Package 'pathological'

February 15, 2017

Type Package

Title Path Manipulation Utilities

Version 0.1-2

Date 2017-02-14

Author Richard Cotton [aut, cre], Janko Thyson [ctb]

Maintainer Richard Cotton <richierocks@gmail.com>

Description Utilities for paths, files and directories.

URL https://github.com/richierocks/pathological

BugReports https://github.com/richierocks/pathological/issues

Depends R (>= 2.15.0)

Imports assertive.base (>= 0.0-3), assertive.files, assertive.numbers, assertive.properties, assertive.reflection, assertive.strings, assertive.types, magrittr, plyr, stringi, utils

Suggests knitr, rmarkdown, testthat, withr

License Unlimited

LazyLoad yes

Acknowledgments Development of this package was partially funded by the Proteomics Core at Weill Cornell Medical College in Qatar <http://qatar-weill.cornell.edu>. The Core is supported by 'Biomedical Research Program' funds, a program funded by Qatar Foundation.

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-15 15:14:03

R topics documented:

`choose_dir` *Choose files interactively*

Description

Choose one or more files or a directory interactively using a pop-up dialog.

Usage

```
choose_dir(default = "", sep = c("/", "\\"))

choose_files(default = "", multi = FALSE, sep = c("/", "\\"))
```

Arguments

`default`	The default file to be selected. See the Details section of `choose.files` for how to specify. Only on Windows.
`sep`	String separator between directory levels in the output.
`multi`	Logical value indicating if multiple files can be selected. Only on Windows.

Value

A character vector of standardized file paths that were chosen.

Note

choose_files uses choose.files under Windows and file.choose under other platforms.

Examples

```
if(interactive())
{
  choose_files()
  if(assertive.reflection::is_windows())
  {
    choose_dir()
  }
}
```

---

copy_dir *Copy the contents of a directory*

Description

Copies the contents of a directory, possibly recursively.

Usage

```
copy_dir(source_dir, target_dir, pattern = NULL, overwrite = FALSE,
  recursive = TRUE)

dir_copy(...)
```

Arguments

source_dir	String of directory to copy from.
target_dir	String of directory to copy to.
pattern	String regex or NULL. A filter for filenames, passed to dir.
overwrite	Logical value. Should existing files be overwritten?
recursive	Logical value. Should subdirectories and their contents be copied?
...	Passed from the defunct dir_copy to copy_dir.

Value

A logical vector of whether or not each file was successfully copied is invisibly returned.

Note

Target directories that don't exist are created, silently (assuming write permission).

See Also

basename

Examples

```
## Not run:
#Copy subdirs by default
copy_dir(R.home("etc"), file.path(tempdir(), "etc"))
#Just copy the top level
copy_dir(R.home("etc"), file.path(tempdir(), "etc2"), recursive = FALSE)
#Now copy deeper levels, without overwriting.
copy_dir(R.home("etc"), file.path(tempdir(), "etc2"), overwrite = FALSE)
#Cleanup
unlink(file.path(tempdir(), "etc"), recursive = TRUE)
unlink(file.path(tempdir(), "etc2"), recursive = TRUE)

## End(Not run)
```

---

create_dirs *Create or remove files and directories*

Description

A vectorized version of dir.create, and file.create and unlink with more convenient defaults.

Usage

```
create_dirs(x = temp_file(pattern = "dir"))

create_files(x = temp_file())

remove_dirs(x)
```

Arguments

x	A character vector of paths of directories to create/remove. For create_dirs, it defaults to a directory inside tempdir().

Value

A logical vector of successes of failures.

Note

create_dirs will only attempt to create directories that don't already exist.

See Also

dir.create, unlink

Examples

```
dirs <- temp_dir(c("foo", "bar/baz"))
create_dirs(dirs)

# Check this worked:
assertive.files::assert_all_are_dirs(dirs)

files <- temp_dir("blah/blah/blah", LETTERS)
create_files(files)

assertive.files::assert_all_are_existing_files(files)

# Clean up
remove_dirs(temp_dir(c("foo", "bar", "blah")))
```

---

cygwinify_path *Make a path suitable for cygwin*

Description

By default, cygwin complains about standard paths. This function converts paths to a form that cygwin likes.

Usage

```
cygwinify_path(x = dir())
```

Arguments

x	A character vector of file paths. Defaults to files in the current directory.

Value

A character vector of the cygwinified inputs.

See Also

`standardize_path`

Examples

```
# Connecting to a non-existent network drive is slow
cygwinify_path(c("c:/Program Files", "\\\\some/network/drive"))
```

decompose_path *Split a path into its components*

Description

decompose_path splits a path into the directory name, filename without extension, and extension. strip_extension, get_extension and replace_extension provide shortcuts to manipulate the file extension. recompose_path takes the result of decompose_path and returns complete paths.

Usage

```
decompose_path(x = dir())

get_extension(x = dir())

recompose_path(x, ...)

## S3 method for class 'decomposed_path'
recompose_path(x, ...)

replace_extension(x = dir(), new_extension, include_dir = NA)

strip_extension(x = dir(), include_dir = NA)
```

Arguments

x	A character vector of file paths. Defaults to files in the current directory.
...	Not currently used.
new_extension	A new extension to replace the existing ones.
include_dir	Should the directory part of the path be included? If NA, the default, keep the directory from the input. If TRUE, standardize the directory. If FALSE, strip the directory.

Value

decompose_path returns a character matrix with three columns named "dirname", "filename" and "extension". strip_extension returns a character vector of the filename, possibly with a directory (see include_dir argument). replace_extension returns a character vector of the filename with a newextension, possibly with a directory (see include_dir argument). get_extension returns a character vector of the third column. recompose_path returns a character vector of paths.

Warning

A few of the tests for this function don't pass under the CRAN Windows machine. It is unclear exactly why this is happening, and the failing tests have not been reproduced elsewhere. If you have unexpected behaviour with this function, please report it on the package issue tracker. https://github.com/richierocks/pathological/issues

Note

Decomposing and then recomposing a path is usually equivalent to standardizing that path (though slower). That is, usually `recompose_path(decompose_path(x)) == standardize_path(x)`. One exception to this is when the directory of x is a symbolic link to another directory. In this case `decompose_path` will follow the link but `standardize_path` won't.

See Also

`file_ext`, a primitive version of `get_extension`

Examples

```
x <- c(
  "somedir/foo.tgz",         # single extension
  "another dir\\bar.tar.gz", # double extension
  "baz",                     # no extension
  "quux. quuux.tbz2",        # single ext, dots in filename
  R.home(),                  # a dir
  "~",                       # another dir
  "~/quuuux.tar.xz",         # a file in a dir
  "",                        # empty
  ".",                       # current dir
  "..",                      # parent dir
  NA_character_              # missing
)
(decomposed <- decompose_path(x))
get_extension(x)
strip_extension(x)
strip_extension(x, FALSE)
recompose_path(decomposed)
```

---

get_libraries *Get the libraries on your machine*

Description

Wrapper to `.libPaths` that gets all the libraries that R knows about on your machine.

Usage

```
get_libraries(index = TRUE, sep = c("/", "\\"))
```

Arguments

index	A numeric or logical vector specifying the index of the libraries to return. By default, all libraries are returned.
sep	String separator between directory levels in the output.

Value

A character vector of paths to libraries.

References

https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-the-difference-between-package-and-library_003f

See Also

.libPaths

Examples

```
get_libraries()
get_libraries(1)
```

---

get_windows_drive *On Windows, return the drive of the path*

Description

On a Windows system, this returns the drive letter of the path followed by a colon. On other systems, it returns a single forward slash.

Usage

```
get_windows_drive(x = getwd())

get_drive(x = getwd())
```

Arguments

x	A character vector of file paths. Defaults to the current directory.

Value

A character vector of drive paths on Windows systems, or forward slashes on Unix-based systems.

See Also

is_windows_drive

Examples

```
 # often takes > 5s to run
get_windows_drive(c(".", "~", r_home(), temp_dir(), "\\\\foo/bar"))
```

`is_windows_drive` *Is the path a Windows drive?*

Description

Checks to see if the path is a Windows drive.

Usage

```
is_windows_drive(x)
```

Arguments

`x`	A character vector of file paths. Defaults to files in the current directory.

Value

A logical vector, `TRUE` when the path is a Windows drive name. On non-Windows machines, the return value is `FALSE` everywhere.

Note

The check is done by regular expression: values are considered to be Windows drive name if they consist of a letter followed by a colon, optionally followed by a slash or backslash. Paths are standardardized before checking, so `.` and `..` are resolved to their actual locations rather than always returning `FALSE`.

See Also

`get_drive`

Examples

```
x <- c("c:", "c:/", "c:\\", "C:", "C:/", "C:\\", "c:/c", "cc:", NA)
# Warnings about OS suppressed so package checks pass on non-Windows systems.
suppressWarnings(is_windows_drive(x))
```

os_path *The OS path*

Description

The locations in the operating system PATH environment variable.

Usage

```
os_path(sep = c("/", "\\"), standardize = TRUE, splitter = if
  (is_windows()) ";" else ":")
```

Arguments

sep	String separator between directory levels in the output.
standardize	Should the paths be standardized?
splitter	The character to split the PATH environment variable on. Defaults to a semicolon on Windows systems and a colon elsewhere.

Value

A character vector of paths.

See Also

Sys.getenv

Examples

```
os_path()
```

parent_dir *Get the parent dir*

Description

Gets the parent directory of the input.

Usage

```
parent_dir(x = ".", sep = c("/", "\\"))
```

Arguments

x	A character vector of file paths.
sep	String separator between directory levels in the output.

Value

A character vector of parent directories of the input.

Note

Missing values are returned as missing. On Windows, the parent of a drive, e.g., `"c:/"` is itself. Likewise, under Unix, the parent of `"/"` is itself.

Examples

```
(x <- c(
  sys_which("R"),
  r_home(),
  r_profile_site(),
  "c:/",  # different behaviour under Windows/Unix
  "~",
  "/",
  "foo/bar/nonexistent",
  NA
))
parent_dir(x)
```

`pathological` *pathological: utilities for paths, files and directories.*

Description

This package contains utilities for manipulating paths, files and directories.

Details

`decompose_path` splits a path into the directory name, filename without extension, and extension. `strip_extension` and `get_extension` provide shortcuts to the second and third parts of the filename. `recompose_path` takes the result of `decompose_path` and returns complete paths.

`copy_dir` copies the contents of a directory, possibly recursively.

Author(s)

Richie Cotton

rstudio_project_dir *Get the RStudio project directory*

Description

Gets the current RStudio project directory.

Usage

```
rstudio_project_dir(sep = c("/", "\\"))
```

Arguments

sep	String separator between directory levels in the output.

Value

A string giving the path to the current RStudio project directory, or character vector with length zero if you are running RStudio without a project open.

Note

This only works when your IDE is RStudio. Otherwise an error is thrown.

Examples

```
assertive.base::dont_stop(rstudio_project_dir())
```

r_environ *Get the location of the R profile/environ*

Description

Gets the location of the user or site R profile and environ startup files.

Usage

```
r_environ(sep = c("/", "\\"))

r_environ_site(sep = c("/", "\\"))

r_profile(sep = c("/", "\\"))

r_profile_site(sep = c("/", "\\"))
```

Arguments

sep	String separator between directory levels in the output.

Value

A string giving the path the ".Rprofile", ".Renviron", "Rprofile.site", or ".Renviron.site". If the file cannot be found, NA is returned.

See Also

Startup for how this is calculated.

Examples

```
r_environ()
r_environ_site()
r_profile()
r_profile_site()
```

---

r_home *The R home directory*

Description

Return a path to a file in the R home directory. A vectorized, standardized version of R.home.

Usage

```
r_home(component = "home", ..., sep = c("/", "\\"))
```

Arguments

component	"home" for the root of the R installation directory, or the name of a subdirectory.
...	Further subdirectories passed to file.path.
sep	String separator between directory levels in the output.

Value

A character vector of paths inside the R installation dir.

Note

The component argument has special behaviour for the values "home", "bin", "doc", "etc", "include", "modules", and "share". See the help page for R.home.

See Also

R.home

Examples

```
r_home()
r_home("etc", "Rprofile.site")
r_home(c("home", "bin", "share"), c("", "i386", "zoneinfo"))
```

`split_path` *Split a path into directory components*

Description

`split_path` splits a character vector of paths into directory components. The opposite of `file.path`. `split_dir` is a convenience wrapper equivalent to `dir` + `split_path`, making it easy to `View` directory contents.

Usage

```
split_path(x = dir(), simplify = FALSE)

split_dir(x = ".", pattern = NULL, all.files = TRUE, recursive = TRUE,
  simplify = TRUE)
```

Arguments

`x`	A character vector. For `split_path`, this should contain file paths, and it defaults to files in the current directory. For `split_dir`, this should contain directory paths, and it defaults to the current directory.
`simplify`	A logical value. If `TRUE`, the return value is simplified from a list to a matrix.
`pattern`	A string containing a regular expression, to filter the files that are returned. Passed to `dir`.
`all.files`	Logical. If `TRUE`, files whose name starts with a dot are included. Passed to `dir`.
`recursive`	Logical. If `TRUE`, files in subdirectories are included. Passed to `dir`.

Value

Either a named list of character vectors containing the split paths, or a matrix. See `simplify` argument in Usage section.

Note

Paths are split on forward and back slashes, except for double forward or back slashes at the start of (UNC) paths. These are included in the first element of that split path.

See Also

`file.path`, `dir`

Examples

```
(splits <- split_path(c(getwd(), "~", r_home())))
# Reverse the operation
sapply(splits, paste, collapse = "/")

base_r_files <- split_dir(R.home(), pattern = "\\.R$")

  # Viewing not needed for testing purposes
  utils::View(base_r_files)
```

`standardize_path` *Standardize paths*

Description

Standardi[sz]e path names so that they can be more easily compared.

Usage

```
standardize_path(x = dir(), sep = c("/", "\\"), include_names = TRUE)

standardise_path(x = dir(), sep = c("/", "\\"), include_names = TRUE)
```

Arguments

`x`	A character vector of file paths. Defaults to files in the current directory.
`sep`	String separator between directory levels in the output.
`include_names`	A logical value indicating whether the output should be named with the input file paths.

Details

`standardize_path` wraps `normalizePath`, providing additional tweaks to the output.

- Missing inputs always return `NA_character_`.
- Leading double back slashes are preserved under all OSes regardless of the values of `sep`.
- Leading double forward slashes are converted to double back slash under Windows (they are likely UNC paths), and a single forward slash under Unixes (they are likely absolute paths).
- Other back and forward slashes are replaced by `sep`.
- Paths are always made absolute.
- Trailing slashes are always stripped, except for root (`"/"`) and Windows drives (`"C:/"`, etc.).
- Windows drives are always capitalized.

Value

A character vector of paths, pointing to the same locations as the input, but in a standardized form.

See Also

normalizePath, path.expand, file_path_as_absolute, getAbsolutePath

Examples

```
standardize_path(c(".", "..", "~", R.home(), NA))
standardize_path(c(".", "..", "~", R.home(), NA), "\\")
```

---

system_file *Find a file in a package*

Description

Wrapper to system.file that returns standardized paths.

Usage

```
system_file(..., package = "base", library_location = NULL,
  must_work = FALSE, sep = c("/", "\\"))
```

Arguments

...	Character vectors specifying subdirectories and files within some package. The default, none, returns the root of the package. Wildcards are not supported.
package	A string with the name of a single package. An error occurs if more than one package name is given.
library_location	a character vector with path names of R libraries. See the 'Details section of system.file for the meaning of the default value of NULL.
must_work	If TRUE, an error is given if there are no matching files.
sep	String separator between directory levels in the output.

Value

A character vector of positive length, containing the file paths that matched . . . , or a missing string, NA, if none matched (unless mustWork = TRUE). (This behaviour for missing paths differs from system.file.) If matching the root of a package, there is no trailing separator. system.file() with no arguments gives the root of the base package.

See Also

system.file

Examples

```
# Examples taken from ?system.file
system_file()                 # The root of the 'base' package
system_file(package = "stats") # The root of package 'stats'
system_file("INDEX")
system_file("help", "AnIndex", package = "splines")
```

`sys_which` *Find paths to executables*

Description

Wrapper to `Sys.which`, that returns standardized paths.

Usage

```
sys_which(x, sep = c("/", "\\"))
```

Arguments

`x`	A character vector of executables.
`sep`	String separator between directory levels in the output.

Value

A character vector of paths to those executables, or `NA` if it doesn't exist. (This behaviour for missing executables differs from `Sys.which`.)

See Also

`Sys.which`

Examples

```
sys_which("R")              # R executable
sys_which(c("make", "gcc")) # tools for running Rcpp
```

temp_dir *Return paths to files or dirs within the temp dir*

Description

Vectorized wrappers to `tempdir` and `tempfile` that return standardized paths.

Usage

```
temp_dir(..., sep = c("/", "\\"))

temp_file(..., pattern = "file", fileext = "", sep = c("/", "\\"))
```

Arguments

`...`	Character vectors of further directories within the temp directory. Passed to `file.path`.
`sep`	String separator between directory levels in the output.
`pattern`	Character vector of prefixes for the temp file name. Passed to `tempfile`.
`fileext`	Character vector of file extensions for the temp file. Passed to `tempfile`.

Value

For `temp_file` a character vector giving the names of possible (temporary) files. Note that no files are generated by `temp_file`. For `temp_dir`, the path of the per-session temporary directory.

See Also

`tempdir`

Examples

```
temp_dir(c("foo", "bar/baz"))
temp_file(c("foo", "bar/baz"), fileext = c(".txt", ".R"))
```

Index