Package ‘fritools’

March 9, 2022

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

Version 3.2.0

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

License BSD_2_clause + file LICENSE

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

Depends R (>= 3.3.0)

Imports methods, stats, utils

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

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

Config/testthat/edition 3

NeedsCompilation no

Author Andreas Dominik Cullmann [aut, cre]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository CRAN

Date/Publication 2022-03-09 15:40:02 UTC
R topics documented:

- fritools-package .................................................. 3
- bulk_read_csv ..................................................... 4
- bulk_write_csv .................................................... 5
- call_conditionally .................................................. 6
- call_safe ............................................................ 7
- check_ascii_file ................................................... 8
- compare_vectors .................................................... 9
- convert_umlauts_to_ascii ....................................... 9
- convert_umlauts_to_tex ......................................... 10
- csv ................................................................. 11
- csv2csv ............................................................ 12
- file_modified_last ................................................ 13
- find_files .......................................................... 14
- fromto ............................................................... 15
- get_boolean_envvar ................................................ 17
- get_mtime ........................................................... 18
- get_options ........................................................ 18
- get_package_version .............................................. 19
- get_rscript_script_path ........................................ 20
- get_run_r_tests ................................................... 20
- get_r_cmd_batch_script_path .................................... 21
- get_script_name ................................................... 22
- get_script_path .................................................... 23
- get_unique_string ................................................ 23
- golden_ratio ....................................................... 24
- index_groups ...................................................... 24
- is_batch .......................................................... 25
- is_cran ............................................................. 26
- is_difftime_less ................................................... 27
- is_false ............................................................ 28
- is_files_current ................................................... 29
- is_force ............................................................ 30
- is_installed ....................................................... 31
- is_not_false ........................................................ 31
- is_null_or_true ................................................... 33
- is_of_length_zero ................................................ 34
- is_path ............................................................. 34
- is_running_on_fvafrcu_machines ................................ 35
- is_running_on_github_com ........................................ 36
- is_r_cmd_check .................................................... 37
- is_r_package_installed ......................................... 37
- is_success ........................................................ 38
- is_valid_primary_key ............................................. 39
- is_version_sufficient ............................................ 40
- is_windows ......................................................... 41
- load_internal_functions ......................................... 41
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("An_Introduction_to_fritools",package = "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(), csv2csv(), csv
**bulk_write_csv**

### Description

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

### Usage

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

### Arguments

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

### Value

The list holding the return values of `write_csv`.

### See Also

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

```r
unlink(dir(tempdir(), full.names = TRUE))
data(mtcars)
mt_german <- mtcars
rownames(mt_german)[1] <- "Mazda Rö4"
names(mt_german)[1] <- "mgü00dc"
for (i in 1:10) {
  f <- file.path(tempdir(), paste0("f", i, ".csv"))
  write.csv(mtcars[1:5, TRUE], file = f)
  f <- file.path(tempdir(), paste0("f", i, ",_german.csv"))
  write.csv2(mt_german[1:7, TRUE], file = f, fileEncoding = "Latin1")
}
# read
bulk <- bulk_read_csv(tempdir())
print(mtime <- file.info(list.files(tempdir(), full.names = TRUE))['mtime'])
bulk['["f2"]'][3, 5] <- bulk['["f2"]'][3, 5] + 2
Sys.sleep(2) # make sure the mtimes would change
result <- bulk_write_csv(bulk)
print(new_times <- file.info(dir(tempdir(), full.names = TRUE))['mtime'])
index_change <- grep("f2\.csv", rownames(mtime))
if (requireNamespace("digest", quietly = TRUE)) {
  only_f2_changed <- all((mtime == new_times)[-c(index_change)]) &&
  (mtime < new_times)[c(index_change)]
  RUnit::checkTrue(only_f2_changed)
} else {
  RUnit::checkTrue(all(mtime < new_times))
}
```

call_conditionally  Call a Function Conditionally

Description

**whoami** 1.3.0 uses things like `system("getent passwd $(whoami)", intern = TRUE)` which I cannot use `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.
call_safe

Value
The return value of f or fallback.

See Also
Other call functions: call_safe()

Examples

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

---

Description
Just a specialized version of call_conditionally.

Usage

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

Arguments

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

Value
The return value of f or fallback.
check_ascii_file  

Description

Check the Number of Lines and Fields in a File

Usage

check_ascii_file(path, sep = ";")

Arguments

path  
Path to a file.

sep  
A character separating the fields in the file.

Value

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

See Also

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

Examples

f <- tempfile()
write.csv2(mtcars, file = f)
check_ascii_file(f)
**compare_vectors**  
*Compare Two Vectors*

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

**Usage**
```r
compare_vectors(x, y, 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()`, `missing_docs`, `search_files()`, `search_rows()`, `summary.filesearch()`

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

---

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

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

Usage

convert_umlauts_to_ascii(x)

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

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

Arguments

x  A string or data.frame.

Value

x with the umlauts converted to ascii.

See Also

Other German umlaut converters: convert_umlauts_to_tex()

Examples

```r
string <- paste("this is ä string")
print(string)
print(convert_umlauts_to_ascii(string))
string <- paste("this is \u00e4 string")
df <- data.frame(v1 = c(string, "foobar"),
                 v2 = c("foobar", string), v3 = 3:4)
names(df)[3] <- "\u00dfy"
convert_umlauts_to_ascii(df)
```

---

convert_umlauts_to_tex

*Tex Codes for German Umlauts*

Description

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

Usage

convert_umlauts_to_tex(x)

Arguments

x  A string.
Value

A string with the umlauts converted to plain TeX.

See Also

Other German umlaut converters: convert_umlauts_to_ascii()

Examples

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

---

**Description**

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

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

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

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

**Usage**

```r
read_csv(file, ...)

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

**Arguments**

- **file** The path to the file to be read or written.
- **...** Arguments passed to `utils::read.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

```r
# read from standard CSV
f <- tempfile()
write.csv(mtcars, file = f)
str(read_csv(f))
f <- tempfile()
write.csv2(mtcars, file = f)
str(read_csv(f))
# write to standard CSV
f <- tempfile()
d <- mtcars
str(d <- write_csv(d, file = f))
file.mtime(f)
Sys.sleep(2) # make sure the mtime would have changed
write.csv(d, file = f)
file.mtime(f)
```

---

csv2csv  

Convert a German Comma Separated File into a Comma Separated File

Description

Convert a German Comma Separated File into a Comma Separated File

Usage

```r
csv2csv(file, ...)
```

Arguments

- `file`  
  Path to the file.
- `...`  
  Arguments passed to `read_csv`

Value

Invisibly the return value of `write_csv`, but called for its side effect.
file_modified_last

**See Also**

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

**Examples**

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

---

**file_modified_last**  
*Get the File Modified Last*

**Description**

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

**Usage**

```r
file_modified_last(...)```

**Arguments**

```r
...  # Arguments passed to `list.files`.```

**Value**

The path to the file last modified.

**See Also**

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

Other file utilities: `find_files()`, `get_mtime()`, `get_unique_string()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

**Examples**

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

Find Files on Disk

Description

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

Usage

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

Arguments

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

Details

This is a wrapper to either `file.exists` or `list.files`, that ensures that (some) files 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 \texttt{file.exists} or \texttt{list.files}, depending on whether file names is given.

See Also

Other searching functions: \texttt{compare_vectors()}, \texttt{file_modified_last()}, \texttt{fromto()}, \texttt{missing_docs}, \texttt{search_files()}, \texttt{search_rows()}, \texttt{summary.filesearch()}

Other file utilities: \texttt{file_modified_last()}, \texttt{get_mtime()}, \texttt{get_unique_string()}, \texttt{is_files_current()}, \texttt{is_path()}, \texttt{paths}, \texttt{search_files()}, \texttt{split_code_file()}, \texttt{touch()}

Examples

\verbatim
#% 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]\.$|\.\.[RrSs]nw\.$|\.[RrSs]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(), pattern = ".*")
find_files(path = tempdir(), pattern = ".*",
    select = list(size = c(min = 1000)))
  )
\endverbatim
Usage

```r
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.
- **to**: Another pattern.
- **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()`, `missing_docs`, `search_files()`, `search_rows()`, `summary.filesearch()`

Examples

```r
rep("D", 4), "t3")
fromto(foo, "^f", "^t")
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

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

Arguments

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

Details

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

Value

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

See Also

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

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

Examples

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

Get the mtime Attribute to or 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: `file_modified_last()`, `find_files()`, `get_unique_string()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`, `touch()`

Examples

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

get_options

Get Options For Packages

Description

A convenience function for `getOption`.

Usage

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

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

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

---

**get_rscript_script_path**

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

Description

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

Usage

```r
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

```r
get_rscript_script_path()
```

---

**get_run_r_tests**

*Get System Variable RUN_R_TESTS*

Description

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

Usage

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

stop_on_failure

Throw an error instead of returning 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: get_boolean_envvar(), is_cran(), is_r_cmd_check(), is_running_on_fvafrucu_machines(), is_running_on_gitlab_com(), run_r_tests_for_known_hosts(), set_run_r_tests()

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

Other logical helpers: 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_fvafrucu_machines(), is_running_on_gitlab_com(), is_success(), 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_path

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_unique_string

Create a Fairly Unique String

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

Usage
get_unique_string()

Value
A fairly unique string.

See Also
Other file utilities: file_modified_last(), find_files(), get_mtime(), is_files_current(), is_path(), paths, search_files(), split_code_file(), touch()
Examples
  replicate(20, get_unique_string())

golden_ratio      Calculate the Golden Ratio

Description
  Divide a length using the golden ratio.

Usage
  golden_ratio(x)

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

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

See Also
  Other bits and pieces: is_difftime_less(), is_valid_primary_key(), r_cmd_install(), round_half_away_from_zero(), str2num(), strip_off_attributes(), tapply(), throw(), weighted_variance()

Examples
  golden_ratio(10)

index_groups      Determine Indices and Sizes of Subsets

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

Usage
  index_groups(n, k)

Arguments
  n       The size of the set.
  k       The number of subsets.
is_batch

Value

A matrix with starting index, size, and stopping index for each subset.

See Also

Other subsetting functions: `subset_sizes()`

Examples

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

is_batch

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

Description

Just a wrapper to `interactive`.

Usage

```r
is_batch()
```

Value

`TRUE` on success, `FALSE` otherwise.

See Also

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

Examples

```r
is_batch()
```
is_cran

Is R Running on CRAN?

Description

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

Usage

is_cran(cran_pattern, n_r_check4cran)

Arguments

cran_pattern A regular expressions to apply to the names of Sys.getenv() to identify possible CRAN parameters. Defaults to Sys.getenv('_CRAN_pattern_') if available and '^_R_' if not.

n_r_check4cran Assume this is CRAN if at least n_R_CHECK4CRAN elements of Sys.getenv() have names matching x. Defaults to Sys.getenv('_n_R_CHECK4CRAN_') if available and 5 if not.

Details

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

R CMD check --as-cran

due to compute time constraints with CRAN tests.

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

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

'--as-cran'

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

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

Value

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

Check Whether Two Times Differ Less Than A Given Value

Description

This is just a wrapper to `difftime`.

Usage

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

Arguments

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

Value

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

See Also

Other test helpers: `get_boolean_envvar()`, `get_run_r_tests()`, `is_r_cmd_check()`, `is_running_on_fvafrcu_machines()`, `run_r_tests_for_known_hosts()`, `set.Run_r_tests()`.

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

Examples

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

See Also

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

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

Description

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

Usage

is_false(x)

Arguments

x

The object to be tested.

Value

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

See Also

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

Examples

is_false("not false")

is_false(FALSE)
is_files_current Check Whether Files are Current

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

Usage

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

Arguments

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

Value

TRUE on success, FALSE otherwise.

See Also

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

Examples

```r
p1 <- tempfile()
p2 <- tempfile()
p3 <- tempfile()
touch(p1)
touch(p2)
Sys.sleep(3)
touch(p3)
is_files_current(p3, newer_than = 1, units = "days",
```

is_force

Opt-out Via Option

Description

Check whether or not a package option (set via \texttt{set_options}) \textit{force} is not set or set to \texttt{TRUE}.

Usage

\begin{verbatim}
  is_force(x = .packages()[1])
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x}
    The option under which an element "force" is to be searched for.
\end{itemize}

Value

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

See Also

Other option functions: \texttt{get_options()}, \texttt{set_options()}

Other logical helpers: \texttt{get_run_r_tests()}, \texttt{is_batch()}, \texttt{is_cran()}, \texttt{is_false()}, \texttt{is_installed()}, \texttt{is_not_false()}, \texttt{is_null_or_true()}, \texttt{is_of_length_zero()}, \texttt{is_r_cmd_check()}, \texttt{is_r_package_installed()}, \texttt{is_running_on_fvafrcu_machines()}, \texttt{is_running_on_gitlab_com()}, \texttt{is_success()}, \texttt{is_version_sufficient()}, \texttt{is_windows()}

Examples

\begin{verbatim}
  is_force()
  set_options(list(force = FALSE))
  get_options(flatten_list = FALSE)
  is_force()
\end{verbatim}
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: get_run_r_tests(), is_batch(), is_cran(), is_false(), is_force(), is_not_false(), is_null_or_true(), is_of_length_zero(), is_r_cmd_check(), is_r_package_installed(), is_running_on_fvafrcu_machines(), is_running_on_gitlab_com(), is_success(), is_version_sufficient(), is_windows()

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

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

```r
is_null_or_true(x)
```

**Arguments**

- `x` The object to be tested.

**Value**

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

**See Also**

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

**Examples**

```r
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**

```r
is_of_length_zero(x, class = NULL)
```

**Arguments**

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

**Value**

`TRUE` on success, `FALSE` otherwise.

**See Also**

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

**Examples**

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

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

---

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

**Description**

Check Whether an Object Contains a Valid File System Path

**Usage**

```r
is_path(x)
```
is_running_on_fvafrcu_machines

Arguments

x

The object.

Value

TRUE on success, FALSE otherwise.

See Also

Other file utilities: file_modified_last(), find_files(), get_mtime(), get_unique_string(), is_files_current(), paths, 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", "fvafr"))

Arguments

type

An optional selection.

Value

TRUE on success, FALSE otherwise.

See Also

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

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

**Examples**

```r
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**

```r
is_running_on_gitlab_com(verbosed = TRUE)
```

**Arguments**

- `verbose`: Be verbose?

**Value**

`TRUE` on success, `FALSE` otherwise.

**See Also**

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

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

**Examples**

```r
is_running_on_gitlab_com()
```
is_r_cmd_check

Is the Current R Process an R CMD check?

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

Usage
is_r_cmd_check()

Value
TRUE on success, FALSE otherwise.

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

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

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

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

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

Examples

```r
is_r_package_installed("base", "3.00.0.0")
is_r_package_installed("fritools", "1.0.0")
```

### Description

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

### Usage

```r
is_success(x)
```

### Arguments

- **x**  
  The external commands return value.

### Value

`TRUE` on success, `FALSE` otherwise.

### See Also

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

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

Examples

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

is_valid_primary_key  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(), r_cmd_install(), round_half_away_from_zero(), str2num(), strip_off_attributes(), tapply(), throw(), weighted_variance()

Examples

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

Description

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

Usage

is_version_sufficient(installed, required)

Arguments

installed  The version available.
required  The version required.

Value

TRUE, if so, FALSE otherwise.

See Also

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

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

Other version functions: get_package_version(), is_r_package_installed()

Examples

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

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

**Description**

Is the system running a windows machine?

**Usage**

```
is_windows()
```

**Value**

`TRUE` if so, `FALSE` otherwise.

**See Also**

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

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

**Examples**

```
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.
memory_hogs

Value

Invisibly TRUE.

See Also

codetools::checkUsageEnv.

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

Examples

load_internal_functions("fritools")

memory_hogs Find Memory Hogs

Description

List objects in an R environment by size.

Usage

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

Arguments

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

Value

A named vector of memory usages.

See Also

Other R memory functions: wipe_clean()
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)
## Not run:
# remove the two largest objects:
rm(list = names(tail(memory_hogs(decreasing = FALSE), n = 2)))
memory_hogs(unit = "Mb")
## End(Not run)

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(), search_files(), search_rows(), summary.filesearch()
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: file_modified_last(), find_files(), get_mtime(), get_unique_string(), is_files_current(), is_path(), search_files(), split_code_file(), touch()

Examples

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

Commerical rounding is done a lot, especially with invoices. There is even standard 1333 by the German Institute for Standardisation. \texttt{round} rounds half to even, see \texttt{round}'s Details section.

\texttt{round\_commercially} is just a link to \texttt{round\_half\_away\_from\_zero}.

Usage

\begin{verbatim}
round\_half\_away\_from\_zero(x, digits = 0)
round\_commercially(x, digits = 0)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A number to be rounded.
  \item \texttt{digits} \hspace{1cm} The number of digits, as in \texttt{round}.
\end{itemize}

Value

The rounded number.

See Also

Other bits and pieces: \texttt{golden\_ratio()}, \texttt{is\_difftime\_less()}, \texttt{is\_valid\_primary\_key()}, \texttt{r\_cmd\_install()}, \texttt{str2num()}, \texttt{strip\_off\_attributes()}, \texttt{tapply()}, \texttt{throw()}, \texttt{weighted\_variance()}.

Examples

\begin{verbatim}
x <- 22.5
round\_half\_away\_from\_zero(x)
round(x)
round\_half\_away\_from\_zero(-x)
round(-x)
\end{verbatim}
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**

```r
run_r_tests_for_known_hosts()
```

**Details**

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

**Value**

*Invisibly NULL.*

**See Also**

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

**Examples**

```r
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**

```r
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()`, `missing_docs`, `search_rows()`, `summary.filesearch()`

Other file utilities: `file_modified_last()`, `find_files()`, `get_mtime()`, `get_unique_string()`, `is_files_current()`, `is_path()`, `paths`, `split_code_file()`, `touch()`

**Examples**

```r
write.csv(mtcars, file.path(tempdir(), "mtcars.csv"))
for (i in 0:9) {
  write.csv(iris, file.path(tempdir(), paste0("iris", i, ".csv"))
}
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$"))
```

**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**

```r
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()`, `missing_docs`, `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()`
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

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

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

*Set the System Variable RUN_R_TESTS*

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**See Also**

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

**Examples**

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

### split_code_file

*Split a Code File Into Multiple Files*

**Description**

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

**Usage**

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

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

Value

Invisibly a vector of paths to the new files.

See Also

Other file utilities: file_modified_last(), find_files(), get_mtime(), get_unique_string(), is_files_current(), is_path(), paths, search_files(), touch()

Examples

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

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

## End(Not run)

---

**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(), r_cmd_install(),
round_half_away_from_zero(), strip_off_attributes(), tapply(), throw(), weighted_variance()

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[7]), digits = 9)
```

---

### 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(), r_cmd_install(),
round_half_away_from_zero(), str2num(), tapply(), throw(), weighted_variance()

#### 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.
**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`.  

---

**Value**

A summarized object.

**See Also**

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

**Examples**

```r
tapply(mtcars, factor(cyl), mean)
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$"
search_files(what = "[Ss]etosa", path = tempdir(), pattern = "^.*\.csv$"
```

```r
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$"))
```
See Also

Other bits and pieces: golden_ratio(), is_difftime_less(), is_valid_primary_key(), r_cmd_install(), round_half_away_from_zero(), str2num(), strip_off_attributes(), throw(), weighted_variance()

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 weights, 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"))
wml <- 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[pmag], subset[awt]) == wml
```
Usage

```r
touch(path)
```

Arguments

- **path**: Path to the file to be touched.

Value

The Path to the file touched.

See Also

Other file utilities: `file_modified_last()`, `find_files()`, `get_mtime()`, `get_unique_string()`, `is_files_current()`, `is_path()`, `paths`, `search_files()`, `split_code_file()`

Examples

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

---

**un_hash**

Separate an Object from its Hash Attribute

---

Description

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

Usage

```r
un_hash(x)
```

Arguments

- **x**: The object.

Value

A list containing the object and its hash attribute.
weighted_variance

See Also

Other hash functions for objects: set_hash()

---

**Description**

Calculate a weighted variance.

**Usage**

weighted_variance(x, ...)

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

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

**Arguments**

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

**Details**

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

**See Also**

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

```r
## GPA from Siegel 1994
wt <- c(5, 5, 4, 1)/15
x <- c(3.7, 3.3, 3.5, 2.8)
var(x)
weighted_variance(x = x)
weighted_variance(x = x, weights = wt)
weighted_variance(x = x, weights = wt, weights_counts = TRUE)
weights <- c(5, 5, 4, 1)
weighted_variance(x = x, weights = weights)
weighted_variance(x = x, weights = weights, weights_counts = FALSE)
weighted_variance(x = data.frame(x, wt), var = "x",
                   weight = "wt")
# apply by groups:
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, typically `.GlobalEnv`, clean.

**Usage**

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

**Arguments**

- `environment` The environment that should be wiped clean. Defaults to `.GlobalEnv`.
- `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()`
Examples

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

* CSV functions
  bulk_read_csv, 4
  bulk_write_csv, 5
  check_ascii_file, 8
  csv, 11
  csv2csv, 12
* German umlaut converters
  convert_umlauts_to_ascii, 9
  convert_umlauts_to_tex, 10
* R memory functions
  memory_hogs, 42
  wipe_clean, 58
* bits and pieces
  golden_ratio, 24
  is_difftime_less, 27
  is_valid_primary_key, 39
  round_half_away_from_zero, 45
  str2num, 51
  strip_off_attributes, 52
  tapply, 54
  weighted_variance, 57
* call functions
  call_conditionally, 6
  call_safe, 7
* file utilities
  file_modified_last, 13
  find_files, 14
  get_mtime, 18
  get_unique_string, 23
  is_files_current, 29
  is_path, 34
  paths, 44
  search_files, 46
  split_code_file, 50
  touch, 55
* hash functions for objects
  set_hash, 48
  un_hash, 56
* logical helpers
  get_run_r_tests, 20
  is_batch, 25
  is_cran, 26
  is_false, 28
  is_force, 30
  is_installed, 31
  is_not_false, 31
  is_null_or_true, 33
  is_of_length_zero, 34
  is_r_cmd_check, 37
  is_r_package_installed, 37
  is_running_on_fvafrcu_machines, 35
  is_running_on_gitlab_com, 36
  is_success, 38
  is_version_sufficient, 40
  is_windows, 41
* operating system functions
  get_boolean_envvar, 17
  get_run_r_tests, 20
  is_installed, 31
  is_r_package_installed, 37
  is_success, 38
  is_windows, 41
  with_dir, 59
* option functions
  get_options, 18
  is_force, 30
  set_options, 49
* package functions
  get_package_version, 19
  is_r_package_installed, 37
  is_version_sufficient, 40
  load_internal_functions, 41
* package
  fritools-package, 3
* script path getter functions
  get_r_cmd_batch_script_path, 21
  get_rscript_script_path, 20
  get_script_name, 22
get_script_path, 23

* searching functions
  compare_vectors, 9
  file_modified_last, 13
  find_files, 14
  fromto, 15
  missing_docs, 43
  search_files, 46
  search_rows, 47
  summary_filesearch, 53

* subsetting functions
  index_groups, 24
  subset_sizes, 53

* test helpers
  get_boolean_envvar, 17
  get.run.r_tests, 20
  is_cran, 26
  is.r_cmd_check, 37
  is_running_on_fvafrcu_machines, 35
  is_running_on_gitlab_com, 36
  run_r_tests_for_known_hosts, 46
  set.run.r_tests, 50

* version functions
  get_package_version, 19
  is.r_package_installed, 37
  is_version_sufficient, 40
  .GlobalEnv, 58
  .onLoad, 46

attr, 18, 44

base::tapply, 54
base::uname, 52
bulk_read_csv, 4, 5, 8, 12, 13
bulk_write_csv, 4, 5, 8, 12, 13

call_conditionally, 6, 7, 8
call_safe, 7, 7
check_ascii_file, 4, 5, 8, 12, 13
class, 17
codetools::checkUsageEnv, 42
compare_vectors, 9, 13, 15, 16, 43, 47, 48, 54
compareVersion, 19, 40
convert_umlauts_to_ascii, 9, 11
convert_umlauts_to_tex, 10, 10
csv, 4, 5, 8, 11, 13
csv2csv, 4, 5, 8, 12, 12
data.frame, 11, 48, 54, 57
difftime, 27, 29
do.call, 6, 7
exists, 32
FALSE, 17, 21, 25, 27–29, 31–42
file.exists, 14, 15
find_files, 4, 9, 13, 14, 16, 18, 23, 29, 35, 43, 44, 47, 48, 51, 54, 56
get_boolean_envvar, 17, 20, 21, 27, 31, 35–38, 41, 46, 50, 59
gmtime, 13, 15, 18, 23, 29, 35, 44, 47, 51, 56
get_options, 18, 30, 49
get_package_version, 19, 38, 40, 42
get_path(paths), 44
get.r.cmd.batch_script_path, 20, 21, 22, 23
get.r_script_script_path, 20, 22, 23
global.r_tests, 17, 20, 25, 27, 28, 30–38, 40, 41, 46, 50, 59
get.script_name, 20, 22, 23
get.script_path, 20, 22, 23
golden_ratio, 24, 28, 39, 45, 52, 55, 57
.index_groups, 24, 53
integer, 34
interactive, 25
invisible, 27
Invisibly, 12, 42, 46, 47, 49, 51
is_batch, 21, 25, 27, 28, 30–38, 40, 41
is_cran, 17, 21, 25, 26, 28, 30–38, 40, 41, 46, 50
is_difftime_less, 24, 27, 39, 45, 52, 55, 57
is_false, 21, 25, 27, 28, 30–38, 40, 41
is_files_current, 13, 15, 18, 23, 29, 35, 44, 47, 51, 56
is_force, 19, 21, 25, 27, 28, 30, 31–38, 40, 41, 49
is_installed, 17, 21, 25, 27, 28, 30, 31, 32–38, 40, 41, 59
is_not_false, 21, 25, 27, 28, 30, 31, 33–38, 40, 41
is_null_or_true, 21, 25, 27, 28, 30–32, 33, 34–38, 40, 41
is_of_length_zero, 21, 25, 27, 28, 30–33, 34, 35–38, 40, 41
is_path, 13, 15, 18, 23, 29, 34, 44, 47, 51, 56
is_r_cmd_check, 17, 21, 25, 27, 28, 30–36, 37, 38, 40, 41, 46, 50
is_r_package_installed, 17, 19, 21, 25, 27, 28, 30–37, 37, 38, 40–42, 59
is_running_on_fvaרכע_machines, 17, 21, 25, 27, 28, 30–34, 35, 36–38, 40, 41, 46, 50
is_running_on_gitlab_com, 17, 21, 25, 27, 28, 30–35, 36, 37, 38, 40, 41, 46, 50
is_success, 17, 21, 25, 27, 28, 30–38, 38, 40, 41, 59
is_valid_primary_key, 24, 28, 39, 45, 52, 55, 57
is_version_sufficient, 19, 21, 25, 27, 28, 30–38, 40, 41, 42
is_windows, 17, 21, 25, 27, 28, 30–38, 40, 41, 59

length, 22, 23
list.files, 13–15, 47
load_internal_functions, 19, 38, 40, 41
ls, 41, 58

matrix, 48
memory_hogs, 42, 58
missing_docs, 9, 13, 15, 16, 43, 47, 48, 54
mode, 20, 22, 23

NULL, 30–33, 46, 50

options, 49
order, 42
package_version, 19
packageDescription, 19
packageVersion, 19
paths, 13, 15, 18, 23, 29, 35, 44, 47, 51, 56

r_cmd_install, 24, 28, 39, 45, 52, 55, 57
read_csv, 15
read_csv, 4, 12, 44
read_csv (csv), 11
round, 45
round_commercially
(round_halfAway_from_zero), 45
round_halfAway_from_zero, 24, 28, 39, 45, 52, 55, 57
run_r_tests_for_known_hosts, 17, 21, 27, 35–37, 46, 50

search_files, 9, 13, 15, 16, 18, 23, 29, 35, 43, 44, 46, 47, 48, 51, 53, 54, 56
search_rows, 9, 13, 15, 16, 43, 47, 47, 54
set_hash, 48, 57
set_options, 19, 30, 49
set_path (paths), 44
set_run_r_tests, 17, 21, 27, 35–37, 46, 50
source, 51

split_code_file, 13, 15, 18, 23, 29, 35, 44, 47, 50, 56
str2num, 24, 28, 39, 45, 51, 52, 55, 57
strip_off_attributes, 24, 28, 39, 45, 52, 52, 55, 57
subset_sizes, 24, 25, 53
summary.filessearch, 9, 13, 15, 16, 43, 47, 48, 53
Sys.getenv, 17, 50

tapply, 24, 28, 39, 45, 52, 54, 57
throw, 24, 28, 39, 45, 52, 55, 57
touch, 13, 15, 18, 23, 29, 35, 44, 47, 51, 55
TRUE, 6, 25, 27–42, 49
tryCatch, 6

un_hash, 48, 56
utils::read.csv, 11
utils::read.csv2, 11
utils:read.csv2, 11

vector, 57

weighted_variance, 24, 28, 39, 45, 52, 55, 57
wipe_clean, 42, 58
with_dir, 17, 21, 31, 38, 41, 59
write_csv, 5, 12, 44
write_csv (csv), 11