Package: libxdiffR (via r-universe)

August 29, 2024

Type Package

Title provides xdiff access

Version 0.1.1

Maintainer Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Description Bundles xdiff to create unidiffs from R.

URL https://janmarvin.github.io/libxdiffR/,
https://github.com/JanMarvin/libxdiffR

License GPL-2

Encoding UTF-8

Language en-US

LazyData true

Imports digest

Suggests curl, fs, knitr, stringi, testthat (>= 3.0.0)

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

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

RemoteUrl https://github.com/JanMarvin/libxdiffR

RemoteRef HEAD

RemoteSha 50626adbf45a408b701a7a2ed0f368d82ee513d5

Contents

`merge3` *Compare Files Using Three-Way Merges*

Description

`merge3()` is a function designed to perform three-way merges between files or text inputs. Three-way merging is commonly used in version control systems to reconcile changes made by multiple contributors, allowing for the integration of modifications from different branches or versions with a shared common ancestor.

Usage

```
merge3(base, merge1, merge2, level = "minimal", favor = NULL, style = NULL)
```

Arguments

`base`	The common ancestor file or text input. This is the original version from which the two modified versions (`merge1` and `merge2`) were derived.
`merge1, merge2`	The first and second modified files or text input to be merged.
`level`	A string specifying the merging aggressiveness level. Options include: • `"minimal"`: Only essential changes are merged, minimizing potential conflicts. • `"eager"`: Merges more eagerly, potentially leading to more conflicts but also more comprehensive merges. • `"zealous"`: Aggressively merges changes, resolving conflicts where possible. • `"zealous_alnum"`: Similar to "zealous", but with additional handling for alphanumeric characters.
`favor`	favor: A string specifying which changes to favor in case of conflicts. Options are: • `"ours"`: Favor the changes from merge1. • `"theirs"`: Favor the changes from merge2. • `"union"`: Combine the changes from both merge1 and merge2 when possible.
`style`	A string specifying the merge style. Options include: • `"diff3"`: Standard diff3-style merge output. • `"zdiff3"`: A zealous variant of the diff3-style merge, offering more comprehensive conflict resolution.

Details

The `merge3()` function takes three inputs—an ancestor version and two modified versions—and attempts to reconcile the differences between them, producing a single merged output. This process is crucial in collaborative development environments where multiple team members work on the same files. The function supports different levels of merging aggressiveness, allowing users to tailor the merge process according to their needs. Additionally, users can specify which version's changes should be favored in case of conflicts, and choose between standard and zealous diff3-style outputs. The function also supports inputs from URLs, making it versatile for remote file merging scenarios.

Value

A character string containing the three way merge.

Examples

```
base <- "function(x) print(x) \n"
merge1 <- "function(x) print(x) \n message(x)\n"
merge2 <- "function(x) print(x) \n cat(x)\n"

cat(merge3(base, merge1, merge2))
```

`unidiff` *Compare Files Using Unified Diffs*

Description

`unidiff()` is a function designed to generate unified diffs between two files or text inputs. Unified diffs are widely used in version control systems to highlight differences between two versions of a file in a compact and readable format.

Usage

```
unidiff(
  old,
  new,
  create_head = TRUE,
  with_context = FALSE,
  context_length = 3,
  ignore_whitespace = NULL,
  algorithm = "minimal",
  indent_heuristic = FALSE,
  ignore = NULL
)
```

Arguments

`old`	The original file or text input. This can be a file path, a URL, or a string of text.
`new`	The modified file or text input to compare against the original. This can also be a file path, a URL, or a string of text.
`create_head`	A logical flag (default `TRUE`) indicating whether to include a header in the diff output. The header contains the filenames or identifiers of the files being compared.
`with_context`	A logical flag (default `FALSE`) specifying whether to include the full context in the diff output. Including context helps to provide a more complete view of the changes.
`context_length`	An integer value (default `3`) that determines the number of lines of context to include around each change. This is only relevant if with_context is `TRUE`.
`ignore_whitespace`	A character vector indicating how whitespace differences should be treated. Options include `"all"`, `"change"`, `"at_eol"`, `"cr_at_eol"`, and `"blank_lines"`, allowing for fine-grained control over which whitespace differences to ignore.
`algorithm`	A string specifying the diff algorithm to use. Options are `"minimal"`, `"patience"`, and `"histogram"`, with `"minimal"` being the default. Each algorithm has different characteristics suited to various types of text comparisons.
`indent_heuristic`	A logical flag (default `FALSE`) that, when enabled, applies an additional heuristic to better handle indented lines, improving the accuracy of the diff in some cases.
`ignore`	A single string specifying the regex pattern to use for ignoring lines in the diff process. If this is a non-empty string, it should be a valid regex pattern. Lines in the texts that match this pattern will be ignored in the diff.

Details

The `unidiff()` function compares two inputs—whether they are files, URLs, or raw text—and generates a unified diff that highlights the differences between them. It supports advanced features like customizable context, whitespace handling, and different diff algorithms to suit various needs. The function can also handle inputs from URLs, making it flexible for remote file comparisons. The output is a unified diff string, which can be directly used in version control systems or for manual inspection of changes. For identical files, no diff is returned, and an empty character string is provided. If regex support is not available on the current platform and a non-empty pattern is provided, an error will be raised.

Value

A character string containing the unified diff.

Examples

```
cat(unidiff(old = "foo bar", new = "foo baz"))
```

unidiff_dir *Compare Files Between Two Directories Using Unified Diffs*

Description

The `unidiff_dir()` function compares files between two directories and generates unified diffs for each differing file. This function is useful for identifying and displaying differences between two directory structures, such as when comparing different versions of a codebase.

Usage

```
unidiff_dir(
  old,
  new,
  pattern = NULL,
  create_head = TRUE,
  with_context = FALSE,
  context_length = 3,
  ignore_whitespace = NULL,
  algorithm = "minimal",
  indent_heuristic = FALSE,
  ignore = NULL
)
```

Arguments

`old`	The directory path of the first directory to compare.
`new`	The directory path of the second directory to compare.
`pattern`	An optional regular expression. Only file names matching the pattern will be included in the comparison. Default is `NULL`, meaning all files are compared.
`create_head`	A logical flag (default `TRUE`) indicating whether to include a header in the diff output. The header contains the filenames or identifiers of the files being compared.
`with_context`	A logical flag (default `FALSE`) specifying whether to include the full context in the diff output. Including context helps to provide a more complete view of the changes.
`context_length`	An integer value (default 3) that determines the number of lines of context to include around each change. This is only relevant if with_context is `TRUE`.
`ignore_whitespace`	A character vector indicating how whitespace differences should be treated. Options include `"all"`, `"change"`, `"at_eol"`, `"cr_at_eol"`, and `"blank_lines"`, allowing for fine-grained control over which whitespace differences to ignore.
`algorithm`	A string specifying the diff algorithm to use. Options are `"minimal"`, `"patience"`, and `"histogram"`, with `"minimal"` being the default. Each algorithm has different characteristics suited to various types of text comparisons.

`indent_heuristic`	A logical flag (default `FALSE`) that, when enabled, applies an additional heuristic to better handle indented lines, improving the accuracy of the diff in some cases.
`ignore`	A single string specifying the regex pattern to use for ignoring lines in the diff process. If this is a non-empty string, it should be a valid regex pattern. Lines in the texts that match this pattern will be ignored in the diff.

Value

A character string containing the unified diffs for all differing files between the two directories.

Index