Package ‘fritools2’

June 9, 2023

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

Version 4.1.0

Description Miscellaneous utilities, tools and helper functions for finding and searching files on disk, searching for and removing R objects from the workspace. 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, covr, desc, devtools, digest, dplyr, pkgload, reshape, rmarkdown, RUnit, testthat (>= 3.0.0), tinytest, whoami

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

VignetteBuilder utils

Config/testthat/edition 3

NeedsCompilation no

Author Andreas Dominik Cullmann [aut, cre]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository CRAN

Date/Publication 2023-06-09 14:00:17 UTC

R topics documented:

fritools2-package .................................................. 3
bulk_read_csv .................................................. 4
R topics documented:

bulk_write_csv ........................................ 5
call_conditionally .................................... 6
call_safe .............................................. 7
check_ascii_file ...................................... 8
clipboard_path ....................................... 9
column_sums .......................................... 9
compare_vectors ...................................... 10
convert_umlauts_to_ascii .............................. 11
convert_umlauts_to_tex ................................ 12
count_groups ........................................... 12
csv ..................................................... 13
csv2csv .................................................. 14
delete_trailing_blank_lines ........................... 15
delete_trailing_whitespace ............................. 16
develop_test .......................................... 16
file_copy ................................................ 17
file_modified_last ..................................... 18
file_save ............................................... 19
find_files .............................................. 20
fromto .................................................. 22
get_boolean_envvar ..................................... 23
get_lines_between_tags ................................ 24
gmtime .................................................. 25
get_options ............................................ 26
get_package_version ................................... 27
get_rscript_script_path ................................ 28
get_run_r_tests ........................................ 28
get_r_cmd_batch_script_path ........................... 29
get_script_name ........................................ 30
get_script_path ........................................ 31
ger_unique_string ...................................... 31
golden_ratio ............................................ 32
grep_file ................................................ 32
index_groups .......................................... 33
is_batch ............................................... 34
is_cran .................................................. 35
is_difftime_less ...................................... 36
is_false .................................................. 37
is_files_current ...................................... 38
is_force ............................................... 39
is_installed .......................................... 40
is_not_false .......................................... 41
is_null_or_true ....................................... 42
is_of_length_zero .................................... 43
is_path .................................................. 44
is_running_on_fvafrcu_machines ....................... 44
is_running_on_gitlab_com .............................. 45
is_r_cmd_check ........................................ 46
fritools2-package

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

Description

Miscellaneous utilities, tools and helper functions.

Details

You will find the details in
vignette("Not_an_Introduction_to_fritools", package = "fritools2").
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`.

**Usage**

```r
bulk_read_csv(
  paths,  
  stop_on_error = FALSE,  
  is_latin1 = TRUE,  
  pattern = ".*\\.csv\$",  
  all_files = TRUE,  
  recursive = FALSE,  
  ignore_case = FALSE,  
  find_all = FALSE,  
  select = NA,  
  ...  
)
```

**Arguments**

- **paths**  
  A vector of file paths or the directory to find files.
- **stop_on_error**  
  Stop if any of the files is not read? Warn and continue otherwise.
- **is_latin1**  
  Are the files encoded in "Latin1"?
- **pattern**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **all_files**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **recursive**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **ignore_case**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **find_all**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **select**  
  see `find_files`. Ignored, if `paths` is not a directory.
- **...**  
  Arguments passed to `read_csv`.

**Value**

A named list, each element holding the contents of one csv file read by `read_csv`.

**See Also**

Other CSV functions: `bulk_write_csv()`, `check_ascii_file()`, `csv2csv()`, `csv`
bulk_write_csv

Examples

```r
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda Rööf64"
names(mt_german)[1] <- "mgūdc"
#% read from directory
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")
}
bulk <- bulk_read_csv(tempdir())

#% pass a path
f <- list.files(tempdir(), pattern = ".*\.csv\$", full.names = TRUE)[1]
bulk <- bulk_read_csv(f)

#% pass multiple path
f <- list.files(tempdir(), pattern = ".*\.csv\$", full.names = TRUE)[2:4]
bulk <- bulk_read_csv(f)
```

bulk_write_csv

_Bulk Write Comma Separated Files_

Description

Write a bunch of objects to disk using `write_csv`.

Usage

```r
bulk_write_csv(x, ...)
```

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()`, `check_ascii_file()`, `csv2csv()`, `csv`
Examples

```
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda Rö4"
names(mt_german)[1] <- "mgüdc"
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_safe

Value
The return value of f or fallback.

See Also
Other call functions: call_safe()

Examples

```r
call_conditionally(get_package_version,
  condition = TRUE,
  args = list(x = "fritools2"),
  fallback = "0.0")
call_conditionally(get_package_version,
  condition = FALSE,
  args = list(x = "fritools2"),
  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

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

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.
check_ascii_file

Check the Number of Lines and Fields in a File

Description

Check the Number of Lines and Fields in a File

Usage

check_ascii_file(path, sep = "");"

Arguments

path Path to a file.
sep A character separating the fields in the file.

Value

A list giving the number of lines, number of fields and an boolean indicating whether all lines have the same number of fields.

See Also

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

Examples

f <- tempfile()
write.csv2(mtcars, file = f)
check_ascii_file(f)
clipboard_path

Copy a Path from Clipboard to ‘R’

Description

I often have to work under Windows, where file paths cannot just be pasted into the code, so I adapted code from https://www.r-bloggers.com/2015/12/stop-fiddling-around-with-copied-paths-in-windows-r/. Under Windows, the de-windowsified path is copied to the clipboard.

Usage

clipboard_path()

Value

The de-windowsified path.

Note

It makes only sense to call clipboard_path in an interactive R session.

See Also

Other operating system functions: file_copy(), file_save(), get_boolean_envvar(), get_run_r_tests(), is_installed(), is_r_package_installed(), is_success(), is_windows(), view(), vim(), wipe_tempdir(), with_dir()

Other file utilities: delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, search_files(), split_code_file(), touch()

column_sums

Sum up the Numeric Columns of a Data Frame

Description

I often need to calculate the sums of the numeric columns of a data.frame. While colSums requires the data frame to be numeric, this is a convenience wrapper to select numeric columns only.

Usage

column_sums(x, ...)

compare_vectors

Arguments

x A data.frame.

... Arguments passed to colSums.

Value

A named vector of column sums (see colSums).

See Also

Other statistics: count_groups(), relative_difference(), round_half_away_from_zero(), weighted_variance()

Examples

try(colSums(iris))
column_sums(iris)
names(iris) # no column sum for 'Species'

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

compare_vectors(x, y, differences_only = FALSE)

Arguments

x, y Two vectors of the same mode.
differences_only

Value

A matrix containing the side-by-side comparison.

See Also

Other searching functions: file_modified_last(), find_files(), fromto(), grep_file(), missing_docs, search_files(), search_rows(), summary.filesearch() Other vector comparing functions: relative_difference()
convert_umlauts_to_ascii

Examples

```r
data(mtcars)
cars <- rownames(mtcars)
carz <- cars[!grepl("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

```
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 ä string")
print(string)
print(convert_umlauts_to_ascii(string))
string <- paste("this is ä string")
df <- data.frame(v1 = c(string, "foobar"),
                 v2 = c("foobour", string),
                 v3 = 3:4)
names(df)[3] <- "y\u00dfy"
convert_umlauts_to_ascii(df)
```
**convert_umlauts_to_tex**

Tex Codes for German Umlauts

**Description**

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

**Usage**

`convert_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))
```

---

**count_groups**

Count Observations per Groups

**Description**

I tend to forget the syntax that works with `stats::aggregate`.

**Usage**

`count_groups(x, ...)`

**Arguments**

- **x**: A `data.frame`.
- **...**: Columns in `x`. 
Value

A `data.frame` with the counts per groups.

See Also

Other statistics: `column_sums()`, `relative_difference()`, `round_half_away_from_zero()`, `weighted_variance()`

Examples

count_groups(mtcars, "am", "gear")
RUnit::checkEquals(dplyr::count(mtcars, am, gear),
count_groups(mtcars, "am", "gear"), checkNames = FALSE)

Description

Functions to read and write CSV files. The objects returned by these functions are `data.frame`s 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.csv`. It sets the above three arguments.

`write_csv` compares the hash value stored in the object’s attribute with the objects 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

```r
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.csv2` 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()`, `check_ascii_file()`, `csv2csv()`

Examples

```r
# 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

**Usage**

```r
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.
**delete_trailing_blank_lines**

**See Also**

Other CSV functions: `bulk_read_csv()`, `bulk_write_csv()`, `check_ascii_file()`, `csv`

**Examples**

```r
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)
```

---

** delete_trailing_blank_lines  
Remove Trailing Blank Lines From Files  

**Description**

Trailing blank lines are classical lints.

**Usage**

```r
delete_trailing_blank_lines(...)```

**Arguments**

... Arguments passed to `find_files`.

**Value**

Invisibly NULL.

**See Also**

Other file utilities: `clipboard_path()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

**Examples**

```r
dir <- tempfile()
dir.create(dir)
file.copy(system.file("tinytest", package = "fritools2"), dir, recursive = TRUE)
delete_trailing_blank_lines(path = dir, recursive = TRUE)
unlink(dir, recursive = TRUE)
```
### delete_trailing_whitespace

#### Remove Trailing Whitespace From Files

**Description**

Trailing whitespace is a classical lint.

**Usage**

```r
delete_trailing_whitespace(...)```

**Arguments**

```r
... Arguments passed to find_files.
```

**Value**

*Invisibly NULL.*

**See Also**

- Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

**Examples**

```r
dir <- tempfile()
dir.create(dir)
file.copy(system.file("tinytest", package = "fritools2"), dir,
    recursive = TRUE)
delete_trailing_whitespace(path = dir, recursive = TRUE)
unlink(dir, recursive = TRUE)
```

---

### develop_test

#### Develop Unit Testing for a Code File

**Description**

Looking at the output of `covr::zero_coverage`, I want to open a code file an the corresponding unit testing file.

**Usage**

```r
develop_test(file, force_runit = FALSE, force_tiny = TRUE)```
Arguments

file  The path to the code file, assuming the working directory to be the root of an R package under development.

force.runit  If there is no corresponding RUnit test file: create one?

force.tiny  If there is no corresponding tinytest test file: create one?

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()`, `run_r_tests_for_known_hosts()`, `set_run_r_tests()`

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

---

**file_copy**  
*Force Copying a File While Backing it up*

**Description**

`file.copy` has an argument `overwrite` that allows for overwriting existing files. But I often want to overwrite an existing file while creating a backup copy of that file.

**Usage**

`file_copy(from, to, stop_on_error = FALSE, ...)`

**Arguments**

from  See `file.copy`.

to  See `file.copy`.

stop_on_error  Throw an exception on error?

...  Arguments passed to `file.copy`.

**Value**

A vector of `boolean` values indicating success or failure.
See Also

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()

Other operating system functions: `clipboard_path()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()` , `with_dir()`

Examples

```r
touch(f1 <- file.path(tempdir(), "first.R"),
     f2 <- file.path(tempdir(), "second.R"))
dir.create(t <- file.path(tempdir(), "foo"))
file_copy(from = c(f2, f1), to = t)
dir(t)
touch(f1)
touch(f2)
file_copy(from = c(f2, f1), to = t)
dir(t)
list.files(tempdir(), pattern = "first.*\.R")
dir <- file.path(tempdir(), "subdir")
dir.create(dir)
file_copy(f1, dir)
touch(f1)
file_copy(f1, dir)
list.files(dir, pattern = "first.*\.R")
```

file_modified_last  Get the File Modified Last

Description

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

Usage

`file_modified_last(...)`

Arguments

...  Arguments passed to `find_files`.

Value

The path to the file last modified.
See Also

Other searching functions: `compare_vectors()`, `find_files()`, `grep_file()`, `missing_docs()`, `search_files()`, `search_rows()`, `summary.filesearch()`

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

Examples

```r
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)
```

---

**file_save**

Create a Copies of Files

Description

I often want a timestamped copies as backup of files or directories.

Usage

```r
file_save(
  ..., 
  file_extension_pattern = "\.[A-z]{1,5}$", 
  force = TRUE, 
  recursive = NA, 
  stop_on_error = TRUE, 
  overwrite = FALSE 
)
```

Arguments

```
... Paths to files.
file_extension_pattern   A Pattern to mark a file extension. If matched, the time stamp will get inserted before that pattern.
force   Force even if file_extension_pattern is not matched. Set to FALSE to skip stamping such files.
```
recursive Passed to `file.copy`. Defaults to ‘if the current path is a directory, then TRUE, else FALSE’.

stop_on_error Throw an exception on error?

overwrite Passed to `file.copy`.

Value
A vector of boolean values indicating success or failure.

See Also
Other operating system functions: `clipboard_path()`, `file_copy()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`, `with_dir()`.

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths()`, `search_files()`, `split_code_file()`, `touch()`.

Examples
```r
f1 <- tempfile()
f2 <- tempfile()
try(file_save(f1))
touch(f1)
file_save(f1, recursive = FALSE)
f2 <- paste0(file.path(tempfile()), ".txt")
touch(f2)
file_save(f1, f2)
file_save(f1, f2)
file_save(f1, f2, overwrite = TRUE)
dir(tempdir())
```

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
```r
find_files(
  path = ".",
  pattern = NULL,
  file_names = NA,
  all_files = TRUE,
)```
find_files

recursive = FALSE,
ignore_case = FALSE,
find_all = FALSE,
select = NA
)

Arguments

path see list.files.

pattern see list.files.

file_names character vector of file names (to be checked if the files exist).

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 exists. 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(), fromto(), grep_file(),
missing_docs, search_files(), search_rows(), summary.filesearch()

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(),
develop_test(), file_copy(), file_modified_last(), file_save(), get_lines_between_tags(),
get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths,
search_files(), split_code_file(), touch()
Examples

```r
#% 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 it's basename
find_files(path = tempdir(), pattern = paste0("^", basename(files[1]), "$"))
#% file_names and path given: file_names beats path
try(find_files(file_names = tempfile(), path = tempdir()))
##% select by file size:
write.csv(mtcars, file.path(tempdir(), "mtcars.csv"))
find_files(path = tempdir())
find_files(path = tempdir(),
    select = list(size = c(min = 1000))
)
```

---

**fromto**

*Extract All Items of a Vector Between Two Patterns*

**Description**

This comes in handy to cut lines from a file read by `readLines`.

**Usage**

```r
fromto(
    x,
    from,
    to,
    from_i = 1,
    to_i = 1,
    shift_from = 0,
    shift_to = 0,
    remove_empty_item = TRUE
)
```
get_boolean_envvar

Arguments

x             A vector.

from          A pattern, use NA to start with the first item.

to            Another pattern, use NA to stop with the last item.

from_i        If the from pattern matches multiple times, which one is to be used.

to_i          Analogously to to_i.

shift_from    The number of items to shift from the item selected via from and from_i.

shift_to      Analogously to shift_from.

remove_empty_item
Remove empty items?

Value

The extracted vector.

See Also

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

Examples

         rep("D", 4), "t3", "Last")

fromto(foo, "^f", "^t")

fromto(foo, NA, "^t")

fromto(foo, "^f", NA)

fromto(foo, "^f", "^t", from_i = 2)

fromto(foo, "^f", "^t", from_i = 2, to_i = 2)

fromto(foo, "^f", "^t", from_i = 2, to_i = 2, shift_from = 1, shift_to = -1)

fromto(foo, "^f", "^t", from_i = 2, to_i = 2, shift_from = -1, shift_to = 2)

get_boolean_envvar  Get a Boolean Environment Variable

Description

A convenience wrapper to Sys.getenv.

Usage

get_boolean_envvar(x, stop_on_failure = FALSE)
get_lines_between_tags

Cut Code Chunks From a File

Description

Get all lines between tagged lines. The tagged lines themselves may be in- or excluded from the selection.

Usage

```r
get_lines_between_tags(
  file_name,
  keep_tagged_lines = TRUE,
  begin_pattern = "ROXYGEN_START",
  end_pattern = "ROXYGEN_STOP",
  from_first_line = TRUE,
  to_last_line = TRUE
)
```

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: `develop_test()`, `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()`, `is_running_on_gitlab_com()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`, `with_dir()`

Examples

```r
message("See\n example("get_run_r_tests", package = "fritools2")")
```
get_mtime

Arguments

- `file_name`: The name of the R code file to be parsed.
- `keep_tagged_lines`: Keep tagged lines output?
- `begin_pattern`: A pattern that marks the line beginning a `roxygen2` chunk.
- `end_pattern`: A pattern that marks the line ending a `roxygen2` chunk.
- `from_first_line`: Use first line as tagged line if first tag found matches the `end_pattern`?
- `to_last_line`: Use last line as tagged line if last tag found matches the `begin_pattern`?

Value

A character vector of matching lines.

Note

If you know the file to contain valid `roxygen2` code only, you do not need to tag any lines if you keep from_first_line and to_last_line both TRUE: in this case the whole file will be returned.

See Also

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

---

get_mtime

Get the mtime Attribute from an Object

Description

We set modification times on some objects, this is a convenience wrappers to `attr`.

Usage

get_mtime(x)

Arguments

- `x`: An object.

Value

The value of `attr(attr(x, "path", "mtime")`. 
get_options

See Also

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(),
develop_test(), file_copy(), file_modified_last(), file_save(), find_files(), get_lines_between_tags(),
get_unique_string(), grep_file(), is_files_current(), is_path(), paths, search_files(),
split_code_file(), touch()

Examples

```r
x <- 2
path <- tempfile()
touch(path)
x <- set_path(x, path)
get_mtime(x)
```

get_options  Get Options For Packages

Description

A convenience function for `getOption`.

Usage

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

Arguments

```r
...  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()
get_package_version

Examples

```r
example("set_options", package = "fritools2")
```

---

**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 \texttt{R} Code File in Case of an \texttt{Rscript} Run*

**Description**

Retrieve the path from parsing the command line arguments of an \texttt{Rscript} run.

**Usage**

```r
get_rscript_script_path()
```

**Value**

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

**See Also**

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

**Examples**

```r
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**

```r
get_run_r_tests(stop_on_failure = FALSE)
```

**Arguments**

- `stop_on_failure`

  Throw an error instead of returning \texttt{FALSE} if the environment variable is not set or cannot be converted to boolean.

**Value**

The value \texttt{RUN_R_TESTS} is set to, converted to boolean. \texttt{FALSE} if \texttt{RUN_R_TESTS} is not set or cannot be converted to boolean.
get_r_cmd_batch_script_path

Description

Retrieve the path from parsing the command line arguments of an 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.
get_script_name

See Also

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

Examples

get_r_cmd_batch_script_path()

give_script_name(default = "/ nutritious.R")

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

give_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

give_script_name(default = 'foozbar.R')
### get_script_path

#### Description
This is just a wrapper for `get_rscript_script_path` and `get_r_cmd_batch_script_path`.

#### Usage
```r
get_script_path()
```

#### 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
```r
get_script_path()
```

### get_unique_string

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

#### Usage
```r
get_unique_string()
```

#### Value
A fairly unique string.

#### See Also
Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `grep_file()`, `is_files_current()`, `is_path()`, `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: is_difftime_less(), is_valid_primary_key(), r_cmd_install(), str2num(), strip_off_attributes(), tapply(), throw()

Examples

golden_ratio(10)

grep_file  Grep a Pattern from Files

Description
This is an approximation of the unix command grep.

Usage
grep_file(paths, pattern, a = 1, b = 1, ...)

Arguments

paths  A vector of file paths.
pattern  The pattern to grep.
a  Number of lines of trailing context before matching lines. Like grep’s -A option.
b  Number of lines of leading context before matching lines. Like grep’s -B option.
...  Arguments passed to list.files.

Value

A named list with one item per file path. Each item consists of a list of row numbers matching the pattern. Each item is a vector of the matching lines and b lines before and a lines after the matching lines.

See Also

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

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), is_files_current(), is_path(), paths, search_files(), split_code_file(), touch()

Examples

file_paths <- list.files(path = system.file("tinytest", package = "fritools2"), pattern = ".*\.R", full.names = TRUE)
res <- grep_file(path = file_paths, pattern = "forSureNotThere", a = 3, b = 2, ignore.case = TRUE)
tinytest::expect_true(all(res == FALSE))

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)
is_batch

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: `subset_sizes()`

Examples

```r
index_groups(n = 100, k = 6)
index_groups(n = 2, k = 6)
```

is_batch

Is ‘R’ Run in Batch Mode (via ‘R CMD BATCH’ or ‘Rscript’)?

Description

Just a wrapper to `interactive`.

Usage

`is_batch()`

Value

`TRUE` on success, `FALSE` otherwise.

See Also

Other logical helpers: `get_run_r_tests()`, `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()`, `is_windows()`

Examples

```r
is_batch()
```
is_cran

Is 'R' Running on CRAN?

Description

This is a verbatim copy of fda::CRAN of fda version 5.1.9.

Usage

is_cran(cran_pattern, n_r_check4cran)

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.
is_difftime_less

See Also

Other test helpers: `develop_test()`, `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()`

Examples

```r
if (!is_cran()) {
  message("Run your tests here."
}
```

is_difftime_less

Check Whether Two Times Differ Less Than A Given Value

Description

This is just a wrapper to `difftime`.

Usage

```r
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`. 
is_false

Value

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

See Also

Other bits and pieces: golden_ratio(), is_valid_primary_key(), r_cmd_install(), str2num(), strip_off_attributes(), tapply(), throw()

Examples

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_difftime_less(a, b)
is_difftime_less(a, c)
print(is_difftime_less(a, b, verbose = TRUE))
print(is_difftime_less(a, c, verbose = TRUE))
try(is_difftime_less(a, b, stop_on_error = TRUE))
is_difftime_less(a, c, verbose = TRUE, stop_on_error = TRUE)

---

is_false

Provide isFALSE for ‘R’ < 3.5.0

Description

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

Usage

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(), 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_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

```r
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.

See Also

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

Examples

```r
p1 <- tempfile()
p2 <- tempfile()
p3 <- tempfile()
touch(p1)
touch(p2)
Sys.sleep(3)
```
is_force

```
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_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()
```
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: `clipboard_path()`, `file_copy()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`, `with_dir()`.

Examples

```r
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

Is an Object Set and not Set to FALSE?

**Description**

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

**Usage**

```r
is_not_false(x, null_is_false = TRUE, ...)
```

**Arguments**

- `x`: The object to be tested.
- `null_is_false`: Should `NULL` be treated as `FALSE`?
- `...`: Parameters passed to `exists`. See Examples.

**Value**

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

**See Also**

Other logical helpers: `get_run_r_tests()`, `is_batch()`, `is_cran()`, `is_false()`, `is_force()`, `is_installed()`, `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
a <- 1
b <- FALSE
c <- NULL
is_not_false(a)
is_not_false(b)
is_not_false(c)
is_not_false(c, null_is_false = FALSE)
is_not_false(not_defined)
f <- function() {
  print(a)
  print(is_not_false(a))
}
f()

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

f <- function() {
  print(a)
  print(is_not_false(a, null_is_false = TRUE,
                  inherits = FALSE))
}

### We use this to check whether an option is set to something different than FALSE:
### Make sure an option is not set:
set_options("test" = NULL, package = "fritools2")
tmp <- get_options("test")
is_not_false(tmp)
is_not_false(tmp, null_is_false = FALSE)
# Does not work on the option directly as it is not an object defined:
options("foo" = NULL)
is_not_false(getOption("foo"), null_is_false = FALSE)
```

---

### is_null_or_true

**Is an Object TRUE or NULL?**

**Description**

Is an object **TRUE** or **NULL**?

**Usage**

```r
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_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `is_success()`, `is_version_sufficient()`, `is_windows()`
is_of_length_zero

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)

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_fvafrceu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(),
is_windows()

Examples

x <- ""; length(x); is_of_length_zero(x)
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_path

**Check Whether an Object Contains a Valid File System Path**

**Description**

Check Whether an Object Contains a Valid File System Path

**Usage**

```r
is_path(x)
```

**Arguments**

- `x`
  - The object.

**Value**

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

**See Also**

Other file utilities: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

**Examples**

```r
is_path(tempdir())
path <- tempfile()
is_path(path)
touch(path)
is_path(path)
```

---

### 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(type = c("any", "cu", "bwi", "fvafr"))
```
is_running_on_gitlab_com

Arguments

type          An optional selection.

Value

TRUE on success, FALSE otherwise.

See Also

Other test helpers: `develop_test()`, `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()`

Examples

is_running_on_fvafrcu_machines()

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(verbose = TRUE)

Arguments

verbose          Be verbose?

Value

TRUE on success, FALSE otherwise.
is_r_cmd_check

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: develop_test(), get_boolean_envvar(), get_run_r_tests(), is_cran(), is_r_cmd_check(), is_r_package_installed(), 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

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: develop_test(), get_boolean_envvar(), get_run_r_tests(), is_cran(), is_r_cmd_check(), is_r_package_installed(), 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.

**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: `clipboard_path()`, `file_copy()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`, `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("fritools2", "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: `clipboard_path()`, `file_copy()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`, `with_dir()`

**Examples**

```r
is_success(0)
is_success(1)
is_success(-1)
```

### is_valid_primary_key

**Is a Key a Valid Potential Primary Key for a data.frame?**

**Description**

I sometimes see tables with obscure structure so I try to guess their primary keys.

**Usage**

```r
is_valid_primary_key(data, key, verbose = TRUE)
```
**is_version_sufficient**

Is a Version Requirement Met?

**Description**

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

**Usage**

```r
is_version_sufficient(installed, required)
```

**Arguments**

- `installed` The version available.
- `required` The version required.

**Value**

`TRUE`, if so, `FALSE` otherwise.
is_windows

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

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: clipboard_path(), file_copy(), file_save(), get_boolean_envvar(), get_run_r_tests(), is_installed(), is_r_package_installed(), is_success(), view(), vim(), wipe_tempdir(), with_dir()

Examples

is_windows()
load_internal_functions

Load a Package’s Internals

Description
Load objects not exported from a package’s namespace.

Usage
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.

Value
Invisibly TRUE.

See Also
codetools::checkUsageEnv.
Other package functions: get_package_version(), is_r_package_installed(), is_version_sufficient()

Examples
load_internal_functions("fritools2")

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 = parent.frame()
)
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(), wipe_tempdir()

Examples

va <- rep(mtcars, 1)
vb <- rep(mtcars, 1000)
vcc <- rep(mtcars, 2000)
vd <- rep(mtcars, 100)
memory_hogs()
memory_hogs(unit = "Mb", decreasing = TRUE)
memory_hogs(unit = "Mb", decreasing = TRUE, return_numeric = FALSE)

Description

For fritools2, we make exhaustive use of categorizing functions into families with the ‘See also’ section of the man pages (which are generated by the @family tags in the code files).

Usage

find_missing_see_also(path, list_families = TRUE)

find_missing_family(path, list_families = TRUE, clean = TRUE)

Arguments

path Path to a (package) directory.
list_families List the function families defined so far.
clean Remove temporary directory?
**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, force = FALSE)

set_path(x, path, action = c(NA, "read", "write"), overwrite = FALSE)

**Arguments**

- **x**: An object.
- **force**: Force the retrieval, even if the path is not valid? Only meant for unit testing, leave alone!
- **path**: The path to be set.
- **action**: Do we have a read or write process? Passed by `read_csv` and `write_csv`. Leave alone otherwise.
- **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: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `search_files()`, `split_code_file()`, `touch()`
Examples

```r
x <- 2
path <- tempfile()
touch(path)
x <- set_path(x, path)
get_path(x)
```

relative_difference

Compute Relative Differences Between the Values of Two Vectors

Description

We often try to compare vectors on near equality. This is a wrapper to `all.equal` for our convenience. It also implements relative difference and change as discussed in [https://en.wikipedia.org/wiki/Relative_change_and_difference](https://en.wikipedia.org/wiki/Relative_change_and_difference).

Usage

```r
relative_difference(
  current,       
  reference,     
  type = c("all.equal", "difference", "change")
)
```

Arguments

- `current` One vector.
- `reference` Another vector, for type = `all.equal`, this is passed as `target`, for type = `all.equal` this can be thought of as the "correct" value or the state "before".
- `type` The method to be used. See Details.

Details

The default method (type = `all.equal`) applies `all.equal` onto the two vectors. Method type = difference is somewhat the same as the default, method type = change takes account of the sign of the differences.

Value

A vector of relative differences.

See Also

Other statistics: `column_sums()`, `count_groups()`, `round_half_away_from_zero()`, `weighted_variance()`

Other vector comparing functions: `compare_vectors()`
Examples

```r
n <- 500
x <- rnorm(n)
y <- x + rnorm(n, sd = 0.0001)
plot(relative_difference(x, y), x)
plot(relative_difference(x, y, "difference"), x)
# They do approximately the same:
max(relative_difference(relative_difference(x, y),
                         relative_difference(x, y, "difference")))
# Takes sign into account:
plot(relative_difference(x, y, "change"), x)
max(relative_difference(relative_difference(x, y),
                        abs(relative_difference(x, y, "change"))))
```

Description

Commercial rounding is done a lot, especially with invoices. There is even standard 1333 by the German Institute for Standardization. `round` rounds half to even, see `round`'s Details section.

`round_commercially` is just a link to `round_half_away_from_zero`.

Usage

```r
round_half_away_from_zero(x, digits = 0)

round_commercially(x, digits = 0)
```

Arguments

- `x`: A number to be rounded.
- `digits`: The number of digits, as in `round`.

Value

The rounded number.

See Also

Other statistics: `column_sums()`, `count_groups()`, `relative_difference()`, `weighted_variance()`
Examples

\[
\begin{align*}
x & \leftarrow 22.5 \\
\text{round} \_\text{half} \_\text{away} \_\text{from} \_\text{zero}(x) \\
\text{round}(x) \\
\text{round} \_\text{half} \_\text{away} \_\text{from} \_\text{zero}(-x) \\
\text{round}(-x)
\end{align*}
\]

```
run_r_tests_for_known_hosts
```

\textit{Force Testing on Known Hosts}

Description

Enforce the environment variable RUN\_R\_TESTS to TRUE on known hosts.

Usage

```r
run_r_tests_for_known_hosts()
```

Details

This should go into \texttt{onLoad} to force tests on known hosts.

Value

\texttt{Invisibly NULL}.

See Also

Other test helpers: \texttt{develop\_test()}, \texttt{get\_boolean\_envvar()}, \texttt{get\_run\_r\_tests()}, \texttt{is\_cran()}, \texttt{is\_r\_cmd\_check()}, \texttt{is\_running\_on\_fvafrcu\_machines()}, \texttt{is\_running\_on\_gitlab\_com()}, \texttt{set\_run\_r\_tests()}

Examples

```r
get\_run\_r\_tests()
if (is\_FALSE(get\_run\_r\_tests())) {
  run\_r\_tests\_for\_known\_hosts()
  get\_run\_r\_tests()
}
```
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(), fromto(), grep_file(), missing_docs, search_rows(), summary.filesearch()

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, split_code_file(), touch()

Examples

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

Examples

```r
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()`

---

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)
get_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: `develop_test()`, `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())
get_run_r_tests()
set_run_r_tests(TRUE, force = TRUE)
get_run_r_tests()
```
split_code_file

**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**

```r
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: `clipboard_path()`, `delete_trailing_blank_lines()`, `delete_trailing_whitespace()`, `develop_test()`, `file_copy()`, `file_modified_last()`, `file_save()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `touch()`

---

str2num

**Convert Character Numbers to Numeric**

**Description**

If you read text containing (possibly German, i.e. the decimals separated by comma and dots inserted for what they think of as readability) numbers, you may want to convert them to numeric.
strip_off_attributes

Usage

str2num(x)

Arguments

x  A string representing a (possibly German) number.

Value

The number as a numeric.

See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), r_cmd_install(), strip_off_attributes(), tapply(), throw()

Examples

line_in_text <- "foo bar 10.303,70 foo bar 1.211.000,55 foo bar"
words <- unlist(strsplit(line_in_text, split = " "))
print(na.omit(sapply(words, str2num)), digits = 9)
print(str2num(words[c(3, 4, 7)]), digits = 9)
print(str2num(words[7]), digits = 9)

strip_off_attributes

Strip Attributes off an Object

Usage

strip_off_attributes(x)

Arguments

x  An object.

Value

The object.

See Also

base::unname

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), r_cmd_install(), str2num(), tapply(), throw()
subset_sizes

Examples

y <- stats::setNames(1:3, letters[1:3])
attr(y, "myattr") <- "qwer"
comment(y) <- "qwer"
strip_off_attributes(y)


describe_subset_sizes <- subset_sizes

Determine Subset Sizes Close to Equality

Description

Determine the sizes of k subsets of a set with n elements in such a way that the sizes are as equal as possible.

Usage

subset_sizes(n, k)

Arguments

n The size of the set.

k The number of subsets.

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**

```r
define_S3_method('filesearch', summary, c("file", "what", "matches"))
```

**Arguments**

- **object**: An object returned by `search_files`.
- **type**: Type of summary.
- **...**: Needed for compatibility.

**Value**

A summarized object.

**See Also**

Other searching functions: `compare_vectors`, `file_modified_last`, `find_files`, `fromto`, `grep_file`, `missing_docs`, `search_files`, `search_rows`

**Examples**

```r
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$")
search_files(what = "[Ss]etosa", path = tempdir(), pattern = "^.*\.csv$")
x <- search_files(path = tempdir(),
  pattern = "^.*\.csv$",
  exclude = "[2-9]\..csv$",
  what = "[Ss]etosa")
summary(x)
summary(x, type = "what")
summary(x, type = "matches")
try(search_files(what = "ABC", path = tempdir(), pattern = "^.*\.csv$"))
```
**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()`, `is_difftime_less()`, `is_valid_primary_key()`, `r_cmd_install()`, `str2num()`, `strip_off_attributes()`, `throw()`

**Examples**

```r
result <- fritools2::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!")
} else
  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.frames:")
try(base::tapply(X = mtcars, INDEX = list(mtcars["cyl"], mtcars["vs"]),
  FUN = weighted_mean, x = "mpg", w = "wt"))
wm <- fritools2::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

---

**touch**

Mock the Unix touch Utility

**Description**

Creating files or ensuring that their file modification times change.
touch2 is an alternate - yet not faster - implementation.

**Usage**

touch(...)
touch2(...)

**Arguments**

... Paths to files.

**Value**

The Paths to the files touched.

**See Also**

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, search_files(), split_code_file()
Examples

```r
file1 <- tempfile()
file2 <- tempfile()
touch(file1, file2)
t1 <- file.mtime(file1, file2)
touch(file2)
t2 <- file.mtime(file1, file2)
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 object changed.

**Usage**

`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()`
view  View a File or Directory

Description
Call shell.exec on windows, mimic shell.exec otherwise.

Usage
view(path, program = NA)

Arguments
- path: A path to a file or directory.
- program: A program to use.

Value
Invisibly NULL.

See Also
Other operating system functions: clipboard_path(), file_copy(), file_save(), get_boolean_envvar(), get_run_r_tests(), is_installed(), is_r_package_installed(), is_success(), is_windows(), vim(), wipe_tempdir(), with_dir()

Examples
path <- file.path(tempdir(), "foo.txt")
writeLines(c("abc", "xyz"), con = path)
view(path)

vim  Edit a File With 'VIM' if Possible

Description
Just a wrapper to file.edit, trying to use [g]vim as editor, if installed.

Usage
vim(...)

Arguments
... See file.edit.
weighted_variance

Value

See `file.edit`.

See Also

Other operating system functions: `clipboard_path()`, `file_copy()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `wipe_tempdir()`, `with_dir()`.

Examples

```r
if (interactive()) {
  path <- file.path(tempdir(), "foo.txt")
  writeLines(c("abc", "xyz"), con = path)
  vim(path)
}
```

---

**weighted_variance**  
*Calculate a Weighted Variance*

Description

Calculate a weighted variance.

Usage

```r
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.

Value

A numeric giving the (weighted) variance of `x`.

See Also

Other statistics: `column_sums()`, `count_groups()`, `relative_difference()`, `round_half_away_from_zero()`

Examples

```r
## 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:
frtools2::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 clean. This is similar to the broom button in RStudio.

Usage

```r
wipe_clean(environment = getOption("wipe_clean_environment"), all_names = TRUE)
```

Arguments

- `environment`: The environment that should be wiped clean.
- `all_names`: See argument `all.names` for `ls`.
**wipe_tempdir**

**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*, *wipe_tempdir*

**Examples**

```r
an_object <- 1
wipe_clean()
ls()
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)
```

**Description**

I often need a clean temporary directory.

**Usage**

```r
wipe_tempdir(recreate = FALSE)
```

**Arguments**

- **recreate**
  Use the method described in the examples section of *tempdir* (using tempdir(check = TRUE), this results in a new path.)

**Value**

The path to the temporary directory.

**See Also**

Other R memory functions: *memory_hogs*, *wipe_clean*

Other operating system functions: *clipboard_path*, *file_copy*, *file_save*, *get_boolean_envvar*, *get_run_r_tests*, *is_installed*, *is_r_package_installed*, *is_success*, *is_windows*, *view*, *vim*, *with_dir*
with_dir  

**Execute Code in a Temporary Working Directory**

**Description**

This is a verbatim copy of `withr::with_dir` from `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**

```r
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: `clipboard_path()`, `file_copy()`, `file_save()`, `get_boolean_envvar()`, `get_run_r_tests()`, `is_installed()`, `is_r_package_installed()`, `is_success()`, `is_windows()`, `view()`, `vim()`, `wipe_tempdir()`

**Examples**

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

* CSV functions
  bulk_read_csv, 4
  bulk_write_csv, 5
  check_ascii_file, 8
  csv, 13
  csv2csv, 14
* German umlaut converters
  convert_umlauts_to_ascii, 11
  convert_umlauts_to_tex, 12
* R memory functions
  memory_hogs, 51
  wipe_clean, 70
  wipe_tempdir, 71
* bits and pieces
  golden_ratio, 32
  is_difftime_less, 36
  is_valid_primary_key, 48
  str2num, 61
  strip_off_attributes, 62
  tapply, 65
* call functions
  call_conditionally, 6
  call_safe, 7
* file utilities
  clipboard_path, 9
  delete_trailing_blank_lines, 15
  delete_trailing_whitespace, 16
  develop_test, 16
  file_copy, 17
  file_modified_last, 18
  file_save, 19
  find_files, 20
  get_lines_between_tags, 24
  get_mtime, 25
  get_unique_string, 31
  grep_file, 32
  is_files_current, 38
  is_path, 44
  paths, 53
  search_files, 57
  split_code_file, 61
  touch, 66
* hash functions for objects
  set_hash, 58
  un_hash, 67
* logical helpers
  get_run_r_tests, 28
  is_batch, 34
  is_cran, 35
  is_false, 37
  is_force, 39
  is_installed, 40
  is_not_false, 41
  is_null_or_true, 42
  is_of_length_zero, 43
  is_r_cmd_check, 46
  is_r_package_installed, 47
  is_running_on_fvafrcu_machines, 44
  is_running_on_gitlab_com, 45
  is_success, 48
  is_version_sufficient, 49
  is_windows, 50
* operating system functions
  clipboard_path, 9
  file_copy, 17
  file_save, 19
  get_boolean_envvar, 23
  get_run_r_tests, 28
  is_installed, 40
  is_r_package_installed, 47
  is_success, 48
  is_windows, 50
  view, 68
  vim, 68
  wipe_tempdir, 71
  with_dir, 72
* option functions
  get_options, 26
is_force, 39
set_options, 59

* package functions
    get_package_version, 27
    is_r_package_installed, 47
    is_version_sufficient, 49
    load_internal_functions, 51

* package
    fritools2-package, 3

* script path getter functions
    get_r_cmd_batch_script_path, 29
    get_rscript_script_path, 28
    get_script_name, 30
    get_script_path, 31

* searching functions
    compare_vectors, 10
    file_modified_last, 18
    fromto, 22
    grep_file, 32
    missing_docs, 52
    search_files, 57
    search_rows, 58
    summary.filesearch, 64

* statistics
    column_sums, 9
    count_groups, 12
    relative_difference, 54
    round_half_away_from_zero, 55
    weighted_variance, 69

* subsetting functions
    index_groups, 33
    subset_sizes, 63

* test helpers
    develop_test, 16
    get_boolean_envvar, 23
    get_run_r_tests, 28
    is_cran, 35
    is_r_cmd_check, 46
    is_running_on_fvafrcu_machines, 44
    is_running_on_gitlab_com, 45
    run_r_tests_for_known_hosts, 56
    set_run_r_tests, 60

* vector comparing functions
    compare_vectors, 10
    relative_difference, 54

* version functions
    get_package_version, 27
    is_r_package_installed, 47
    is_version_sufficient, 49
    .onLoad, 56
    all.equal, 54
    attr, 25, 53
    base::tapply, 65
    base::unname, 62
    boolean, 17, 20
    bulk_read_csv, 4, 5, 8, 14, 15
    bulk_write_csv, 4, 5, 8, 14, 15
    call_conditionally, 6, 7, 8
    call_safe, 7.7
    check_ascii_file, 4, 5, 8, 14, 15
    class, 24
    clipboard_path, 9, 15–21, 24–26, 29, 31, 33, 38, 40, 44, 47, 48, 50, 53, 57, 61, 66, 68, 69, 71, 72
    codetools::checkUsageEnv, 51
    colSums, 9, 10
    column_sums, 9, 13, 54, 55, 70
    compare_vectors, 10, 19, 21, 23, 33, 53, 54, 57, 58, 64
    compareVersion, 27, 49
    convert_umlauts_to_ascii, 11, 12
    convert_umlauts_to_tex, 11, 12
    count_groups, 10, 12, 54, 55, 70
    covr::zero_coverage, 16
    csv, 4, 5, 8, 13, 15
csv2csv, 4, 5, 8, 14, 15

data.frame, 9, 10, 12, 13, 58, 65, 69, 70
delete_trailing_blank_lines, 9, 15, 16–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
delete_trailing_whitespace, 9, 15, 16, 17–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
develop_test, 9, 15, 16, 18–21, 24–26, 29, 31, 33, 36, 38, 44–46, 53, 56, 57, 60, 61, 66
difftime, 36, 38
do.call, 6, 7
exists, 41
FALSE, 19, 24, 28, 34, 36–38, 40–50, 52
file.copy, 17, 20
file.exists, 21
file_copy, 9, 15–17, 17, 19–21, 24–26, 29, 31, 33, 38, 40, 44, 47, 48, 50, 53, 57, 61, 66, 68, 69, 71, 72
file_modified_last, 9, 10, 15–18, 18, 20, 21, 23, 25, 26, 31, 33, 38, 44, 53, 57, 58, 61, 64, 66
file_save, 9, 15–19, 19, 21, 24–26, 29, 31, 33, 38, 40, 44, 47, 48, 50, 53, 57, 61, 66, 68, 69, 71, 72
find_files, 4, 9, 10, 15–20, 20, 23, 25, 26, 31, 33, 38, 44, 53, 57, 58, 61, 64, 66
find_missing_family (missing_docs), 52
find_missing_see_also (missing_docs), 52
fritools2–package, 3
fromto, 10, 19, 21, 22, 33, 53, 57, 58, 64
get_boolean_envvar, 9, 17, 18, 20, 23, 28, 29, 36, 40, 45–48, 50, 56, 60, 68, 69, 71, 72
get_lines_between_tags, 9, 15–21, 24, 26, 31, 33, 38, 44, 53, 57, 61, 66
get_mtime, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
get_options, 26, 39, 59
get_package_version, 35, 47, 50, 51
get_path (paths), 53
get_r_cmd_batch_script_path, 28, 29, 30, 31
get_rscript_script_path, 28, 30, 31
get_run_r_tests, 9, 17, 18, 20, 24, 28, 34, 36, 37, 39–43, 45–48, 50, 56, 60, 68, 69, 71, 72
get_script_name, 28, 30, 31
get_script_path, 28, 30, 31
get_unique_string, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
getOption, 26
golden_ratio, 32, 37, 49, 62, 65
grep_file, 9, 10, 15–21, 23, 25, 26, 31, 32, 38, 44, 53, 57, 58, 61, 64, 66
index_groups, 33, 63
integer, 43
interactive, 34
invisible, 36
Invisibly, 14–17, 51, 56, 57, 59, 61, 68
is_batch, 29, 34, 36, 37, 39–43, 45–48, 50
is_cran, 17, 24, 29, 34, 35, 37, 39–43, 45–48, 50, 56, 60
is_difftime_less, 32, 36, 49, 62, 65
is_false, 29, 34, 36, 37, 39–43, 45–48, 50
is_files_current, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
is_force, 26, 29, 34, 36, 37, 39, 40–43, 45–48, 50, 59
is_installed, 9, 18, 20, 24, 29, 34, 36, 37, 39, 40, 41–43, 45–48, 50, 68, 69, 71, 72
is_not_false, 29, 34, 36, 37, 39, 40, 41, 42, 43, 45–48, 50
is_null_or_true, 29, 34, 36, 37, 39–41, 42, 43, 45–48, 50
is_of_length_zero, 29, 34, 36, 37, 39–42, 43, 45–48, 50
is_path, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57, 61, 66
is_r_cmd_check, 17, 24, 29, 34, 36, 37, 39–43, 45, 46, 47, 48, 50, 56, 60
is_r_package_installed, 9, 18, 20, 24, 27, 29, 34, 36, 37, 39–43, 45, 46, 47, 48, 50, 51, 68, 69, 71, 72
is_running_on_fvafrcu_machines, 17, 24, 29, 34, 36, 37, 39–43, 44, 46–48, 50, 56, 60
is_running_on_gitlab_com, 17, 24, 29, 34, 36, 37, 39–43, 45, 46–48, 50, 56, 60
is_success, 9, 18, 20, 24, 29, 34, 36, 37, 39–43, 45–48, 50, 68, 69, 71, 72
is_valid_primary_key, 32, 37, 48, 62, 65
is_version_sufficient, 27, 29, 34, 36, 37, 39–43, 45–48, 49, 50, 51
is_windows, 9, 18, 20, 24, 29, 34, 36, 37, 39–43, 45–48, 50, 50, 68, 69, 71, 72
length, 30, 31
list.files, 20, 21, 33, 57
load_internal_functions, 27, 47, 50, 51
ls, 51, 70
matrix, 58
memory_hogs, 51, 71
missing_docs, 10, 19, 21, 23, 33, 52, 57, 58, 64
mode, 28–31
INDEX

NULL, 15–17, 39, 41, 42, 56, 60, 68
options, 59
order, 52
package_version, 27
packageDescription, 27
packageVersion, 27
paths, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57,
61, 66
r_cmd_install, 32, 37, 49, 62, 65
read_csv, 4, 14, 53
read_csv(csv), 13
readLines, 22
relative_difference, 10, 13, 54, 55, 70
round, 55
round_commercially
  (round_halfawayfromzero), 55
round_halfawayfromzero, 10, 13, 54, 55,
70
run_r_tests_for_known_hosts, 17, 24, 29,
36, 43, 46, 56, 60
search_files, 9, 10, 15–21, 23, 25, 26, 31,
33, 38, 44, 53, 57, 58, 61, 64, 66
search_rows, 10, 19, 21, 23, 33, 53, 57, 58, 64
set_hash, 58, 67
set_options, 26, 39, 59
set_path(paths), 53
set_run_r_tests, 17, 24, 29, 36, 45, 46, 56,
60
source, 61
split_code_file, 9, 15–21, 25, 26, 31, 33,
38, 44, 53, 57, 61, 66
stats::aggregate, 12
str2num, 32, 37, 49, 61, 62, 65
strip_off_attributes, 32, 37, 49, 62, 62, 65
subset_sizes, 33, 34, 63
summary.filesearch, 10, 19, 21, 23, 33, 35,
57, 58, 64
Sys.getenv, 23, 24, 60
tapply, 32, 37, 49, 62, 65, 70
tempdir, 71
throw, 32, 37, 49, 62, 65
touch, 9, 15–21, 25, 26, 31, 33, 38, 44, 53, 57,
61, 66
touch2(touch), 66
TRUE, 6, 34, 37–51, 59
tryCatch, 6

un_hash, 59, 67
utils::read.csv, 13
utils::read.csv2, 13
utils:read.csv, 13
utils:read.csv2, 13

vector, 69
view, 9, 18, 20, 24, 29, 40, 47, 48, 50, 68, 69,
71, 72
vim, 9, 18, 20, 24, 29, 40, 47, 48, 50, 68, 68,
71, 72

weighted_variance, 10, 13, 54, 55, 69
wipe_clean, 52, 70, 71
wipe_tempdir, 9, 18, 20, 24, 29, 40, 47, 48,
50, 52, 68, 69, 71, 71, 72
with_dir, 9, 18, 20, 24, 29, 40, 47, 48, 50, 68,
69, 71, 72
write_csv, 5, 14, 53
write_csv(csv), 13