Package ‘rhandsontable’

May 27, 2021

Type  Package
Title  Interface to the 'Handsontable.js' Library
Version  0.3.8
Maintainer  Jonathan Owen <jonathanro@gmail.com>
Description  An R interface to the 'Handsontable' JavaScript library, which is a minimalistic Excel-like data grid editor. See <https://handsontable.com/> for details.
License  MIT + file LICENSE
URL  http://jrowen.github.io/rhandsontable/
BugReports  https://github.com/jrowen/rhandsontable/issues
Imports  jsonlite, htmlwidgets (>= 0.3.3), magrittr, methods, utils
Suggests  knitr, rmarkdown, shiny (>= 0.13), miniUI (>= 0.1.1), rstudioapi (>= 0.6), htmltools
VignetteBuilder  knitr
RoxygenNote  7.1.1
Encoding  UTF-8
NeedsCompilation  no
Author  Jonathan Owen [aut, cre, cph],
Jeff Allen [ctb],
Yihui Xie [ctb],
Enzo Martoglio [ctb],
Inberg Ger [ctb],
Warpechowski Marcin [ctb, cph] (Handsontable.js library),
Handsoncode sp. z o.o. [ctb, cph] (Handsontable.js library),
Aisch Gregor [ctb, cph] (Chroma.js library),
Företagsplatsen [ctb, cph] (Numbro.js library),
Draper Adam [ctb, cph] (Numeral.js library),
Wood Tim [ctb, cph] (Moment.js library),
Chernev Iskren [ctb, cph] (Moment.js library),
Moment.js contributors [ctb, cph] (Moment.js library),
Bushell David [ctb, cph] (Pikaday.js library),
jQuery Foundation [ctb, cph] (jQuery.js library),
Description

R interface for creating tables using Handsontable, urlhttps://handsontable.com/

Details

For full documentation on the package, visit https://jrowen.github.io/rhandsontable/
editAddin

### Description

Interactively edit a `data.frame` or `data.table`. The resulting code will be emitted as a call to reload the data from a temp RDS file.

### Usage

```r
editAddin()
```

### Details

This addin can be used to interactively edit. The intended way to use this is as follows:

1. Highlight a symbol naming a `data.frame` or `data.table` in your R session, e.g. `mtcars`
2. Execute this addin, to interactively edit it.

When you’re done, the code performing this operation will be emitted at the cursor position.

This function borrows heavily from `rstudio/addinexamples/subsetAddin`

---

**hot_cell**  
*Handsontable widget*

### Description

Configure single cell. See `Handsontable.js` for details.

### Usage

```r
hot_cell(hot, row, col, comment = NULL, readOnly = NULL)
```

### Arguments

- **hot**: rhandsontable object
- **row**: numeric row index
- **col**: column name or index
- **comment**: character comment to add to cell
- **readOnly**: logical making the cell read-only

### See Also

- `hot_cols`, `hot_rows`
Examples

```r
library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
               small = letters[1:10],
               dt = seq(from = Sys.Date(), by = "days", length.out = 10),
               stringsAsFactors = FALSE)
rhandsontable(DF) %>%
    hot_cell(1, 1, comment = "Test comment") %>%
    hot_cell(2, 3, readOnly = TRUE)
```

---

**hot_col**

**Handsontable widget**

**Description**

Configure single column.

**Usage**

```r
hot_col(
    hot,  # rhandsontable object
    col,  # vector of column names or indices
    type = NULL,  # character specify the data type. Options include: numeric, date, checkbox, select, dropdown, autocomplete, password, and handsontable (not implemented yet)
    format = NULL,
    source = NULL,
    strict = NULL,
    readOnly = NULL,
    validator = NULL,
    allowInvalid = NULL,
    halign = NULL,
    valign = NULL,
    renderer = NULL,
    copyable = NULL,
    dateFormat = NULL,
    default = NULL,
    language = NULL,
    ...
)
```

**Arguments**

- **hot** rhandsontable object
- **col** vector of column names or indices
- **type** character specify the data type. Options include: numeric, date, checkbox, select, dropdown, autocomplete, password, and handsontable (not implemented yet)
format character specifying column format. See Cell Types at Handsontable.js for the formatting options for each data type. Numeric columns are formatted using Numbro.js.

source a vector of choices for select, dropdown and autocomplete column types

strict logical specifying whether values not in the source vector will be accepted

readOnly logical making the column read-only

validator character defining a Javascript function to be used to validate user input. See hot_validate_numeric and hot_validate_character for pre-build validators.

allowInvalid logical specifying whether invalid data will be accepted. Invalid data cells will be color red.

halign character defining the horizontal alignment. Possible values are htLeft, htCenter, htRight and htJustify

valign character defining the vertical alignment. Possible values are htTop, htMiddle, htBottom

renderer character defining a Javascript function to be used to format column cells. Can be used to implement conditional formatting.

copyable logical defining whether data in a cell can be copied using Ctrl + C

dateFormat character defining the date format. See Moment.js for details.

default default column value for new rows (NA if not specified; shiny only)

language locale passed to Numbro.js; default is 'en-US'.

See Also

hot_cols, hot_rows, hot_cell

Examples

library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
small = letters[1:10],
dt = seq(from = Sys.Date(), by = "days", length.out = 10),
stringsAsFactors = FALSE)
rhandsontable(DF, rowHeaders = NULL) %>%
hot_col(col = "big", type = "dropdown", source = LETTERS) %>%
hot_col(col = "small", type = "autocomplete", source = letters,
strict = FALSE)
hot_cols

*Handsontable widget*

Description

Configure multiple columns.

Usage

```r
hot_cols(
  hot,
  colWidths = NULL,
  columnSorting = NULL,
  manualColumnMove = NULL,
  manualColumnResize = NULL,
  fixedColumnsLeft = NULL,
  ...
)
```

Arguments

- `hot`  
  rhandsontable object
- `colWidths`  
  a scalar or numeric vector of column widths
- `columnSorting`  
  logical enabling row sorting. Sorting only alters the table presentation and the original dataset row order is maintained. The sorting will be done when a user click on column name
- `manualColumnMove`  
  logical enabling column drag-and-drop reordering
- `manualColumnResize`  
  logical enabline column width resizing
- `fixedColumnsLeft`  
  a scalar indicating the number of columns to freeze on the left
- `...`  
  passed to hot_col

See Also

- `hot_col`, `hot_rows`, `hot_cell`

Examples

```r
library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
                 small = letters[1:10],
                 dt = seq(from = Sys.Date(), by = "days", length.out = 10),
                 stringsAsFactors = FALSE)
rhandsontable(DF) %&gt;
hot_cols(columnSorting = TRUE)
```
hot_context_menu

Description

Configure the options for the right-click context menu

Usage

hot_context_menu(
  hot,
  allowRowEdit = TRUE,
  allowColEdit = TRUE,
  allowReadOnly = FALSE,
  allowComments = FALSE,
  allowCustomBorders = FALSE,
  customOpts = NULL,
  ...
)

Arguments

- **hot**: rhandsontable object
- **allowRowEdit**: logical enabling row editing
- **allowColEdit**: logical enabling column editing. Note that Handsontable does not support column add/remove when column types are defined (i.e. useTypes == TRUE in rhandsontable).
- **allowReadOnly**: logical enabling read-only toggle
- **allowComments**: logical enabling comments
- **allowCustomBorders**: logical enabling custom borders
- **customOpts**: list
- **...**: ignored

Examples

```
library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
                small = letters[1:10],
                dt = seq(from = Sys.Date(), by = "days", length.out = 10),
                stringsAsFactors = FALSE)
rhandsontable(DF) %>%
  hot_context_menu(allowRowEdit = FALSE, allowColEdit = FALSE)
```
### hot_heatmap

**Handsontable widget**

**Description**
Add heatmap to table.

**Usage**

```r
hot_heatmap(hot, cols, color_scale = c("#ED6D47", "#17F556"), renderer = NULL)
```

**Arguments**

- `hot`: handsontable object
- `cols`: numeric vector of columns to include in the heatmap. If missing all columns are used.
- `color_scale`: character vector that includes the lower and upper colors
- `renderer`: character defining a Javascript function to be used to determine the cell colors. If missing, `rhandsontable:::renderer_heatmap` is used.

**Examples**

```r
MAT = matrix(rnorm(50), nrow = 10, dimnames = list(LETTERS[1:10], letters[1:5]))

rhandsontable(MAT) %>%
  hot_heatmap()
```

---

### hot_row

**Handsontable widget**

**Description**
Configure properties of all cells in a given row(s). Note that `hot_row` is not to be confused with `hot_rows`. See Handsontable.js for details.

**Usage**

```r
hot_row(hot, row, readOnly = NULL)
```

**Arguments**

- `hot`: handsontable object
- `row`: numeric vector of row indexes
- `readOnly`: logical making the row(s) read-only
hot_rows

See Also

hot_cols, hot_cell, hot_rows

Examples

library(rhandsontable)
MAT = matrix(rnorm(50), nrow = 10, dimnames = list(LETTERS[1:10],
letters[1:5]))

rhandsontable(MAT, width = 300, height = 150) %>%
  hot_row(c(1,3:5), readOnly = TRUE)

rhandsontable(MAT, width = 300, height = 150) %>%
  hot_cols(colWidths = 100, fixedColumnsLeft = 1) %>%
  hot_rows(rowHeights = 50, fixedRowsTop = 1)

Description

Configure row settings that pertain to the entire table. Note that hot_rows is not to be confused with
hot_row. See Handsontable.js for details.

Usage

hot_rows(hot, rowHeights = NULL, fixedRowsTop = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot</td>
<td>rhandsontable object</td>
</tr>
<tr>
<td>rowHeights</td>
<td>a scalar or numeric vector of row heights</td>
</tr>
<tr>
<td>fixedRowsTop</td>
<td>a scaler indicating the number of rows to freeze on the top</td>
</tr>
</tbody>
</table>

See Also

hot_cols, hot_cell

Examples

library(rhandsontable)
MAT = matrix(rnorm(50), nrow = 10, dimnames = list(LETTERS[1:10],
letters[1:5]))

rhandsontable(MAT, width = 300, height = 150) %>%
  hot_cols(colWidths = 100, fixedColumnsLeft = 1) %>%
  hot_rows(rowHeights = 50, fixedRowsTop = 1)
hot_table

Handsontable widget

Description

Configure table. See Handsontable.js for details.

Usage

```r
hot_table(
  hot,
  contextMenu = TRUE,
  stretchH = "none",
  customBorders = NULL,
  highlightRow = NULL,
  highlightCol = NULL,
  enableComments = FALSE,
  overflow = NULL,
  rowHeaderWidth = NULL,
  ...
)
```

Arguments

- **hot**: rhandsontable object
- **contextMenu**: logical enabling the right-click menu
- **stretchH**: character describing column stretching. Options are 'all', 'right', and 'none'
- **customBorders**: json object
- **highlightRow**: logical enabling row highlighting for the selected cell
- **highlightCol**: logical enabling column highlighting for the selected cell
- **enableComments**: logical enabling comments in the table
- **overflow**: character setting the css overflow behavior. Options are auto (default), hidden and visible
- **rowHeaderWidth**: numeric width (in px) for the rowHeader column
- **...**: passed to Handsontable.js constructor

See Also

- rhandsontable
Examples

```r
code
library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
                small = letters[1:10],
                dt = seq(from = Sys.Date(), by = "days", length.out = 10),
                stringsAsFactors = FALSE)
rhandsontable(DF) %>%
hot_table(highlightCol = TRUE, highlightRow = TRUE)
```

---

**hot_to_r**  
**Handsonable widget**

**Description**

Convert handsontable data to R object. Can be used in a shiny app to convert the input json to an R dataset.

**Usage**

```r
hot_to_r(...)
```

**Arguments**

```r
...
```

passed to rhandsontable:::toR

**See Also**

`rHandsontableOutput`

---

**hot_validate_character**  
**Handsonable widget**

**Description**

Add numeric validation to a column

**Usage**

```r
hot_validate_character(hot, cols, choices, allowInvalid = FALSE)
```
**Arguments**

- **hot**: rhandsontable object
- **cols**: vector of column names or indices
- **choices**: a vector of acceptable numeric choices. It will be evaluated after min and max if specified.
- **allowInvalid**: logical specifying whether invalid data will be accepted. Invalid data cells will be color red.

**See Also**

- `hot_validate_numeric`

**Examples**

```r
library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
    small = letters[1:10],
    dt = seq(from = Sys.Date(), by = "days", length.out = 10),
    stringsAsFactors = FALSE)
rhandsontable(DF) %>%
    hot_validate_character(col = "big", choices = LETTERS[1:10])
```

---

**Description**

Add numeric validation to a column

**Usage**

```r
hot_validate_numeric(
    hot,
    cols,
    min = NULL,
    max = NULL,
    choices = NULL,
    exclude = NULL,
    allowInvalid = FALSE
)
```
**renderRHandsontable**

**Arguments**

- **hot**: rhandsontable object
- **cols**: vector of column names or indices
- **min**: minimum value to accept
- **max**: maximum value to accept
- **choices**: a vector of acceptable numeric choices. It will be evaluated after min and max if specified.
- **exclude**: a vector of unacceptable numeric values
- **allowInvalid**: logical specifying whether invalid data will be accepted. Invalid data cells will be color red.

**See Also**

- `hot_validate_character`

**Examples**

```r
library(rhandsontable)
MAT = matrix(rnorm(50), nrow = 10, dimnames = list(LETTERS[1:10],
letters[1:5]))

rhandsontable(MAT * 10) %>%
  hot_validate_numeric(col = 1, min = -50, max = 50, exclude = 40)

rhandsontable(MAT * 10) %>%
  hot_validate_numeric(col = 1, choices = c(10, 20, 40))
```

**Description**

Shiny bindings for rhandsontable

**Usage**

```r
renderRHandsontable(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

- **expr**: an expression that generates an rhandsontable.
- **env**: the environment in which to evaluate expr.
- **quoted**: is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

**See Also**

- `rHandsontableOutput`, `hot_to_r`
Handsontable widget

Description

Create a Handsontable.js widget.

Usage

```r
rhandsontable(
  data, colHeaders, rowHeaders, comments = NULL, useTypes = TRUE, readOnly = NULL, selectCallback = FALSE, width = NULL, height = NULL, digits = 4, debug = NULL, search = FALSE,
  ...
)
```

Arguments

- `data`: a `data.table`, `data.frame` or matrix
- `colHeaders`: a vector of column names. If missing `colnames` will be used. Setting to `NULL` will omit.
- `rowHeaders`: a vector of row names. If missing `rownames` will be used. Setting to `NULL` will omit.
- `comments`: matrix or `data.frame` of comments; NA values are ignored
- `useTypes`: logical specifying whether column classes should be mapped to equivalent Javascript types. Note that Handsontable does not support column add/remove when column types are defined (i.e. `useTypes == TRUE` in `rhandsontable`).
- `readOnly`: logical specifying whether the table is editable
- `selectCallback`: logical enabling the afterSelect event to return data. This can be used with shiny to tie updates to a selected table cell.
- `width`: numeric table width
- `height`: numeric table height
- `digits`: numeric passed to `jsonlite::toJSON`
- `debug`: numeric Javascript log level
- `search`: logical specifying if the data can be searched (see [https://jrowen.github.io/rhandsontable/#Customizing](https://jrowen.github.io/rhandsontable/#Customizing) and Shiny example in inst/examples/rhandsontable_search)
- `...`: passed to `hot_table` and to the `params` property of the widget
Details

For full documentation on the package, visit https://jrowen.github.io/rhandsontable/

See Also

hot_table, hot_cols, hot_rows, hot_cell

Examples

library(rhandsontable)
DF = data.frame(val = 1:10, bool = TRUE, big = LETTERS[1:10],
small = letters[1:10],
dt = seq(from = Sys.Date(), by = "days", length.out = 10),
stringsAsFactors = FALSE)
rhandsontable(DF, rowHeaders = NULL)

Description

The following functions are imported and then re-exported from the rhandsontable package to enable use of the magrittr pipe operator with no additional library calls

rHandsontableOutput Handsontable widget

Description

Shiny bindings for rhandsontable

Usage

rHandsontableOutput(outputId, width = "100\%", height = "100\%")

Arguments

outputId output variable to read from
width, height must be a valid CSS unit in pixels or a number, which will be coerced to a string and have "px" appended.

See Also

renderRHandsontable
set_data

Description
Set data inside a Handsontable instance without recreating the widget. Send the new values as a vector of rows, a vector of columns, and a vector of values. If different length vectors are supplied then the shorter ones are recycled to match the length of the longest.

Usage

set_data(id, row, col, val, session, zero_indexed = F)

Arguments

<table>
<thead>
<tr>
<th>id</th>
<th>The id of the table to interact with.</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Integer vector of row indexes.</td>
</tr>
<tr>
<td>col</td>
<td>Integer vector the column indexes.</td>
</tr>
<tr>
<td>val</td>
<td>Vector of values to set at each row-col pair.</td>
</tr>
<tr>
<td>session</td>
<td>The session that is associated with your shiny server function. The table is only interactive when used in shiny so we only use set_data when the table is in shiny.</td>
</tr>
<tr>
<td>zero_indexed</td>
<td>Default FALSE. Set to TRUE if you are supplying row and col indexes that are already 0-based.</td>
</tr>
</tbody>
</table>
Index

%>% (rhandsontable-exports), 15
editAddin, 3
hot_cell, 3, 5, 6, 9, 15
hot_col, 4, 6
hot_cols, 3, 5, 6, 9, 15
hot_context_menu, 7
hot_heatmap, 8
hot_row, 8, 9
hot_rows, 3, 5, 6, 8, 9, 9, 15
hot_table, 10, 15
hot_to_r, 11, 13
hot_validate_character, 11, 13
hot_validate_numeric, 12, 12
renderRHandsontable, 13, 15
rhandsontable, 10, 14
rhandsontable-exports, 15
rhandsontable-package, 2
rHandsontableOutput, 11, 13, 15
set_data, 16