# Package 'filesstrings'

May 4, 2019

**Type** Package

**Title** Handy File and String Manipulation

**Version** 3.0.0

**Maintainer** Rory Nolan <rorynoolan@gmail.com>

**Description** This started out as a package for file and string manipulation. Since then, the 'fs' and 'strex' packages emerged, offering functionality previously given by this package (but it's done better in these new ones). Those packages have hence almost pushed 'filesstrings' into extinction. However, it still has a small number of unique, handy file manipulation functions which can be seen in the vignette.
One example is a function to remove spaces from all file names in a directory.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** magrittr, tools, purrr, checkmate, strex (>= 0.1.2), matrixStats, Rcpp, ore (>= 1.4.0), tibble, rlang

**RoxygenNote** 6.1.0

**Suggests** testthat, covr, knitr, rmarkdown, spelling

**Depends** stringr

**URL** https://www.github.com/rorynolan/filesstrings

**BugReports** https://www.github.com/rorynolan/filesstrings/issues

**VignetteBuilder** knitr

**LinkingTo** Rcpp

**Language** en-US

**NeedsCompilation** yes

**Author** Rory Nolan [aut, cre, cph] (<https://orcid.org/0000-0002-5239-4043>), Sergi Padilla-Parra [ths] (<https://orcid.org/0000-0002-8010-9481>)

**Repository** CRAN

**Date/Publication** 2018-10-30 22:40:03 UTC

# R topics documented:



---

`all_equal` *A more flexible version of all.equal for vectors.*

---

## Description

This function will return `TRUE` whenever `base::all.equal()` would return `TRUE`, however it will also return `TRUE` in some other cases:

- If `a` is given and `b` is not, `TRUE` will be returned if all of the elements of `a` are the same.
- If `a` is a scalar and `b` is a vector or array, `TRUE` will be returned if every element in `b` is equal to `a`.
- If `a` is a vector or array and `b` is a scalar, `TRUE` will be returned if every element in `a` is equal to `b`.

When this function does not return TRUE, it returns FALSE (unless it errors). This is unlike base::all.equal().

## Usage

```
all_equal(a, b = NULL)
```

## Arguments

| | |
|---|---|
| a | A vector, array or list. |
| b | Either NULL or a vector, array or list of length either 1 or length(a). |

## Value

TRUE if "equality of all" is satisfied (as detailed in 'Description' above) and FALSE otherwise.

## Note

- This behaviour is totally different from base::all.equal().
- There's also dplyr::all_equal(), which is different again. To avoid confusion, always use the full filesstrings::all_equal() and never library(filesstrings) followed by just all_equal().

## Examples

```
all_equal(1, rep(1, 3))
all_equal(2, 1:3)
all_equal(1:4, 1:4)
all_equal(1:4, c(1, 2, 3, 3))
all_equal(rep(1, 10))
all_equal(c(1, 88))
all_equal(1:2)
all_equal(list(1:2))
all_equal(1:4, matrix(1:4, nrow = 2))  # note that this gives TRUE
```

---

# before_last_dot *Get the part of a string before the last period.*

---

## Description

See strex::str_before_last_dot().

## Usage

```
before_last_dot(string)
```

## Arguments

| | |
|---|---|
| string | A character vector. |

## can_be_numeric *Check if a string could be considered as numeric.*

### Description

See `strex::str_can_be_numeric()`.

### Usage

```
can_be_numeric(string)
```

### Arguments

| | |
|---|---|
| `string` | A character vector. |

## create_dir *Create directories if they don't already exist*

### Description

Given the names of (potential) directories, create the ones that do not already exist.

### Usage

```
create_dir(...)
```

### Arguments

| | |
|---|---|
| `...` | The names of the directories, specified via relative or absolute paths. Duplicates are ignored. |

### Value

Invisibly, a vector with a `TRUE` for each time a directory was actually created and a `FALSE` otherwise. This vector is named with the paths of the directories that were passed to the function.

### Examples

```
## Not run:
create_dir(c("mydir", "yourdir"))
remove_dir(c("mydir", "yourdir"))
## End(Not run)
```

## currency *Get the currencies of numbers within a string.*

### Description

See `strex::str_get_currency()`.

### Usage

```
get_currencies(string)

get_currency(string)
```

### Arguments

| | |
|---|---|
| `string` | A string for `get_currencies()` and a character vector for `get_currency()`. |

## extend_char_vec *Pad a character vector with empty strings.*

### Description

Extend a character vector by appending empty strings at the end.

### Usage

```
extend_char_vec(char_vec, extend_by = NA, length_out = NA)
```

### Arguments

| | |
|---|---|
| `char_vec` | A character vector. The thing you wish to expand. |
| `extend_by` | A non-negative integer. By how much do you wish to extend the vector? |
| `length_out` | A positive integer. How long do you want the output vector to be? |

### Value

A character vector.

### Examples

```
extend_char_vec(1:5, extend_by = 2)
extend_char_vec(c("a", "b"), length_out = 10)
```

## extract_non_numerics *Extract non-numbers from a string.*

### Description

See strex::str_extract_non_numerics()

### Usage

```
extract_non_numerics(string, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE)

nth_non_numeric(string, n, decimals = FALSE, leading_decimals = FALSE,
  negs = FALSE)

first_non_numeric(string, decimals = FALSE, leading_decimals = FALSE,
  negs = FALSE)

last_non_numeric(string, decimals = FALSE, leading_decimals = FALSE,
  negs = FALSE)
```

### Arguments

| | |
|---|---|
| string | A string. |
| decimals | Do you want to include the possibility of decimal numbers (TRUE) or not (FALSE, the default). |
| leading_decimals | Do you want to allow a leading decimal point to be the start of a number? |
| negs | Do you want to allow negative numbers? Note that double negatives are not handled here (see the examples). |
| n | The index of the number (or non-numeric) that you seek. Negative indexing is allowed i.e. n = 1 (the default) will give you the first number (or non-numeric) whereas n = -1 will give you the last number (or non-numeric), n = -2 will give you the second last number and so on. The function is vectorized over this argument. |

## extract_numbers *Extract numbers from a string.*

### Description

See strex::str_extract_numbers().

## Usage

```
extract_numbers(string, leave_as_string = FALSE, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE)

nth_number(string, n, leave_as_string = FALSE, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE)

first_number(string, leave_as_string = FALSE, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE)

last_number(string, leave_as_string = FALSE, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE)
```

## Arguments

| | |
|---|---|
| `string` | A string. |
| `leave_as_string` | Do you want to return the number as a string (`TRUE`) or as numeric (`FALSE`, the default)? |
| `decimals` | Do you want to include the possibility of decimal numbers (`TRUE`) or not (`FALSE`, the default). |
| `leading_decimals` | Do you want to allow a leading decimal point to be the start of a number? |
| `negs` | Do you want to allow negative numbers? Note that double negatives are not handled here (see the examples). |
| `n` | The index of the number (or non-numeric) that you seek. Negative indexing is allowed i.e. `n = 1` (the default) will give you the first number (or non-numeric) whereas `n = -1` will give you the last number (or non-numeric), `n = -2` will give you the second last number and so on. The function is vectorized over this argument. |

---

`filesstrings` `filesstrings`*: handy file and string manipulation*

---

## Description

This started out as a package for file and string manipulation. Since then, the `fs` file manipulation package and the 'strex string manipulation package emerged, offering functionality previously given by this package (but slightly better). Those packages have hence almost pushed 'filesstrings' into extinction. However, it still has a small number of unique, handy file manipulation functions which can be seen in the vignette. One example is a function to remove spaces from all file names in a directory.

## References

Rory Nolan and Sergi Padilla-Parra (2017). filesstrings: An R package for file and string manipulation. The Journal of Open Source Software, 2(14). doi: 10.21105/joss.00260.

## filesstrings-defunct *Defunct functions in* filesstrings

### Description

These functions are gone, no longer available.

### Usage

```
str_with_patterns(...)

count_matches(...)
```

### Arguments

| | |
|---|---|
| ... | Arguments to defunct functions. |

## give_ext *Ensure a file name has the intended extension.*

### Description

See strex::str_give_ext().

### Usage

```
give_ext(string, ext, replace = FALSE)
```

### Arguments

| | |
|---|---|
| string | The intended file name. |
| ext | The intended file extension (with or without the "."). |
| replace | If the file has an extension already, replace it (or append the new extension name)? |

## group_close *Group together close adjacent elements of a vector.*

### Description

Given a strictly increasing vector (each element is bigger than the last), group together stretches of the vector where *adjacent* elements are separated by at most some specified distance. Hence, each element in each group has at least one other element in that group that is *close* to it. See the examples.

### Usage

```
group_close(vec_ascending, max_gap = 1)
```

### Arguments

| | |
|---|---|
| vec_ascending | A strictly increasing numeric vector. |
| max_gap | The biggest allowable gap between adjacent elements for them to be considered part of the same *group*. |

### Value

A where each element is one group, as a numeric vector.

### Examples

```
group_close(1:10, 1)
group_close(1:10, 0.5)
group_close(c(1, 2, 4, 10, 11, 14, 20, 25, 27), 3)
```

## locate_braces *Locate the braces in a string.*

### Description

See strex::str_locate_braces().

### Usage

```
locate_braces(string)
```

### Arguments

| | |
|---|---|
| string | A character vector |

## match_arg *Argument Matching*

### Description

See strex::match_arg().

### Usage

```
match_arg(arg, choices, index = FALSE, several_ok = FALSE,
  ignore_case = FALSE)
```

### Arguments

| | |
|---|---|
| arg | A character vector (of length one unless several_ok = TRUE). |
| choices | A character vector of candidate values. |
| index | Return the index of the match rather than the match itself? Default no. |
| several_ok | Allow arg to have length greater than one to match several arguments at once? Default no. |
| ignore_case | Ignore case while matching. Default no. If this is TRUE, the returned value is the matched element of choices (with its original casing). |

## move_files *Move files around.*

### Description

Move specified files into specified directories

### Usage

```
move_files(files, destinations, overwrite = FALSE)

file.move(files, destinations, overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| files | A character vector of files to move (relative or absolute paths). |
| destinations | A character vector of the destination directories into which to move the files. |
| overwrite | Allow overwriting of files? Default no. |

### Details

If there are $n$ files, there must be either 1 or $n$ directories. If there is one directory, then all $n$ files are moved there. If there are $n$ directories, then each file is put into its respective directory. This function also works to move directories.

If you try to move files to a directory that doesn't exist, the directory is first created and then the files are put inside.

## Value

Invisibly, a logical vector with a TRUE for each time the operation succeeded and a FALSE for every fail.

## Examples

```
## Not run:
dir.create("dir")
files <- c("1litres_1.txt", "1litres_30.txt", "3litres_5.txt")
file.create(files)
file.move(files, "dir")
## End(Not run)
```

---

# nice_file_nums *Make file numbers comply with alphabetical order*

---

## Description

If files are numbered, their numbers may not *comply* with alphabetical order, i.e. "file2.ext" comes after "file10.ext" in alphabetical order. This function renames the files in the specified directory such that they comply with alphabetical order, so here "file2.ext" would be renamed to "file02.ext".

## Usage

```
nice_file_nums(dir = ".", pattern = NA)
```

## Arguments

| | |
|---|---|
| dir | Path (relative or absolute) to the directory in which to do the renaming (default is current working directory). |
| pattern | A regular expression. If specified, files to be renamed are restricted to ones matching this pattern (in their name). |

## Details

It works on file names with more than one number in them e.g. "file01part3.ext" (a file with 2 numbers). All the file names that it works on must have the same number of numbers, and the non-number bits must be the same. One can limit the renaming to files matching a certain pattern. This function wraps nice_nums(), which does the string operations, but not the renaming. To see examples of how this function works, see the examples in that function's documentation.

## Value

A logical vector with a TRUE for each successful rename (should be all TRUEs) and a FALSE otherwise.

## Examples

```
## Not run:
dir.create("NiceFileNums_test")
setwd("NiceFileNums_test")
files <- c("1litres_1.txt", "1litres_30.txt", "3litres_5.txt")
file.create(files)
nice_file_nums()
nice_file_nums(pattern = "\\.txt$")
setwd("..")
dir.remove("NiceFileNums_test")
## End(Not run)
```

---

`nice_nums` *Make string numbers comply with alphabetical order*

---

## Description

See `strex::str_alphord_nums()`.

## Usage

```
nice_nums(strings)
```

## Arguments

| | |
|---|---|
| `strings` | A vector of strings. |

---

`nth_number_after_mth` *Find the $n$th number after the $m$th occurrence of a pattern.*

---

## Description

See `strex::str_nth_number_after_mth()`.

## Usage

```
nth_number_after_mth(string, pattern, n, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

nth_number_after_first(string, pattern, n, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

nth_number_after_last(string, pattern, n, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

first_number_after_mth(string, pattern, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_after_mth(string, pattern, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)
```

```
first_number_after_first(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

first_number_after_last(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_after_first(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_after_last(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)
```

## Arguments

| | |
|---|---|
| string | A character vector. |
| pattern | A character vector. Pattern(s) specified like the pattern(s) in the stringr package (e.g. look at stringr::str_locate()). If this has length >1 its length must be the same as that of string. |
| n | Natural numbers. |
| m | Natural numbers. |
| decimals | Do you want to include the possibility of decimal numbers (TRUE) or not (FALSE, the default). |
| leading_decimals | Do you want to allow a leading decimal point to be the start of a number? |
| negs | Do you want to allow negative numbers? Note that double negatives are not handled here (see the examples). |
| leave_as_string | Do you want to return the number as a string (TRUE) or as numeric (FALSE, the default)? |

---

# nth_number_before_mth *Find the $n$th number before the $m$th occurrence of a pattern.*

---

## Description

Given a string, a pattern and natural numbers n and m, find the nth number that comes before the mth occurrence of the pattern.

## Usage

```
nth_number_before_mth(string, pattern, n, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

nth_number_before_first(string, pattern, n, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

nth_number_before_last(string, pattern, n, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)
```

```
first_number_before_mth(string, pattern, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_before_mth(string, pattern, m, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

first_number_before_first(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

first_number_before_last(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_before_first(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)

last_number_before_last(string, pattern, decimals = FALSE,
  leading_decimals = FALSE, negs = FALSE, leave_as_string = FALSE)
```

## Arguments

| | |
|---|---|
| `string` | A character vector. |
| `pattern` | A character vector. Pattern(s) specified like the pattern(s) in the stringr package (e.g. look at `stringr::str_locate()`). If this has length >1 its length must be the same as that of `string`. |
| `n, m` | Natural numbers. |
| `decimals` | Do you want to include the possibility of decimal numbers (`TRUE`) or not (`FALSE`, the default). |
| `leading_decimals` | Do you want to allow a leading decimal point to be the start of a number? |
| `negs` | Do you want to allow negative numbers? Note that double negatives are not handled here (see the examples). |
| `leave_as_string` | Do you want to return the number as a string (`TRUE`) or as numeric (`FALSE`, the default)? |

## Value

A numeric vector.

## Examples

```
string <- c("abc1abc2abc3abc4def5abc6abc7abc8abc9",
            "abc1def2ghi3abc4def5ghi6abc7def8ghi9")
nth_number_before_mth(string, "def", 1, 1)
nth_number_before_mth(string, "abc", 2, 3)
nth_number_before_first(string, "def", 2)
nth_number_before_last(string, "def", -1)
first_number_before_mth(string, "abc", 2)
last_number_before_mth(string, "def", 1)
first_number_before_first(string, "def")
first_number_before_last(string, "def")
```

```
last_number_before_first(string, "def")
last_number_before_last(string, "def")
```

---

`put_in_pos` *Put specified strings in specified positions in an otherwise empty character vector.*

---

## Description

Create a character vector with a set of strings at specified positions in that character vector, with the rest of it taken up by empty strings.

## Usage

```
put_in_pos(strings, positions)
```

## Arguments

| | |
|---|---|
| `strings` | A character vector of the strings to put in positions (coerced by as.character if not character already). |
| `positions` | The indices of the character vector to be occupied by the elements of strings. Must be the same length as strings or of length 1. |

## Value

A character vector.

## Examples

```
put_in_pos(1:3, c(1, 8, 9))
put_in_pos(c("Apple", "Orange", "County"), c(5, 7, 8))
put_in_pos(1:2, 5)
```

---

`remove_dir` *Remove directories*

---

## Description

Delete directories and all of their contents.

## Usage

```
remove_dir(...)

dir.remove(...)
```

## Arguments

| | |
|---|---|
| `...` | The names of the directories, specified via relative or absolute paths. |

## Value

Invisibly, a logical vector with TRUE for each success and FALSE for failures.

## Examples

```
## Not run:
sapply(c("mydir1", "mydir2"), dir.create)
remove_dir(c("mydir1", "mydir2"))
## End(Not run)
```

---

`remove_filename_spaces` *Remove spaces in file names*

---

## Description

Remove spaces in file names in a specified directory, replacing them with whatever you want, default nothing.

## Usage

```
remove_filename_spaces(dir = ".", pattern = "", replacement = "")
```

## Arguments

| | |
|---|---|
| `dir` | The directory in which to perform the operation. |
| `pattern` | A regular expression. If specified, only files matching this pattern will be treated. |
| `replacement` | What do you want to replace the spaces with? This defaults to nothing, another sensible choice would be an underscore. |

## Value

A logical vector indicating which operation succeeded for each of the files attempted. Using a missing value for a file or path name will always be regarded as a failure.

## Examples

```
## Not run:
dir.create("RemoveFileNameSpaces_test")
setwd("RemoveFileNameSpaces_test")
files <- c("1litres 1.txt", "1litres 30.txt", "3litres 5.txt")
file.create(files)
remove_filename_spaces()
list.files()
setwd("..")
dir.remove("RemoveFileNameSpaces_test")
## End(Not run)
```

## remove_quoted *Remove the quoted parts of a string.*

### Description

See `strex::str_remove_quoted()`.

### Usage

```
remove_quoted(string)
```

### Arguments

| | |
|---|---|
| `string` | A character vector. |

## rename_with_nums *Replace file names with numbers*

### Description

Rename the files in the directory, replacing file names with numbers only.

### Usage

```
rename_with_nums(dir = ".", pattern = NULL)
```

### Arguments

| | |
|---|---|
| `dir` | The directory in which to rename the files (relative or absolute path). Defaults to current working directory. |
| `pattern` | A regular expression. If specified, only files with names matching this pattern will be treated. |

### Value

A logical vector with a `TRUE` for each successful renaming and a `FALSE` otherwise.

### Examples

```
## Not run:
dir.create("RenameWithNums_test")
setwd("RenameWithNums_test")
files <- c("1litres 1.txt", "1litres 30.txt", "3litres 5.txt")
file.create(files)
rename_with_nums()
list.files()
setwd("..")
dir.remove("RenameWithNums_test")
## End(Not run)
```

## singleize *Remove back-to-back duplicates of a pattern in a string.*

### Description

See strex::str_singleize().

### Usage

```
singleize(string, pattern)
```

### Arguments

| | |
|---|---|
| string | A character vector. The string(s) to be purged of duplicates. |
| pattern | A character vector. Pattern(s) specified like the pattern(s) in the stringr package (e.g. look at stringr::str_locate()). If this has length >1 its length must be the same as that of string. |

## str_after_nth *Text before or after $n$th occurrence of pattern.*

### Description

See strex::str_after_nth().

### Usage

```
str_after_nth(strings, pattern, n)

str_after_first(strings, pattern)

str_after_last(strings, pattern)

str_before_nth(strings, pattern, n)

str_before_first(strings, pattern)

str_before_last(strings, pattern)
```

### Arguments

| | |
|---|---|
| strings | A character vector. |
| pattern | A character vector. Pattern(s) specified like the pattern(s) in the stringr package (e.g. look at stringr::str_locate()). If this has length >1 its length must be the same as that of string. |
| n | A natural number to identify the $n$th occurrence (defaults to first (n = 1)). This can be negatively indexed, so if you wish to select the *last* occurrence, you need n = -1, for the second-last, you need n = -2 and so on. |

## str_elem *Extract a single character from a string, using its index.*

### Description

See strex::str_elem().

### Usage

```
str_elem(string, index)
```

### Arguments

| | |
|---|---|
| string | A string. |
| index | An integer. Negative indexing is allowed as in stringr::str_sub(). |

## str_nth_instance_indices *Get the indices of the $n$th instance of a pattern.*

### Description

See strex::str_locate_nth().

### Usage

```
str_nth_instance_indices(string, pattern, n)

str_first_instance_indices(string, pattern)

str_last_instance_indices(string, pattern)
```

### Arguments

| | |
|---|---|
| string | A character vector. These functions are vectorized over this argument. |
| pattern | A character vector. Pattern(s) specified like the pattern(s) in the stringr package (e.g. look at stringr::str_locate()). If this has length >1 its length must be the same as that of string. |
| n | Then $n$ for the $n$th instance of the pattern. |

## str_paste_elems *Extract bits of a string and paste them together*

### Description

See strex::str_paste_elems().

### Usage

```
str_paste_elems(string, indices)
```

### Arguments

| | |
|---|---|
| string | A string. |
| indices | A numeric vector of positive integers detailing the indices of the characters of string that we wish to paste together. |

## str_split_by_nums *Split a string by its numeric characters.*

### Description

See strex::str_split_by_nums().

### Usage

```
str_split_by_nums(string, decimals = FALSE, leading_decimals = FALSE,
  negs = FALSE)
```

### Arguments

| | |
|---|---|
| string | A string. |
| decimals | Do you want to include the possibility of decimal numbers (TRUE) or not (FALSE, the default). |
| leading_decimals | Do you want to allow a leading decimal point to be the start of a number? |
| negs | Do you want to allow negative numbers? Note that double negatives are not handled here (see the examples). |

## str_split_camel_case *Split a string based on CamelCase*

### Description

See strex::str_split_camel_case().

### Usage

```
str_split_camel_case(string, lower = FALSE)
```

### Arguments

| | |
|---|---|
| string | A character vector. |
| lower | Do you want the output to be all lower case (or as is)? |

## str_to_vec *Convert a string to a vector of characters*

### Description

See strex::str_to_vec().

### Usage

```
str_to_vec(string)
```

### Arguments

| | |
|---|---|
| string | A string. |

## trim_anything *Trim something other than whitespace*

### Description

See strex::str_trim_anything().

### Usage

```
trim_anything(string, pattern, side = "both")
```

### Arguments

| | |
|---|---|
| string | A string. |
| pattern | A string. The pattern to be trimmed (*not* interpreted as regular expression). So to trim a period, use `char = "."` and not `char = "\\."`). |
| side | Which side do you want to trim from? `"both"` is the default, but you can also have just either `"left"` or `"right"` (or optionally the shortened `"b"`, `"l"` and `"r"`). |

`unitize_dirs` *Put files with the same unit measurements into directories*

## Description

Say you have a number of files with "5min" in their names, number with "10min" in the names, a number with "15min" in their names and so on, and you'd like to put them into directories named "5min", "10min", "15min" and so on. This function does this, but not just for the unit "min", for any unit.

## Usage

```
unitize_dirs(unit, pattern = NULL, dir = ".")
```

## Arguments

| | |
|---|---|
| `unit` | The unit upon which to base the categorizing. |
| `pattern` | If set, only files with names matching this pattern will be treated. |
| `dir` | In which directory do you want to perform this action (defaults to current)? |

## Details

This function takes the number to be the last number (as defined in `nth_number()`) before the first occurrence of the unit name. There is the option to only treat files matching a certain pattern.

## Value

Invisibly `TRUE` if the operation is successful, if not there will be an error.

## Examples

```
## Not run:
dir.create("UnitDirs_test")
setwd("UnitDirs_test")
files <- c("1litres_1.txt", "1litres_3.txt", "3litres.txt", "5litres_1.txt")
file.create(files)
unitize_dirs("litres", "\\.txt")
setwd("..")
dir.remove("UnitDirs_test")
## End(Not run)
```

# Index