goal of this function is to take text input and convert it to an R vector.

Usage

```
text_to_vector(text_input)
```

Arguments

text_input	A string input to be converted to a vector.

Value

A vector parsed from the input string.

Examples

```
text_to_vector("1,2,3,4,5")

text_to_vector("1:5")

text_to_vector("rep(1:5, times = 2)")

text_to_vector("seq(1,10,2)")
```

vec_null *Handle Null Values for Text to Vector Conversion*

Description

Handle Null Values for Text to Vector Conversion

Usage

```
vec_null(par_input = "", alt_na = NULL)
```

Arguments

par_input	A string input, default is "".
alt_na	If alt_na is not set to NULL (the default), then it is an alternative string used to represent NA. Usually, this is a string such as "NA", "NaN", etc.

Value

NULL if input is NA, if input is empty, or if input is alt_na (and alt_na is not NULL). Otherwise, return a vector.

Examples

```
# Convert missing value to NULL
vec_null()
vec_null(NA)
vec_null("na", alt_na="na")

# Convert string to vector when input is not missing
num_vec <- vec_null("2,8,3,7")

# Convert string to NULL when a single element is missing
vec_null("2,3,NA,5", "NA")
```

Index