Package 'fritools'

May 31, 2021

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

Version 2.0.0

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

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

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Andreas Dominik Cullmann [aut, cre]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository CRAN

Date/Publication 2021-05-31 21:50:05 UTC

R topics documented:

`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")`.

`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")
```

`compare_vectors` *Compare Two Vectors*

Description

Side-by-side comparison of two vectors.

Usage

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

Arguments

`x, y`	Two vectors of the same mode.

Value

A matrix containing the side-by-side comparison.

Examples

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

`convert_umlauts_to_tex` *Tex Codes for German Umlauts*

Description

Tex Codes for German Umlauts

Usage

```
convert_umlauts_to_tex(x)
```

Arguments

x	A string.

Value

A string with the umlauts converted to plain TeX.

`find_files` *Find Files on Disk*

Description

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

Usage

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

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?

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: search_files(), search_rows(), summary.filesearch()

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()))
```

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.

See Also

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

Examples

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

`get_options` *Get Options For Packages*

Description

A convenience function for `getOption`.

Usage

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

Arguments

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

Value

A (possibly named) list or a vector.

See Also

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

Examples

```
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.

Examples

```
get_package_version("base")
try(get_package_version("mgcv"))
```

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

Description

Get the Path of the R Code File in Case of an 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.

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: `get_boolean_envvar()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `run_r_tests_for_known_hosts()`, `set_run_r_tests()`

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

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

Usage

```
get_r_cmd_batch_script_path()
```

Value

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

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.

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.

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.

Examples

```
replicate(20, get_unique_string())
```

golden_ratio *Calculate the Golden Ratio*

Description

Calculate 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.

Examples

```
golden_ratio(10)
```

index_groups *Determine Indices and Sizes of Subsets*

Description

Determine Indices and Sizes of Subsets

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.

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

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

Usage

```
is_batch()
```

Value

A boolean.

Examples

```
is_batch()
```

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 codeFALSE, FALSE otherwise.

See Also

Other logical helpers: `is_not_false()`, `is_null_or_true()`

Examples

```
is_false("not false")
is_false(FALSE)
```

is_force *Opt-out Via Option*

Description

Opt-out Via Option

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()

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.

Examples

```
is_installed("R")
is_installed("probably_not_installed")
```

---

is_not_false *Test if an Object is Not* FALSE

Description

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

Usage

```
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: is_false(), is_null_or_true()

Examples

```
a  <- 1
is_not_false(a)
f <- function() {
    print(a)
    print(is_not_false(a))
}
f()

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

```
}
f()

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

`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 codeTRUE or codeNULL, `FALSE` otherwise.

See Also

`is_force`

Other logical helpers: `is_false()`, `is_not_false()`

Examples

```
is_false("not false")
is_false(FALSE)
```

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.

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

Description

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

Usage

```
is_running_on_fvafrcu_machines()
```

Value

`TRUE` on success, `FALSE` otherwise.

See Also

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

---

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 test helpers: get_boolean_envvar(), get_run_r_tests(), is_running_on_fvafrcu_machines(), run_r_tests_for_known_hosts(), set_run_r_tests()

---

is_r_cmd_check *Is the Current R Process an* R CMD check*?*

Description

Is the Current R Process an R CMD check?

Usage

```
is_r_cmd_check()
```

Value

TRUE on success, FALSE otherwise.

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.

Examples

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

is_success *Does the Return Value of a Command Signal Success?*

Description

This is just a wrapper to ease the evaluation of retrun 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.

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.

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.

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.

Details

The files to be checked get sourced, which means they have to contain R code producing no errors. If we want to check the source code of a package, we need to load the package *and* be able to run all its internals in our environment.

Value

invisible(TRUE)

See Also

checkUsageEnv in codetools.

Examples

```
load_internal_functions("fritools")
```

memory_hogs *Find Memory Hogs*

Description

Find Memory Hogs

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.

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)
```

---

paths *Set or Get the* path *Attribute to or from an Object*

Description

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

Usage

```
get_path(x)

set_path(x, path, must_work = TRUE)
```

Arguments

x	An object.
path	The path to be set.
must_work	See normalizePath's argument mustWork.

Value

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

For set_path the modified object.

Examples

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

---

run_r_tests_for_known_hosts

Enforce the Environment Variable RUN_R_TESTS to TRUE on Known Hosts

---

Description

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

Usage

```
run_r_tests_for_known_hosts()
```

Value

Invisibly NULL.

See Also

Other test helpers: get_boolean_envvar(), get_run_r_tests(), 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: find_files(), search_rows(), summary.filesearch()

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 whether all rows of a matrix-like structure containing 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: `find_files()`, `search_files()`, `summary.filesearch()`

Examples

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

set_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

invisible(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 System Variable RUN_R_TESTS*

Description

Set System Variable 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: `get_boolean_envvar()`, `get_run_r_tests()`, `is_running_on_fvafrcu_machines()`, `is_running_on_gitlab_com()`, `run_r_tests_for_known_hosts()`

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()
```

---

split_code_file *Split a Code File Into Multiple Files Each Containing a Single Function*

Description

I tend to find files with dozens of functions. They don't read well.

Usage

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

Arguments

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

Value

`Invisibly` a vector of paths to the new files.

Examples

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

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

## End(Not run)
```

`strip_off_attributes` *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`

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.

Examples

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

summary.filesearch *Summarize File Searches*

Description

Summarize File Searches

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: `find_files()`, `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.frame`s to be passed as `X`.

Usage

```
tapply(X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)
```

Arguments

`X`	See `base::tapply`.
`INDEX`	See `base::tapply`.
`FUN`	See `base::tapply`.
`...`	See `base::tapply`.
`default`	See `base::tapply`.
`simplify`	See `base::tapply`.

Value

See `base::tapply`.

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) {
    weighted.mean(df[[x]], df[[w]])
}
if (RUnit::checkIdentical(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(X = mtcars, INDEX = list(mtcars[["cyl"]],
                                                mtcars[["vs"]]),
                       FUN = 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 a file or ensuring a file's modification time changes.

Usage

```
touch(path)
```

Arguments

path	Path to the file to be touched.

Value

The Path to the file touched.

Examples

```
file <- tempfile()
touch(file)
t1 <- file.mtime(file)
touch(file)
t2 <- file.mtime(file)
t1 < t2
```

---

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*.

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(X = mtcars, INDEX = list(mtcars[["cyl"]], mtcars[["vs"]]),
                       FUN = weighted_variance, var = "mpg", w = "wt")
```

`wipe_clean` *Remove All Objects From an Environment*

Description

Remove All Objects From an Environment

Usage

```
wipe_clean(environment)
```

Arguments

`environment`	The environment that should be wiped clean.

Value

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

Examples

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

`with_dir` *Execute Code in a Temporary Working Directory*

Description

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

Usage

```
with_dir(new, code)
```

Arguments

`new`	The new working directory.
`code`	Code to execute in the temporary working directory.

Value

The results of the evaluation of the `code` argument.

Examples

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

Index