Package 'fozziejoin'

March 9, 2026

Title Utilities for Joining Dataframes with Inexact Matching

Version 0.0.13

Description Provides functions for joining data frames based on inexact
criteria, including string distance, Manhattan distance, Euclidean
distance, and interval overlap. This API is designed as a modern,
performance-oriented alternative to the 'fuzzyjoin' package (Robinson 2026)
<doi:10.32614/CRAN.package.fuzzyjoin>. String distance functions utilizing
'q-grams' are adapted with permission from the 'textdistance' 'Rust' crate
(Orsinium 2024) <https://docs.rs/textdistance/latest/textdistance/>. Other
string distance calculations rely on the 'rapidfuzz' 'Rust' crate (Bachmann
2023) <https://docs.rs/rapidfuzz/0.5.0/rapidfuzz/>. Interval joins are
backed by a Adelson-Velsky and Landis tree as implemented by the 'interavl'
'Rust' crate <https://docs.rs/interavl/0.5.0/interavl/>.

License MIT + file LICENSE

Depends R (>= 4.2)

Imports stats, tibble, utils

Suggests babynames, dplyr, fuzzyjoin, knitr, microbenchmark,
qdapDictionaries, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/CodeOfConduct
https://github.com/fozzieverse/fozzieverse/blob/main/CODE_OF_CONDUCT.md

Config/rextendr/version 0.4.2

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

URL https://github.com/fozzieverse/fozziejoin

BugReports https://github.com/fozzieverse/fozziejoin/issues

SystemRequirements Cargo (Rust's package manager), rustc, xz

NeedsCompilation yes

Author Jon Downs [aut, cre],
    The authors of the dependency Rust crates [ctb, cph] (see inst/AUTHORS
      file for details)

Maintainer Jon Downs <jon@jondowns.net>

Repository CRAN

Date/Publication 2026-03-09 16:20:02 UTC

Contents

---

fozzie_difference_join_family

Perform a fuzzy join between two data frames using numeric difference matching.

Description

fozzie_difference_join() and its directional variants (fozzie_difference_inner_join(),
fozzie_difference_left_join(), fozzie_difference_right_join(), fozzie_difference_anti_join(),
fozzie_difference_full_join()) enable approximate matching of numeric fields in two data
frames based on absolute difference thresholds. These joins are analogous to fuzzyjoin::difference_join,
but implemented in Rust for performance.

Usage

```
fozzie_difference_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)
```

```
fozzie_difference_inner_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_difference_left_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_difference_right_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_difference_anti_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_difference_full_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_difference_semi_join(
  df1,
```

```
    df2,
    by = NULL,
    max_distance = 1,
    distance_col = NULL,
    nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A named list or character vector indicating the matching columns. Can be a character vector of length 2, e.g. c("col1", "col2"), or a named list like list(col1 = "col2").
how	A string specifying the join mode. One of: • "inner": matched pairs only. • "left": all rows from df1, unmatched rows filled with NAs. • "right": all rows from df2, unmatched rows filled with NAs. • "full": all rows from both df1 and df2. • "anti": rows from df1 not matched in df2. • "semi": rows from df1 that matched with one or more matches in df2.
max_distance	A numeric threshold for allowable absolute difference between values (lower is stricter).
distance_col	Optional name of column to store computed differences.
nthread	Optional integer specifying the number of threads to use for parallelization. If not provided, the value is determined by options("fozzie.nthread"). The package default is inherited from Rayon, the multithreading library used throughout the package.

Value

A data frame with approximately matched rows depending on the join type. See individual functions like fozzie_difference_inner_join() for examples. If distance_col is specified, an additional numeric column is included.

Examples

```
df1 <- data.frame(x = c(1.0, 2.0, 3.0))
df2 <- data.frame(x = c(1.05, 2.1, 2.95))

fozzie_difference_inner_join(df1, df2, by = c("x"), max_distance = 0.1)
fozzie_difference_left_join(df1, df2, by = c("x"), max_distance = 0.2)
fozzie_difference_right_join(df1, df2, by = c("x"), max_distance = 0.05)
```

```
fozzie_distance_join_family
```
                            *Perform a fuzzy join between two data frames using vector distance
                            matching.*

Description

fozzie_distance_join() and its directional variants (fozzie_distance_inner_join(), fozzie_distance_left_join(
fozzie_distance_right_join(), fozzie_distance_anti_join(), fozzie_distance_full_join())
enable approximate matching of numeric fields in two data frames based on vector distance thresh-
olds. These joins are analogous to fuzzyjoin::distance_join, but implemented in Rust for
performance.

Usage

```
fozzie_distance_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_inner_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_left_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_right_join(
```

```
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_full_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_anti_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_distance_semi_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  method = "manhattan",
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A character vector of column names to match on. These columns must be numeric and present in both data frames.
how	A string specifying the join mode. One of:
	• "inner": matched pairs only.
	• "left": all rows from df1, unmatched rows filled with NAs.

- "right": all rows from df2, unmatched rows filled with NAs.
- "full": all rows from both df1 and df2.
- "anti": rows from df1 not matched in df2.
- "semi": rows from df1 that matched with one or more matches in df2.

max_distance   A numeric threshold for allowable vector distance between rows.

method   A string specifying the distance metric. One of:

- "manhattan": sum of absolute differences.
- "euclidean": square root of sum of squared differences.

distance_col   Optional name of column to store computed distances.

nthread   Optional integer specifying the number of threads to use for parallelization. If
                 not provided, the value is determined by options("fozzie.nthread"). The
                 package default is inherited from Rayon, the multithreading library used through-
                 out the package.

Value

A data frame with approximately matched rows depending on the join type. If distance_col is
specified, an additional numeric column is included.

Examples

```
df1 <- data.frame(x = c(1.0, 2.0), y = c(3.0, 4.0))
df2 <- data.frame(x = c(1.1, 2.1), y = c(3.1, 4.1))

fozzie_distance_inner_join(df1, df2, by = c("x", "y"), max_distance = 0.3, method = "euclidean")
```

---

fozzie_interval_join_family

Perform a fuzzy join between two data frames using interval overlap matching.

Description

fozzie_interval_join() and its directional variants (fozzie_interval_inner_join(), fozzie_interval_left_join()
etc.) enable approximate matching of interval columns in two data frames based on overlap logic.
These joins are conceptually similar to data.table::foverlaps() and Bioconductor's IRanges::findOverlaps(),
supporting both continuous and discrete interval semantics.

Usage

```
fozzie_interval_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
```

```
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  interval_mode = c("auto", "real", "integer"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_inner_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  interval_mode = "auto",
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_left_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  interval_mode = "auto",
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_right_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  interval_mode = "auto",
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_full_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  interval_mode = "auto",
```

```
    nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_anti_join(
    df1,
    df2,
    by = NULL,
    overlap_type = "any",
    maxgap = 0,
    minoverlap = 0,
    interval_mode = "auto",
    nthread = getOption("fozzie.nthread", NULL)
)

fozzie_interval_semi_join(
    df1,
    df2,
    by = NULL,
    overlap_type = "any",
    maxgap = 0,
    minoverlap = 0,
    interval_mode = "auto",
    nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A named list mapping left and right interval columns. Must contain two entries: start and end.
how	A string specifying the join mode. One of:
	• "inner": matched pairs only.
	• "left": all rows from df1, unmatched rows filled with NAs.
	• "right": all rows from df2, unmatched rows filled with NAs.
	• "full": all rows from both df1 and df2.
	• "anti": rows from df1 not matched in df2.
	• "semi": rows from df1 that matched with one or more matches in df2.
overlap_type	A string specifying the overlap logic. One of:
	• "any": any overlap.
	• "within": left interval fully within right.
	• "start": left start within right.
	• "end": left end within right.
maxgap	Maximum allowed gap between intervals (non-negative).
minoverlap	Minimum required overlap length (non-negative).

interval_mode   A string specifying how interval boundaries should be interpreted. One of:

- "auto": automatically infer mode based on column types.
- "real": treat interval boundaries as continuous numeric values (e.g., double). Overlaps are computed using strict inequality and floating-point arithmetic.
- "integer": treat interval boundaries as discrete integer ranges. This mode behaves similarly to Bioconductor's IRanges — intervals are inclusive and defined over integer coordinates, so [start, end] includes both endpoints. This affects how overlaps, gaps, and minimum overlap lengths are calculated, especially when maxgap or minoverlap are used.

nthread   Optional integer specifying the number of threads to use for parallelization. If not provided, the value is determined by options("fozzie.nthread"). The package default is inherited from Rayon, the multithreading library used throughout the package.

Value

A data frame with approximately matched rows depending on the join type.

Note

When interval_mode = "real", interval boundaries are treated as continuous values and matched using floating-point arithmetic. Due to precision limitations, a small threshold (typically around 1e-6) is internally added to the query range to ensure adjacent or near-touching intervals are considered for matching. This is especially relevant for timestamp-based joins, where intervals like [14:00:00, 14:00:01] and [13:00:00, 14:00:00] may fail to match unless a sufficient maxgap or internal epsilon is applied.

Examples

```
df1 <- data.frame(start = c(1, 5), end = c(3, 7))
df2 <- data.frame(start = c(2, 6), end = c(4, 8))

fozzie_interval_inner_join(df1, df2, by = c(start = "start", end = "end"), overlap_type = "any")
```

---

fozzie_regex_join_family

Perform a fuzzy join between two data frames using regex pattern matching.

Description

fozzie_regex_join() and its directional variants (fozzie_regex_inner_join(), fozzie_regex_left_join(), fozzie_regex_right_join(), fozzie_regex_anti_join(), fozzie_regex_full_join(), fozzie_regex_semi_join()) enable approximate matching of string fields in two data frames using regular expressions. These joins are analogous to fuzzyjoin::regex_join, but implemented in Rust for performance.

Usage

```
fozzie_regex_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_regex_inner_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_regex_left_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_regex_right_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_regex_anti_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_regex_full_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
```

```
)

fozzie_regex_semi_join(
  df1,
  df2,
  by = NULL,
  ignore_case = FALSE,
  nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1   A data frame to join from (left table).

df2   A data frame to join to (right table).

by   A named list or character vector indicating the matching columns. Can be a
                character vector of length 2, e.g. c("col1", "col2"), or a named list like
                list(col1 = "col2").

how   A string specifying the join mode. One of:
                - "inner": matched pairs only.
                - "left": all rows from df1, unmatched rows filled with NAs.
                - "right": all rows from df2, unmatched rows filled with NAs.
                - "full": all rows from both df1 and df2.
                - "anti": rows from df1 not matched in df2.
                - "semi": rows from df1 that matched with one or more matches in df2.

ignore_case   Should be case insensitive. Default is FALSE.

nthread   Optional integer specifying the number of threads to use for parallelization. If
                not provided, the value is determined by options("fozzie.nthread"). The
                package default is inherited from Rayon, the multithreading library used through-
                out the package.

Details

The right-hand column (from df2) is treated as a vector of regex patterns, and each value in the
left-hand column (from df1) is matched against those patterns.

Value

A data frame with approximately matched rows depending on the join type. See individual functions
like fozzie_regex_inner_join() for examples.

Examples

```
df1 <- data.frame(name = c("apple", "banana", "cherry"))
df2 <- data.frame(pattern = c("^a", "an", "rry$"))

fozzie_regex_inner_join(df1, df2, by = c("name" = "pattern"))
fozzie_regex_left_join(df1, df2, by = c("name" =  "pattern"))
```

---

fozzie_string_join_family

Perform a fuzzy join between two data frames using approximate string matching.

Description

fozzie_string_join() and its directional variants (fozzie_string_inner_join(), fozzie_string_left_join(), fozzie_string_right_join(), fozzie_string_anti_join(), fozzie_string_full_join()) enable approximate matching of string fields in two data frames. These joins support multiple string distance and similarity algorithms including Levenshtein, Jaro-Winkler, q-gram similarity, and others.

Usage

```
fozzie_string_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
  how = "inner",
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_inner_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_left_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
```

```
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_right_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_anti_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_full_join(
  df1,
  df2,
  by = NULL,
  method = "levenshtein",
  max_distance = 1,
  distance_col = NULL,
  q = NULL,
  max_prefix = 0,
  prefix_weight = 0,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_string_semi_join(
```

```
    df1,
    df2,
    by = NULL,
    method = "levenshtein",
    max_distance = 1,
    distance_col = NULL,
    q = NULL,
    max_prefix = 0,
    prefix_weight = 0,
    nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A named list or character vector indicating the matching columns. Can be a character vector of length 2, e.g. c("col1", "col2"), or a named list like list(col1 = "col2").
method	A string indicating the fuzzy matching method. Supported methods:

- "levenshtein": Levenshtein edit distance (default).
- "osa": Optimal string alignment.
- "damerau_levensthein" or "dl": Damerau-Levenshtein distance.
- "hamming": Hamming distance (equal-length strings only).
- "lcs": Longest common subsequence.
- "qgram": Q-gram similarity (requires q).
- "cosine": Cosine similarity (requires q).
- "jaccard": Jaccard similarity (requires q).
- "jaro": Jaro similarity.
- "jaro_winkler" or "jw": Jaro-Winkler similarity.
- "soundex": Soundex codes based on the National Archives standard.

how	A string specifying the join mode. One of:

- "inner": matched pairs only.
- "left": all rows from df1, unmatched rows filled with NAs.
- "right": all rows from df2, unmatched rows filled with NAs.
- "full": all rows from both df1 and df2.
- "anti": rows from df1 not matched in df2.
- "semi": rows from df1 that matched with one or more matches in df2.

max_distance	A numeric threshold for allowable string distance or dissimilarity (lower is stricter).
distance_col	Optional name of column to store computed string distances.
q	Integer. Size of q-grams for "qgram", "cosine", or "jaccard" methods.
max_prefix	Integer (for Jaro-Winkler) specifying the prefix length influencing similarity boost.

prefix_weight   Numeric (for Jaro-Winkler) specifying the prefix weighting factor.

nthread   Optional integer specifying the number of threads to use for parallelization. If
                 not provided, the value is determined by options("fozzie.nthread"). The
                 package default is inherited from Rayon, the multithreading library used through-
                 out the package.

Value

A data frame with fuzzy-matched rows depending on the join type. See individual functions like
fozzie_string_inner_join() for examples. If distance_col is specified, an additional numeric
column is included.

See [fozzie_string_join()](#)

See [fozzie_string_join()](#)

See [fozzie_string_join()](#)

See [fozzie_string_join()](#)

See [fozzie_string_join()](#)

See [fozzie_string_join()](#)

Examples

```
df1 <- data.frame(name = c("Alice", "Bob", "Charlie"))
df2 <- data.frame(name = c("Alicia", "Robert", "Charles"))

fozzie_string_inner_join(
  df1, df2, by = c("name"), method = "levenshtein", max_distance = 2
)
fozzie_string_left_join(
  df1, df2, by = c("name"), method = "jw", max_distance = 0.2
)
fozzie_string_right_join(
  df1, df2, by = c("name"), method = "cosine", q = 2, max_distance = 0.1
 )
```

---

fozzie_temporal_interval_join_family
                    Perform a fuzzy join between two data frames using time-based inter-
                    val overlap matching.

Description

fozzie_temporal_interval_join() and its directional variants (fozzie_temporal_interval_inner_join(),
fozzie_temporal_interval_left_join(), etc.) enable approximate matching of time-based in-
tervals in two data frames using continuous overlap logic.

Usage

```
fozzie_temporal_interval_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_inner_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_left_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_right_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_full_join(
  df1,
```

```
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_anti_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_interval_semi_join(
  df1,
  df2,
  by = NULL,
  overlap_type = "any",
  maxgap = 0,
  minoverlap = 0,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A named list mapping left and right interval columns. Must contain two entries: start and end.
how	A string specifying the join mode. One of:

- "inner": matched pairs only.
- "left": all rows from df1, unmatched rows filled with NAs.
- "right": all rows from df2, unmatched rows filled with NAs.
- "full": all rows from both df1 and df2.
- "anti": rows from df1 not matched in df2.
- "semi": rows from df1 that matched with one or more matches in df2.

overlap_type	A string specifying the overlap logic. One of:

- "any": any overlap.

- "within": left interval fully within right.
- "start": left start within right.
- "end": left end within right.

maxgap	Maximum allowed gap between intervals, expressed in the specified time unit.
minoverlap	Minimum required overlap length, expressed in the specified time unit.
unit	A string specifying the time unit for maxgap and minoverlap. One of: "days", "hours", "minutes", "seconds", "ms", "us", "ns".
nthread	Optional integer specifying the number of threads to use for parallelization. If not provided, the value is determined by options("fozzie.nthread"). The package default is inherited from Rayon, the multithreading library used throughout the package.

Details

All interval columns must be of the same type — either Date or POSIXct — across both data frames. Mixed types are not supported. Overlaps are computed using real-valued time semantics, allowing for fractional gaps and overlaps. This is useful for calendar intervals (Date) as well as precise timestamp ranges (POSIXct).

Value

A data frame with approximately matched rows depending on the join type.

Examples

```
df1 <- data.frame(
  start = as.Date(c("2023-01-01", "2023-01-05")),
  end = as.Date(c("2023-01-03", "2023-01-07"))
)
df2 <- data.frame(
  start = as.Date(c("2023-01-02", "2023-01-06")),
  end = as.Date(c("2023-01-04", "2023-01-08"))
)

fozzie_temporal_interval_inner_join(
  df1, df2,
  by = list(start = "start", end = "end"),
  overlap_type = "any",
  maxgap = 0.5,
  unit = "days"
)
```

---

fozzie_temporal_join_family

Perform a fuzzy join between two data frames using temporal difference matching.

Description

fozzie_temporal_join() and its directional variants (fozzie_temporal_inner_join(), fozzie_temporal_left_join(
etc.) enable approximate matching of temporal columns in two data frames based on absolute time
difference thresholds. These joins are conceptually similar to fozzie_difference_join(), but
specialized for temporal data types (Date and POSIXct).

Usage

```
fozzie_temporal_join(
  df1,
  df2,
  by = NULL,
  how = "inner",
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_inner_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_left_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_right_join(
  df1,
```

```
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_full_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_anti_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)

fozzie_temporal_semi_join(
  df1,
  df2,
  by = NULL,
  max_distance = 1,
  unit = c("days", "hours", "minutes", "seconds", "ms", "us", "ns"),
  distance_col = NULL,
  nthread = getOption("fozzie.nthread", NULL)
)
```

Arguments

df1	A data frame to join from (left table).
df2	A data frame to join to (right table).
by	A named list indicating the matching temporal columns, e.g. list(time1 = "time2").
how	A string specifying the join mode. One of:

- "inner": matched pairs only.
- "left": all rows from df1, unmatched rows filled with NAs.

- "right": all rows from df2, unmatched rows filled with NAs.
- "full": all rows from both df1 and df2.
- "anti": rows from df1 not matched in df2.
- "semi": rows from df1 that matched with one or more matches in df2.

max_distance   Maximum allowed time difference between values.

unit   A string specifying the time unit for max_distance. One of: "days", "hours", "minutes", "seconds", "ms", "us", "ns". If joining on Date columns, only "days" is allowed.

distance_col   Optional name of column to store computed time differences (in seconds or days).

nthread   Optional integer specifying the number of threads to use for parallelization. If not provided, the value is determined by options("fozzie.nthread"). The package default is inherited from Rayon, the multithreading library used throughout the package.

Details

All join columns must be either Date or POSIXct, and must be consistent across both data frames. Mixed types (e.g., Date in one and POSIXct in the other) are not allowed.

Value

A data frame with approximately matched rows depending on the join type. If distance_col is specified, an additional numeric column is included.

Examples

```
df1 <- data.frame(time = as.POSIXct(c("2023-01-01 12:00:00", "2023-01-01 13:00:00")))
df2 <- data.frame(time = as.POSIXct(c("2023-01-01 12:00:05", "2023-01-01 14:00:00")))

fozzie_temporal_inner_join(df1, df2, by = list(time = "time"), max_distance = 10, unit = "seconds")

df1 <- data.frame(date = as.Date(c("2023-01-01", "2023-01-03")))
df2 <- data.frame(date = as.Date(c("2023-01-02", "2023-01-04")))

fozzie_temporal_inner_join(df1, df2, by = list(date = "date"), max_distance = 1, unit = "days")
```

---

get_nthread_default   *Get Number of Threads in the Global Thread Pool*

Description

This function retrieves the current number of threads allocated by the Rayon thread pool. Understanding this value can be useful for optimizing the parallelization capacity of your computations.

Usage

```
get_nthread_default()
```

Value

A single numeric value indicating the number of threads in the global thread pool.

---

normalize_by   *Normalize Join Columns*

Description

Join columns expect a named list, where names are left-hand columns to join on, and values are right-hand columns to join on. This function ensures a fuzzy-like syntax to the user while producing the correct output for the Rust join utilities.

Usage

```
normalize_by(df1, df2, by)
```

Arguments

df1	A data frame representing the left-hand side of the join.
df2	A data frame representing the right-hand side of the join.
by	A named list or character vector specifying join columns. If NULL, shared column names between df1 and df2 are used.

Value

A named list mapping left-hand columns to right-hand columns.

---

test_df   *Baby Names Dataset*

Description

A small example dataset containing fictional baby names and various column types for testing joins, type handling, and metadata preservation.

Usage

```
test_df
```

Format

A data frame with 10 rows and 8 columns:

Name Character. Baby name.

int_col Integer. Some missing values.

real_col Numeric. Some missing values.

logical_col Logical. TRUE/FALSE with NA.

date_col Date. Sequential from 2020-01-01.

posixct_col POSIXct. Hourly timestamps.

posixlt_col POSIXlt. Same as above, different class.

factor_col Factor. Five levels: A–E.

Source

Created manually for testing purposes.

Index