Package ‘fritools’

November 15, 2021

Title Utilities for the Forest Research Institute of the State Baden-Wuerttemberg

Version 2.4.0

Description Miscellaneous utilities, tools and helper functions for finding and searching files on disk, searching for and removing R objects from the workspace.
These are utilities for packages
<https://CRAN.R-project.org/package=cleanr>,
<https://CRAN.R-project.org/package=document>,
<https://CRAN.R-project.org/package=fakemake>,
<https://CRAN.R-project.org/package=packager> and
<https://CRAN.R-project.org/package=rasciidoc>.
Does not import or depend on any third party package, but on core R only (i.e. it may depend on packages with priority 'base').

License BSD_2_clause + file LICENSE

URL https://gitlab.com/fvafrcu/fritools

Depends R (>= 3.3.0)

Imports methods, stats, utils

Suggests callr, checkmate, desc, devtools, digest, knitr, packager (>= 1.9.0), pkglint, reshape, rmarkdown, RUnit, testthat (>= 3.0.0), tinytest, whoami

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

Config/testthat/edition 3

NeedsCompilation no

Author Andreas Dominik Cullmann [aut, cre]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository CRAN

Date/Publication 2021-11-15 12:50:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fritools-package</td>
<td>3</td>
</tr>
<tr>
<td>bulk_read_csv</td>
<td>3</td>
</tr>
<tr>
<td>bulk_write_csv</td>
<td>4</td>
</tr>
<tr>
<td>call_conditionally</td>
<td>5</td>
</tr>
<tr>
<td>call_safe</td>
<td>6</td>
</tr>
<tr>
<td>compare_vectors</td>
<td>7</td>
</tr>
<tr>
<td>convert_umlauts_to_ascii</td>
<td>8</td>
</tr>
<tr>
<td>convert_umlauts_to_tex</td>
<td>9</td>
</tr>
<tr>
<td>csv</td>
<td>9</td>
</tr>
<tr>
<td>csv2csv</td>
<td>10</td>
</tr>
<tr>
<td>file_modified_last</td>
<td>11</td>
</tr>
<tr>
<td>find_files</td>
<td>12</td>
</tr>
<tr>
<td>get_boolean_envvar</td>
<td>14</td>
</tr>
<tr>
<td>get_options</td>
<td>15</td>
</tr>
<tr>
<td>get_package_version</td>
<td>16</td>
</tr>
<tr>
<td>get_rscript_script_path</td>
<td>17</td>
</tr>
<tr>
<td>get_run_r_tests</td>
<td>17</td>
</tr>
<tr>
<td>get_r_cmd_batch_script_path</td>
<td>18</td>
</tr>
<tr>
<td>get_script_name</td>
<td>19</td>
</tr>
<tr>
<td>get_script_path</td>
<td>19</td>
</tr>
<tr>
<td>get_unique_string</td>
<td>20</td>
</tr>
<tr>
<td>golden_ratio</td>
<td>21</td>
</tr>
<tr>
<td>index_groups</td>
<td>21</td>
</tr>
<tr>
<td>is_batch</td>
<td>22</td>
</tr>
<tr>
<td>is_cran</td>
<td>22</td>
</tr>
<tr>
<td>is_difftime_less</td>
<td>24</td>
</tr>
<tr>
<td>is_false</td>
<td>25</td>
</tr>
<tr>
<td>is_files_current</td>
<td>26</td>
</tr>
<tr>
<td>is_force</td>
<td>27</td>
</tr>
<tr>
<td>is_installed</td>
<td>28</td>
</tr>
<tr>
<td>is_not_false</td>
<td>28</td>
</tr>
<tr>
<td>is_null_or_true</td>
<td>30</td>
</tr>
<tr>
<td>is_of_length_zero</td>
<td>30</td>
</tr>
<tr>
<td>is_running_on_fvafrcu_machines</td>
<td>31</td>
</tr>
<tr>
<td>is_running_on_gitlab_com</td>
<td>32</td>
</tr>
<tr>
<td>is_r_cmd_check</td>
<td>33</td>
</tr>
<tr>
<td>is_r_package_installed</td>
<td>33</td>
</tr>
<tr>
<td>is_success</td>
<td>34</td>
</tr>
<tr>
<td>is_valid_primary_key</td>
<td>35</td>
</tr>
<tr>
<td>is_version_sufficient</td>
<td>36</td>
</tr>
<tr>
<td>is_windows</td>
<td>37</td>
</tr>
<tr>
<td>load_internal_functions</td>
<td>37</td>
</tr>
<tr>
<td>memory_hogs</td>
<td>38</td>
</tr>
<tr>
<td>paths</td>
<td>39</td>
</tr>
<tr>
<td>run_r_tests_for_known_hosts</td>
<td>40</td>
</tr>
<tr>
<td>search_files</td>
<td>41</td>
</tr>
</tbody>
</table>
Description

Miscellaneous utilities, tools and helper functions.

Details

You will find the details in
vignette("An_Introduction_to_fritools",package = "fritools").

Usage

bulk_read_csv(paths, pattern = ".*\.csv$", is_latin1 = TRUE, ...)

bulk_read_csv

Bulk Read Comma Separated Files

Description

Import a bunch of comma separated files or all comma separated files below a directory using read_csv.
bulk_write_csv

Bulk Write Comma Separated Files

Description

Write a bunch of objects to disk using write_csv.

Usage

bulk_write_csv(x, ...)
call_conditionally

Arguments

- **x**: A list of objects to be written to csv.
- **...**: Arguments passed to `write_csv`.

Value

The list holding the return values of `write_csv`.

See Also

Other CSV functions: `bulk_read_csv()`, `csv2csv()`, `csv`

Examples

```r
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda Rö4"
names(mt_german)[1] <- "mg\u00dc"
for (i in 1:10) {
  f <- file.path(tempdir(), paste0("f", i, ".csv"))
  write.csv(mtcars[1:5, TRUE], file = f)
  f <- file.path(tempdir(), paste0("f", i, ".german.csv"))
  write.csv2(mt_german[1:7, TRUE], file = f, fileEncoding = "Latin1")
}
#% read
bulk <- bulk_read_csv(tempdir())

print(mtime <- file.info(list.files(tempdir(), full.names = TRUE))[, "mtime"])
bulk[,"f2"][3, 5] <- bulk[,"f2"][3, 5] + 2
Sys.sleep(2) # make sure the mtimes would change
result <- bulk_write_csv(bulk)
print(new_times <- file.info(dir(tempdir(), full.names = TRUE))[, "mtime"])
index_change <- grep("f2\.csv", rownames(mtime))
if (requireNamespace("digest", quietly = TRUE)) {
  only_f2_changed <- all((mtime == new_times)[-c(index_change)]) &&
  (mtime < new_times)[c(index_change)]
  RUnit::checkTrue(only_f2_changed)
} else {
  RUnit::checkTrue(all(mtime < new_times))
}
```

---

call_conditionally  
Call a Function Conditionally

Description

`whoami` 1.3.0 uses things like `system("getent passwd $(whoami)", intern = TRUE)` which I cannot tryCatch, as it gives no error nor warning. So this function returns a fallback if the condition given is not `TRUE`. 
**Usage**

call_conditionally(f, condition, fallback, ..., harden = FALSE)

**Arguments**

- **f**: The function passed to `do.call`.
- **condition**: An expression.
- **fallback**: See Description.
- **...**: arguments passed to `do.call`.
- **harden**: Set to TRUE to return fallback if `do.call` fails.

**Value**

The return value of `f` or `fallback`.

**See Also**

Other call functions.: `call_safe()`

**Examples**

call_conditionally(get_package_version,  
  condition = TRUE,  
  args = list(x = "fritools"),  
  fallback = "0.0")

call_conditionally(get_package_version,  
  condition = FALSE,  
  args = list(x = "fritools"),  
  fallback = "0.0")

call_conditionally(get_package_version,  
  condition = TRUE,  
  args = list(x = "not_there"),  
  harden = TRUE,  
  fallback = "0.0")

---

**Description**

Just a specialized version of `call_conditionally`.

**Usage**

call_safe(f, dependency, fallback = "Fallback", ...)

---

**call_safe**

*Call a Function Given an External Dependency on Non-Windows Systems*

**Usage**

`call_safe(f, dependency, fallback = "Fallback", ...)"
**compare_vectors**

**Arguments**

- **f**
  - The function passed to `do.call`.
- **dependency**
  - The external dependency, see **Examples**.
- **fallback**
  - See **Description**.
- ... arguments passed to `do.call`.

**Value**

The return value of `f` or `fallback`.

**See Also**

Other call functions: `call_conditionally()`

**Examples**

```r
call_safe(whoami::email_address, dependency = "whoami",
  args = list(fallback = "foobar@nowhere.com"),
  fallback = "nobar@nowhere.com")
call_safe(whoami::email_address, dependency = "this_is_not_installed",
  args = list(fallback = "foobar@nowhere.com"),
  fallback = "nobar@nowhere.com")
```

---

**compare_vectors**  
*Compare Two Vectors*

**Description**

Side-by-side comparison of two vectors. The vectors get sorted and are compared element-wise. So the result will be as long as the union of the two vectors plus their number of values unique to one of them.

**Usage**

```r
compare_vectors(x, y)
```

**Arguments**

- **x, y**
  - Two vectors of the same mode.

**Value**

A matrix containing the side-by-side comparison.

**See Also**

Other searching functions: `file_modified_last()`, `find_files()`, `search_files()`, `search_rows()`, `summary.filesearch()`
Examples

```r
data(mtcars)
cars <- rownames(mtcars)
carz <- cars[-grep("Merc", cars)]
cars <- cars[nchar(cars) < 15]
cars <- c(cars, "foobar")
compare_vectors(cars, carz)
```

```
convert_umlauts_to_ascii

Convert German umlauts to a more or less suitable ascii representation.
```

Description

Convert German umlauts to a more or less suitable ascii representation.

Usage

```r
convert_umlauts_to_ascii(x)
```

## S3 method for class 'character'
```
convert_umlauts_to_ascii(x)
```

## S3 method for class 'data.frame'
```
convert_umlauts_to_ascii(x)
```

Arguments

- `x` A string or data.frame.

Value

x with the umlauts converted to ascii.

See Also

Other German umlaut converters: `convert_umlauts_to_tex()`

Examples

```r
string <- paste("this is \u00e4 string")
print(string)
print(convert_umlauts_to_ascii(string))
string <- paste("this is \u00e4 string")
df <- data.frame(v1 = c(string, "foobar"),
    v2 = c("foobar", string), v3 = 3:4)
names(df)[3] <- "y\u00dffy"
convert_umlauts_to_ascii(df)
```
**convet_umlauts_to_tex**

*Tex Codes for German Umlauts*

**Description**

Convert German umlauts in a string to their plain TeX representation.

**Usage**

```r
clean_umlauts_to_tex(x)
```

**Arguments**

- `x` A string.

**Value**

A string with the umlauts converted to plain TeX.

**See Also**

Other German umlaut converters: `convert_umlauts_to_ascii()`

**Examples**

```r
string <- paste("this is \u00e4 string")
print(string)
print(convert_umlauts_to_tex(string))
```

---

**csv**

*Read and Write a Comma Separated File*

**Description**

Functions to read and write CSV files. The objects returned by these functions are `data.frames` with the following attributes:

- `path` The path to the file on disk.
- `csv` The type of CSV: either `standard` or `german`.
- `hash` The hash value computed with `digest`'s digest function, if `digest` is installed.

`read_csv` is a wrapper to determine whether to use `utils:read.csv2` or `utils:read.csv2`. It sets the above three arguments.

`write_csv` compares the hash value stored in the object's attribute with the object's current hash value. If they differ, it writes the object to the `file` argument or, if not given, to the `path` stored in the object's attribute. If no `csv_type` is given, it uses the `csv` type stored in object's attribute. If `digest` is not installed, the object will (unconditionally) be written to disk.
Usage

read_csv(file, ...)

write_csv(x, file = NULL, csv_type = c(NA, "standard", "german"))

Arguments

file The path to the file to be read or written.

... Arguments passed to utils::read.csv or utils::read.csv2.

x The object to write to disk.

csv_type Which csv type is to be used. If NA, the csv attribute is read from the object.

Value

For read_csv: An object read from the file.

For write_csv: The object with updated hash (and possibly path and csv) attribute.

See Also

Other CSV functions: bulk_read_csv(), bulk_write_csv(), csv2csv()

Examples

# read from standard CSV
f <- tempfile()
write.csv(mtcars, file = f)
str(read_csv(f))

f <- tempfile()
write.csv2(mtcars, file = f)
str(read_csv(f))

# write to standard CSV
f <- tempfile()
d <- mtcars
str(d <- write_csv(d, file = f))
file.mtime(f)
Sys.sleep(2) # make sure the mtime would have changed
write_csv(d, file = f)
file.mtime(f)

---

csv2csv Convert a German Comma Separated File into a Comma Separated File

Description

Convert a German Comma Separated File into a Comma Separated File
file\_modified\_last

Usage

csv2csv(file, ...)

Arguments

file Path to the file.
...
Arguments passed to read\_csv

Value

Invisibly the return value of write\_csv, but called for its side effect.

See Also

Other CSV functions: bulk\_read\_csv(), bulk\_write\_csv(), csv

Examples

f <- tempfile()
write.csv2(mtcars, file = f)
res <- csv2csv(f)
readLines(get_path(res), n = 1)
write.csv(mtcars, file = f)
readLines(get_path(res), n = 1)

file\_modified\_last Get the File Modified Last

Description

I often look for the file modified last under some directory.

Usage

file\_modified\_last(...)
See Also

Other searching functions: compare_vectors(), find_files(), search_files(), search_rows(), summary.filesearch()
Other file utilities: find_files(), get_unique_string(), is_difftime_less(), paths, search_files(), split_code_file(), touch()

Examples

for (suffix in c(".txt", ".ascii"))
  for (f in file.path(tempdir(), letters))
    touch(paste0(f, suffix))
list.files(tempdir())
file_modified_last(path = tempdir(), pattern = "\.txt$")
dir.create(file.path(tempdir(), "new"))
touch(file.path(tempdir(), "new", "file.txt"))
file_modified_last(path = tempdir(), pattern = "\.txt$")
file_modified_last(path = tempdir(), pattern = "\.txt$", recursive = TRUE)

---

find_files

Find Files on Disk

Description

Look for files on disk, either scanning a vector of names or searching for files with list.files and throw an error if no files are found.

Usage

find_files(
  file_names = NA,
  path = ".",
  pattern = ".*[RrSs]$|.*[RrSs]nw$",
  all_files = TRUE,
  recursive = FALSE,
  ignore_case = FALSE,
  find_all = FALSE,
  select = NA
)

Arguments

file_names character vector of file names (to be checked if the files exist).
path see list.files.
pattern see list.files.
all_files see list.files, argument all.files.
recursive see list.files.
ignore_case see list.files, argument ignore.case.
find_all Throw an error if not all files (given by file_names) are found?
select A named list of numerical vectors of maximum length 2 named min and/or max.
If given, file searching will be restricted to file attributes corresponding to the
names in the list ranging between min and max. See examples.

Details
This is a wrapper to either file.exists or list.files, that ensures that (some) files exist. This
may come handy if you want to perform some kind of file manipulation e.g. with one of the
functions listed under
See Also Other file utilities:

Value
A character vector of file names.

Note
This is merely a wrapper around file.exists or list.files, depending on whether file_names
is given.

See Also
Other searching functions.: compare.vectors(), file_modified_last(), search_files(), search_rows(),
summary.filesearch()
Other file utilities.: file_modified_last(), get_unique_string(), is_difftime_less(), paths,
search_files(), split_code_file(), touch()

Examples
## create some files
files <- unname(sapply(file.path(tempdir(), paste0(sample(letters, 10),
                 ".", c("R", "Rnw", "txt"))),
           touch))
print(files)
print(list.files(tempdir(), full.names = TRUE)) # same as above
## file names given
find_files(file_names = files[1:3])
## some do not exist:
find_files(file_names = c(files[1:3], replicate(2, tempfile())))
try(find_files(file_names = c(files[1:3], replicate(2, tempfile())),
       find_all = TRUE))
## all do not exist:
try(find_files(file_names = replicate(2, tempfile())))
## path given
find_files(path = tempdir())
## change pattern
find_files(path = tempdir(),
          pattern = ".\.[RrSs]$\|.\.[RrSs]nw$|.*\.[txt]"
## Find a specific file by its basename

```r
find_files(path = tempdir(), pattern = paste0("^", basename(files[1]), ")
```

### file_names and path given: file_names beats path

```r
try(find_files(file_names = tempfile(), path = tempdir()))
```

### Select by file size:

```r
write.csv(mtcars, file.path(tempdir(), "mtcars.csv"))
find_files(path = tempdir(), pattern = ".*"
find_files(path = tempdir(), pattern = ".*",
          select = list(size = c(min = 1000))
)
```

---

### `get_boolean_envvar`

Get a Boolean Environment Variable

**Description**

A convenience wrapper to `Sys.getenv`.

**Usage**

```r
get_boolean_envvar(x, stop_on_failure = FALSE)
```

**Arguments**

- `x`: The name of the Environment Variable.
- `stop_on_failure`: Throw an error instead of returning `FALSE` if the environment variable is not set or cannot be converted to boolean.

**Details**

As `Sys.getenv` seems to always return a character vector, the `class` of the value you set it to does not matter.

**Value**

The value the environment variable is set to, converted to boolean. `FALSE` if the environment variable is not set or cannot be converted to boolean. But see `Arguments: stop_on_failure`.

**See Also**

Other test helpers: `get_run_r_tests()`, `is_cran()`, `is_r_cmd_check()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `run_r_tests_for_known_hosts()`, `set_run_r_tests()`

Other operating system functions: `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `with_dir()`

**Examples**

```r
message("See\n example("get_run_r_tests", package = "fritools")")
```
get_options

Get Options For Packages

Description

A convenience function for getOption.

Usage

get_options(
  ..., 
  package_name = .packages()[1],
  remove_names = FALSE,
  flatten_list = TRUE
)

Arguments

... See getOption.
package_name The package's name.
remove_names [boolean(1)]
  Remove the names?
flatten_list [boolean(1)]
  Return a vector?

Value

A (possibly named) list or a vector.

See Also

Other option functions.: is_force(), set_options()

Examples

eample("set_options", package = "fritools")
get_package_version

Query Installed Package Version

Description

`packageVersion` converts to class `package_version`, which then again would need to be converted for `compareVersion`. So this is a modified copy of `packageVersion` skipping the conversion to `package_version`.

Usage

```r
get_package_version(x, lib_loc = NULL)
```

Arguments

- `x`: A character giving the package name.
- `lib_loc`: See argument `lib.loc` in `packageDescription`.

Value

A character giving the package version.

See Also

Other version functions: `is_r_package_installed()`, `is_version_sufficient()`

Other package functions: `is_r_package_installed()`, `is_version_sufficient()`, `load_internal_functions()`

Examples

```r
get_package_version("base")
try(get_package_version("mgcv"))
utils::compareVersion("1000.0.0", get_package_version("base"))
utils::compareVersion("1.0", get_package_version("base"))
# from ?is_version_sufficient:
is_version_sufficient(installed = get_package_version("base"),
                    required = "1.0")
```
get_rscript_script_path

Get the Path of the R Code File in Case of an Rscript Run

Description
Retrieve the path from parsing the command line arguments of a Rscript run.

Usage
get_rscript_script_path()

Value
A vector of mode character giving the name of the R code file. Will be character(0) if not in an Rscript run.

See Also
Other script path getter functions: get_r_cmd_batch_script_path(), get_script_name(), get_script_path()

Examples
get_rscript_script_path()

get_run_r_tests

Get System Variable RUN_R_TESTS

Description
A convenience wrapper to get_boolean_envvar("RUN_R_TESTS").

Usage
get_run_r_tests(stop_on_failure = FALSE)

Arguments
stop_on_failure
Throw an error instead of returning FALSE if the environment variable is not set or cannot be converted to boolean.

Value
The value RUN_R_TESTS is set to, converted to boolean. FALSE if RUN_R_TESTS is not set or cannot be converted to boolean.
get_r_cmd_batch_script_path

Get the Path of the R Code File in Case of an R CMD BATCH Run

Description

Retrieve the path from parsing the command line arguments of a R CMD BATCH run.

Usage

get_r_cmd_batch_script_path()

Value

A vector of mode character giving the name of the R code file. Will be character(0) if not in an R CMD BATCH run.

See Also

Other script path getter functions: get_rscript_script_path(), get_script_name(), get_script_path()
get_script_name

Examples

get_r_cmd_batch_script_path()

get_script_name
Get the Name of the R Code File or set it to default

Description

The code file name is retrieved only for R CMD BATCH and Rscript, if R is used interactively, the
name is set to default, even if you’re working with code stored in a (named) file on disk.

Usage

get_script_name(default = "interactive_R_session")

Arguments
default the name to return if R is run interactively.

Value

A vector of length 1 and mode character giving the name of the R code file if R was run via R CMD
BATCH or Rscript, the given default otherwise.

See Also

Other script path getter functions: get_r_cmd_batch_script_path(), get_rscript_script_path(),
get_script_path()

Examples

get_script_name(default = 'foobar.R')

get_script_path
Get the Path of the R Code File

Description

This is just a wrapper for get_rscript_script_path and get_r_cmd_batch_script_path.

Usage

get_script_path()
get_unique_string

Value

A vector of length 1 and mode character giving the name of the R code file if R was run via R CMD BATCH or Rscript.

See Also

Other script path getter functions: get_r_cmd_batch_script_path(), get_rscript_script_path(), get_script_name()

Examples

get_script_path()

get_unique_string Create a Fairly Unique String

Description

I sometimes need a fairly unique string, mostly for file names, that should start with the current date.

Usage

get_unique_string()

Value

A fairly unique string.

See Also

Other file utilities: file_modified_last(), find_files(), is_difftime_less(), paths, search_files(), split_code_file(), touch()

Examples

replicate(20, get_unique_string())
golden_ratio

Calculate the Golden Ratio

Description
Divide a length using the golden ratio.

Usage
golden_ratio(x)

Arguments
x
The sum of the two quantities to be in the golden ratio.

Value
A numeric vector of length 2, containing the two quantities $a$ and $b$, $a$ being the larger.

See Also
Other bits and pieces: strip_off_attributes(), tapply(), weighted_variance()

Examples
golden_ratio(10)

index_groups

Determine Indices and Sizes of Subsets

Description
Create starting and stopping indices for subsets defined by subset_sizes.

Usage
index_groups(n, k)

Arguments
n
The size of the set.

k
The number of subsets.

Value
A matrix with starting index, size, and stopping index for each subset.
See Also
Other subsetting functions: \texttt{subset_sizes()}

Examples
\begin{verbatim}
index_groups(n = 100, k = 6)
index_groups(n = 2, k = 6)
\end{verbatim}

\begin{tabular}{ll}
\texttt{is\_batch} & \textit{Is R Run in Batch Mode (via R CMD BATCH or Rscript)?} \\
\end{tabular}

Description
Just a wrapper to \texttt{interactive}.

Usage
\begin{verbatim}
is\_batch()
\end{verbatim}

Value
\texttt{TRUE} on success, \texttt{FALSE} otherwise.

See Also
Other logical helpers: \texttt{get\_run\_r\_tests()}, \texttt{is\_cran()}, \texttt{is\_false()}, \texttt{is\_force()}, \texttt{is\_installed()},
\texttt{is\_not\_false()}, \texttt{is\_null\_or\_true()}, \texttt{is\_of\_length\_zero()}, \texttt{is\_r\_cmd\_check()}, \texttt{is\_r\_package\_installed()},
\texttt{is\_running\_on\_fvafrcu\_machines()}, \texttt{is\_running\_on\_gitlab\_com()}, \texttt{is\_success()}, \texttt{is\_version\_sufficient()},
\texttt{is\_windows()}

Examples
\begin{verbatim}
is\_batch()
\end{verbatim}

\begin{tabular}{ll}
\texttt{is\_cran} & \textit{Is R Running on CRAN?} \\
\end{tabular}

Description
\textit{This is a verbatim copy of fda::CRAN of fda version 5.1.9.}

Usage
\begin{verbatim}
is\_cran(cran\_pattern, n\_r\_check4cran)
\end{verbatim}
is_cran

Arguments

cran_pattern  A regular expressions to apply to the names of Sys.getenv() to identify possible CRAN parameters. Defaults to Sys.getenv('_CRAN_pattern_') if available and '^_R_' if not.
n_r_check4cran  Assume this is CRAN if at least n_R_CHECK4CRAN elements of Sys.getenv() have names matching x. Defaults to Sys.getenv('_n_R_CHECK4CRAN_') if available and 5 if not.

Details

This function allows package developers to run tests themselves that should not run on CRAN or with

R CMD check --as-cran

because of compute time constraints with CRAN tests.

The "Writing R Extensions" manual says that R CMD check can be customized "by setting environment variables _R_CHECK_*:_, as described in" the Tools section of the "R Internals" manual.

R CMD check was tested with R 3.0.1 under Fedora 18 Linux and with Rtools 3.0 from April 16, 2013 under Windows 7. With the

'--as-cran'

option, 7 matches were found; without it, only 3 were found. These numbers were unaffected by the presence or absence of the '-timings' parameter. On this basis, the default value of n_R_CHECK4CRAN was set at 5.

1. x. <-Sys.getenv()
2. Fix CRAN_pattern and n_R_CHECK4CRAN if missing.
3. Let i be the indices of x. whose names match all the patterns in the vector x.
4. Assume this is CRAN if length(i) >= n_R_CHECK4CRAN

Value

A logical scalar with attributes '"Sys.getenv '" containing the results of Sys.getenv() and 'matches' containing 1 per step 3 above.

See Also

Other test helpers: get_boolean_envvar(), get_run_r_tests(), is_r_cmd_check(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), run_r_tests_for_known_hosts(), set_run_r_tests()

Other logical helpers: get_run_r_tests(), is_batch(), is_false(), is_force(), is_installed(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(), is_windows()
is_difftime_less

Check Whether Two Times Differ Less Than A Given Value

Description

This is just a wrapper to difftime.

Usage

is_difftime_less(
  time1,
  time2,
  less_than = 1,
  units = "days",
  verbose = FALSE,
  visible = !verbose,
  stop_on_error = FALSE
)

Arguments

time1 See difftime.
time2 See difftime.
less_than The number of units that would be too much of a difference.
units See difftime.
verbose Be verbose?
visible Set to FALSE to return invisible.
stop_on_error Throw an error if the time lag is not less than less_than.

Value

TRUE if the times do not differ ‘that much’, but see stop_on_error.

See Also

Other file utilities: file_modified_last(), find_files(), get_unique_string(), paths, search_files(), split_code_file(), touch()
is_false

Examples

```r
a <- as.POSIXct(0, origin = "1970-01-01", tz = "GMT")
b <- as.POSIXct(60*60*24, origin = "1970-01-01", tz = "GMT")
c <- as.POSIXct(60*60*24 - 1, origin = "1970-01-01", tz = "GMT")
is_diffftime_less(a, b)
is_diffftime_less(a, c)
print(is_diffftime_less(a, b, verbose = TRUE))
print(is_diffftime_less(a, c, verbose = TRUE))
try(is_diffftime_less(a, b, stop_on_error = TRUE))
is_diffftime_less(a, c, verbose = TRUE, stop_on_error = TRUE)
```

Description

I still use R 3.3.3 for testing, `isFALSE()` was introduced in R 3.5.0.

Usage

```r
is_false(x)
```

Arguments

- `x` The object to be tested.

Value

`TRUE` if the object is set to `FALSE`, `FALSE` otherwise.

See Also

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_force()`, `isInstalled()`, `is_not_false()`, `is_null_or_true()`, `is_of_length_zero()`, `is_r_cmd_check()`, `is_r_package_installed()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`, `is_windows()`

Examples

```r
is_false("not false")
is_false(FALSE)
```
is_files_current  
Check Whether Files are Current

Description

I sometimes produce a couple of files by some kind of process and need to check whether they are fairly current and probably product of the same run. So I need to know whether a bunch of files was modified within the last, say, 7 days and that their modification dates do not differ by more than, say, 24 hours.

Usage

is_files_current(
  ..., 
  newer_than = 1,
  units = "week",
  within = 1,
  within_units = "days"
)

Arguments

...   File paths.
newer_than The number of units the files need to be newer than.
units    The unit of newer_than. See difftime.
within   The number of units the files need to be modified within.
within_units The unit of within. See difftime.

Value

TRUE on success, FALSE otherwise.

Examples

p1 <- tempfile()
p2 <- tempfile()
p3 <- tempfile()
touch(p1)
touch(p2)
Sys.sleep(3)
touch(p3)
is_files_current(p3, newer_than = 1, units = "days",
    within = 4, within_units = "secs")
is_files_current(p1, p2, p3, newer_than = 1, units = "days",
    within = 4, within_units = "secs")
is_files_current(p1, p2, p3, newer_than = 1, units = "days",
    within = 1, within_units = "secs")
is_force

is_files_current(p1, p2, p3, newer_than = 1, units = "secs",
within = 4, within_units = "secs")

---

is_force  Opt-out Via Option

Description

Check whether or not a package option (set via set_options) force is not set or set to TRUE.

Usage

is_force(x = .packages()[1])

Arguments

x
The option under which an element "force" is to be searched for.

Value

TRUE if option x["force"] is either TRUE or NULL (i.e. not set at all).

See Also

Other option functions: get_options(), set_options()
Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_installed(),
is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(),
is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(),
is_windows()

Examples

is_force()
set_options(list(force = FALSE))
get_options(flatten_list = FALSE)
is_force()
is_installed  

Description
Is an external program installed?

Usage
is_installed(program)

Arguments
program  Name of the program.

Value
TRUE on success, FALSE otherwise.

See Also
Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(), is_windows()

Other operating system functions: get_boolean_envvar(), get_run_r_tests(), is_r_package_installed(), is_success(), is_windows(), with_dir()

Examples
if (is_running_on_fvafrcu_machines() || is_running_on_gitlab_com()) {
    # NOTE: There are CRAN machines where neither "R" nor "R-devel" is in
    # the path, so we skip this example on unknown machines.
    is_installed("R")
}
is_installed("probably_not_installed")

is_not_false  

Description
Sometimes you need to know whether or not an object exists and is not set to FALSE (and possibly not NULL).
Usage

\texttt{is\_not\_false(x, null\_is\_false = TRUE, ...)}

Arguments

\texttt{x} \quad \text{The object to be tested.}
\texttt{null\_is\_false} \quad \text{Should NULL be treated as FALSE?}
\texttt{...} \quad \text{Parameters passed to exists. See Examples.}

Value

\texttt{TRUE} if the object is set to something different than FALSE, FALSE otherwise.

See Also

Other logical helpers: \texttt{get\_run\_r\_tests()}, \texttt{is\_batch()}, \texttt{is\_cran()}, \texttt{is\_false()}, \texttt{is\_force()}, \texttt{is\_installed()}, \texttt{is\_null\_or\_true()}, \texttt{is\_of\_length\_zero()}, \texttt{is\_r\_cmd\_check()}, \texttt{is\_r\_package\_installed()}, \texttt{is\_running\_on\_fvafrcu\_machines()}, \texttt{is\_running\_on\_gitlab\_com()}, \texttt{is\_success()}, \texttt{is\_version\_sufficient()}, \texttt{is\_windows()}

Examples

\begin{verbatim}
a <- 1
is_not_false(a)
f <- function() {
  print(a)
  print(is_not_false(a))
}
f()

f <- function() {
a <- FALSE
  print(a)
  print(is_not_false(a))
}
f()

f <- function() {
  print(a)
  print(is_not_false(a, null\_is\_false = TRUE,
               inherits = FALSE))
}
f()
\end{verbatim}
is_null_or_true  Is an Object TRUE or NULL?

Description
Is an object TRUE or NULL?

Usage
is_null_or_true(x)

Arguments
x  The object to be tested.

Value
TRUE if the object is set to TRUE or NULL, FALSE otherwise.

See Also
Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_installed(), is_not_false(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrceu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(), is_windows()

Examples
is_null_or_true("true") # FALSE
is_null_or_true(TRUE) # TRUE
is_null_or_true(NULL) # TRUE
suppressWarnings(rm("not_defined"))
try(is_null_or_true(not_defined)) # error

is_of_length_zero  Is an Object of Length Zero?

Description
Some expressions evaluate to integer(0) or the like.

Usage
is_of_length_zero(x, class = NULL)
is_running_on_fvafrcu_machines

Arguments

- `x`: The object.
- `class`: An optional character vector of length 1 giving the class. See examples.

Value

- TRUE on success, FALSE otherwise.

See Also

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `is_not_false()`, `is_null_or_true()`, `is_r_cmd_check()`, `is_r_package_installed()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`, `is_windows()`

Examples

```r
x <- ""; length(x); is_of_length_zero(x)
```

```r
x <- grep(" ","")
print(x)
is_of_length_zero(x)
is_of_length_zero(x, "character")
is_of_length_zero(x, "numeric")
is_of_length_zero(x, "integer")
```

is_running_on_fvafrcu_machines

*Is the Machine Running the Current R Process Owned by FVAFRCU?*

Description

Is the machine running the current R process known to me?

Usage

```r
is_running_on_fvafrcu_machines()
```

Value

- TRUE on success, FALSE otherwise.

See Also

Other test helpers: `get_boolean_envvar()`, `get_run_r_tests()`, `is_cran()`, `is_r_cmd_check()`, `is_running_on_gitlab_com()`, `run_r_tests_for_known_hosts()`, `set_run_r_tests()`

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `is_not_false()`, `is_null_or_true()`, `is_of_length_zero()`, `is_r_cmd_check()`, `is_r_package_installed()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`, `is_windows()`
is_running_on_gitlab_com

Is the Current Machine Owned by https://about.gitlab.com?

Description

Check whether the current machine is located on https://about.gitlab.com. This check is an approximation only.

Usage

is_running_on_gitlab_com(verbos = TRUE)

Arguments

verbose Be verbose?

Value

TRUE on success, FALSE otherwise.

See Also

Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_installed(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_success(), is_version_sufficient(), is_windows()

Other test helpers: get_boolean_envvar(), get_run_r_tests(), is_cran(), is_r_cmd_check(), is_running_on_fvafrcu_machines(), run_r_tests_for_known_hosts(), set_run_r_tests()

Examples

is_running_on_gitlab_com()
**is_r_cmd_check**  
*Is the Current R Process an R CMD check?*

**Description**
Check for system variables to guess whether or not this is an R CMD check.

**Usage**
```r
is_r_cmd_check()
```

**Value**
TRUE on success, FALSE otherwise.

**See Also**
Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_installed(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(), is_windows()

Other test helpers: get_boolean_envvar(), get_run_r_tests(), is_cran(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), run_r_tests_for_known_hosts(), set_run_r_tests()

---

**is_r_package_installed**  
*Is an R Package Installed?*

**Description**
Is an R package installed?

**Usage**
```r
is_r_package_installed(x, version = "0")
```

**Arguments**

- **x**  
  Name of the package as character string.

- **version**  
  Required minimum version of the package as character string.

**Value**
TRUE on success, FALSE otherwise.
is_success

See Also

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `is_not_false()`, `is_null_or_true()`, `is_of_length_zero()`, `is_r_cmd_check()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`, `is_windows()`

Other operating system functions: `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_success()`, `is_windows()`, `with_dir()`

Other package functions: `get_package_version()`, `is_version_sufficient()`, `load_internal_functions()`

Other version functions: `get_package_version()`, `is_version_sufficient()`

Examples

```r
is_r_package_installed("base", "300.0.0")
is_r_package_installed("fritools", "1.0.0")
```

---

is_success  Does the Return Value of a Command Signal Success?

---

Description

This is just a wrapper to ease the evaluation of return values from external commands: External commands return 0 on success, which is `FALSE`, when converted to logical.

Usage

```r
is_success(x)
```

Arguments

- `x`  The external commands return value.

Value

- `TRUE` on success, `FALSE` otherwise.

See Also

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `is_not_false()`, `is_null_or_true()`, `is_of_length_zero()`, `is_r_cmd_check()`, `is_r_package_installed()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_version_sufficient()`, `is_windows()`

Other operating system functions: `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_windows()`, `with_dir()`
is_valid_primary_key

Examples

is_success(0)
is_success(1)
is_success(-1)

is_valid_primary_key(data, key, verbose = TRUE)

Arguments

data
key
verbose

The data.frame for which you want to find valid potential primary key.
Character vector containing a subset of the columns names of data.
Be verbose?

Value

TRUE, if key is a valid primary key, FALSE otherwise.

Examples

is_valid_primary_key(mtcars, "qsec")
is_valid_primary_key(mtcars, "carb")
is_valid_primary_key(mtcars, c("qsec", "gear"))
is_valid_primary_key(mtcars, c("qsec", "carb"))
cars <- mtcars
cars$id <- seq_len(nrow(cars))
is_valid_primary_key(cars, "id")
is_version_sufficient  Is a Version Requirement Met?

Description

Just a wrapper to compareVersion. I regularly forget how to use it.

Usage

is_version_sufficient(installed, required)

Arguments

installed  The version available.
required  The version required.

Value

TRUE, if so, FALSE otherwise.

See Also

Other logical helpers: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_installed(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_windows()

Other package functions: get_package_version(), is_r_package_installed(), load_internal_functions()

Other version functions: get_package_version(), is_r_package_installed()

Examples

is_version_sufficient(installed = "1.0.0", required = "2.0.0")
is_version_sufficient(installed = "1.0.0", required = "1.0.0")
is_version_sufficient(installed = get_package_version("base"), required = "3.5.2")
### is_windows

**Is the System Running a Windows Machine?**

**Description**

Is the system running a windows machine?

**Usage**

```r
is_windows()
```

**Value**

`TRUE` if so, `FALSE` otherwise.

**See Also**

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `is_not_false()`, `is_null_or_true()`, `is_of_length_zero()`, `is_r_cmd_check()`, `is_r_package_installed()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`

Other operating system functions: `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `with_dir()`

**Examples**

```r
is_windows()
```

---

### load_internal_functions

**Load a Package’s Internals**

**Description**

Load objects not exported from a package’s namespace.

**Usage**

```r
load_internal_functions(package, ...)
```

**Arguments**

- `package` The name of the package as a string.
- `...` Arguments passed to `ls`, `all.names = TRUE` could be a good idea.
memory_hogs

Value

Invisibly TRUE.

See Also

codetools::checkUsageEnv.

Other package functions: get_package_version(), is_r_package_installed(), is_version_sufficient()

Examples

load_internal_functions("fritools")

memory_hogs Find Memory Hogs

Description

List objects in an R environment by size.

Usage

memory_hogs(
  unit = c("b", "Kb", "Mb", "Gb", "Tb", "Pb"),
  return_numeric = TRUE,
  ..., 
  envir = .GlobalEnv
)

Arguments

unit The unit to use.
return_numeric Return a numeric vector? If set to FALSE, a character vector including the unit will be returned, which might be less usable but easier to read.
... Arguments passed to order, defaults to decreasing = FALSE.
envir The environment where to look for objects.

Value

A named vector of memory usages.

See Also

Other R memory functions: wipe_clean()
paths

Set or Get the path Attribute to or from an Object

Description

We set paths on some objects, these are convenience wrappers to attr.

Usage

get_path(x)

set_path(x, path, overwrite = FALSE)

Arguments

x An object.

path The path to be set.

overwrite Overwrite an existing path attribute instead of throwing an error?

Value

For get_path the value of attr(x,"path").

For set_path the modified object.

See Also

Other file utilities: file_modified_last(), find_files(), get_unique_string(), is_difftime_less(), search_files(), split_code_file(), touch()
run_r_tests_for_known_hosts

Force Testing on Known Hosts

Description

Enforce the environment variable RUN_R_TESTS to TRUE on known hosts.

Usage

run_r_tests_for_known_hosts()

Details

This should go into .onLoad to force tests on known hosts.

Value

Invisibly NULL.

See Also

Other test helpers: get_boolean_envvar(), get_run_r_tests(), is_cran(), is_r_cmd_check(),
is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), set_run_r_tests()

Examples

get_run_r_tests()
if (isFALSE(get_run_r_tests())) {
  run_r_tests_for_known_hosts()
  get_run_r_tests()
}
**search_files**

**Search Files for a Pattern**

**Description**

This is an approximation of unix find and grep.

**Usage**

```
search_files(what, verbose = TRUE, exclude = NULL, ...)  
```

**Arguments**

- `what` A regex pattern for which to search.
- `verbose` Be verbose?
- `exclude` A regular expression for excluding files.
- `...` Arguments passed to `list.files`.

**Value**

*Invisibly* a vector of names of files containing the pattern given by `what`.

**See Also**

Other searching functions: `compare_vectors()`, `file_modified_last()`, `find_files()`, `search_rows()`, `summary.filesearch()`

Other file utilities: `file_modified_last()`, `find_files()`, `get_unique_string()`, `is_difftime_less()`, `paths`, `split_code_file()`, `touch()`

**Examples**

```
write.csv(mtcars, file.path(tempdir(), "mtcars.csv"))
for (i in 0:9) {
  write.csv(iris, file.path(tempdir(), paste0("iris", i, ", .csv"))
}
search_files(what = "Mazda", path = tempdir(), pattern = ".*\.csv$")
search_files(what = "[Ss]etosa", path = tempdir(), pattern = ".*\.
search_files(path = tempdir(),
  pattern = ".*\.
  exclude = "[2-9]\.
  what = ")
summary(x)
summary(x, type = "what")
summary(x, type = "matches")
try(search_files(what = "ABC", path = tempdir(), pattern = ".*\.
```
search_rows

Search All Rows Across Columns of a Matrix-like Structure

Description

I sometimes need to see which rows of a matrix-like structure contain a string matched by a search pattern. This somewhat similar to writing a matrix-like structure to disk and then using search_files on it.

Usage

search_rows(x, pattern = ".*", include_row_names = TRUE)

Arguments

x A matrix or data.frame.
pattern A pattern.
include_row_names Include row names into the search?

Value

All rows where the pattern was found in at least one column.

See Also

Other searching functions: compare_vectors(), file_modified_last(), find_files(), search_files(), summary.filesearch()

Examples

p <- "\<4.0[[:alpha:]]\""
search_rows(x = mtcars, pattern = p)
search_rows(x = mtcars, pattern = p, include_row_names = FALSE)
try(search_rows(x = mtcars, pattern = "ABC"))

set_hash

Set a Hash Attribute on an Object

Description

Set a Hash Attribute on an Object

Usage

set_hash(x)
set_options

Arguments

- `x`  The object.

Value

The modified object.

See Also

Other hash functions for objects.: un_hash()

---

**set_options**  
Set Options For Packages

Description

A convenience function for options.

Usage

```r
set_options(..., package_name = .packages()[1], overwrite = TRUE)
```

Arguments

- `...`  See options.
- `package_name`  The package's name.
- `overwrite`  [boolean(1)]  
  Overwrite options already set?

Value

Invisibly TRUE.

See Also

Other option functions.: get_options(), is_force()

Examples

```r
options("cleanr" = NULL)
defaults <- list(max_file_width = 80, max_file_length = 300,
                 max_lines = 65, max_lines_of_code = 50,
                 max_num_arguments = 5, max_nesting_depth = 3,
                 max_line_width = 80, check_return = TRUE)

set_options(package_name = "cleanr", defaults)
getOption("cleanr")
set_options(package_name = "cleanr", list(max_line_width = 3,
```
max_lines = "This is nonsense!")
set_options(package_name = "cleanr", check_return = NULL, max_lines = 4000)
geet_options(package_name = "cleanr")

---

set_run_r_tests  Set the System Variable RUN_R_TESTS

### Description

A convenience wrapper to `Sys.getenv` for setting RUN_R_TESTS.

### Usage

```r
set_run_r_tests(x, force = FALSE)
```

### Arguments

- **x**: A logical, typically some function output.
- **force**: Overwrite the variable if already set?

### Value

The value RUN_R_TESTS is set to, `NULL` if nothing is done.

### See Also

Other test helpers: `get_boolean_envvar()`, `get_run_r_tests()`, `is_cran()`, `is_r_cmd_check()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `run_r_tests_for_known_hosts()`

### Examples

```r
set_run_r_tests(is_running_on_fvafrcu_machines())
geet_run_r_tests()
set_run_r_tests(TRUE, force = TRUE)
geet_run_r_tests()
```
split_code_file

Split a Code File Into Multiple Files

Description

I tend to find files with dozens of functions. They don’t read well. So I split a code file into multiple files each containing a single function.

Usage

split_code_file(
  file,
  output_directory = tempdir(),
  encoding = getOption("encoding"),
  write_to_disk = getOption("write_to_disk")
)

Arguments

file The code file to be split.
output_directory Where to create the new files.
encoding The encoding passed to source.
write_to_disk Set the output_directory to dirname(file)? Just a shortcut.

Value

Invisibly a vector of paths to the new files.

See Also

Other file utilities: file_modified_last(), find_files(), get_unique_string(), is_diffmtime_less(), paths, search_files(), touch()

Examples

infile <- system.file("files", "test_helpers.R", package = "fritools")
## Not run:
  file.show(infile)

## End(Not run)
paths <- split_code_file(file = infile)
## Not run:
  file.show(paths[2])

## End(Not run)
### strip_off_attributes

**Description**
Strip Attributes off an Object

**Usage**

\[
\text{strip\_off\_attributes}(x)
\]

**Arguments**

- \(x\)  
  An object.

**Value**

The object.

**See Also**

- `base::unname`
- Other bits and pieces: `golden_ratio()`, `tapply()`, `weighted_variance()`

**Examples**

\[
y \leftarrow \text{stats::setNames}(1:3, \text{letters}[1:3])
\]
\[
\text{attr}(y, "myattr") \leftarrow "qwer"
\]
\[
\text{comment}(y) \leftarrow "qwer"
\]
\[
\text{strip\_off\_attributes}(y)
\]

### subset_sizes

**Description**

Determine Subset Sizes Close to Equality

**Usage**

\[
\text{subset\_sizes}(n, k)
\]

**Arguments**

- \(n\)  
  The size of the set.
- \(k\)  
  The number of subsets.
summary.filesearch

Value

A vector of k sizes of the subsets.

See Also

Other subsetting functions: index_groups()

Examples

subset_sizes(n = 100, k = 6)
subset_sizes(n = 2, k = 6)

summary.filesearch Summarize File Searches

Description

A custom summary function for objects returned by search_files.

Usage

## S3 method for class 'filesearch'
summary(object, ..., type = c("file", "what", "matches"))

Arguments

object An object returned by search_files.

... Needed for compatibility.

type Type of summary.

Value

A summarized object.

See Also

Other searching functions: compare_vectors(), file_modified_last(), find_files(), search_files(), search_rows()
tapply

Apply a Function Over a Ragged Array

Description

This is a modified version of base::tapply to allow for data.frames to be passed as X.

Usage

tapply(object, index, func = NULL, ..., default = NA, simplify = TRUE)

Arguments

object See base::tapply X.
index See base::tapply INDEX.
func See base::tapply FUN.
... See base::tapply.
default See base::tapply.
simplify See base::tapply.

Value

See base::tapply.

See Also

Other bits and pieces: golden_ratio(), strip_off_attributes(), weighted_variance()
Examples

```r
result <- fritools::tapply(warpbreaks["breaks"], warpbreaks[, -1], sum)
expectation <- base::tapply(warpbreaks["breaks"], warpbreaks[, -1], sum)
RUnit::checkIdentical(result, expectation)
data("mtcars")
s <- stats::aggregate(x = mtcars["mpg"],
                     by = list(mtcars["cyl"], mtcars["vs"]),
                     FUN = mean)
t <- base::tapply(X = mtcars["mpg"],
                     INDEX = list(mtcars["cyl"], mtcars["vs"]),
                     FUN = mean)
if (require("reshape", quietly = TRUE)) {
  suppressWarnings(tm <- na.omit(reshape::melt(t)))
  if (RUnit::checkEquals(s, tm, check.attributes = FALSE))
    message("Works!"
  }
message("If you don't pass weigths, this is equal to:"
w <- base::tapply(X = mtcars["mpg"], INDEX = list(mtcars["cyl"], mtcars["vs"]),
                   FUN = stats::weighted.mean)
all.equal(w, t, check.attributes = FALSE)
message("But how do you pass those weights?"
# we define a wrapper to pass the column names for a data.frame:
weighted_mean <- function(df, x, w) {
  stats::weighted.mean(df[[x]], df[[w]])
}
if (RUnit::checkIdentical(stats::weighted.mean(mtcars["mpg"],
                           mtcars["wt"]),
                         weighted_mean(mtcars, "mpg", "wt")))
  message("Works!"
message("base::tapply can't deal with data.frame:"
try(base::tapply(X = mtcars, INDEX = list(mtcars["cyl"], mtcars["vs"]),
               FUN = weighted_mean, x = "mpg", w = "wt")
wm <- fritools::tapply(object = mtcars, index = list(mtcars["cyl"],
                         mtcars["vs"]),
                       func = weighted_mean, x = "mpg", w = "wt")
subset <- mtcars[mtcars["cyl"] == 6 & mtcars["vs"] == 0, c("mpg", "wt")]
stats::weighted.mean(subset["mpg"], subset["wt"]) == wm
```

Mock the Unix `touch` Utility

Description

Creating a file or ensuring a file's modification time changes.

Usage

```
touch(path)
```
Arguments

path Path to the file to be touched.

Value

The Path to the file touched.

See Also

Other file utilities: file_modified_last(), find_files(), get_unique_string(), is_difftime_less(), paths, search_files(), split_code_file()

Examples

```r
file <- tempfile()
touch(file)
t1 <- file.mtime(file)
touch(file)
t2 <- file.mtime(file)
t1 < t2
file < file.path(tempfile(), "path", "not", "there.txt")
touch(file)
file.exists(file)
```

---

**un_hash**

*Separate an Object from its Hash Attribute*

Description

We calculate a hash value of an object and store it as an attribute of the objects, the hash value of that object will change. So we need to split the hash value from the object to see whether or not the objected changed.

Usage

```r
un_hash(x)
```

Arguments

x The object.

Value

A list containing the object and its hash attribute.

See Also

Other hash functions for objects: set_hash()
weighted_variance

Calculate a Weighted Variance

Description

Calculate a weighted variance.

Usage

weighted_variance(x, ...)

## S3 method for class 'numeric'
weighted_variance(x, weights, weights_counts = NULL, ...)

## S3 method for class 'data.frame'
weighted_variance(x, var, weight, ...)

Arguments

x A numeric vector or data.frame.
...
Other arguments ignored.
weights A vector of weights.
weights_counts Are the weights counts of the data? If so, we can calculate the unbiased sample variance, otherwise we calculate the biased (maximum likelihood estimator of the) sample variance.
var The name of the column in x giving the variable of interest.
weight The name of the column in x giving the weights.

Details

The data.frame method is meant for use with tapply, see examples.

See Also

Other bits and pieces: golden_ratio(), strip_off_attributes(), tapply()

Examples

## GPA from Siegel 1994
wt <- c(5, 5, 4, 1)/15
x <- c(3.7,3.3,3.5,2.8)
var(x)
weighted_variance(x = x)
weighted_variance(x = x, weights = wt)
weighted_variance(x = x, weights = wt, weights_counts = TRUE)
weights <- c(5, 5, 4, 1)
weighted_variance(x = x, weights = weights)
weighted_variance(x = x, weights = weights, weights_counts = FALSE)
weighted_variance(x = data.frame(x, wt), var = "x",
              weight = "wt")

# apply by groups:
frtools::tapply(object = mtcars,
              index = list(mtcars[["cyl"]], mtcars[["vs"]]),
              func = weighted_variance, var = "mpg", w = "wt")

---

## wipe_clean

### Remove All Objects From an Environment

#### Description

Wipe an environment, typically `.GlobalEnv`, clean.

#### Usage

```r
wipe_clean(environment)
```

#### Arguments

- `environment` The environment that should be wiped clean.

#### Value

A character vector containing the names of objects removed, but called for its side effect of removing all objects from the environment.

#### See Also

Other R memory functions: [memory_hogs()](#)

#### Examples

```r
e <- new.env()
assign("a", 1, envir = e)
assign("b", 1, envir = e)
ls(envir = e)
wipe_clean(envir = e)
ls(envir = e)
RUnit::checkIdentical(length(ls(envir = e)), 0L)
```
with_dir

Execute Code in a Temporary Working Directory

Description

This is a verbatim copy of withr::with_dir from of withr's version 2.4.1. I often need withr only to import withr::with_dir, which is a really simple function. So I just hijack withr::with_dir.

Usage

with_dir(new, code)

Arguments

new The new working directory.

code Code to execute in the temporary working directory.

Value

The results of the evaluation of the code argument.

See Also

Other operating system functions: get_boolean_envvar(), get_run_r_tests(), is_installed(), is_r_package_installed(), is_success(), is_windows()

Examples

temp_dir <- file.path(tempfile())
dir.create(temp_dir)
with_dir(temp_dir, getwd())
Index

* CSV functions.
  bulk_read_csv, 3
  bulk_write_csv, 4
  csv, 9
  csv2csv, 10
* German umlaut converters.
  convert_umlauts_to_ascii, 8
  convert_umlauts_to_tex, 9
* R memory functions.
  memory_hogs, 38
  wipe_clean, 52
* bits and pieces.
  golden_ratio, 21
  strip_off_attributes, 46
  tapply, 48
  weighted_variance, 51
* call functions.
  call_conditionally, 5
  call_safe, 6
* file utilities.
  file_modified_last, 11
  find_files, 12
  get_unique_string, 20
  is_difftime_less, 24
  paths, 39
  search_files, 41
  split_code_file, 45
  touch, 49
* hash functions for objects.
  set_hash, 42
  un_hash, 50
* logical helpers.
  get_run_r_tests, 17
  is_batch, 22
  is_cran, 22
  is_false, 25
  is_force, 27
  is_installed, 28
  is_not_false, 28
  is_null_or_true, 30
  is_of_length_zero, 30
  is_r_cmd_check, 33
  is_r_package_installed, 33
  is_running_on_fvafrcu_machines, 31
  is_running_on_gitlab_com, 32
  is_success, 34
  is_version_sufficient, 36
  is_windows, 37
* operating system functions.
  get_boolean_envvar, 14
  get_run_r_tests, 17
  is_installed, 28
  is_r_package_installed, 33
  is_success, 34
  is_windows, 37
  with_dir, 53
* option functions.
  get_options, 15
  is_force, 27
  set_options, 43
* package functions.
  get_package_version, 16
  is_r_package_installed, 33
  is_version_sufficient, 36
  load_internal_functions, 37
* package
  fritools-package, 3
* script path getter functions.
  get_r_cmd_batch_script_path, 18
  get_rscript_script_path, 17
  get_script_name, 19
  get_script_path, 19
* searching functions.
  compare_vectors, 7
  file_modified_last, 11
  find_files, 12
  search_files, 41
  search_rows, 42
INDEX

summary_filesearch, 47
* subsetting functions.
  index_groups, 21
  subset_sizes, 46
* test helpers.
  get_boolean_envvar, 14
  get_run_r_tests, 17
  is_cran, 22
  is_r_cmd_check, 33
  is_running_on_fvafrcu_machines, 31
  is_running_on_gitlab_com, 32
  run_r_tests_for_known_hosts, 40
  set_run_r_tests, 44
* version functions.
  get_package_version, 16
  is_r_package_installed, 33
  is_version_sufficient, 36
  .GlobalEnv, 52
  .onLoad, 40
  attr, 39

base::tapply, 48
base::unname, 46
bulk_read_csv, 3, 5, 10, 11
bulk_write_csv, 4, 4, 10, 11

call_conditionally, 5, 6, 7
call_safe, 6, 6
class, 14
codetools::checkUsageEnv, 38
compare_vectors, 7, 12, 13, 41, 42, 47
compareVersion, 16, 36
convert_umlauts_to_ascii, 8, 9
convert_umlauts_to_tex, 8, 9
csv, 4, 5, 9, 11
csv2csv, 4, 5, 10, 10
data.frame, 9, 42, 48, 51
difftime, 24, 26
do.call, 6, 7
exists, 29
FALSE, 14, 17, 22, 24–26, 28–38
file.exists, 13
file_modified_last, 7, 11, 13, 20, 24, 39, 41, 42, 45, 47, 50
find_files, 4, 7, 12, 12, 20, 24, 39, 41, 42, 45, 47, 50

fritools-package, 3
get_boolean_envvar, 14, 17, 18, 23, 28, 31–34, 37, 40, 44, 53
get_options, 15, 27, 43
get_package_version, 16, 34, 36, 38
get_path (paths), 39
get_r_cmd_batch_script_path, 17, 18, 19, 20
get_rscript_script_path, 14, 17, 22, 23, 25, 27–34, 36, 37, 40, 44, 53
get_script_name, 17, 18, 19, 20
get_script_path, 17–19, 19
get_unique_string, 12, 13, 20, 24, 39, 41, 45, 50
getOption, 15
golden_ratio, 30

index_groups, 21, 47
integer, 30
interactive, 22
invisible, 24
Invisibly, 11, 38, 40, 41, 43, 45
is.batch, 18, 22, 23, 25, 27–34, 36, 37
is.cran, 14, 18, 22, 22, 25, 27–34, 36, 37, 40, 44
is_difftime_less, 12, 13, 20, 24, 39, 41, 45, 50
is_false, 18, 22, 23, 25, 27–34, 36, 37
is_files_current, 26
is_force, 15, 18, 22, 23, 25, 27, 28–34, 36, 37
is_installed, 14, 18, 22, 23, 25, 27, 28, 29–34, 36, 37, 53
is_not_false, 18, 22, 22, 25, 27, 28, 28, 30–34, 36, 37
is_null_or_true, 18, 22, 23, 25, 27–29, 30, 31–34, 36, 37
is_of_length_zero, 18, 22, 23, 25, 27–30, 30, 31–34, 36, 37
is_r_cmd_check, 14, 18, 22, 23, 25, 27–32, 33, 34, 36, 37, 40, 44
is_r_package_installed, 14, 16, 18, 22, 23, 25, 27–33, 33, 34, 36–38, 53
is_running_on_fvafrcu_machines, 14, 18, 22, 23, 25, 27–31, 31, 32–34, 36, 37, 40, 44
INDEX

is_running_on_gitlab_com, 14, 18, 22, 23, 25, 27–31, 32, 33, 34, 36, 37, 40, 44
is_success, 14, 18, 22, 23, 25, 27–34, 34, 36, 37, 53
is_valid_primary_key, 35
is_version_sufficient, 16, 18, 22, 23, 25, 27–34, 36, 37, 38
is_windows, 14, 18, 22, 23, 25, 27–34, 36, 37, 53
length, 19, 20
list.files, 11–13, 41
load_internal_functions, 16, 34, 36, 37
ls, 37

matrix, 42
memory_hogs, 38, 52
mode, 17–20

NULL, 27–30, 40, 44

options, 43
order, 38

package_version, 16
packageDescription, 16
packageVersion, 16
paths, 12, 13, 20, 24, 39, 41, 45, 50

read_csv, 3, 4, 11
read_csv(csv), 9
run_r_tests_for_known_hosts, 14, 18, 23, 31–33, 40, 44

search_files, 7, 12, 13, 20, 24, 39, 41, 42, 45, 47, 50
search_rows, 7, 12, 13, 41, 42, 47
set_hash, 42, 50
set_options, 15, 27, 43
set_path(paths), 39
set_run_r_tests, 14, 18, 23, 31–33, 40, 44
source, 45
split_code_file, 12, 13, 20, 24, 39, 41, 45, 50
strip_off_attributes, 21, 46, 48, 51
subset_sizes, 21, 46, 48, 51
summary.filesearch, 7, 12, 13, 41, 42, 47
Sys.getenv, 14, 44
tapply, 21, 46, 48, 51
touch, 12, 13, 20, 24, 39, 41, 45, 49
TRUE, 5, 6, 22, 24–38, 43
tryCatch, 5
un_hash, 43, 50
utils::read.csv, 10
utils::read.csv2, 10
utils::read.csv2, 9

vector, 51

weighted_variance, 21, 46, 48, 51
wipe_clean, 38, 52
with_dir, 14, 18, 28, 34, 37, 53
write_csv, 4, 5, 11
write_csv(csv), 9