Package ‘clipr’

January 11, 2019

Type Package

Title Read and Write from the System Clipboard

Version 0.5.0

Description Simple utility functions to read from and write to the Windows, OS X, and X11 clipboards.

License GPL-3

URL https://github.com/mdlincoln/clipr

BugReports https://github.com/mdlincoln/clipr/issues

Imports utils

Suggests covr, rstudioapi (>= 0.5), testthat (>= 2.0.0)

Encoding UTF-8

Language en-US

LazyData TRUE

RoxygenNote 6.1.1

SystemRequirements xclip (https://github.com/astrand/xclip) or xsel (http://www.vergenet.net/~conrad/software/xsel/) for accessing the X11 clipboard

NeedsCompilation no

Author Matthew Lincoln [aut, cre] (<https://orcid.org/0000-0002-4387-3384>), Louis Maddox [ctb], Steve Simpson [ctb], Jennifer Bryan [ctb]

Maintainer Matthew Lincoln <matthew.d.lincoln@gmail.com>

Repository CRAN

Date/Publication 2019-01-11 14:50:02 UTC
\section*{R topics documented:}

- \texttt{clear\_clip} .................................................. 2
- \texttt{clipr} .......................................................... 2
- \texttt{clipr\_available} ........................................ 3
- \texttt{read\_clip} .................................................. 4
- \texttt{read\_clip\_tbl} .............................................. 4
- \texttt{write\_clip} .................................................. 5

\section*{Index}

\begin{tabular}{ll}
\texttt{clear\_clip} & \emph{Clear clipboard} \\
\end{tabular}

\section*{Description}

Clear the system clipboard.

\section*{Usage}

\begin{verbatim}
clear\_clip(...) 
\end{verbatim}

\section*{Arguments}

\begin{verbatim}
  ...  
\end{verbatim}

Pass other options to \texttt{write\_clip}.

\section*{Note}

This is a simple wrapper function for \texttt{write\_clip("")}

\section*{clipr \texttt{clipr: Read and Write from the System Clipboard}}

\section*{Description}

Simple utility functions to read from and write to the Windows, OS X, and X11 clipboards.

\section*{Details}

The basic functions \texttt{read\_clip} and \texttt{write\_clip} wrap platform-specific functions for writing values from R to the system clipboard. \texttt{read\_clip\_tbl} will attempt to process the clipboard content like a table copied from a spreadsheet program.\texttt{clipr\_available} is useful when building packages that depend on \texttt{clipr} functionality.
Is the system clipboard available?

Description

Checks to see if the system clipboard is write-able/read-able. This may be useful if you are developing a package that relies on clipr and need to ensure that it will skip tests on machines (e.g. CRAN, Travis) where the system clipboard may not be available.

Usage

clipr_available(...)

dr_clipr(...)  

Arguments

...  

Pass other options to write_clip. Generally only used to pass the argument allow_non_interactive_use = TRUE.

Details

If you are trying to call this in a non-interactive session, be sure to call using clipr_available(allow_non_interactive = TRUE) or by setting the environment variable CLIPBOARD_AVAILABLE=TRUE

Value

clipr_available returns a boolean value.

dr_clipr prints an informative message to the console with software and system configuration requirements if clipr is not available (invisibly returns the same string)

Examples

## Not run:
# When using testthat:
library(testthat)
skip_if_not(clipr_available())

## End(Not run)
read_clip

Read clipboard

Description

Read the contents of the system clipboard into a character vector.

Usage

read_clip()

Value

A character vector with the contents of the clipboard. If the system clipboard is empty, returns NULL.

Note

read_clip will not try to guess at how to parse copied text. If you are copying tabular data, it is suggested that you use read_clip_tbl.

Examples

```r
## Not run:
clip_text <- read_clip()

## End(Not run)
```

read_clip_tbl

Transforms output of read_clip into data frame.

Description

Transforms clipped content into a data frame by putting read_clip output by using read.table.

Usage

read_clip_tbl(x = read_clip(), ...)

write_clip

Arguments

x
- Defaults to reading from the clipboard, but can be substituted by a character vector already generated by `read_clip`.

... Options to pass to `read.table`. The following `read.table` arguments will be passed by default, but can be overridden by specifying them when calling `read_clip_tbl`:
  - header: TRUE
  - sep: "\t"
  - stringsAsFactors: FALSE
  - na.strings: c("NA", "")
  - strip.white: TRUE

Value

A data frame with the contents of the clipboard. If the system clipboard is empty, returns NULL

write_clip  Write clipboard

Description

Write a character vector to the system clipboard

Usage

`write_clip(content, object_type = c("auto", "character", "table"), breaks = NULL, eos = NULL, return_new = TRUE, allow_non_interactive = Sys.getenv("CLIPR_ALLOW", interactive()), ...)`

Arguments

content An object to be written to the system clipboard.

object_type  write_clip() tries to be smart about writing objects in a useful manner. If passed a data.frame or matrix, it will format it using `write.table` for pasting into an external spreadsheet program. It will otherwise coerce the object to a character vector. auto will check the object type, otherwise table or character can be explicitly specified.

breaks The separator to be used between each element of the character vector being written. NULL defaults to writing system-specific line breaks between each element of a character vector, or each row of a table.

eos The terminator to be written after each string, followed by an ASCII null. Defaults to no terminator character, indicated by NULL.

return_new If true, returns the rendered string; if false, returns the original object
allow_non_interactive

By default, clipr will throw an error if run in a non-interactive session. Set the environment variable CLIPR_ALLOW=TRUE in order to override this behavior, however see the advisory below before doing so.

... Custom options to be passed to write.table (if x is a table-like). Defaults to sane line-break and tab standards based on the operating system. By default colnames = FALSE, however you may override this by passing colnames = TRUE.

Value

Invisibly returns the original object

Note

On X11 systems, write_clip will cause either xclip (preferred) or xsel to be called. Be aware that, by design, these processes will fork into the background. They will run until the next paste event, when they will then exit silently. (See the man pages for xclip and xsel for more on their behaviors.) However, this means that even if you terminate your R session after running write_clip, those processes will continue until you access the clipboard via another program. This may be expected behavior for interactive use, but is generally undesirable for non-interactive use. For this reason you must not run write_clip on CRAN, as the nature of xsel has caused issues in the past.

Call clipr_available to safely check whether the clipboard is readable and writable.

Examples

```r
## Not run:
text <- "Write to clipboard"
write_clip(text)

multiline <- c("Write", "to", "clipboard")
write_clip(multiline)
# Write
to
# clipboard

write_clip(multiline, breaks = ",")
# write,to,clipboard

tbl <- data.frame(a=c(1,2,3), b=c(4,5,6))
write_clip(tbl)

## End(Not run)
```
Index

clear_clip, 2
clipr, 2, 2.3
clipr-package (clipr), 2
clipr_available, 2, 3, 6

dr_clipr (clipr_available), 3
read.table, 4, 5
read_clip, 2, 4, 4, 5
read_clip_tbl, 2, 4, 4

write.table, 5, 6
write_clip, 2, 3, 5