Package ‘DataEditR’

October 12, 2022

Title  An Interactive Editor for Viewing, Entering, Filtering & Editing Data
Version 0.1.5
Date 2022-03-08
Description An interactive editor built on ‘rhandsontable’ to allow the interactive viewing, entering, filtering and editing of data in R <https://dillonhammill.github.io/DataEditR/>.

BugReports https://github.com/DillonHammill/DataEditR/issues
Depends R(>= 3.5.0)
Imports shiny (>= 1.5.0), shinyBS, shinyjs, bslib, rhandsontable (>= 0.3.8), rstudioapi, htmltools, miniUI, utilties
License GPL-2
Encoding UTF-8
RoxygenNote 7.1.2
Language en-GB
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Dillon Hammill [aut, cre] (<https://orcid.org/0000-0002-1407-7223>)
Maintainer Dillon Hammill <Dillon.Hammill@anu.edu.au>
Repository CRAN
Date/Publication 2022-03-08 12:00:02 UTC

R topics documented:

dataEdit ................................................................. 2
dataFilter .............................................................. 4
dataInput ............................................................... 5
**Shiny module for data editing**

**Description**

Shiny module for data editing

**Usage**

```r
dataEditUI(id)

dataEditServer(
  id,
  data = reactive(NULL),
  col_bind = NULL,
  col_edit = TRUE,
  col_options = NULL,
  col_stretch = FALSE,
  col_names = TRUE,
  col_readonly = NULL,
  col_factor = FALSE,
  row_bind = NULL,
  row_edit = TRUE,
  row_index = reactive(NULL),
  quiet = FALSE,
  read_fun = "read.csv",
  read_args = NULL,
  ...
)
```

**Arguments**

- **id**
  - unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.

- **data**
  - a reactive expression containing an array (e.g. data.frame, matrix or data.table) or a vector indicating the dimensions of the array (e.g. c(10,10)) or column names to construct a new template for editing. If no data is supplied a template with 10 rows and columns will be generated for editing.

- **col_bind**
  - additional columns to add to the data prior to loading into editor, can be either an array containing the new data, a vector containing the new column names for empty columns or a named list containing a vector for each new column.
`dataEdit` 3

- **col_edit**: logical indicating whether columns can be added or removed, set to TRUE by default.
- **col_options**: a list named with valid columns names and either `c(TRUE, FALSE)` for checkboxes, a vector of options for dropdowns, "date" for date input or "password" for password input.
- **col_stretch**: logical indicating whether columns should be stretched to fill the full width of the display, set to FALSE by default.
- **col_names**: logical indicating whether column names can be edited or a vector of column names that cannot be edited, set to TRUE by default to allow editing of column names.
- **col_readonly**: names of columns that cannot be edited. Users will be able to edit values but these will be reverted to the original values. Column names for these column cannot be edited either.
- **col_factor**: logical indicating whether character columns should be converted to factors prior to returning the edited data, set to FALSE by default.
- **row_bind**: additional rows to add to the data prior to loading into editor, can be either an array containing the new data, a vector containing the new row names for empty rows or a named list containing a vector for each new column.
- **row_edit**: logical indicating whether rows can be added or removed, set to TRUE by default.
- **row_index**: indicates the starting index for new rows when the data supplied to `DataEdit()` is a subset of a larger dataset, i.e. `row_index` indicates the number of rows present in the parental dataset.
- **quiet**: logical to suppress warnings when using `col_options`.
- **read_fun**: name of the function to use to read in the data when a file is selected, set to `read.csv` by default.
- **read_args**: a named list of additional arguments to pass to `read_fun` when reading in files.
- **...**: additional arguments passed to `rhandsontable`.

**Value**

reactive expression containing the edited data.

**Author(s)**

Dillon Hammill, <Dillon.Hammill@anu.edu.au>

**Examples**

```r
if (interactive()) {
  ui <- fluidPage(
    dataInputUI("input-1"),
    dataOutputUI("output-1"),
    dataEditUI("edit-1")
  )
} 
```
server <- function(input, output, session) {
  data_to_edit <- dataInputServer("input-1")
  data_edit <- dataEditServer("edit-1",
    data = data_to_edit
  )
  dataOutputServer("output-1",
    data = data_edit
  )
}

shinyApp(ui, server)

---

dataFilter

**Shiny module for filtering data**

**Description**

Shiny module for filtering data

**Usage**

dataFilterUI(id)

dataFilterServer(id, data = reactive(NULL), hide = FALSE, hover_text = NULL)

**Arguments**

- **id** unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.
- **data** an array wrapped in reactive() containing the data to be filtered.
- **hide** logical indicating whether the data filtering user interface should be hidden from the user, set to FALSE by default.
- **hover_text** text to display on download button when user hovers cursor over button, set to NULL by default to turn off hover text.

**Value**

a list of reactive objects containing the filtered data and indices for filtered rows.

**Author(s)**

Dillon Hammill, <Dillon.Hammill@anu.edu.au>
Examples

```r
if (interactive()) {
  library(shiny)
  library(rhandsontable)
  library(shinyjs)

  ui <- fluidPage(
    useShinyjs(),
    dataInputUI("input1"),
    dataFilterUI("filter1"),
    rHandsontableOutput("data1")
  )

  server <- function(input, output, session) {
    data_input <- dataInputServer("input1")

    # list with slots data and rows (indices)
    data_filter <- dataFilterServer("filter1",
      data = data_input
    )

    output$data1 <- renderRHandsontable(
      if (!is.null(data_filter$data())) {
        rhandsontable(data_filter$data())
      }
    )

    shinyApp(ui, server)
  }
}
```

**dataInput**  
*Shiny module for data input*

**Description**

Shiny module for data input

**Usage**

```r
dataInputUI(id, cellWidths = c("50%", "48%"))

dataInputServer(
  id,
  data = NULL,
)```
read_fun = "read.csv",
read_args = NULL,
hide = FALSE,
envir = parent.frame()
)

Arguments

id unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.

cellWidths a vector of length 2 to control the relative widths of the fileInput and textInput, set to c("50\%", "50\%") by default.

data can be either the name of a dataset or file as a character string (e.g. "mtcars" or "mtcars.csv") or a vector column names (e.g. c("A", "B", "C")) or template dimensions (e.g. c(10,10)).

read_fun name of the function to use to read in the data when a file is selected, set to read.csv by default.

read_args a named list of additional arguments to pass to read_fun when reading in files.

hide logical indicating whether the data input user interface should be hidden from the user, set to FALSE by default.

envir the environment in which to search for the supplied data, set to the parent.frame() of dataInput by default.

Author(s)

Dillon Hammill, <Dillon.Hammill@anu.edu.au>

Examples

if (interactive()) {
  library(shiny)
  library(rhandsontable)

  ui <- fluidPage(
    dataInputUI("input1"),
    rhandsontableOutput("data1")
  )

  server <- function(input,
                      output,
                      session) {
    data_input1 <- dataInputServer("input1")

    output$data1 <- renderRHandsontable({
      if (!is.null(data_input1())) {
        rhandsontable(data_input1())
      }
    })
  }
}
dataOutput

shinyApp(ui, server)
}

dataOutput Shiny module for data output

Description

Shiny module for data output

Usage

dataOutputUI(id, icon = "download")

dataOutputServer(
  id,
  data = reactive(NULL),
  save_as = NULL,
  write_fun = "write.csv",
  write_args = NULL,
  hide = FALSE,
  hover_text = NULL
)

Arguments

id unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.
icon supplied to dataOutputUI to control the appearance of the icon displayed on the download button, set to "download" by default.
data an object of class data.frame wrapped in reactive to be saved to file.
save_as name of the file to which the data should be saved, overrides input file path if supplied.
write_fun name of the function to use when writing the data to file, set to "write.csv" by default.
write_args a named list of additional arguments to pass to write_fun when reading in files.
hide logical indicating whether the data input user interface should be hidden from the user, set to FALSE by default.
hover_text text to display on download button when user hovers cursor over button, set to NULL by default to turn off hover text.

Author(s)

Dillon Hammill, <Dillon.Hammill@anu.edu.au>
```r
if (interactive()) {
  library(shiny)
  library(rhandsontable)
  library(shinyjs)

  ui <- fluidPage(
    useShinyjs(),
    dataInputUI("input1"),
    dataOutputUI("output1"),
    rHandsontableOutput("data1")
  )

  server <- function(input, output, session) {
    data_input1 <- dataInputServer("input1")

    output$data1 <- renderRHandsontable({
      if (!is.null(data_input1())) {
        rhandsontable(data_input1())
      }
    })

    dataOutputServer("output1",
                    data = data_input1)
  }

  shinyApp(ui, server)
}
```

**dataSelect**

*Shiny module for selecting data*

**Description**

Shiny module for selecting data

**Usage**

```r
dataSelectUI(id)
```

```r
dataSelectServer(id, data = reactive(NULL), hide = FALSE, hover_text = NULL)
```

**Arguments**

- `id` unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.
dataSelect

data an array wrapped in reactive() containing the data to be filtered.
hide logical indicating whether the data selection user interface should be hidden from the user, set to FALSE by default.
hover_text text to display on download button when user hovers cursor over button, set to NULL by default to turn off hover text.

Value

a list of reactive objects containing the filtered data and indices for selected columns.

Author(s)

Dillon Hammill, <Dillon.Hammill@anu.edu.au>

Examples

if (interactive()) {
  library(shiny)
  library(rhandsontable)
  library(shinyjs)

  ui <- fluidPage(
    useShinyjs(),
    dataInputUI("input1"),
    dataSelectUI("select1"),
    rHandsontableOutput("data1")
  )

  server <- function(input, output, session) {
    data_input <- dataInputServer("input1")

    data_select <- dataSelectServer("select1",
      data = data_input
    )

    output$data1 <- renderRHandsontable({
      if (!is.null(data_select$data())) {
        rhandsontable(data_select$data())
      }
    })

    shinyApp(ui, server)
  }
}
**dataSync**

*A shiny module to synchronise datasets*

### Description

The purpose of this module is to merge changes made to a subset of the data with the master copy of the data.

### Usage

```r
dataSyncUI(id)

dataSyncServer(
  id,
  data = reactive(NULL),
  data_subset = reactive(NULL),
  rows = reactive(NULL),
  columns = reactive(NULL),
  hide = FALSE,
  hover_text = NULL
)
```

### Arguments

- **id**: unique identifier for the module to prevent namespace clashes when making multiple calls to this shiny module.
- **data**: master copy of the data.
- **data_subset**: subset of data with altered entries.
- **rows**: the row indices of `data_subset` within `data`.
- **columns**: the column indices of `data_subset` within `data`.
- **hide**: logical indicating whether the data synchronisation user interface should be hidden from the user, set to FALSE by default.
- **hover_text**: text to display on download button when user hovers cursor over button, set to NULL by default to turn off hover text.

### Author(s)

Dillon Hammill, <Dillon.Hammill@anu.edu.au>

### Examples

```r
if(interactive()){
  library(shiny)
  library(rhandsontable)
  library(shinyjs)
```
**data_edit**

An interactive editor for viewing, entering and editing data

**Description**

codedata_edit is a shiny application built on rhandsonetable that is designed to make it easy to interactively view, enter or edit data without any coding. data_edit is also a wrapper for any reading or writing function to make it easy to interactively update data saved to file.

**Usage**

data_edit(
    x = NULL,
)
col_bind = NULL,
col_edit = TRUE,
col_options = NULL,
col_stretch = FALSE,
col_factor = FALSE,
col_names = TRUE,
col_readonly = NULL,
row_bind = NULL,
row_edit = TRUE,
save_as = NULL,
title = NULL,
logo = NULL,
logo_size = 30,
logo_side = "left",
viewer = "dialog",
viewer_height = 800,
viewer_width = 1200,
theme = "yeti",
read_fun = "read.csv",
read_args = NULL,
write_fun = "write.csv",
write_args = NULL,
quiet = FALSE,
hide = FALSE,
code = FALSE,
...
)

Arguments

x a matrix, data.frame, data.table or the name of a csv file to edit. Tibbles are also supported but will be coerced to data.frames. An empty table can be created by specifying the dimensions in a vector of the form c(nrow, ncol) or the names of the columns to include in the template.

col_bind additional columns to add to the data prior to loading into editor, can be either an array containing the new data, a vector containing the new column names for empty columns or a named list containing a vector for each new column.

col_edit logical indicating whether columns can be added or removed, set to TRUE by default.

col_options named list containing the options for columns that use dropdown menus, dates, checkboxes or passwords.

col_stretch logical indicating whether columns should be stretched to fill the full width of the display, set to FALSE by default.

col_factor logical indicating whether character columns should be converted to factors prior to returning the edited data, set to FALSE by default.

col_names logical indicating whether column names can be edited or a vector of column names that cannot be edited, set to TRUE by default to allow editing of column names.
col_readonly names of columns that cannot be edited. Users will be able to edit values but these will be reverted to the original values. Column names for these column cannot be edited either.

row_bind additional rows to add to the data prior to loading into editor, can be either an array containing the new data, a vector containing the new row names for empty rows or a named list containing a vector for each new column.

row_edit logical indicating whether rows can be added or removed, set to TRUE by default.

save_as name of a csv file to which the edited data should be saved.

title optional title to include above the data editor.

logo optional package logo to include in title above the data editor, must be supplied as path to logo png.

logo_size width of the logo in pixels, set to 30 pixels by default.

logo_side can be either "left" or "right" to determine the position of the logo relative to the title, set to "left" by default.

viewer can be either "dialog", "browser" or "pane" to open the application in a dialog box, browser or RStudio viewer pane. First letter abbreviations are allowed, set to "dialog" by default.

viewer_height numeric to control the height of the viewer in pixels when viewer is set to "dialog", set 800 by default.

viewer_width numeric to control the width of the viewer in pixels when viewer is set to "dialog", set to 1200 by default.

theme valid shinytheme name, set to "yeti" by default.

read_fun name of the function to use to read in the data when x is the name of a file, set to read.csv by default.

read_args a named list of additional arguments to pass to read_fun.

write_fun name of the function to use to write the edited version of x to a file, set to write.csv by default. Only requirement is that the first argument accepts the edited data and the second argument accepts the file name supplied to save_as.

write_args a named list of additional arguments to pass to write_fun.

quiet logical indicating whether messages should be suppressed, set to FALSE by default.

hide logical indicating whether the dataInput and dataOutput modules should be visible to the user within the application. If hide = FALSE and save_as is specified, the edited data will be written to file after the application is closed.

code logical indicating whether the code required to generate the edited data should be printed to the console, set to FALSE by default. Alternatively, users can supply the name of an R script to create and store this code.

Value

the edited data as a matrix or data.frame.
Author(s)

Dillon Hammill, <Dillon.Hammill@anu.edu.au>

Examples

```r
if(interactive()) {
  data_edit(mtcars)
}
```
Index

data_edit, 11
dataEdit, 2
dataEditServer (dataEdit), 2
dataEditUI (dataEdit), 2
dataFilter, 4
dataFilterServer (dataFilter), 4
dataFilterUI (dataFilter), 4
dataInput, 5
dataInputServer (dataInput), 5
dataInputUI (dataInput), 5
dataOutput, 7
dataOutputServer (dataOutput), 7
dataOutputUI (dataOutput), 7
dataSelect, 8
dataSelectServer (dataSelect), 8
dataSelectUI (dataSelect), 8
dataSync, 10
dataSyncServer (dataSync), 10
dataSyncUI (dataSync), 10

rhandsontable, 3