

Package ‘fuzzysting’

July 24, 2026

Title Fast Fuzzy String Joins for Data Frames

Version 0.0.6

Description Perform fuzzy joins on data frames using approximate string matching. Implements inner, left, right, full, semi, and anti joins with string distance metrics from the 'stringdist' package, including Optimal String Alignment, Levenshtein, Damerau-Levenshtein, Jaro-Winkler, q-gram, cosine, Jaccard, and Soundex. Uses a 'data.table' backend plus compiled 'C++' result assembly to reduce overhead in large joins, while adaptive candidate planning avoids unnecessary distance evaluations in single-column string joins. Suitable for reconciling misspellings, inconsistent labels, and other near-match identifiers while optionally returning the computed distance for each match. Bibliographic references include the 'stringdist' package documentation (2014) <<https://CRAN.R-project.org/package=stringdist>> and Robinson, D. (2015) <<https://github.com/dgrtwo/fuzzyjoin>>.

License MIT + file LICENSE

Depends R (>= 4.1)

Imports data.table, Rcpp, stringdist

LinkingTo Rcpp

Suggests dplyr, ggplot2, knitr, qdapDictionaries, readr, rmarkdown, rvest, stringr, testthat (>= 3.0.0), tibble, tidyr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

URL <https://github.com/PaulESantos/fuzzysting>,
<https://paulasantos.github.io/fuzzysting/>

BugReports <https://github.com/PaulESantos/fuzzysting/issues>

VignetteBuilder knitr

Maintainer Paul E. Santos Andrade <paulefrens@gmail.com>

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Paul E. Santos Andrade [aut, cre, cph] (ORCID:
 <<https://orcid.org/0000-0002-6635-0375>>),
 David Robinson [ctb] (aut of fuzzyjoin)

Repository CRAN

Date/Publication 2026-07-24 06:40:02 UTC

Contents

fuzzystring	2
fuzzystring_join	3
misspellings	5
Index	6

fuzzystring	<i>fuzzystring: Fast fuzzy string joins for data frames</i>
-------------	---

Description

fuzzystring provides fuzzy inner, left, right, full, semi, and anti joins for `data.frame` and `data.table` objects using approximate string matching. It combines `stringdist` metrics with a `data.table` backend and compiled 'C++' result assembly to reduce overhead in large joins while preserving familiar join semantics.

Details

Main entry points are `fuzzystring_join()` and the convenience wrappers `fuzzystring_inner_join()`, `fuzzystring_left_join()`, `fuzzystring_right_join()`, `fuzzystring_full_join()`, `fuzzystring_semi_join()`, and `fuzzystring_anti_join()`.
The package also includes the example dataset `misspellings`.

Author(s)

- Maintainer:** Paul E. Santos Andrade <paulefrens@gmail.com> (ORCID) [copyright holder]
Authors:
- Paul E. Santos Andrade <paulefrens@gmail.com> (ORCID) [copyright holder]
- Other contributors:
- David Robinson <admiral.david@gmail.com> (aut of fuzzyjoin) [contributor]

See Also

- Useful links:
- <https://github.com/PaulESantos/fuzzystring>
 - <https://paulesantos.github.io/fuzzystring/>
 - Report bugs at <https://github.com/PaulESantos/fuzzystring/issues>

fuzzystring_join	<i>Join two tables based on fuzzy string matching</i>
------------------	---

Description

Uses `stringdist::stringdist()` to compute distances and a `data.table`-orchestrated backend with compiled 'C++' assembly to produce the final result. This is the main user-facing entry point for fuzzy joins on strings.

Usage

```
fuzzystring_join(
  x,
  y,
  by = NULL,
  max_dist = 2,
  method = c("osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw",
    "soundex"),
  mode = "inner",
  ignore_case = FALSE,
  distance_col = NULL,
  ...
)

fuzzystring_inner_join(x, y, by = NULL, distance_col = NULL, ...)

fuzzystring_left_join(x, y, by = NULL, distance_col = NULL, ...)

fuzzystring_right_join(x, y, by = NULL, distance_col = NULL, ...)

fuzzystring_full_join(x, y, by = NULL, distance_col = NULL, ...)

fuzzystring_semi_join(x, y, by = NULL, distance_col = NULL, ...)

fuzzystring_anti_join(x, y, by = NULL, distance_col = NULL, ...)
```

Arguments

<code>x</code>	A <code>data.frame</code> or <code>data.table</code> .
<code>y</code>	A <code>data.frame</code> or <code>data.table</code> .
<code>by</code>	Columns by which to join the two tables. You can supply a character vector of common names (e.g. <code>c("name")</code>), or a named vector mapping <code>x</code> to <code>y</code> (e.g. <code>c(name = "approx_name")</code>).
<code>max_dist</code>	Maximum distance to use for joining. Smaller values are stricter.
<code>method</code>	Method for computing string distance, see <code>?stringdist::stringdist</code> and the <code>stringdist</code> package vignettes.

mode	One of "inner", "left", "right", "full", "semi", or "anti".
ignore_case	Logical; if TRUE, comparisons are case-insensitive.
distance_col	If not NULL, adds a column with this name containing the computed distance for each matched pair (or NA for unmatched rows in outer joins).
...	Additional arguments passed to stringdist .

Details

If method = "soundex", max_dist is automatically set to 0.5, since Soundex distance is 0 (match) or 1 (no match).

When by maps multiple columns, the same method, max_dist, and any additional stringdist arguments are applied independently to each mapped column, and a row pair is kept only when all mapped columns satisfy the distance threshold. When distance_col is requested for a multi-column join, it contains the maximum distance across the mapped columns for each matched row pair.

For single-column joins, fuzzystring uses adaptive candidate planning before calling `stringdist::stringdist()`. For Levenshtein-like methods ("osa", "lv", "dl"), a fast prefilter is applied: if `abs(nchar(v1) - nchar(v2)) > max_dist`, the pair cannot match, so distance is not computed for that pair. For low-duplication workloads, the planner can also evaluate larger dense blocks of unique values to reduce orchestration overhead while preserving the same matching semantics. When `max_dist = 0` and no additional metric arguments are supplied, methods whose zero distance implies string equality use an exact keyed join instead of evaluating the full candidate distance set.

Value

A joined table that preserves the container class of `x`: `data.table` inputs return `data.table`, `tibble` inputs return `tibble`, and plain `data.frame` inputs return plain `data.frame`. See [fuzzystring_join_backend](#) for details on output structure.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  d <- data.table::data.table(approximate_name = c("Idea", "Premiom"))
  # Match diamonds$cut to d$approximate_name
  res <- fuzzystring_inner_join(ggplot2::diamonds, d,
    by = c(cut = "approximate_name"),
    max_dist = 1
  )
  head(res)
}
```

misspellings*A corpus of common misspellings, for examples and practice*

Description

This is a `tbl_df` mapping misspellings of their words, compiled by Wikipedia, where it is licensed under the CC-BY SA license. (Three words with non-ASCII characters were filtered out). If you'd like to reproduce this dataset from Wikipedia, see the example code below.

Usage

```
misspellings
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 4505 rows and 2 columns.

Source

https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines

Examples

```
if (interactive()) {
  library(rvest)
  library(readr)
  library(dplyr)
  library(stringr)
  library(tidyr)

  u <- "https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines"
  h <- read_html(u)

  misspellings <- h %>%
    html_nodes("pre") %>%
    html_text() %>%
    read_delim(col_names = c("misspelling", "correct"),
               delim = ">",
               skip = 1) %>%
    mutate(misspelling = str_sub(misspelling,
                                1, -2)) |>
    separate_rows(correct, sep = ", ") |>
    filter(Encoding(correct) != "UTF-8")
}
```

Index

- * **datasets**
 - misspellings, [5](#)
- * **package**
 - fuzzyststring, [2](#)
- fuzzyststring, [2](#)
- fuzzyststring-package (fuzzyststring), [2](#)
- fuzzyststring_anti_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_anti_join(), [2](#)
- fuzzyststring_full_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_full_join(), [2](#)
- fuzzyststring_inner_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_inner_join(), [2](#)
- fuzzyststring_join, [3](#)
- fuzzyststring_join(), [2](#)
- fuzzyststring_join_backend, [4](#)
- fuzzyststring_left_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_left_join(), [2](#)
- fuzzyststring_right_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_right_join(), [2](#)
- fuzzyststring_semi_join
 - (fuzzyststring_join), [3](#)
- fuzzyststring_semi_join(), [2](#)
- misspellings, [2](#), [5](#)
- stringdist, [4](#)