Package ‘fritools’

November 3, 2026

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

Version 4.7.0

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, microbenchmark, pkgload, reshape, rmarkdown, RUnit, testthat (>= 3.0.0), tinytest, whoami

Encoding UTF-8

Language en-US

VignetteBuilder utils

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Andreas Dominik Cullmann [aut, cre]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository CRAN

Date/Publication 2026-11-03 14:00:02 UTC

Contents

fritools-package 4
bulk_read_csv 4

bulk_write_csv . . . 6
call_conditionally . . . 7
call_safe . . . 8
char2factor . . . 8
check_ascii_file . . . 9
clipboard_path . . . 10
columns2numeric . . . 10
column_sums . . . 11
compare_vectors . . . 12
convert_umlauts_to_ascii . . . 13
convert_umlauts_to_tex . . . 14
convert_umlauts_to_utf8 . . . 14
count_groups . . . 15
csv . . . 16
csv2csv . . . 17
delete_trailing_blank_lines . . . 18
delete_trailing_whitespace . . . 19
develop_test . . . 19
escape_non_ascii . . . 20
file_copy . . . 21
file_modified_last . . . 22
file_save . . . 23
file_string . . . 24
find_files . . . 25
fromto . . . 27
get_boolean_envvar . . . 28
get_german_umlauts . . . 29
get_lines_between_tags . . . 30
get_mtime . . . 31
get_options . . . 31
get_package_version . . . 32
get_rscript_script_path . . . 33
get_run_r_tests . . . 33
get_r_cmd_batch_script_path . . . 34
get_script_name . . . 35
get_script_path . . . 36
get_session_string . . . 36
get_unique_string . . . 37
golden_ratio . . . 37
grep_file . . . 38
index_groups . . . 39
is_batch . . . 40
is_cran . . . 40
is_difftime_less . . . 42
is_false . . . 43
is_files_current . . . 44
is_force . . . 45
is_installed . . . 46

is_not_false . . . 47
is_null_or_true . . . 48
is_of_length_zero . . . 49
is_path . . . 50
is_running_on_fvafrcu_machines . . . 50
is_running_on_gitlab_com . . . 51
is_r_cmd_check . . . 52
is_r_package_installed . . . 53
is_scalar . . . 54
is_scalar_convertible2numeric . . . 55
is_success . . . 56
is_true . . . 57
is_valid_primary_key . . . 58
is_version_sufficient . . . 58
is_windows . . . 59
load_internal_functions . . . 60
memory_hogs . . . 61
missing_docs . . . 62
paths . . . 62
pause . . . 63
powers_of_ten . . . 64
relative_difference . . . 65
round_half_away_from_zero . . . 66
rownames2col . . . 67
runsed . . . 68
run_r_tests_for_known_hosts . . . 69
search_files . . . 70
search_rows . . . 71
set_hash . . . 71
set_options . . . 72
set_run_r_tests . . . 73
sloboda . . . 74
split_code_file . . . 75
str2num . . . 76
string2words . . . 76
strip_off_attributes . . . 77
subset_sizes . . . 78
summary.filesearch . . . 79
tapply . . . 80
touch . . . 81
un_hash . . . 82
view . . . 83
vim . . . 83
weighted_variance . . . 84
wipe_clean . . . 85
wipe_tempdir . . . 86
with_dir . . . 87

Index **88**

fritools-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 = "fritools")`.

Author(s)

Maintainer: Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Authors:

- Andreas Dominik Cullmann <fvafrcu@mailbox.org>

See Also

Useful links:

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

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

```
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(), csv, csv2csv()

Examples

```
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda R\u00f64"
names(mt_german)[1] <- "mg\u00dc"
#% 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

```
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(), csv, csv2csv()

Examples

```
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda R\u00f64"
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 can not `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")
```

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

Description

Just a specialized version of `call_conditionally`.

Usage

```
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`.

See Also

Other call functions: `call_conditionally()`

Examples

```
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")
```

char2factor — *Convert a Character Vector Into a Factor*

Description

I often need a factor with levels the unique values of a character vector (for example: to prevent ggplot2 from sorting the character vector).

Usage

```
char2factor(x, levels = unique(x))
```

Arguments

x	A character vector.
levels	The levels to use, see factor.

Value

A factor.

See Also

Other vector functions: escape_non_ascii(), file_string(), powers_of_ten

Examples

```
x <- c("beech", "oak", "spruce", "fir")
char2factor(x)
```

---

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(), csv, csv2csv()

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 which is down now. 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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

columns2numeric *Check Whether a Scalar is Convertible to Numeric*

Description

Check Whether a Scalar is Convertible to Numeric

Usage

```
columns2numeric(x)
```

Arguments

x	A data.frame.

Value

A data.frame.

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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), is_version_sufficient(), is_windows()

Other data.frame helpers: powers_of_ten

Examples

```
df <- data.frame(a = 1:24,
                 b = as.character(1:24),
                 c = paste(" ", 1:24, " "),
                 d = LETTERS[1:24]
                 )
df[2, 2]  <- df[3, 3] <- ""
df[3, 2]  <- df[4, 3] <- " "
columns2numeric(df)
```

---

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, ...)
```

Arguments

x	A data.frame.
...	Arguments passed to colSums.

Value

A named vector of column sums (see colSums).

See Also

Other statistics: count_groups(), powers_of_ten, relative_difference(), round_half_away_from_zero(), sloboda(), 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`	Report only the differences?

Value

A matrix containing the side-by-side comparison.

See Also

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

Other vector comparing functions: `relative_difference()`

Examples

```
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

```
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()`, `convert_umlauts_to_utf8()`, `get_german_umlauts()`

Examples

```
string <- "this is \u00e4 string"
print(string)
print(convert_umlauts_to_ascii(string))
string <- "this is \u00e4 string"
df <- data.frame(v1 = c(string, "foobar"),
                 v2 = c("foobar", 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(), convert_umlauts_to_utf8(), get_german_umlauts()

Examples

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

convert_umlauts_to_utf8 *Convert German Umlauts to a More or Less Suitable 'utf8' Representation*

Description

Convert German Umlauts to a More or Less Suitable 'utf8' Representation

Usage

```
convert_umlauts_to_utf8(x)

## S3 method for class 'character'
convert_umlauts_to_utf8(x)

## S3 method for class 'data.frame'
convert_umlauts_to_utf8(x)
```

Arguments

x	A string or `data.frame`.

Value

x with the umlauts converted to utf8.

See Also

Other German umlaut converters: `convert_umlauts_to_ascii()`, `convert_umlauts_to_tex()`, `get_german_umlauts()`

Examples

```
string <- "_(\xdcLH)"
print(string)
print(convert_umlauts_to_utf8(string))
string <- "this is _(\xdcLH) string"
df <- data.frame(v1 = c(string, "foobar"),
                 v2 = c("foobar", string), v3 = 3:4)
names(df)[3] <- "y_(\xdcLH)"
convert_umlauts_to_utf8(df)
convert_umlauts_to_ascii(convert_umlauts_to_utf8(df))
```

---

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(), powers_of_ten, relative_difference(), round_half_away_from_zero(), sloboda(), weighted_variance()

Examples

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

---

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.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

```
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()`, `check_ascii_file()`, `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

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(), check_ascii_file(), 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)
```

---

delete_trailing_blank_lines

Remove Trailing Blank Lines From Files

Description

Trailing blank lines are classical lints.

Usage

```
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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

Examples

```
dir <- tempfile()
dir.create(dir)
file.copy(system.file("tinytest", package = "fritools"), 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

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

Arguments

...	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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

Examples

```
dir <- tempfile()
dir.create(dir)
file.copy(system.file("tinytest", package = "fritools"), 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 and the corresponding unit testing file.

Usage

```
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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

---

escape_non_ascii *Escape non-ASCII Characters*

Description

I often get code with german umlauts that need to be escaped.

Usage

```
escape_non_ascii(x)
```

Arguments

x	A character vector.

Value

A character vector.

See Also

Other vector functions: char2factor(), file_string(), powers_of_ten

Examples

```
x <- c("foo", "djörman", "bar", "djörman bar")
escape_non_ascii(x)
# change file
f <- tempfile()
writeLines(x, f)
writeLines(escape_non_ascii(readLines(f)), f)
```

`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()`, `file_string()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `runsed()`, `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

```
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(), fromto(), grep_file(), missing_docs, runsed(), search_files(), search_rows(), summary.filesearch()

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

`file_save` *Create a Copies of Files*

Description

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

Usage

```
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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

Examples

```
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())
```

---

file_string *Substitute All Blanks and Punctuations in a String with an Underscore*

Description

Need to store stuff on disk. Replacement may also be a minus sign instead of underscore.

Usage

```
file_string(x, replacement = c("_", "-"))
```

Arguments

x	A string.
replacement	The replacement character.

Value

A 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(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

Other vector functions: char2factor(), escape_non_ascii(), powers_of_ten

Examples

```
file_string("foo:bar$ this, indeed(!) is # a number 7")
file_string("foo:bar$ this, indeed(!) is # a number 7", replacement = "-")
```

---

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(
  path = ".",
  pattern = NULL,
  file_names = NA,
  all_files = TRUE,
  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`, `runsed()`, `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()`, `file_string()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `runsed()`, `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 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

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

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 from_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`, `runsed()`, `search_files()`, `search_rows()`, `summary.filesearch()`

Examples

```
foo <- c("First", "f1", "A", "f2", rep("B", 4), "t1", "f3", "C", "t2",
         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)
```

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_mac
is_running_on_gitlab_com(), run_r_tests_for_known_hosts(), set_run_r_tests()

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

Examples

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

`get_german_umlauts` *Get German Umlauts*

Description

I often need German umlauts in reporting. So I need either a UTF-8 or LaTeX representation.

Usage

```
get_german_umlauts(
  which = NULL,
  type = c("utf-8", "latex"),
  strip_names = TRUE
)
```

Arguments

`which`	A character vector specifying a subset of the result vector.
`type`	UTF-8 or LaTeX?
`strip_names`	Return an unnamed vector?

Value

A (possibly named) vector of UTF-8 representations of german umlauts.

See Also

Other German umlaut converters: `convert_umlauts_to_ascii()`, `convert_umlauts_to_tex()`, `convert_umlauts_to_utf8()`

Examples

```
get_german_umlauts()
get_german_umlauts(type = "latex")
get_german_umlauts(strip_names = FALSE)
get_german_umlauts(which = c("sz", "Ae"))
try(get_german_umlauts(which = c("sz", "foo", "Ae", "bar")))
paste0("Cologne is K", get_german_umlauts("oe"), "ln. In LaTeX it's K",
       get_german_umlauts("oe", "latex"), "ln")
```

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

```
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

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(), file_string(), find_files(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), 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").

See Also

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

Examples

```
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

```
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

```
example("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

```
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: get_session_string(), is_r_package_installed(), is_version_sufficient()

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

Examples

```
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.

See Also

Other test helpers: develop_test(), get_boolean_envvar(), 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: clipboard_path(), file_copy(), file_save(), get_boolean_envvar(), is_installed(), is_r_package_installed(), is_success(), is_windows(), view(), vim(), wipe_tempdir(), with_dir()

Other logical helpers: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), is_version_sufficient(), is_windows()

Examples

```
set_run_r_tests("", force = TRUE) # make sure it is not set.
get_run_r_tests()
try(get_run_r_tests(stop_on_failure = TRUE))
set_run_r_tests("A", force = TRUE) # "A" is not boolean.
get_run_r_tests()
try(get_run_r_tests(stop_on_failure = TRUE))
set_run_r_tests(4213, force = TRUE) # All numbers apart from 0 are TRUE
get_run_r_tests()
set_run_r_tests("0", force = TRUE) # 0 (and "0") is FALSE
get_run_r_tests()
set_run_r_tests("FALSE", force = TRUE)
get_run_r_tests()
set_run_r_tests(TRUE, force = TRUE)
get_run_r_tests()
```

---

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()

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()
```

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_session_string *Get a Session String*

Description

I sometimes wan't to document the R session used in a string, so a need an excerpt of `sessionInfo` an `Sys.time`.

Usage

```
get_session_string()
```

Value

An excerpt of `sessionInfo` as a string.

See Also

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

Examples

```
get_session_string()
```

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: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), grep_file(), is_files_current(), is_path(), paths, runsed(), 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(), pause(), r_cmd_install(), rownames2col(), str2num(), string2words(), 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, runsed(), 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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file(), touch()

Examples

```
file_paths <- list.files(path = system.file("tinytest",
                                            package = "fritools"),
                         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)
```

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

```
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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

```
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 `i` per step 3 above.

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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

```
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

```
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 bits and pieces: `golden_ratio()`, `is_valid_primary_key()`, `pause()`, `r_cmd_install()`, `rownames2col()`, `str2num()`, `string2words()`, `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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `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

```
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()`, `file_string()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_path()`, `paths`, `runsed()`, `search_files()`, `split_code_file()`, `touch()`

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_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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

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

`is_installed` *Is an External Program 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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `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

```
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 skipt this example on unkown 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

```
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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

```
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)
    print(is_not_false(a))
}
f()

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

```
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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `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)
```

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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeri`, `is_success()`, `is_true()`, `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

```
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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), paths, runsed(), search_files(), split_code_file(), touch()

Examples

```
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

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

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: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), 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.

See Also

Other logical helpers: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), is_version_sufficient(), is_windows()

Other test helpers: develop_test(), 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

```
is_r_cmd_check()
```

Value

TRUE on success, FALSE otherwise.

See Also

Other logical helpers: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), is_version_sufficient(), is_windows()

Other test helpers: develop_test(), 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

```
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: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `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()`, `rename_package()`

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

Examples

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

`is_scalar` *Check Whether an R Object is Scalar*

Description

R is vector based. But I often come across vectors of length 1 or arrays and matrices with a single element.

Usage

```
is_scalar(x)
```

Arguments

x	An R object.

Value

A boolean.

See Also

Other logical helpers: `columns2numeric()`, `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_scalar_convertible2numeric()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

```
x <- "C"
is_scalar(x)
x <- LETTERS[1:24]
!is_scalar(x)
is_scalar(x[3])
dim(x) <- c(6, 4)
!is_scalar(x)
is_scalar(x[1, 2])
dim(x) <- c(2, 3, 4)
!is_scalar(x)
is_scalar(x[1, 2, 3])
is_scalar(list(1))
```

`is_scalar_convertible2numeric` *Check Whether a Scalar is Convertible to Numeric*

Description

Check Whether a Scalar is Convertible to Numeric

Usage

```
is_scalar_convertible2numeric(x)
```

Arguments

x	A Scalar.

Value

A boolean.

See Also

Other logical helpers: `columns2numeric()`, `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_scalar()`, `is_success()`, `is_true()`, `is_version_sufficient()`, `is_windows()`

Examples

```
x <- "3"
tinytest::expect_true(is_scalar_convertible2numeric(as.vector(x)))
tinytest::expect_true(is_scalar_convertible2numeric(as.list(x)))
tinytest::expect_true(is_scalar_convertible2numeric(as.array(x)))
tinytest::expect_true(is_scalar_convertible2numeric(as.matrix(x)))
x <- as.character(1:24)
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_true(is_scalar_convertible2numeric(x[3]))
tinytest::expect_true(all(sapply(x, is_scalar_convertible2numeric)))
dim(x) <- c(6, 4)
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_true(is_scalar_convertible2numeric(x[1, 2]))
dim(x) <- c(2, 3, 4)
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_true(is_scalar_convertible2numeric(x[1, 2, 3]))

x <- LETTERS[1:24]
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_false(is_scalar_convertible2numeric(x[3]))
dim(x) <- c(6, 4)
```

```
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_false(is_scalar_convertible2numeric(x[1, 2]))
dim(x) <- c(2, 3, 4)
tinytest::expect_error(is_scalar_convertible2numeric(x))
tinytest::expect_false(is_scalar_convertible2numeric(x[1, 2, 3]))
```

`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

```
is_success(x)
```

Arguments

`x`	The external commands return value.

Value

`TRUE` on success, `FALSE` otherwise.

See Also

Other logical helpers: `columns2numeric()`, `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_scalar()`, `is_scalar_convertible2numeric()`, `is_true()`, `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

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

is_true *Convert a* logical *Array to a Binary Boolean Array*

Description

I often use mathematical expressions to index data by its values. But when the data contain missing values, the logical indices do not index the data, so I need to convert them to boolean.

Usage

```
is_true(x)
```

Arguments

x	An logical array, probably the result of some kind of mathematical expression.

Value

A binary boolean array indicating where the logical array is TRUE.

See Also

Other logical helpers: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_version_sufficient(), is_windows()

Examples

```
x <- array(1:24, dim = c(2,3,4))
x[2,2,3] <- NA
print(x)
x < 20 # An array containing NA
is_true(x < 20) # An array boolean only, NA converted to FALSE
print(x <- x[2, TRUE, TRUE])
is_true(x < 20) # A matrix
x <- x[2, TRUE]
is_true(x < 20) # A vector
x <- x[3]
is_true(x < 20) # A scalar
```

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

```
is_valid_primary_key(data, key, verbose = TRUE)
```

Arguments

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

Value

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

See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), pause(), r_cmd_install(), rownames2col(), str2num(), string2words(), strip_off_attributes(), tapply(), throw()

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: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), is_windows()

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

Other version functions: get_package_version(), get_session_string(), 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: columns2numeric(), 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_scalar(), is_scalar_convertible2numeric(), is_success(), is_true(), 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()`, `rename_package()`

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 = 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)
vc <- rep(mtcars, 2000)
vd <- rep(mtcars, 100)
memory_hogs()
memory_hogs(unit = "Mb", decreasing = TRUE)
memory_hogs(unit = "Mb", decreasing = TRUE, return_numeric = FALSE)
```

missing_docs *Find Missing Documentation*

Description

For **fritools**, 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?

Value

For 'find_missing_see_also': a character vector of man pages with missing 'See also' sections.

For 'find_missing_family': a character vector of function names with missing '@family' tags.

See Also

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

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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), runsed(), search_files(), split_code_file(), touch()

Examples

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

---

pause *Pause*

Description

Pause

Usage

```
pause()
```

Value

A data.frame.

Invisibly NULL.

See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), r_cmd_install(), rownames2col(), str2num(), string2words(), strip_off_attributes(), tapply(), throw()

Examples

```
pause()
```

---

powers_of_ten *Conversions from or to powers of ten.*

Description

I often need to table big numbers in non-scientific notation.

Usage

```
convert_to_power_of_ten(x, exponent = NULL)

convert_from_power_of_ten(x)

df_to_powers_of_ten(x, is_individual = FALSE)

df_from_powers_of_ten(x)
```

Arguments

x	A data frame with attributed numeric columns.
exponent	Specify an exponent instead of deriving one from the data.
is_individual	Use individual powers of ten for each numeric column in x? But why would you want to?

Value

An attributed numeric vector.

A numeric vector.

A data frame with attributed numeric columns.

A data frame.

See Also

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

Other vector functions: char2factor(), escape_non_ascii(), file_string()

Other data.frame helpers: columns2numeric()

Examples

```
# Using vectors

print(x <- (5 + rnorm(15)) * 10^11)
convert_to_power_of_ten(x, 6)
print(y <- convert_to_power_of_ten(x))
all.equal(x, convert_from_power_of_ten(y))

# Using data frames
## same exponent for all numeric columns
x <- (5 + rnorm(15)) * 10^11
df <- data.frame(x, y = x * 10^3, z = letters[seq_along(x)],
                 row.names = as.character(seq_along(x)))
x <- df_to_powers_of_ten(df)
str(x)
y <- df_from_powers_of_ten(x)
identical(df, y)
## individual exponents for different columns - but what for?
### automatically
### manually
df1 <- df
df1[["x"]] <- convert_to_power_of_ten(df1[["x"]])
df1[["y"]] <- convert_to_power_of_ten(df1[["y"]])
str(df1)
print(df2 <- df_from_powers_of_ten(df1))
identical(df, df2)
```

---

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.

Usage

```
relative_difference(
  current,
  reference,
  type = c("all.equal", "difference", "change", "change2")
)
```

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()`, `powers_of_ten`, `round_half_away_from_zero()`, `sloboda()`, `weighted_variance()`

Other vector comparing functions: `compare_vectors()`

Examples

```
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")))
# But "all.equal" is _much_ slower:
microbenchmark::microbenchmark(all_equal = relative_difference(x, y),
                               difference = 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"))))
```

`round_half_away_from_zero`

Round Half Away From Zero

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

```
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(), powers_of_ten, relative_difference(), sloboda(), weighted_variance()

Examples

```
x <- 22.5
round_half_away_from_zero(x)
round(x)
round_half_away_from_zero(-x)
round(-x)
```

---

rownames2col *Add a Column Containing the Row Names to a Data Frame*

Description

Add a Column Containing the Row Names to a Data Frame

Usage

```
rownames2col(x, column_name)
```

Arguments

x	A data.frame.
column_name	The name of the new column containing the row.names.

Value

A data.frame.

See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), pause(), r_cmd_install(), str2num(), string2words(), strip_off_attributes(), tapply(), throw()

Examples

```
rownames2col(mtcars, column_name = "model")
```

---

runsed *Replace a Pattern in Files with a Replacement String*

Description

This function mimics the runsed script published in *Unix Power Tools*.

Usage

```
runsed(files, pattern, replacement)
```

Arguments

files	A list of file names in which to replace.
pattern	A regex pattern, see gsub.
replacement	A string, see gsub.

Value

Invisibly the vector of names of files changed.

References

Shelley Powers, Jerry Peek, Tim O'Reilly and Mike Loukides, 2002, Unix Power Tools, 3rd edition, O'Reilly Media, Inc.

See Also

Other file utilities: clipboard_path(), delete_trailing_blank_lines(), delete_trailing_whitespace(), develop_test(), file_copy(), file_modified_last(), file_save(), file_string(), 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 searching functions: compare_vectors(), file_modified_last(), find_files(), fromto(), grep_file(), missing_docs, search_files(), search_rows(), summary.filesearch()

Examples

```
source_files <- list.files(system.file(package = "fritools", "source", "R"),
                           pattern = ".*\\.R$", full.names = TRUE)
file.copy(source_files, tempdir(), overwrite = TRUE)
files <- find_files(file_names = file.path(tempdir(),
                                           basename(source_files)))
print(f <- runsed(files, pattern = "_clean", replacement = "_cleanr"))
print(f <- runsed(files, pattern = "_cleanr\\>", replacement = "_cleaner"))
```

---

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: `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()`, `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()`, `fromto()`, `grep_file()`, `missing_docs`, `runsed()`, `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()`, `file_string()`, `find_files()`, `get_lines_between_tags()`, `get_mtime()`, `get_unique_string()`, `grep_file()`, `is_files_current()`, `is_path()`, `paths`, `runsed()`, `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$")
search_files(what = "[Ss]etosa", path = tempdir(), pattern = "^.*\\.csv$")
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$"))
```

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, runsed(), 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)
```

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

```
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

```
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

```
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_h`

Examples

```
set_run_r_tests(is_running_on_fvafrcu_machines())
get_run_r_tests()
set_run_r_tests(TRUE, force = TRUE)
get_run_r_tests()
```

sloboda *Sloboda's Growth Function*

Description

Implement the growth function

$$y_t = k^{\beta_1} \times \left(\frac{y_0}{k^{\beta_1}}\right)^{\exp\left[\frac{\beta_2}{(\beta_3-1)\times t^{(\beta_3-1)}} - \frac{\beta_2}{(\beta_3-1)\times t_0^{(\beta_3-1)}}\right]}$$

published in *Sloboda, B., 1971: Zur Darstellung von Wachstumsprozessen mit Hilfe von Differentialgleichungen erster Ordnung. Mitt. d. Baden-Württembergischen Forstlichen Versuchs- und Forschungsanstalt.*

Usage

```
sloboda(a, b, c, y0, t0, t, type = c("classic", "kaendler"), k = 65)
```

Arguments

a	Sloboda's β_3.
b	Sloboda's β_2.
c	Sloboda's β_1.
y0	Sloboda's y_0.
t0	Sloboda's t_0.
t	Sloboda's t.
type	Gerald Kaendler reformulated the algorithm, but it doesn't get faster, see the examples.
k	Sloboda's k.

Value

The value y_t of Sloboda's growth function.

See Also

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

Examples

```
microbenchmark::microbenchmark(cl = sloboda(0.2, 0.7, 3, 30, 30, 35),
                               g =  sloboda(0.2, 0.7, 3, 30, 30, 35,
                                            "kaendler"),
                               check = "equivalent")
```

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"),
  keep_header = TRUE
)
```

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.
keep_header	Keep a header found in your code file?

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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), 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.

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(), pause(), r_cmd_install(), rownames2col(), string2words(), 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)
```

string2words *Convert a Character Vector Into an Enumeration*

Description

Convert a Character Vector Into an Enumeration

Usage

```
string2words(x, separator = ",", last = "and", add_whitespace = TRUE)
```

Arguments

x	A character vector.
separator	A separator used for the enumeration.
last	The separator used last for the enumeration.
add_whitespace	Add whitespace after separators?

Value

A character string of length 1.

See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), pause(), r_cmd_install(), rownames2col(), str2num(), strip_off_attributes(), tapply(), throw()

Examples

```
string2words(c("beech", "oak", "ash"))
```

---

strip_off_attributes *Strip Attributes off an Object*

Description

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(), pause(), r_cmd_install(), rownames2col(), str2num(), string2words(), tapply(), throw()

Examples

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

`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

```
## 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(), fromto(), grep_file(), missing_docs, runsed(), search_files(), search_rows()

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$")
search_files(what = "[Ss]etosa", path = tempdir(), pattern = "^.*\\.csv$")
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(), pause(), r_cmd_install(), rownames2col(), str2num(), string2words(), strip_off_attributes(), throw()

Examples

```
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.frames:")
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
```

---

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(), file_string(), find_files(), get_lines_between_tags(), get_mtime(), get_unique_string(), grep_file(), is_files_current(), is_path(), paths, runsed(), search_files(), split_code_file()

Examples

```
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 objected 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`.

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

```
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

```
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()`, `powers_of_ten`, `relative_difference()`, `round_half_away_from_zero()`, `sloboda()`

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:
fritools::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

```
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`.

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

```
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)
```

---

wipe_tempdir *Wipe Clean the* tempdir()

Description

I often need a clean temporary directory.

Usage

```
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 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: `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

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

Index

∗ **CSV functions**
bulk_read_csv, 4
bulk_write_csv, 6
check_ascii_file, 9
csv, 16
csv2csv, 17
∗ **German umlaut converters**
convert_umlauts_to_ascii, 13
convert_umlauts_to_tex, 14
convert_umlauts_to_utf8, 14
get_german_umlauts, 29
∗ **R memory functions**
memory_hogs, 61
wipe_clean, 85
wipe_tempdir, 86
∗ **bits and pieces**
golden_ratio, 37
is_difftime_less, 42
is_valid_primary_key, 58
pause, 63
rownames2col, 67
str2num, 76
string2words, 76
strip_off_attributes, 77
tapply, 80
∗ **call functions**
call_conditionally, 7
call_safe, 8
∗ **data.frame helpers**
columns2numeric, 10
powers_of_ten, 64
∗ **file utilities**
clipboard_path, 10
delete_trailing_blank_lines, 18
delete_trailing_whitespace, 19
develop_test, 19
file_copy, 21
file_modified_last, 22
file_save, 23
file_string, 24
find_files, 25
get_lines_between_tags, 30
get_mtime, 31
get_unique_string, 37
grep_file, 38
is_files_current, 44
is_path, 50
paths, 62
runsed, 68
search_files, 70
split_code_file, 75
touch, 81
∗ **hash functions for objects**
set_hash, 71
un_hash, 82
∗ **logical helpers**
columns2numeric, 10
get_run_r_tests, 33
is_batch, 40
is_cran, 40
is_false, 43
is_force, 45
is_installed, 46
is_not_false, 47
is_null_or_true, 48
is_of_length_zero, 49
is_r_cmd_check, 52
is_r_package_installed, 53
is_running_on_fvafrcu_machines, 50
is_running_on_gitlab_com, 51
is_scalar, 54
is_scalar_convertible2numeric, 55
is_success, 56
is_true, 57
is_version_sufficient, 58
is_windows, 59
∗ **operating system functions**
clipboard_path, 10

file_copy, 21
file_save, 23
get_boolean_envvar, 28
get_run_r_tests, 33
is_installed, 46
is_r_package_installed, 53
is_success, 56
is_windows, 59
view, 83
vim, 83
wipe_tempdir, 86
with_dir, 87
∗ **option functions**
get_options, 31
is_force, 45
set_options, 72
∗ **package functions**
get_package_version, 32
is_r_package_installed, 53
is_version_sufficient, 58
load_internal_functions, 60
∗ **package**
fritools-package, 4
∗ **script path getter functions**
get_r_cmd_batch_script_path, 34
get_rscript_script_path, 33
get_script_name, 35
get_script_path, 36
∗ **searching functions**
compare_vectors, 12
file_modified_last, 22
find_files, 25
fromto, 27
grep_file, 38
missing_docs, 62
runsed, 68
search_files, 70
search_rows, 71
summary.filesearch, 79
∗ **statistics**
column_sums, 11
count_groups, 15
powers_of_ten, 64
relative_difference, 65
round_half_away_from_zero, 66
sloboda, 74
weighted_variance, 84
∗ **subsetting functions**
index_groups, 39
subset_sizes, 78
∗ **test helpers**
develop_test, 19
get_boolean_envvar, 28
get_run_r_tests, 33
is_cran, 40
is_r_cmd_check, 52
is_running_on_fvafrcu_machines, 50
is_running_on_gitlab_com, 51
run_r_tests_for_known_hosts, 69
set_run_r_tests, 73
∗ **vector comparing functions**
compare_vectors, 12
relative_difference, 65
∗ **vector functions**
char2factor, 8
escape_non_ascii, 20
file_string, 24
powers_of_ten, 64
∗ **version functions**
get_package_version, 32
get_session_string, 36
is_r_package_installed, 53
is_version_sufficient, 58
.onLoad, *69*

all.equal, *65, 66*
attr, *31*, *62*

base::tapply, *80*
base::unname, *77*
boolean, *21*, *23*
bulk_read_csv, 4
bulk_read_csv(), *6*, *9*, *17, 18*
bulk_write_csv, 6
bulk_write_csv(), *5*, *9*, *17, 18*

call_conditionally, 7, *8*
call_conditionally(), *8*
call_safe, 8
call_safe(), *7*
char2factor, 8
char2factor(), *20*, *25*, *64*
character, *77*
check_ascii_file, 9
check_ascii_file(), *5, 6*, *17, 18*
class, *28*
clipboard_path, 10

clipboard_path(), *18–22*, *24–26*, *28*, *30, 31*, *34*, *37, 38*, *44*, *46*, *50*, *53*, *56*, *59*, *63*, *68*, *70*, *75*, *81*, *83, 84*, *86, 87*
codetools::checkUsageEnv, *60*
colSums, *11*
column_sums, 11
column_sums(), *16*, *64*, *66, 67*, *74*, *85*
columns2numeric, 10
columns2numeric(), *34*, *40, 41*, *43*, *45–49*, *51–57*, *59*, *64*
compare_vectors, 12
compare_vectors(), *22*, *26, 27*, *38*, *62*, *66*, *68*, *70, 71*, *79*
compareVersion, *32*, *58*
convert_from_power_of_ten *(*powers_of_ten*)*, 64
convert_to_power_of_ten *(*powers_of_ten*)*, 64
convert_umlauts_to_ascii, 13
convert_umlauts_to_ascii(), *14, 15*, *29*
convert_umlauts_to_tex, 14
convert_umlauts_to_tex(), *13*, *15*, *29*
convert_umlauts_to_utf8, 14
convert_umlauts_to_utf8(), *13, 14*, *29*
count_groups, 15
count_groups(), *11*, *64*, *66, 67*, *74*, *85*
covr::zero_coverage, *19*
csv, *5, 6*, *9*, 16, *18*
csv2csv, 17
csv2csv(), *5, 6*, *9*, *17*

data.frame, *10, 11*, *15, 16*, *63*, *67*, *71*, *80*, *84, 85*
delete_trailing_blank_lines, 18
delete_trailing_blank_lines(), *10*, *19–22*, *24–26*, *30, 31*, *37, 38*, *44*, *50*, *63*, *68*, *70*, *75*, *81*
delete_trailing_whitespace, 19
delete_trailing_whitespace(), *10*, *18*, *20–22*, *24–26*, *30, 31*, *37, 38*, *44*, *50*, *63*, *68*, *70*, *75*, *81*
develop_test, 19
develop_test(), *10*, *18, 19*, *21, 22*, *24–26*, *28*, *30, 31*, *34*, *37, 38*, *41*, *44*, *50–52*, *63*, *68–70*, *73*, *75*, *81*
df_from_powers_of_ten *(*powers_of_ten*)*, 64
df_to_powers_of_ten *(*powers_of_ten*)*, 64
difftime, *42*, *44*
do.call, *7, 8*

escape_non_ascii, 20
escape_non_ascii(), *9*, *25*, *64*
exists, *47*

factor, *9*
FALSE, *23*, *28*, *34*, *40*, *42–44*, *46–53*, *56*, *58, 59*, *61*
file.copy, *21*, *23*
file.edit, *83, 84*
file.exists, *26*
file_copy, 21
file_copy(), *10*, *18–20*, *22*, *24–26*, *28*, *30, 31*, *34*, *37, 38*, *44*, *46*, *50*, *53*, *56*, *59*, *63*, *68*, *70*, *75*, *81*, *83, 84*, *86, 87*
file_modified_last, 22
file_modified_last(), *10*, *12*, *18–21*, *24–27*, *30, 31*, *37, 38*, *44*, *50*, *62, 63*, *68*, *70, 71*, *75*, *79*, *81*
file_save, 23
file_save(), *10*, *18–22*, *25, 26*, *28*, *30, 31*, *34*, *37, 38*, *44*, *46*, *50*, *53*, *56*, *59*, *63*, *68*, *70*, *75*, *81*, *83, 84*, *86, 87*
file_string, 24
file_string(), *9, 10*, *18–22*, *24*, *26*, *30, 31*, *37, 38*, *44*, *50*, *63, 64*, *68*, *70*, *75*, *81*
find_files, *5*, *18, 19*, *22*, 25
find_files(), *10*, *12*, *18–22*, *24, 25*, *27*, *30, 31*, *37, 38*, *44*, *50*, *62, 63*, *68*, *70, 71*, *75*, *79*, *81*
find_missing_family *(*missing_docs*)*, 62
find_missing_see_also *(*missing_docs*)*, 62
fritools *(*fritools-package*)*, 4
fritools-package, 4
fromto, 27
fromto(), *12*, *22*, *26*, *38*, *62*, *68*, *70, 71*, *79*

get_boolean_envvar, 28, *33*
get_boolean_envvar(), *10*, *20, 21*, *24*, *34*, *41*, *46*, *51–53*, *56*, *59*, *69*, *73*, *83, 84*, *86, 87*
get_german_umlauts, 29
get_german_umlauts(), *13–15*
get_lines_between_tags, 30
get_lines_between_tags(), *10*, *18–22*, *24–26*, *31*, *37, 38*, *44*, *50*, *63*, *68*, *70*, *75*, *81*
get_mtime, 31

get_mtime(), *10*, *18–22*, *24–26*, *30*, *37, 38*, *44*, *50*, *63*, *68*, *70*, *75*, *81*
get_options, 31
get_options(), *45*, *72*
get_package_version, 32
get_package_version(), *36*, *53*, *59, 60*
get_path *(*paths*)*, 62
get_r_cmd_batch_script_path, 34, *36*
get_r_cmd_batch_script_path(), *33*, *35*, *36*
get_rscript_script_path, 33, *36*
get_rscript_script_path(), *35, 36*
get_run_r_tests, 33
get_run_r_tests(), *10, 11*, *20, 21*, *24*, *28*, *40, 41*, *43*, *45–49*, *51–57*, *59*, *69*, *73*, *83, 84*, *86, 87*
get_script_name, 35
get_script_name(), *33*, *35, 36*
get_script_path, 36
get_script_path(), *33*, *35*
get_session_string, 36
get_session_string(), *32*, *53*, *59*
get_unique_string, 37
get_unique_string(), *10*, *18–22*, *24–26*, *30, 31*, *38*, *44*, *50*, *63*, *68*, *70*, *75*, *81*
getOption, *31, 32*
golden_ratio, 37
golden_ratio(), *42*, *58*, *64*, *68*, *76, 77*, *80*
grep_file, 38
grep_file(), *10*, *12*, *18–22*, *24–27*, *30, 31*, *37*, *44*, *50*, *62, 63*, *68*, *70, 71*, *75*, *79*, *81*
gsub, *68*

index_groups, 39
index_groups(), *78*
integer, *49*
interactive, *40*
invisible, *42*
Invisibly, *17–20*, *60*, *63*, *68–70*, *72*, *75*, *83*
is_batch, 40
is_batch(), *11*, *34*, *41*, *43*, *45–49*, *51–57*, *59*
is_cran, 40
is_cran(), *11*, *20*, *28*, *34*, *40*, *43*, *45–49*, *51–57*, *59*, *69*, *73*
is_difftime_less, 42
is_difftime_less(), *38*, *58*, *64*, *68*, *76, 77*, *80*
is_false, 43
is_false(), *11*, *34*, *40, 41*, *45–49*, *51–57*, *59*
is_files_current, 44
is_files_current(), *10*, *18–22*, *24–26*, *30, 31*, *37, 38*, *50*, *63*, *68*, *70*, *75*, *81*
is_force, 45
is_force(), *11*, *32*, *34*, *40, 41*, *43*, *46–49*, *51–57*, *59*, *72*
is_installed, 46
is_installed(), *10, 11*, *21*, *24*, *28*, *34*, *40, 41*, *43*, *45*, *47–49*, *51–57*, *59*, *83, 84*, *86, 87*
is_not_false, 47
is_not_false(), *11*, *34*, *40, 41*, *43*, *45, 46*, *48, 49*, *51–57*, *59*
is_null_or_true, 48
is_null_or_true(), *11*, *34*, *40, 41*, *43*, *45–47*, *49*, *51–57*, *59*
is_of_length_zero, 49
is_of_length_zero(), *11*, *34*, *40, 41*, *43*, *45–48*, *51–57*, *59*
is_path, 50
is_path(), *10*, *18–22*, *24–26*, *30, 31*, *37, 38*, *44*, *63*, *68*, *70*, *75*, *81*
is_r_cmd_check, 52
is_r_cmd_check(), *11*, *20*, *28*, *34*, *40, 41*, *43*, *45–49*, *51–57*, *59*, *69*, *73*
is_r_package_installed, 53
is_r_package_installed(), *10, 11*, *21*, *24*, *28*, *32*, *34*, *36*, *40, 41*, *43*, *45–49*, *51, 52*, *54–57*, *59, 60*, *83, 84*, *86, 87*
is_running_on_fvafrcu_machines, 50
is_running_on_fvafrcu_machines(), *11*, *20*, *28*, *34*, *40, 41*, *43*, *45–49*, *52–57*, *59*, *69*, *73*
is_running_on_gitlab_com, 51
is_running_on_gitlab_com(), *11*, *20*, *28*, *34*, *40, 41*, *43*, *45–49*, *51–57*, *59*, *69*, *73*
is_scalar, 54
is_scalar(), *11*, *34*, *40, 41*, *43*, *45–49*, *51–53*, *55–57*, *59*
is_scalar_convertible2numeric, 55
is_scalar_convertible2numeric(), *11*, *34*, *40, 41*, *43*, *45–49*, *51–54*, *56, 57*, *59*
is_success, 56
is_success(), *10, 11*, *21*, *24*, *28*, *34*, *40, 41*, *43*, *45–49*, *51–55*, *57*, *59*, *83, 84*, *86, 87*

is_true, 57
is_true(), *11*, *34*, *40, 41*, *43*, *45–49*, *51–56*, *59*
is_valid_primary_key, 58
is_valid_primary_key(), *38*, *42*, *64*, *68*, *76, 77*, *80*
is_version_sufficient, 58
is_version_sufficient(), *11*, *32*, *34*, *36*, *40, 41*, *43*, *45–49*, *51–57*, *59, 60*
is_windows, 59
is_windows(), *10, 11*, *21*, *24*, *28*, *34*, *40, 41*, *43*, *45–49*, *51–57*, *59*, *83, 84*, *86, 87*

length, *35, 36*
list.files, *25, 26*, *38*, *70*
load_internal_functions, 60
load_internal_functions(), *32*, *53*, *59*
logical, *57*
ls, *60*, *85*

matrix, *71*
memory_hogs, 61
memory_hogs(), *86*
missing_docs, *12*, *22*, *26, 27*, *38*, 62, *68*, *70, 71*, *79*
mode, *33*, *35, 36*

NULL, *18–20*, *45*, *47, 48*, *63*, *69*, *73*, *83*

options, *72*
order, *61*

package_version, *32*
packageDescription, *32*
packageVersion, *32*
paths, *10*, *18–22*, *24–26*, *30, 31*, *37, 38*, *44*, *50*, 62, *68*, *70*, *75*, *81*
pause, 63
pause(), *38*, *42*, *58*, *68*, *76, 77*, *80*
powers_of_ten, *9*, *11*, *16*, *20*, *25*, 64, *66, 67*, *74*, *85*

r_cmd_install(), *38*, *42*, *58*, *64*, *68*, *76, 77*, *80*
read_csv, *4, 5*, *17*, *63*
read_csv *(*csv*)*, 16
readLines, *27*
relative_difference, 65
relative_difference(), *11, 12*, *16*, *64*, *67*, *74*, *85*
rename_package(), *32*, *53*, *59, 60*
round, *66, 67*
round_commercially *(*round_half_away_from_zero*)*, 66
round_half_away_from_zero, 66
round_half_away_from_zero(), *11*, *16*, *64*, *66*, *74*, *85*
row.names, *67*
rownames2col, 67
rownames2col(), *38*, *42*, *58*, *64*, *76, 77*, *80*
run_r_tests_for_known_hosts, 69
run_r_tests_for_known_hosts(), *20*, *28*, *34*, *41*, *51, 52*, *73*
runsed, 68
runsed(), *10*, *12*, *18–22*, *24–27*, *30, 31*, *37, 38*, *44*, *50*, *62, 63*, *70, 71*, *75*, *79*, *81*

search_files, 70, *71*, *79*
search_files(), *10*, *12*, *18–22*, *24–27*, *30, 31*, *37, 38*, *44*, *50*, *62, 63*, *68*, *71*, *75*, *79*, *81*
search_rows, 71
search_rows(), *12*, *22*, *26, 27*, *38*, *62*, *68*, *70*, *79*
sessionInfo, *36*
set_hash, 71
set_hash(), *82*
set_options, *45*, 72
set_options(), *32*, *45*
set_path *(*paths*)*, 62
set_run_r_tests, 73
set_run_r_tests(), *20*, *28*, *34*, *41*, *51, 52*, *69*
sloboda, 74
sloboda(), *11*, *16*, *64*, *66, 67*, *85*
source, *75*
split_code_file, 75
split_code_file(), *10*, *18–22*, *24–26*, *30, 31*, *37, 38*, *44*, *50*, *63*, *68*, *70*, *81*
stats::aggregate, *15*
str2num, 76
str2num(), *38*, *42*, *58*, *64*, *68*, *77*, *80*
string2words, 76
string2words(), *38*, *42*, *58*, *64*, *68*, *76, 77*, *80*
strip_off_attributes, 77
strip_off_attributes(), *38*, *42*, *58*, *64*, *68*, *76, 77*, *80*
subset_sizes, *39*, 78
subset_sizes(), *39*
summary.filesearch, 79

summary.filesearch(), *12*, *22*, *26*, *27*, *38*, *62*, *68*, *70*, *71*
Sys.getenv, *28*, *73*
Sys.time, *36*

tapply, 80, *85*
tapply(), *38*, *42*, *58*, *64*, *68*, *76*, *77*
tempdir, *86*
throw(), *38*, *42*, *58*, *64*, *68*, *76*, *77*, *80*
touch, 81
touch(), *10*, *18–22*, *24–26*, *30*, *31*, *37*, *38*, *44*, *50*, *63*, *68*, *70*, *75*
touch2 *(*touch*)*, 81
TRUE, *7*, *40*, *42–53*, *56–60*, *72*
tryCatch, *7*

un_hash, 82
un_hash(), *72*
utils::read.csv, *16*
utils::read.csv2, *16*
utils:read.csv, *16*
utils:read.csv2, *16*

vector, *84*
view, 83
view(), *10*, *21*, *24*, *28*, *34*, *46*, *53*, *56*, *59*, *84*, *86*, *87*
vim, 83
vim(), *10*, *21*, *24*, *28*, *34*, *46*, *53*, *56*, *59*, *83*, *86*, *87*

weighted_variance, 84
weighted_variance(), *11*, *16*, *64*, *66*, *67*, *74*
wipe_clean, 85
wipe_clean(), *61*, *86*
wipe_tempdir, 86
wipe_tempdir(), *10*, *21*, *24*, *28*, *34*, *46*, *53*, *56*, *59*, *61*, *83*, *84*, *86*, *87*
with_dir, 87
with_dir(), *10*, *21*, *24*, *28*, *34*, *46*, *53*, *56*, *59*, *83*, *84*, *86*
write_csv, *6*, *17*, *63*
write_csv *(*csv*)*, 16