Package ‘rite’

August 29, 2016

Version 0.3.4

Date 2014-08-16

Title The Right Editor to Write R

Author Thomas J. Leeper

Maintainer Thomas J. Leeper <thosjleeper@gmail.com>

Imports tcltk, tcltk2, RCurl, tools, knitr, markdown

Description A simple yet powerful script editor built natively in R with tcltk.

License GPL-2

URL https://github.com/leeper/rite

BugReports https://github.com/leeper/rite/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-19 10:31:53

R topics documented:

`rite-package` *The Right Editor to Write R*

Description

A simple, but powerful script editor for R, built with tcl/tk. `rite` helps ease new R users' transition to R, a major rite of passage.

`rite` is designed to substitute for the built-in script editor provided by R and improve upon it by offering functionality more commonly found in standalone editors and IDEs (e.g., syntax highlighting, command completion, shortcut keys, find and go-to-line commands, direct acces R help files, etc.), as well as an optional output viewer that eliminates the need to toggle between the script editor and the R console (and that additionally facilitates integration with `knitr`.

Author(s)

Thomas J. Leeper

Maintainer: Thomas J. Leeper <thosjleeper@gmail.com>

`rite` *rite*

Description

Open rite

Usage

```
rite(filename = NULL, catchOutput = FALSE, evalenv = .GlobalEnv,
     fontFamily = "Courier", fontSize = 10, orientation = "horizontal",
     fastinsert = FALSE, highlight = "r", color = NULL, autosave = TRUE,
     echo = TRUE, tab = '    ', comment = '#', url = NULL, ...)

riteout(...)
```

Arguments

`filename`	Optionally, a character string specifying a file name in the working directory (or file path) to open in rite.
`catchOutput`	A logical specifying whether output and event handling (errors, warnings, messages, interrupts) should be sent to the rite output viewer panel rather than the R console. Default is `FALSE`.
`evalenv`	The environment in which the script should be evaluated, e.g., `.GlobalEnv`. If `NULL`, the default is a hidden environment, internal to `rite` meaning that, e.g., variables created/modified in `rite` are not accessible via the console.
`fontFamily`	The font family used in rite. Default is "`Courier`". Available fonts can be retrieved by `.Tcl("font families")`.
`fontSize`	The font size used in rite. Default is `10`.

`orientation`	If `catchOutput=TRUE`, whether the output and error panels should be oriented “horizontal” (to the right) or “vertical” (below) the script editing panel. Default is “horizontal”.
`fastinsert`	A logical, specifying whether insertion (from opening a script, appending a script, or pasting a script) should be done “fast” (i.e., without running syntax highlighting. This can significantly improve load times but has the consequence of leaving added text unhighlighted unless modified. Default is `FALSE`.
`highlight`	A character vector containing one or more of “r”, “latex”, “markdown”, “xml”, “roxygen”, “brew”, and “rest” to indicate what should be higlighted in script. Default is “r”. Note that using more than a few of these simultaneously may cause unexpected highlighting.
`color`	Either `NULL`, or a named list specifying Tcl/Tk colors for highlighting. See details below.
`autosave`	A logical specifying whether the script should automatically be saved whenever any of the script is run. Default is `TRUE`. Note: if `filename` is `NULL`, the result is saved in the current R session’s temporary directory and will be deleted when the R session terminates normally.
`echo`	Whether the R calls should be copied to output during `source` (only when `catchOutput=FALSE`). Default is `TRUE`.
`tab`	A character string indicating what text should be inserted when the `<TAB>` key is pressed. Default is the `\t` character. Alternatively, a numeric value (e.g., `4`) is treated as a number of spaces.
`comment`	A character string indicating what text should be used for commenting. Default is the hash/pound character.
`url`	If `filename=NULL`, `url` is considered as the URL for a remote script to load.
`...`	Ignored by `rite`. Used by `riteout` to pass arguments to `rite`.

Details

Create, edit, and save R scripts (or any text-based file) and, optionally, sink output to an output viewer rather than the R console.

Scripts can be loaded, appended, referenced (via `source`) in the current script, and can be saved. As of rite version 0.3, scripts can also be saved to and loaded from anonymous GitHub gists to facilitate code sharing.

Scripts can be run by selection, line, or the entire contents of the script editor from the “Run” menu. Scripts can also be run from the context menu (via a right mouse click) or with `<Control-r>` or `<Control-Return>` to run a selection or the current line. `<F8>` runs the entire script, whereas `<F7>` checks for parsing errors in the entire script without evaluating it (all code is parsed before running, automatically).

Command completion is now fully supported by pressing `<F2>` (a right mouse button click on the menu will close it if no selection is desired). If a function is complete and followed by an open parenetheses (e.g., `data.frame(`, the command completion keys bring up a selectable list named arguments for the function. If a dataframe or environment name is complete and followed by a dollar sign (e.g., `mydf$`), the command completion keys bring up a list of named elements in the dataframe or environment. `<F2>` also opens code chunk arguments after code chunk openings, argument formals after function names, and working directory files and directories after open quotes.

`riteout` is a wrapper for `rite` that defaults to `catchOutput=TRUE`, which provides access to various report generation tools, described in detail below.

Syntax highlighting

rite supports syntax highlighting for R (by default) and, optionally, a number of other R-related languages LaTeX, R-flavored markdown, XML and HTML, roxygen, brew, and restructured text (reST).

Functions and other objects from loaded packages are highlighted by default on-the-fly.

By default, the following colors are used for syntax highlighting: `Normal text (normal)`: "black" `Editor background (background)`: "white" `R functions (functions)`: "purple" (this applies to both base functions and those from packages loaded within `rite`) `R comments (rcomments)`: "darkgreen" `Operators (operators)`: "blue" `Brackets (brackets)`: "darkblue" `Digits (digits)`: "orange" `Character strings (characters)`: "darkgray" `LaTeX macros (latexmacros)`: "darkred" `LaTeX equations (latexequations)`: "black" `LaTeX comments (latexcomments)`: "red" `Sweave/knitr code chunks (rnwchunks)`: "blue" `Rtex code chunks (rtexchunks)`: "blue" `Markdown (rmd)`: "darkred" `Markdown code chunks (rmdchunks)`: "blue" `XML/HTML tags (xml)`: "darkred" `XML/HTML comments (xmlcomments)`: "red" `Roxygen text (roxygentext)`: "black" `Roxygen code chunks (roxygenchunks)`: "blue" `Brew comments (brewcomments)`: "red" `Brew chunks (brewchunks)`: "blue" `Brew templates (brewtemplate)`: "black" `reST chunks (restchunks)`: "blue"

The use of highlighting in general can be regulated by the `highlight` parameter. To specify alternative specific colors for any of the above highlighting rules, the `color` parameter accepts a named list of content types (listed in parentheses above) and their corresponding colors (in quotes). For example, calling `rite(color=list(rcomments='pink'))`, would open `rite` with R comments highlighted in pink but leave all other highlighting rules at their default settings.

Shortcut keys in widget

Key combinations:

`<Ctrl-o>`: Open script

`<Ctrl-s>`: Save script as

`<Ctrl-r>`: Run/evaluate line (or selection, if applicable)

`<Control-Return>`: Run/evaluate line (or selection, if applicable)

`<Ctrl-c>`: Copy

`<Ctrl-x>`: Cut

`<Ctrl-p>`: Paste

`<Ctrl-a>`: Select all

`<Ctrl-e>`: Move cursor to end of current line

`<Shift-e>`: Adjust selection to end of current line

`<Ctrl-f>`: Find/replace

`<Ctrl-g>`: Go to line

`<Ctrl-z>`: Undo

`<Ctrl-y>`: Redo

`<Tab>`: Indent line

`<Ctrl-i>`: Indent line(s)

`<Ctrl-u>`: Unindent line

`<Ctrl-k>`: Toggle comment on/off for line(s)

`<Ctrl-l>`: Clear output panel

`<Shift>` and `<Left>`, `<Right>`, or drag left mouse: Adjust selection by character

`<Control-Shift-Left>` or `<Control-Shift-Right>`: Adjust selection by word

`<Control-Up>` or `<Control-Down>`: Move insertion cursor by paragraphs

`<Control-Shift-Up>` or `<Control-Shift-Down>`: Adjust selection by paragraph

Function keys:

`<F1>`: Open help for current function, if a known function (based on cursor position); or the results of help.search()

`<F2>`: Open command completion context menu (based on cursor position). A right-button mouse click closes the context menu if no selection is desired (not necessary on all platforms).

`<F3>`: Find

`<F7>`: Try to parse the script and return any syntax errors (does not evaluate the script)

`<F8>`: Run/evaluate all code

Mouse shortcuts:

Left mouse click (1 time): Move cursor

Left mouse click (2 times): Select word

Left mouse click (3 times): Select line

Right mouse click (1 time): Open context menu

Report generation with knitr

If `catchOutput=TRUE` (or rite is loaded with `riteout`), `rite` provides a number of report generation capabilities provided by `knitr`. Specifically, a "Report Generation" menu becomes available that includes the following options. By default, the available tools generate reports from the currently open script in rite, but all the tools can also be run on local files (which opens those files into rite). The general pattern of report generation behavior is to load the input file, display the output file in the `riteout` output tab, display any relevant processing information in the `riteout` message tab, and open the resulting output file (in the case of functions that produce PDFs or HTML files). The resulting output files can be saved using "Save Output" from the "Output" menu.

The following tools are available:

`knit`: Runs `knit` on the contents of the script panel and returns them in the output panel. Optionally, an R markdown (Rmd) file can be converted directly to HTML and opened. And, optionally, an Rnw file (for knitr or Sweave) can be converted directly to PDF and opened. Support for converting Sweave documents to knitr format can prevent compatibility issues between the two formats.

`purl`: Runs `purl` on the contents of the script panel and returns them in the output panel. This is the knitr equivalent of Stangle, producing code-only output from the original document. Again, optionally, an Rnw file (for knitr or Sweave) can be converted directly to PDF and opened. Support for converting Sweave documents to knitr format can prevent compatibility issues between the two formats.

`stitch`: Runs `stitch` on the contents of the script panel, embedding it into an LaTeX file (and opens the resulting PDF), markdown file (and opens the resulting HTML), or HTML file (and opens the resulting HTML), based on a simple template. This is helpful for converting an unformatted R script into a simple report.

`spin`: Runs `spin` on the contents of the script panel, which should be a specially formatted roxygen-style script. The output is a knitr file, which can optionally be `knit` or simply displayed into the output tab.

`markdown`: Convert an R markdown document to either a full HTML document or an HTML fragment (without `header` and `body` tags) to embed, e.g., in a blog post.

`LaTeX` and `XeLaTeX`: Run LaTeX or XeLaTeX and, optionally, BibTex as a system call. knitr compiles LaTeX to PDF via DVI using the functionality supplied by the `tools` package, so these report generation tools provide functionality for .tex scripts that are not compatible with DVI.

Author(s)

Thomas J. Leeper

Examples

```
## Not run:
# run the simple script editor
rite()

## End(Not run)

## Not run:
# run the script editor with output tools
riteout()

## End(Not run)
```

`ritesink` *ritesink*

Description

An experimental tcl/tk output widget

Usage

```
sinkstart(echo = TRUE, split = FALSE,
          fontFamily = 'Courier', fontSize = 10,
          col.bg = 'white', col.call = c('black',col.bg),
          col.result = c('black',col.bg), col.err = c('red',col.bg),
          col.warn = c('purple',col.bg), col.msg = c('blue',col.bg))

sinkstop(quiet = TRUE)
```

Arguments

`echo`	A logical indicating whether calls should be output to the sink. Default is `TRUE`.
`split`	A logical indicating whether output (but not messages) should be split between the console and the sink. Default is `FALSE`.
`fontFamily`	The font family used in rite. Default is “`Courier`”. Available fonts can be retrieved by `.Tcl("font families")`.
`fontSize`	The font size used in rite. Default is `10`.
`col.bg`	A one-element character string indicating a tcl/tk color for the background color of the sink. A list of available tcl/tk colors can be found here: `http://www.tcl.tk/man/tcl8.5/TkCmd/colors.htm`.
`col.call`	A two-element character vector indicating tcl/tk colors for the foreground and background, respectively, of evaluated R calls (only visible if `echo=TRUE`..
`col.result`	A two-element character vector indicating tcl/tk colors for the foreground and background, respectively, of standard output.
`col.err`	A two-element character vector indicating tcl/tk colors for the foreground and background, respectively, of errors.
`col.warn`	A two-element character vector indicating tcl/tk colors for the foreground and background, respectively, of warnings.
`col.msg`	A two-element character vector indicating tcl/tk colors for the foreground and background, respectively, of messages.
`quiet`	A logical indicating whether to suppress confirmation that everything is cleaned up. Default is `TRUE`.

Details

These functions make use of a couple of different R features to build a color-coded output window for the R console. While the console is limited to displaying output in monochrome plain text, the rite sink allows multi-colored output and messages to be piped to a single widget that highlights errors, warning, and messages. Unlike a traditional sink, rite sink is a tcl/tk widget that updates output as commands and messages occur. Accomplishing this requires the use of `sink`, task callbacks, and a custom error handler. (This is similar to the R2HTML package.)

`sinkstart` starts the sink and `sinkstop` stops the sink without destroying the widget. Closing the sink widget invisibly calls `sinkstop` and cleans up. The sink can be turned on and off repeatedly without closing the widget.

Value

NULL

Shortcut keys in widget

`<Ctrl-c>`: Copy

`<Ctrl-x>`: Cut

`<Ctrl-p>`: Paste

`<Ctrl-a>`: Select all

`<Ctrl-s>`: Save output

`<Ctrl-l>`: Clear output

Author(s)

Thomas J. Leeper

References

Top-level Task Callbacks in R

R2HTML package

Examples

```
## Not run:
sinkstart() # open the sink
sinkstop() # close the sink

## End(Not run)
```

Index