Package ‘clipr’

June 23, 2018

Type Package
Title Read and Write from the System Clipboard
Version 0.4.1
Description Simple utility functions to read from and write to the Windows, OS X, and X11 clipboards.
Imports utils
SystemRequirements xclip (https://github.com/astrand/xclip) or xsel (http://www.vergenet.net/~conrad/software/xsel/) for accessing the X11 clipboard
License GPL-3
LazyData TRUE
Suggests rstudioapi (>= 0.5), testthat, covr
URL https://github.com/mdlincoln/clipr
BugReports https://github.com/mdlincoln/clipr/issues
RoxygenNote 6.0.1
NeedsCompilation no
Author Matthew Lincoln [aut, cre], Louis Maddox [ctb], Steve Simpson [ctb], Jennifer Bryan [ctb]
Maintainer Matthew Lincoln <matthew.d.lincoln@gmail.com>
Repository CRAN
Date/Publication 2018-06-23 19:47:55 UTC

R topics documented:
clear_clip ................................................................. 2
clipr ............................................................... 2
clipr_available .......................................................... 2
read_clip ............................................................. 3
read_clip_tbl ........................................................... 4
write_clip ............................................................ 4
Index

| clear_clip | Clear clipboard |

Description

Clear the system clipboard.

Usage

clear_clip()

Note

This is a simple wrapper function for write_clip(""")

---

clipr

clipr: Read and Write from the System Clipboard

Description

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

Details

The basic functions read_clip and write_clip wrap platform-specific functions for writing values from R to the system clipboard. read_clip_tbl will attempt to process the clipboard content like a table copied from a spreadsheet program. clipr_available is useful when building packages that depend on clipr functionality.

---

clipr_available

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()
**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**

```r
## 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**

```r
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(), ...)`

**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, ...)
```
write_clip

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 ele-
ment of a character vector, or each row of a table.

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

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

...  Custom options to be passed to write.table (if the object is a table-like) De-
faults to sane line-break and tab standards based on the operating system.

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 until the next paste event, when they will
then exit silently.

Examples

## 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
clipr-package (clipr), 2
clipr_available, 2

dr_clipr (clipr_available), 2

read.table, 4
read_clip, 2, 3, 4
read_clip_tbl, 2, 3, 4

write.table, 5
write_clip, 2, 4