Package: kwb.pathdict (via r-universe)

October 11, 2024

Title Functions to Work with Path Dictionaries

Version 0.1.1

Description This package provides functions to work with what I call path dictionaries. Path dictionaries are lists defining file and folder paths. In order not to repeat sub-paths, placeholders can be used. The package provides functions to find duplicated sub-paths and to define placeholders accordingly.

License MIT + file LICENSE

URL https://github.com/KWB-R/kwb.pathdict

BugReports https://github.com/KWB-R/kwb.pathdict/issues

Encoding UTF-8

LazyData true

Suggests covr, knitr, rmarkdown, testthat

RoxygenNote 7.0.2

Imports fs, kwb.file, kwb.utils, qdapDictionaries

Remotes github::kwb-r/kwb.file, github::kwb-r/kwb.utils

VignetteBuilder knitr

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

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

RemoteRef HEAD

RemoteSha b107485e18408a1bfb0346c785ce28d90fa6b087

Contents

`get_dictionary_one_by_one` *Get a Path Dictionary*

Description

Get a Path Dictionary

Usage

```
get_dictionary_one_by_one(paths, n = 10)
```

Arguments

`paths`	vector of character strings representing file or folder paths
`n`	number of compression levels

`named_vector_to_data_frame` *Convert a Named Vector to a Data Frame*

Description

Convert a Named Vector to a Data Frame

Usage

```
named_vector_to_data_frame(x)
```

Arguments

`x`	named vector

Value

data frame with columns `name`, containing the names of `x` and `value`, containing the values of `x`

Examples

```
kwb.pathdict:::named_vector_to_data_frame(c(a = 1, b = 2, c = 3))
```

order_decreasingly_by *Order Data Frame Decreasingly by one Column*

Description

Order Data Frame Decreasingly by one Column

Usage

```
order_decreasingly_by(df, column)
```

Arguments

df	data frame
column	name of column by which to order decreasingly.

Examples

```
(df <- data.frame(a = 1:3, b = 11:13))
kwb.pathdict:::order_decreasingly_by(df, "a")
```

random_paths *Create Random File Paths Using English Words*

Description

Create Random File Paths Using English Words

Usage

```
random_paths(
  max_depth = 5,
  min_chars = 5,
  max_elements = 10,
  depth_to_leaf_weight = function(depth) 1.2^depth
)
```

Arguments

max_depth	maximum path depth
min_chars	least number of characters per folder or file name
max_elements	maximum number of elements (files or subfolders) in a folder
depth_to_leaf_weight	function that calculates a weight from the given path depth. The weight is used to increase the probability of a folder element to be a file and not a subdirectory. By default the weight is calculated by 1.2^depth, i.e. for a folder in depth 10 it is about six times (1.2^10 = 6.19) more probable of its elements to be files instead of subfolders

Examples

```
# Make this example reproducible
set.seed(12059)

# Create random paths
paths <- kwb.pathdict::random_paths(max_depth = 5)

# Show the random paths
paths

# Frequency of path depths
table(lengths(kwb.file::split_paths(paths)))
```

`rescore_and_reorder_frequency_data` *Rescore and Reorder Frequency Data*

Description

Rescore and Reorder Frequency Data

Usage

```
rescore_and_reorder_frequency_data(frequency_data, placeholder_size)
```

Arguments

`frequency_data`	data frame with columns `length` and `count`
`placeholder_size`	size of placeholder in number of characters. The path length will be reduced by this value before being multiplied with the count to calculate the score.

`sorted_importance` *Importance of Strings*

Description

Decreasingly sorted frequencies of strings, by default weighted by their length. This function can be used to find the most "important" folder paths in terms of frequency and length.

Usage

```
sorted_importance(x, weighted = TRUE)
```

Arguments

x	vector of character strings
weighted	if TRUE (default) the frequencies of strings are multiplied by the corresponding string lengths

Value

named integer vector (of class table) containing the decreasingly sorted importance values of the elements in x. The importance of a string is either its frequency in x (if weighted is FALSE) or the product of this frequency and the string length (if weighted is TRUE)

Examples

```
strings <- c("a", "a", "a", "bc", "bc", "cdefg")

(importance <- kwb.pathdict:::sorted_importance(strings))

# Check that each input element is mentioned in the output
all(unique(strings) %in% names(importance))

# weighted = FALSE just returns the frequencies of strings in x
(importance <- kwb.pathdict:::sorted_importance(strings, weighted = FALSE))

# Check if the sum of frequencies is the number of elements in x
sum(importance) == length(strings)

# You may use the function to assess the "importance" of directory paths
kwb.pathdict:::sorted_importance(dirname(kwb.pathdict:::example_paths()))
```

---

starts_with_parts *Do Subfolder List Elements Start with Given Folder Names?*

Description

Do Subfolder List Elements Start with Given Folder Names?

Usage

```
starts_with_parts(parts, elements)
```

Arguments

parts	list of list of character as returned by split_paths or a matrix of character representing the subfolder names at the different folder depths as returned by to_subdir_matrix.
elements	vector of character giving the sequence of strings to be found in parts

Value

vector of logical as long as `parts` containing `TRUE` at positions `i` for which `all(parts[[i]][seq_along(elements)] == elements)` is `TRUE`

Examples

```
parts <- strsplit(c("a/b/c", "a/b/d", "b/c"), "/")
starts_with_parts(parts, elements = c("a", "b"))
starts_with_parts(parts, elements = c("b", "c"))

subdir_matrix <- kwb.file::to_subdir_matrix(parts)
starts_with_parts(subdir_matrix, elements = c("a", "b"))
starts_with_parts(subdir_matrix, elements = c("b", "c"))
```

---

to_cumulative_id *Give Each Field in a Subdirectory Matrix an ID*

Description

In the `subdir` matrix, each row represents a file path. The different parts of the paths (the folder names) appear in the different columns. For example, the paths "a/b/c" and "d/e" are represented by a matrix with values "a", "b", "c" in the first and "d", "e", "" in the second row. Each cell of the `subdir` matrix that is not empty gets a number. If two cells of one column have the same number, this means that the paths to the cells are the same. See example.

Usage

```
to_cumulative_id(subdirs)
```

Arguments

`subdirs`	matrix of subdirectory names, as returned by `to_subdir_matrix`

Examples

```
# Create a very simple subdirectory matrix
(subdirs <- matrix(byrow = TRUE, ncol = 4, c(
  "a", "b", "c", "d",
  "a", "b", "d", "",
  "a", "c", "d", "e"
)))

# Give each non-empty cell of the matrix an ID
kwb.pathdict:::to_cumulative_id(subdirs)

# You can read the matrix column by column. The highest number represents the
# number of different paths that reach up to the corresponding path level.
# 1st column: The starting parts of the paths in depth 1 are the same: "a".
```

```
#  All cells have ID = 1.
# 2nd column: There are two different paths to the folders in depth 2:
#   "a/b" (ID = 1) and "a/c" (ID = 2).
# 3rd column: There are three different paths to the folders in depth 3:
#   "a/b/c" (ID = 1), "a/b/d" (ID = 2), "a/c/d" (ID = 3).
# 4th column: There are only two out of three paths that reach depth 4:
#   "a/b/c/d" (ID = 1), "a/c/d/e" (ID = 2)
```

`to_dictionary` *Create Dictionary from Unique Strings*

Description

Create Dictionary from Unique Strings

Usage

```
to_dictionary(x, prefix = "a", leading_zeros = FALSE)
```

Arguments

`x`	vector of strings
`prefix`	prefix to be given to the keys in the dictionary. Default: "a"
`leading_zeros`	whether to make all keys in the dictionary have same length by adding leading zeros to the keys. Default: `FALSE`

Examples

```
# Define input strings
x <- c("elephant", "mouse", "cat", "cat", "cat", "mouse", "cat", "cat")

# Create a dictionary for the unique values in x
kwb.pathdict:::to_dictionary(x)

# Note that "cat" is the first entry because it has the highest "importance"
kwb.pathdict:::sorted_importance(x)
```

use_dictionary *Substitute Values that are in a Dictionary with their Keys*

Description

Substitute Values that are in a Dictionary with their Keys

Usage

```
use_dictionary(x, dict, method = "full")
```

Arguments

x	vector of character
dict	list of key = value pairs. Values of this list that are found in x are replaced by the placeholder "<key>" in x.
method	method to be applied, must be one of "full" or "part". If "full", the full values must match, otherwise the values in dict are interpreted as patterns that are matched against the values in x and the matching parts are replaced with the corresponding "<key>" placeholder

Value

x in which values or parts of the values are replaced with their short forms as they are defined in the dictionary dict

Examples

```
# Define a vector of long values
x <- c("What a nice day", "Have a nice day", "Good morning")

# Define short forms for full or partial values
dict_full <- list(wand = "What a nice day", gm = "Good morning")
dict_part <- list(w = "What", nd = "nice day", g = "Good")

# Replace long form values with their short forms
kwb.pathdict:::use_dictionary(x, dict_full, method = "full")
kwb.pathdict:::use_dictionary(x, dict_part, method = "part")
```

Index