Package: kwb.fakin (via r-universe)

August 24, 2024

Type Package

Title Functions Used in Our Fakin Project

Version 0.4.2

Description This package contains all functions and evaluations related to KWB project ``FAKIN''. Set the Subversion setting 'enable-auto-props' to 'yes' so that the 'auto-probs' options are considered.

Encoding UTF-8

Imports cowplot (>= 0.9.4), data.table (>= 1.11.8), digest (>= 0.6.18), dplyr (>= 0.7.8), fakin.path.app, fs (>= 1.2.6), ggplot2 (>= 3.2.0), kwb.code (>= 0.1.0), kwb.file (>= 0.2.1), kwb.plot (>= 0.2.0), kwb.prep, kwb.utils (>= 0.12.0.9000), methods, magrittr (>= 1.5), pathlist (>= 0.2.1), rlang (>= 0.4.0), stringr (>= 1.3.1), yaml (> 2.1)

Remotes github::hsonne/pathlist, github::kwb-r/fakin.path.app, github::kwb-r/kwb.code, github::kwb-r/kwb.file, github::kwb-r/kwb.pathdict, github::kwb-r/kwb.plot, github::kwb-r/kwb.prep, github::kwb-r/kwb.utils

Suggests covr (>= 3.2.1), jsonlite (>= 1.6), knitr (>= 1.20), kwb.pathdict (>= 0.1.0), networkD3 (>= 0.4), rmarkdown (>= 1.10), testthat (>= 2.0.1), treemap (>= 2.4.2)

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 7.1.2

VignetteBuilder knitr

URL https://github.com/kwb-r/kwb.fakin

BugReports https://github.com/kwb-r/kwb.fakin/issues

Repository https://kwb-r.r-universe.dev

RemoteUrl https://github.com/KWB-R/kwb.fakin

RemoteRef HEAD

RemoteSha 17ab0e6e9a63a03c6cb40ef29ee3899c2b2724a0

Contents

`all_path_levels` *All Paths to Parent Folders*

Description

For a given path `a/b/c/d`, all the parent paths `a`, `a/b`, `a/b/c` and the path itself (`a/b/c/d`) are returned.

Usage

```
all_path_levels(path)
```

Arguments

`path`	one character string representing a file path

Value

vector of character representing all parent paths and the `path` itself as the last element

Examples

```
paths <- kwb.fakin:::all_path_levels("this/is/a/long/path")
kwb.file:::to_subdir_matrix(paths)
```

`build_folders_from_file`

Create Folder Structure from Paths in File

Description

Create Folder Structure from Paths in File

Usage

```
build_folders_from_file(
  file,
  target_dir,
  pattern = NULL,
  max_depth = NULL,
  encoding = "Latin-1"
)
```

Arguments

`file`	path to file containing path strings
`target_dir`	path to target directory in which to create the folder structure
`pattern`	regular expression matching the paths from `file` to be considered
`max_depth`	maximum folder depth to be considered
`encoding`	encoding used when reading `file`

Examples

```
# Create a vector of example paths
paths <- c("a1/b1", "a1/b2", "a2/b1", "a2/b1/c1")

# Write the example paths to a temporary file
writeLines(paths, file <- tempfile())

# Create a temporary target directory
target_dir <- kwb.utils::createDirectory(file.path(tempdir(), "test"))

# Create the folder structure as defined by the paths in the temporary file
```

```
kwb.fakin::build_folders_from_file(file, target_dir)

# List the directory paths below the target directory
paths_reread <- list.dirs(target_dir, recursive = TRUE, full.names = FALSE)

# Stop if not all paths have been created
stopifnot(all(paths %in% paths_reread))
```

---

createLocalProject *Create Empty Project Folder Structure Locally*

Description

Copy the pure folder structure (without files) of a given KWB project from the server to a new local folder below "C:/Users/<user_name>/Documents/Projekte"

Usage

```
createLocalProject(
  project,
  start_directory = options()$kwb.fakin.paths$projects
)
```

Arguments

`project`	Project name. Must correspond with the name of a folder below one of the server locations returned by `getProjectPaths`
`start_directory`	Path to the project network drive at KWB

---

cut.path_tree *Cut a Path Tree*

Description

Reduce a path tree to its first levels.

Usage

```
## S3 method for class 'path_tree'
cut(x, n_levels = 2, depth = 0, ...)
```

Arguments

x	tree object as returned by kwb.fakin:::to_tree
n_levels	number of levels to which the tree is cut
depth	current depth level
...	further arguments (currently not used)

---

extdata_file *Path to File in extdata Folder of this Package*

Description

Path to File in extdata Folder of this Package

Usage

```
extdata_file(file)
```

Arguments

file	filename or path to the file, relative to extdata/

---

extract_properties *Extract Pattern-Defined Properties from Strings*

Description

Extract Pattern-Defined Properties from Strings

Usage

```
extract_properties(x, patterns, replacements, as_data_frame = FALSE)
```

Arguments

x	vector of character
patterns	vector of character
replacements	vector of character
as_data_frame	logical. If TRUE (the default is FALSE), a data frame is returned.

Examples

```
# Define patterns to be matched against
patterns <- c(
  "[Bb]ericht",
  "[- ](\\d+)$",
  "Abschluss",
  "Zwischen",
  "_HS$"
)

# Define property:value pairs (or even prop1:value1+prob2:value2+...)
# referring to parts of the pattern enclosed in parentheses with \1, \2, ...
replacements <- c(
  "type:report",
  "number:\\1",
  "stage:final",
  "stage:intermediate",
  "author:Sonnenberg+reviewed:true"
)

# Define strings in which to look for properties and their values
x <- c("Bericht", "Bericht 1", "Abschlussbericht", "Zwischenbericht_HS")

# Extract property values as strings
extract_properties(x = x, patterns, replacements)

# Arrange the properties in a data frame
extract_properties(x = x, patterns, replacements, as_data_frame = TRUE)
```

---

getProjectPaths *Paths where to find project on the KWB server*

Description

Paths where to find project on the KWB server

Usage

```
getProjectPaths(start_directory, as_list = FALSE, skip_pattern = "/_")
```

Arguments

start_directory	Path to the project network drive at KWB
as_list	If TRUE (the default is FALSE) the paths are returned as a list with the folder names as list element names.
skip_pattern	pattern matching paths to be removed from the returned path list. By default all paths containing underscore at the beginning of a subdirectory name are removed.

Value

full paths to project folders as a vector of character or as a named list if `as_list = TRUE`.

---

get_depth_summaries *Get File Number and Size Summary per Folder Depth*

Description

Get File Number and Size Summary per Folder Depth

Usage

```
get_depth_summaries(file_data, project_dir, max_depth = NULL)
```

Arguments

`file_data`	data frame as returned by `read_file_info`
`project_dir`	path by which to filter the paths in `file_data`, passed to `fakin.path.app:::prepare_for_treemap`
`max_depth`	maximum depth for which to calculate a summary. If `NULL` (default), all summaries are created for all available path depths

Value

list of data frames

---

get_example_read_and_write_matrices *Get Example Matrices of Read- and Write-Permissions*

Description

Get Example Matrices of Read- and Write-Permissions

Usage

```
get_example_read_and_write_matrices()
```

get_file_duplicates *Get list of paths containing files of the same name*

Description

Get list of paths containing files of the same name

Usage

```
get_file_duplicates(paths, pattern, ...)
```

Arguments

paths	vector of character representing full file paths
pattern	paths is matched against this pattern before the matching paths are split and analysed for duplicated files
...	arguments passed to grep, e.g. ignore.case

Examples

```
paths <- c("/a/b/c.exe", "/a/b/d.exe", "/A/B/c.exe", "/A/B/d.exe")
get_file_duplicates(paths, pattern = "\\.exe$")
```

get_function_call_frequency *Which Function is Called How Often?*

Description

Which Function is Called How Often?

Usage

```
get_function_call_frequency(tree, simple = FALSE, dbg = TRUE)
```

Arguments

tree	parse tree as returned by parse_scripts
simple	if TRUE, a simple approach using a simple regular expression is used. This approach is fast but not correct as it e.g. counts function calls that are commented out or even string expressions that just look like function calls. Leaving this argument to its default, FALSE, will return only real function calls by evaluating the full structure of parse tree.
dbg	if TRUE, debug messages are shown

Value

data frame with columns `name` (name of function), `count` (number of times the function is called)

`get_package_function_usage`
How Often Are the Functions of a Package Used?

Description

How Often Are the Functions of a Package Used?

Usage

```
get_package_function_usage(tree, package, simple = FALSE, by_script = FALSE)
```

Arguments

`tree`	parse tree as returned by `parse_scripts`
`package`	name of the package (must be installed)
`simple`	if `TRUE`, a simple approach using a simple regular expression is used. This approach is fast but not correct as it e.g. counts function calls that are commented out or even string expressions that just look like function calls. Leaving this argument to its default, `FALSE`, will return only real function calls by evaluating the full structure of parse tree.
`by_script`	if `TRUE` the functions are counted and returned by script, otherwise they are counted over all scripts

Value

data frame with columns `name` (name of the function), `prefixed` (number of function calls prefixed with `<package>::` or `<package>:::`), `non_prefixed` (number of function calls that are not prefixed with the package name) and `total` (total number of function calls)

Examples

```
# Read all scripts that are provided in the kwb.fakin package
tree <- kwb.code::parse_scripts(root = system.file(package = "kwb.fakin"))

# Check which functions from kwb.utils are used and how often
get_package_function_usage(tree, package = "kwb.utils")

# Hm, this does not seem to be the whole truth...
```

get_path_stat_matrix *How Many Folders of the Template are in the Project Folder?*

Description

How Many Folders of the Template are in the Project Folder?

Usage

```
get_path_stat_matrix(project_folder, template_folders)
```

Arguments

project_folder	folder in which to look for the projects of one KWB department, e.g. ".../SUW_Department/Projects"
template_folders	vector of relative paths of folders expected to be contained within each folder within project_folder

get_special_character_info *Get Special Characters and Their Byte Codes*

Description

Get Special Characters and Their Byte Codes

Usage

```
get_special_character_info(text, context_length = 7, bytes_per_char = 2)
```

Arguments

text	vector of character of length one
context_length	number of characters left and right of special character to be put into column context
bytes_per_char	number of bytes per character

Value

data frame with columns special (special characters) and bytes (hexadecimanl byte codes as a space separated string), context (strings "around" the special characters)

Examples

```
(text <- kwb.fakin:::example_string_with_specials("de"))

get_special_character_info(text)
```

list_files *Get File Paths Recursively (Windows only!)*

Description

Get a full list of files below a `root` directory using the `dir` function and write it to a given `file`

Usage

```
list_files(root, file, use_batch = TRUE)
```

Arguments

`root`	path to the directory from which to start "downwards" and recursively for files and folders.
`file`	path to the result file (text) to which the paths are to be written.
`use_batch`	if `TRUE` (default), a batch file is written (by default to `list_files.bat` in `tempdir()`) and run to perform the dir command

merge_read_and_write_matrices *Merge Read and Write Permission Matrices*

Description

Merge Read and Write Permission Matrices

Usage

```
merge_read_and_write_matrices(matrix_read, matrix_write)
```

Arguments

`matrix_read`	matrix of 0 with 1 at positions with read-permissions
`matrix_write`	matrix of 0 with 2 at positions with write-permissions

Value

matrix of character with "" at positions without permissions, "-" at positions with read-permissions, "|" at positions with write-permissions and "+" at positions with read- and write-permissions

Examples

```
# Get example matrices
matrices <- get_example_read_and_write_matrices()

# Overlay example matrices
merge_read_and_write_matrices(matrices$read, matrices$write)
```

`plot_biggest_folders` *Plot Folders with Maximum Number of Files*

Description

Plot Folders with Maximum Number of Files

Usage

```
plot_biggest_folders(tree, max_depth = 5, to_pdf = TRUE)
```

Arguments

`tree`	tree list structure as returned by `to_tree`
`max_depth`	maximum number of folder depth
`to_pdf`	if `TRUE` the output is directed to a temporary pdf file

`plot_files_in_depth` *Plot File Sizes over Folder Depth*

Description

Plot File Sizes over Folder Depth

Usage

```
plot_files_in_depth(depth_summaries, project)
```

Arguments

`depth_summaries`	list as returned by `get_depth_summaries`
`project`	name of project (in fact folder name in folder depth 1)

plot_file_distribution *Plot Distributions of Files in Folder Depths and Name Summaries*

Description

Plot Distributions of Files in Folder Depths and Name Summaries

Usage

```
plot_file_distribution(file_data, start_path, n_root_parts, ..., to_pdf = TRUE)
```

Arguments

file_data	data frame with columns path, type, size
start_path	only paths from file_data are considered that start with this path
n_root_parts	number of first path segments to be considered as "root"
...	further arguments passed to kwb.fakin:::plot_file_size_in_depth
to_pdf	if TRUE (default) the results are plotted into a temporary pdf file that is opened in a pdf viewer

Examples

```
# Set a seed for the random number generator
set.seed(20190625)

# Create random paths
paths <- kwb.pathdict::random_paths()

# Number of paths
n <- length(paths)

# Create artificial file data (invent sizes)
file_data <- kwb.utils::noFactorDataFrame(
  path = paths,
  type = "file",
  size = abs(rnorm(n)) * kwb.fakin:::bytes_to_mib(
    2^sample(30, n, replace = TRUE)
  )
)

kwb.fakin::plot_file_distribution(
  file_data, start_path = "reason", n_root_parts = 1, to_pdf = FALSE
)
```

plot_number_of_elements_per_folder *Plot Folders with Number of Direct Children*

Description

Plot Folders with Number of Direct Children

Usage

```
plot_number_of_elements_per_folder(
  x,
  main = "",
  to_pdf = FALSE,
  max_chars = 20
)
```

Arguments

x	tree list structure as returned by to_tree
main	title of the plot
to_pdf	if TRUE the output is directed to a temporary pdf file
max_chars	maximum number of characters to be used for file or folder names

prepare_path_data *Select and Filter for Relevant Path Information*

Description

This function gets a data frame containing path information as input. It filters for rows with value "file" in column type and keeps only the columns path and size. If pattern is not NULL, the data frame is then filtered for rows in which path matches the given pattern. Finally, the common root of all paths in column path is removed and the resulting data frame is returned.

Usage

```
prepare_path_data(path_info, pattern = NULL)
```

Arguments

path_info	data frame containing file path information as returned by read_file_info
pattern	pattern by which to select a subset of paths or NULL (default) if all paths in path_info are to be considered. By setting the pattern to "^/path/to/start/directory" you can "zoom into" the path tree, returning only the contents of "/path/to/start/directory".

Value

data frame with columns `path` and `size`. See Description.

Examples

```
path_info <- kwb.utils::noFactorDataFrame(
  path = c("/path/to/root/", "/path/to/root/file_1", "/path/to/root/file_2"),
  type = c("directory", "file", "file"),
  size = c(0L, 10L, 20L)
)

path_info

kwb.fakin:::prepare_path_data(path_info)
```

`print.path_tree` *Print a tree*

Description

Print a tree

Usage

```
## S3 method for class 'path_tree'
print(x, max_depth = 2, ...)
```

Arguments

`x`	tree object as returned by `kwb.fakin:::to_tree`
`max_depth`	number of depth levels to be printed
`...`	further arguments (currently not used)

`read_csv` *Read Data from CSV File*

Description

Read Data from CSV File

Usage

```
read_csv(file, sep = ";", version = 2, fileEncoding = NULL, ...)
```

Arguments

file	path to CSV file
sep	column separator
version	determines which function to use for reading the CSV file 1: read.table, 2: fread
fileEncoding	passed to read.table or as encoding to fread
...	further arguments passed to read.table or fread

---

read_path_information *Read Files Containing File Path Information from Folder*

Description

Read Files Containing File Path Information from Folder

Usage

```
read_path_information(file_info_dir, pattern = "^path-info", ...)
```

Arguments

file_info_dir	path to folder in which to look for files matching pattern
pattern	pattern to match against the file names to be read. Default: "^path-info"
...	further arguments passed to read_file_paths

Examples

```
# Set root directory (here: package installation directory of kwb.fakin)
root_dir <- system.file(package = "kwb.fakin")

# Set output directory
output_dir <- tempdir()

# Write all paths below root_dir into a "path-info"-file
fakin.path.app::get_and_save_file_info(root_dir, output_dir)

# Read the "path-info"-files that are (now) found in output_dir
path_info <- kwb.fakin:::read_path_information(output_dir)
```

`report_about_github_package` *Report about the Functions in an R-Package on GitHub*

Description

Report about the Functions in an R-Package on GitHub

Usage

```
report_about_github_package(repo, ...)
```

Arguments

`repo`	path to the GitHub repository
`...`	arguments passed to `report_about_r_scripts`

`report_about_r_scripts` *Create a HTML Report About R Scripts*

Description

Create a HTML Report about each R script below a root directory. The report will contain an overview plot showing the number or rows for each script. In addition, one plot is generated per script, showing for each function defined in the script, the number of expressions contained in the function.

Usage

```
report_about_r_scripts(
  root,
  scripts = dir(root, "\\.R$", ignore.case = TRUE, recursive = TRUE),
  show = TRUE
)
```

Arguments

`root`	path to directory from which to start looking for R scripts
`scripts`	optional. Paths to R scripts, relative to the path given in `root`
`show`	if `TRUE` the created HTML file is opened in your default browser

Value

path to the created HTML file, invisibly

summary.path_tree *Get Statistics on Path Tree Nodes*

Description

Get Statistics on Path Tree Nodes

Usage

```
## S3 method for class 'path_tree'
summary(object, ...)
```

Arguments

object	tree object as returned by kwb.fakin:::to_tree
...	further arguments (currently not used)

to_tree *Convert Paths to Tree List*

Description

Convert Paths to Tree List

Usage

```
to_tree(x, dbg = FALSE)
```

Arguments

x	list as returned by strsplit
dbg	if TRUE, debug messages are shown

Index