Package 'tidyEmoji'

August 4, 2026

Type Package

Title Discover, Count, Categorise, Score, Translate and Relate Emoji in Text

Version 0.3.0

Description A tidy toolkit for working with the emoji in any text column, such as social-media posts, product reviews, chat logs or survey responses. Unicode is awkward to handle and not every code point is an emoji, which makes emoji statistics fiddly to obtain. 'tidyEmoji' extracts, counts, categorises, sentiment-scores and emotion-scores emoji, converts them to and from text (for accessibility and NLP preprocessing), searches the emoji catalogue, maps emoji co-occurrence and sequences (graph-ready edge lists and n-grams), measures where and how densely emoji are used, and builds document-by-emoji feature tables for machine learning, with grapheme-aware detection (so skin-tone and multi-person sequences stay intact), returning tidy data frames that slot straight into a 'tidyverse' workflow. The bundled emoji sentiment lexicon is from the Emoji Sentiment Ranking of Kralj Novak et al. (2015) <doi:10.1371/journal.pone.0144296>, released under CC BY-SA 4.0; the emotion lexicon is from EmoTag1200 of Shoeb & de Melo (2020) <https://aclanthology.org/2020.emnlp-main.720/>, released under the MIT licence.

License GPL (>= 3)

URL https://pursuitofdatascience.github.io/tidyEmoji/

BugReports https://github.com/PursuitOfDataScience/tidyEmoji/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports dplyr (>= 1.1.0), emoji, lifecycle, rlang, stats, tibble, tidyr, utils

Suggests rmarkdown, knitr, testthat (>= 3.0.0), ggplot2, readr, forcats, stringr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Youzhi Yu [aut, cre]

Maintainer Youzhi Yu <yuyouzhi666@icloud.com>

Repository CRAN

Date/Publication 2026-08-04 17:50:24 UTC

Contents

as_emoji_name *Vector helpers: convert emoji to/from names and shortcodes*

Description

Small vector-level helpers for ad-hoc use. They do not take a data frame.

Usage

```
as_emoji_name(x)

as_emoji_shortcode(x)

as_emoji(x)
```

Arguments

x	A character vector of emoji glyphs (for as_emoji_name, as_emoji_shortcode) or of shortcodes/names (for as_emoji).

Details

- as_emoji_name(x) maps emoji glyphs to their Unicode names.
- as_emoji_shortcode(x) maps emoji glyphs to their first shortcode.
- as_emoji(x) maps shortcodes/names to the emoji glyph (emojize).

All three resolve through emoji_key(), so qualified emoji (carrying U+FE0F) and unqualified forms resolve identically. Unmatched inputs return NA.

Value

A character vector the same length as x.

See Also

emoji_to_text(), text_to_emoji() for the data-frame verbs.

Examples

```
as_emoji_name(c("\U0001f600", "\u2764\ufe0f"))
as_emoji_shortcode(c("\U0001f600", "\u2764\ufe0f"))
as_emoji(c("grinning", "heart"))
```

category_unicode_crosswalk *Emoji category to unicode crosswalk*

Description

A table with one row per Unicode category, listing every emoji glyph in that category as a single |-separated string.

Usage

```
category_unicode_crosswalk
```

Format

A data frame with two columns:

category The Unicode category (10 categories).

unicodes The emoji glyphs in the category, separated by |.

Source

Derived from the emojis table of the **emoji** package; rebuilt by data-raw/crosswalks.R.

emoji_categorize *Categorise each row by the emoji categories it contains*

Description

emoji_categorize() keeps the rows of data that contain emoji and adds a .emoji_category column listing the distinct Unicode categories present in that row (for example "Smileys & Emotion"), separated by | when a row spans more than one category.

Usage

```
emoji_categorize(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Value

data, as a tibble, filtered to the rows containing emoji and with an added .emoji_category column.

Examples

```
df <- data.frame(text = c("smile \U0001f600",
                          "flag \U0001f3c1\U0001f600",
                          "nothing"))
emoji_categorize(df, text)
```

`emoji_cooccurrence` *Emoji co-occurrence counts, with an optional diagonal*

Description

`emoji_cooccurrence()` is `emoji_pairs()` under another name, with one addition: `diagonal = TRUE` also returns the `item1 == item2` rows, whose `n` is the number of documents containing that emoji (the diagonal of the co-occurrence matrix, i.e. its document frequency).

Usage

```
emoji_cooccurrence(data, text, doc_id = NULL, diagonal = FALSE, sort = TRUE)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`doc_id`	Optional unquoted column identifying documents. Rows sharing a value are treated as one document. Default: each row is a document.
`diagonal`	If `TRUE`, include one `item1 == item2` row per emoji with its document frequency. Default `FALSE`.
`sort`	If `TRUE` (default), sort by descending `n` (ties broken by `item1`, `item2` so the order is deterministic).

Value

A tibble with columns `item1`, `item2` and `n`.

See Also

`emoji_pairs()`, `emoji_ngrams()`.

Examples

```
df <- data.frame(text = c("\U0001f602\U0001f60d", "\U0001f602"))
emoji_cooccurrence(df, text, diagonal = TRUE)
```

emoji_density *Emoji density per character and per token*

Description

`emoji_density()` measures how emoji-heavy each text is: the number of emoji per character and per whitespace-delimited token. Rows with no emoji get densities of 0; rows whose text is `NA` or empty get `NA`.

Usage

```
emoji_density(data, text)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.

Value

`data`, as a tibble, with added columns `.emoji_n`, `.emoji_per_char` (emoji per character of text) and `.emoji_per_token` (emoji per whitespace-delimited token).

See Also

`emoji_position()`, `emoji_ratio()`.

Examples

```
df <- data.frame(text = c("hi \U0001f600", "\U0001f600\U0001f600", "plain"))
emoji_density(df, text)
```

emoji_dfm *Document-by-emoji feature matrix*

Description

`emoji_dfm()` turns a text column into a wide, model-ready table with one row per document and one column per emoji, weighted by raw counts, binary presence or tf-idf. All documents are kept, including those with no emoji (all-zero rows), so the result aligns row-for-row with the corpus and can be bound to outcome columns for tidymodels-style workflows.

Usage

```
emoji_dfm(data, text, doc_id = NULL, weighting = c("count", "binary", "tfidf"))
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.
doc_id	Optional unquoted column identifying documents; rows sharing a value are aggregated into one document. Default: each row is a document.
weighting	One of "count" (default), "binary" or "tfidf".

Details

By default every row of data is a document and the first output column, .row_number, is its position in data (matching emoji_extract_unnest()). Give doc_id to aggregate rows sharing an id into one document; the id column keeps its name. Emoji columns are named by the glyph itself, canonicalised through the package's codepoint key (so qualified and unqualified forms count as one feature), and ordered by descending total count (ties broken by glyph).

For weighting = "tfidf", the cell for emoji *e* in document *d* is count(d, e) * log(N / df(e)), where N is the number of documents and df(e) the number of documents containing *e*. An emoji that appears in every document therefore scores 0.

Value

A tibble with one row per document: .row_number (or the doc_id column) followed by one numeric column per emoji. Zero emoji in the corpus yields just the document column.

See Also

emoji_frequency() for corpus totals; emoji_tokens() for the long form this widens.

Examples

```
df <- data.frame(text = c("\U0001f600\U0001f600 fun", "\U0001f621",
                          "no emoji"))
emoji_dfm(df, text)
emoji_dfm(df, text, weighting = "binary")
emoji_dfm(df, text, weighting = "tfidf")
```

---

emoji_emotion *Emoji emotion profiles (the 8 Plutchik emotions)*

Description

emoji_emotion() scores each row's emoji across the eight Plutchik emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) using the bundled EmoTag1200 lexicon (Shoeb & de Melo, 2020). Scores each range from 0 to 1 and are averaged over the emoji in the row that appear in the lexicon.

Usage

```
emoji_emotion(data, text, lexicon = "emotag1200", long = FALSE)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.
lexicon	Lexicon to use. Either a string naming a bundled lexicon ("emotag1200", the default), the name of a registered lexicon (see register_emoji_lexicon()), or a data frame. A custom lexicon must have an emoji column and one column per emotion (any subset of the eight Plutchik emotions); it is joined through the same codepoint-normalised key as the bundled one.
long	If TRUE, return one row per (row, emotion) in long form with columns .emoji_emotion (the emotion name) and .emoji_score (its mean). Default FALSE adds eight .emoji_<emotion> columns plus .emoji_n and .emoji_n_scored.

Value

data, as a tibble, with emotion columns added. Rows without emoji, or whose emoji are absent from the lexicon, receive NA scores.

References

Shoeb AAM, de Melo G (2020). EmoTag1200: Understanding the Association between Emojis and Emotions. *EMNLP 2020*. https://aclanthology.org/2020.emnlp-main.720/. Data released under the MIT licence.

See Also

emoji_emotion_lexicon for the underlying scores; emoji_emotion_label() for the dominant emotion per row; emoji_sentiment() for valence.

Examples

```
df <- data.frame(text = c("love it \U0001f60d", "scary \U0001f628", "meh"))
emoji_emotion(df, text)
emoji_emotion(df, text, long = TRUE)
```

---

emoji_emotion_label *The dominant emoji emotion per row*

Description

emoji_emotion_label() adds .emoji_emotion, the emotion with the highest mean score among the row's emoji (using emoji_emotion()). Ties are broken in Plutchik order; rows with no scored emoji receive NA.

Usage

```
emoji_emotion_label(data, text, lexicon = "emotag1200")
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.
lexicon	Passed to emoji_emotion().

Value

data, as a tibble, with a .emoji_emotion column added.

Examples

```
df <- data.frame(text = c("love it \U0001f60d", "scary \U0001f628", "meh"))
emoji_emotion_label(df, text)
```

---

emoji_emotion_lexicon *Emoji emotion lexicon (EmoTag1200)*

Description

Human-annotated emotion-association scores (each from 0 to 1) for the eight Plutchik emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust), for the 150 most popular Twitter emoji, from EmoTag1200.

Usage

```
emoji_emotion_lexicon
```

Format

A data frame with one row per emoji and the columns:

key Codepoint-normalised key (U+FE0F stripped) for robust joining.

emoji The emoji glyph (unqualified form, as stored by the source).

name The emoji's Unicode name.

anger, anticipation, disgust, fear, joy, sadness, surprise, trust Emotion-association scores, each from 0 to 1.

Source

Shoeb AAM, de Melo G (2020). EmoTag1200: Understanding the Association between Emojis and Emotions. *EMNLP 2020*. https://aclanthology.org/2020.emnlp-main.720/. Data from https://github.com/abushoeb/EmoTag, released under the MIT licence. Processed by data-raw/emoji_emotion_lexicon.R.

emoji_extract_nest *Add a list-column of the emoji found in each row*

Description

`emoji_extract_nest()` returns `data` unchanged except for an added list-column, `.emoji_unicode`, holding the emoji found in each row. Detection is grapheme-aware, so skin-tone modifiers and ZWJ sequences (for example family emoji) are kept intact as a single emoji.

Usage

```
emoji_extract_nest(data, text)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.

Value

`data` with an added list-column `.emoji_unicode`.

See Also

`emoji_extract_unnest()` for a long, counted form and `emoji_tokens()` for one row per emoji with metadata.

Examples

```
df <- data.frame(text = c("hi \U0001f600\U0001f603", "none"))
emoji_extract_nest(df, text)
```

emoji_extract_unnest *Emoji counts per row, in long (tidy) form*

Description

`emoji_extract_unnest()` returns one row per (row, emoji) pair with a count, dropping rows that contain no emoji. `.row_number` refers to the position of the entry in `data`.

Usage

```
emoji_extract_unnest(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Value

A tibble with columns .row_number, .emoji_unicode and .emoji_count.

Examples

```
df <- data.frame(text = c("hi \U0001f600\U0001f600", "none", "\U0001f44b"))
emoji_extract_unnest(df, text)
```

---

emoji_filter *Keep only the rows whose text contains emoji*

Description

emoji_filter() returns the rows of data whose text column contains at least one emoji, preserving every original column. emoji_tweets() is a synonym retained for backward compatibility.

Usage

```
emoji_filter(data, text)

emoji_tweets(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Value

A tibble containing only the rows with at least one emoji. The result is always a plain (ungrouped) tibble, whatever the class or grouping of the input.

Examples

```
df <- data.frame(text = c("hi \U0001f600", "no emoji", "bye \U0001f44b"))
emoji_filter(df, text)
```

emoji_frequency *Frequency of every emoji in a text column*

Description

`emoji_frequency()` counts how often each emoji appears across the whole text column (an entry containing the same emoji twice contributes 2) and returns a tibble sorted by descending count, with each emoji's name, shortcode and category.

Usage

```
emoji_frequency(data, text)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.

Value

A tibble with columns `emoji`, `name`, `shortcode`, `group` and `n`, sorted by descending `n` with ties broken by the glyph so the order is deterministic.

See Also

`top_n_emojis()` for just the most frequent emoji.

Examples

```
df <- data.frame(text = c("\U0001f600\U0001f600", "\U0001f621"))
emoji_frequency(df, text)
```

emoji_lexicons *List bundled emoji lexicons*

Description

`emoji_lexicons()` returns a tibble describing the lexicons bundled with tidyEmoji and any user-registered ones: their name, type (sentiment or emotion), dimensions, number of emoji, source and licence.

Usage

```
emoji_lexicons()
```

Value

A tibble with columns `name`, `type`, `dimensions`, `n`, `source`, `licence`.

See Also

`register_emoji_lexicon()` to add your own; `emoji_score()` to score text against any lexicon.

Examples

```
emoji_lexicons()
```

---

emoji_ngrams *Consecutive emoji sequences (n-grams)*

Description

`emoji_ngrams()` slides a window of `n` over each row's emoji, in reading order (any text between the emoji is ignored), and returns one row per n-gram occurrence. Repeated emoji are kept: a row containing the same emoji twice in a row yields a bigram of that emoji with itself. This is the emoji analogue of `tidytext::unnest_tokens(..., token = "ngrams")` and feeds sequence / Markov-style analyses of how emoji chain together.

Usage

```
emoji_ngrams(data, text, n = 2, sep = " ")
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`n`	Length of the n-gram window. Default 2 (bigrams).
`sep`	Separator between the glyphs of an n-gram. Default a space.

Value

A tibble with columns `.row_number` (position of the entry in `data`), `.position` (where the n-gram starts within the row's emoji sequence) and `.emoji_ngram`. Rows with fewer than `n` emoji contribute nothing.

See Also

`emoji_pairs()` for order-free co-occurrence; `emoji_extract_unnest()` for the underlying one-emoji-per-row form.

Examples

```
df <- data.frame(text = c("\U0001f602\U0001f60d\U0001f389", "\U0001f602"))
emoji_ngrams(df, text)
emoji_ngrams(df, text, n = 3)
```

`emoji_pairs` *Co-occurring emoji pairs*

Description

`emoji_pairs()` returns a tidy edge list of the emoji that appear together in the same document: one row per pair with the number of documents in which the pair co-occurs. By default every row of `data` is a document; give `doc_id` to treat all rows sharing an id (a conversation, a user, a day) as one document. The output mirrors `widyr::pairwise_count()` (`item1`, `item2`, `n`) and pipes straight into `igraph::graph_from_data_frame()`, tidygraph or ggraph.

Usage

```
emoji_pairs(data, text, doc_id = NULL, directed = FALSE, sort = TRUE)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`doc_id`	Optional unquoted column identifying documents. Rows sharing a value are treated as one document. Default: each row is a document.
`directed`	If `TRUE`, pairs are ordered by first appearance: a document where the tears-of-joy emoji appears before the heart-eyes emoji counts towards (tears-of-joy, heart-eyes), not the reverse. Default `FALSE` (unordered pairs, with `item1` sorted before `item2`).
`sort`	If `TRUE` (default), sort by descending `n` (ties broken by `item1`, `item2` so the order is deterministic).

Details

Glyphs are canonicalised through the package's codepoint key, so qualified and unqualified forms of the same emoji (with/without `U+FE0F`) count as one node. Pairs are between *distinct* emoji: repeats of the same emoji in a document do not pair with themselves (see `emoji_cooccurrence()` for the diagonal).

Value

A tibble with columns `item1`, `item2` and `n`. Empty (but typed) when no document contains two distinct emoji.

See Also

`emoji_cooccurrence()` for the same counts with an optional diagonal; `emoji_ngrams()` for consecutive sequences.

Examples

```
df <- data.frame(text = c("fun \U0001f602\U0001f60d",
                          "\U0001f602\U0001f60d\U0001f389",
                          "just \U0001f602"))
emoji_pairs(df, text)
emoji_pairs(df, text, directed = TRUE)
```

`emoji_position` *Where do emoji sit within each text?*

Description

`emoji_position()` reports, for each row, the character position of the first and last emoji and the mean *relative* position of all emoji occurrences, from 0 (the very start of the text) to 1 (the very end). The Emoji Sentiment Ranking (Kralj Novak et al., 2015) tracks the same relative position, and it is a studied signal: emoji cluster near the end of messages.

Usage

```
emoji_position(data, text)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.

Details

The relative position of an occurrence starting at character `s` in a text of `L` characters is `(s - 1) / (L - 1)` (taken as 0 when `L <= 1`). Positions are counted in characters (code points), the same unit as `substr()`.

Value

`data`, as a tibble, with added columns `.emoji_n`, `.emoji_first` and `.emoji_last` (character positions where the first/last emoji start) and `.emoji_rel_position` (mean relative position in `[0, 1]`). Rows without emoji get `NA` positions.

See Also

`emoji_density()` and `emoji_ratio()` for intensity metrics.

Examples

```
df <- data.frame(text = c("\U0001f600 leading", "trailing \U0001f600",
                          "none"))
emoji_position(df, text)
```

`emoji_ratio` *What share of the text is emoji — and is it emoji-only?*

Description

`emoji_ratio()` reports, per row, the share of the text's characters that belong to emoji, and whether the text is emoji-only (nothing left after removing emoji and whitespace). "Emoji-only" messages are a studied signal in social-media research and a useful filter in practice.

Usage

```
emoji_ratio(data, text)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.

Details

The ratio is computed over characters (code points), so a multi-code-point emoji (a ZWJ family, a skin-tone sequence) contributes all of its characters.

Value

`data`, as a tibble, with added columns `.emoji_ratio` (emoji characters / all characters, 0 when there are no emoji) and `.emoji_only` (`TRUE` when the text contains emoji and nothing else but whitespace). `NA` text gets `NA` in both.

See Also

`emoji_position()`, `emoji_density()`; `emoji_filter()` to keep emoji-bearing rows.

Examples

```
df <- data.frame(text = c("\U0001f600\U0001f389", "half \U0001f600", "no"))
emoji_ratio(df, text)
```

`emoji_score` *Score emoji in a text column against any lexicon*

Description

`emoji_score()` is the generic scorer that the friendly verbs (`emoji_sentiment()`, `emoji_emotion()`) sit on top of. It joins each row's emoji to `lexicon` through `emoji_key()` and returns the per-row mean of the `score` column, plus the number of emoji scored. Bring your own lexicon, or name a bundled / registered one.

Usage

```
emoji_score(data, text, lexicon, by = "emoji", score = NULL)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`lexicon`	Either a string naming a bundled or registered lexicon, or a data frame. For data frames, `by` names the glyph column and `score` the score column.
`by`	Glyph column name when `lexicon` is a data frame. Default `"emoji"`.
`score`	Score column name when `lexicon` is a data frame. If `NULL`, `"sentiment_score"` then `"score"` are tried.

Value

`data`, as a tibble, with `.emoji_score` (per-row mean), `.emoji_n_scored` (emoji found in the lexicon) and `.emoji_n` (total emoji) added. For the multi-dimensional `"emotag1200"` lexicon the score is the mean over its eight emotion dimensions; use `emoji_emotion()` for the per-emotion profile.

See Also

`emoji_lexicons()`, `register_emoji_lexicon()`.

Examples

```
df <- data.frame(text = c("love \U0001f60d", "angry \U0001f621", "meh"))
emoji_score(df, text, lexicon = "novak2015")

# a bring-your-own lexicon
own <- data.frame(emoji = c("\U0001f600", "\U0001f621"),
                  score = c(0.9, -0.8))
emoji_score(df, text, lexicon = own)
```

emoji_search *Search emoji by keyword, name or shortcode*

Description

`emoji_search()` finds emoji whose Unicode keywords, name or shortcodes match a query (case-insensitive, substring match). It returns a tidy tibble of matches with the glyph, name, shortcode, category and the matching keywords, ready for further inspection or piping into other verbs.

Usage

```
emoji_search(query)
```

Arguments

`query`	A search string, matched as a case-insensitive substring against keywords, name and shortcodes.

Value

A tibble with columns `emoji`, `name`, `shortcode`, `group` and `keyword` (the keywords of the emoji that contained the match, collapsed with `, `).

Examples

```
emoji_search("happy")
emoji_search("heart")
```

emoji_sentiment *Score the sentiment of the emoji in each row*

Description

`emoji_sentiment()` adds the mean emoji sentiment of each row, based on the Emoji Sentiment Ranking lexicon (see emoji_sentiment_lexicon). Scores range from -1 (negative) through 0 (neutral) to +1 (positive). Rows that contain no emoji, or whose emoji are absent from the lexicon, receive `NA`.

Usage

```
emoji_sentiment(data, text, lexicon = "novak2015")
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`lexicon`	Lexicon to use. The default, `"novak2015"`, uses the bundled emoji_sentiment_lexicon. A registered lexicon (see `register_emoji_lexicon()`) or a data frame can also be supplied; see `emoji_score()` for the generic scorer.

Details

Detection is grapheme-aware. Some lexicon entries are stored as unqualified, text-presentation code points (notably the bare heart, `U+2764`, without the `U+FE0F` variation selector); those are not treated as emoji in your text, so they are neither counted nor scored. Supply the emoji-presentation (qualified) form and it resolves normally. See emoji_sentiment_lexicon for the full picture.

Value

`data`, as a tibble, with added columns `.emoji_n` (the number of emoji in the row), `.emoji_n_scored` (the number of emoji that actually appear in the lexicon), and `.emoji_sentiment` (the mean sentiment of the scored emoji).

References

Kralj Novak P, Smailovic J, Sluban B, Mozetic I (2015) Sentiment of Emojis. PLoS ONE 10(12): e0144296. doi:10.1371/journal.pone.0144296

See Also

emoji_sentiment_lexicon for the underlying scores; `emoji_score()` for scoring against any lexicon; `emoji_emotion()` for discrete emotions.

Examples

```
df <- data.frame(text = c("love it \U0001f60d", "awful \U0001f621", "meh"))
emoji_sentiment(df, text)
```

`emoji_sentiment_lexicon`

Emoji Sentiment Ranking lexicon

Description

Sentiment scores for emoji, from the *Emoji Sentiment Ranking 1.0*, computed from ~70,000 tweets in 13 European languages annotated for sentiment. The `sentiment_score` is `(positive - negative) / occurrences`, ranging from -1 (negative) to +1 (positive); `sentiment_label` is derived from its sign.

Usage

```
emoji_sentiment_lexicon
```

Format

A data frame with one row per emoji and the columns:

emoji The emoji glyph.

occurrences Number of times the emoji was observed.

position Mean position of the emoji within its text (0-1).

negative, neutral, positive Annotation counts for each class.

sentiment_score Sentiment score from -1 to 1.

sentiment_label "negative", "neutral" or "positive".

unicode_name The official Unicode character name.

unicode_block The Unicode block.

Detection limitations

Many of the glyphs in this lexicon are stored in their *unqualified*, text-presentation form: a single code point with no `U+FE0F` emoji-presentation variation selector. The best-known is the bare heart, `U+2764`; others include the white smiling face (`U+263A`), the heavy check mark (`U+2714`) and the black rightwards arrow (`U+27A1`). The lexicon also contains characters that are not emoji at all (box-drawing characters, the copyright and registered signs, the replacement character), inherited from the tweets it was built from.

The grapheme-aware detection used throughout the package does not treat these text-presentation code points as emoji, so a row whose only "emoji" is one of them is not counted or scored – it behaves as if it contained no emoji. This affects detection only, never the join: supply the qualified form (the red heart `U+2764` `U+FE0F`, say) and it resolves to the same lexicon entry, because every lookup goes through a codepoint key that ignores `U+FE0F`.

Source

Kralj Novak P, Smailovic J, Sluban B, Mozetic I (2015) Sentiment of Emojis. PLoS ONE 10(12): e0144296. doi:10.1371/journal.pone.0144296. Data from `https://hdl.handle.net/11356/1048`, released under the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) licence. Processed by `data-raw/emoji_sentiment_lexicon.R`.

`emoji_summary` *Summarise emoji presence in a text column*

Description

`emoji_summary()` reports how many entries in a text column contain at least one emoji, alongside the total number of entries. An entry is counted once regardless of how many emoji it holds.

Usage

```
emoji_summary(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Value

A one-row tibble with columns `n_with_emoji` (entries containing at least one emoji) and `n_total` (all entries).

See Also

`emoji_filter()` to keep the emoji-bearing rows themselves.

Examples

```
df <- data.frame(text = c("I love R \U0001f600",
                          "no emoji here",
                          "flags \U0001f3c1\U0001f600"))
emoji_summary(df, text)
```

---

emoji_tokens *Tidy emoji tokens, one row per occurrence with metadata*

Description

`emoji_tokens()` expands `data` to one row per emoji occurrence (in reading order), keeping the original columns and adding the glyph together with its name, category and sentiment score. This mirrors the one-token-per-row shape familiar from tidy text mining and is convenient for counting, joining and plotting.

Usage

```
emoji_tokens(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Value

A tibble with the original columns plus `.emoji`, `.emoji_name`, `.emoji_category` and `.emoji_sentiment`. Rows without emoji are dropped.

See Also

emoji_frequency() for corpus-level counts and emoji_sentiment() for per-row sentiment.

Examples

```
df <- data.frame(id = 1:2, text = c("great \U0001f600", "bad \U0001f621"))
emoji_tokens(df, text)
```

---

emoji_to_text *Replace emoji in a text column with words (demojize)*

Description

emoji_to_text() returns a copy of data with its text column rewritten so that every emoji is replaced by its name or shortcode. This is useful for accessibility (screen readers) and as an NLP normalisation step before tokenising. Detection is grapheme-aware and joins go through emoji_key(), so emoji carrying the U+FE0F variation selector still resolve.

Usage

```
emoji_to_text(data, text, format = c("name", "shortcode"), wrap = ":{x}:")
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.
format	Output form: "name" (the Unicode name, e.g. "grinning face") or "shortcode" (the canonical GitHub-style alias, e.g. "grinning", wrapped as ":grinning:"). Default "name".
wrap	When format = "shortcode", the wrapper applied to each shortcode, written as a template with {x} standing for the shortcode. Default ":{x}:". Ignored for format = "name".

Value

data, as a tibble, with the text column rewritten in place (same column name). NA entries stay NA, and emoji with no known name are left in place unchanged.

See Also

text_to_emoji() for the inverse (emojize); as_emoji_name(), as_emoji_shortcode(), as_emoji() for vector helpers.

Examples

```
df <- data.frame(text = "great \U0001f600 love \u2764\ufe0f")
emoji_to_text(df, text, format = "name")
emoji_to_text(df, text, format = "shortcode")
```

`emoji_unicode_crosswalk` *Emoji name, unicode and category crosswalk*

Description

A table with one row per emoji *name*: each emoji glyph appears once for every GitHub-style name it is known by, so a single unicode can occur on several rows (for example the grinning face is both "grinning" and "grinning_face").

Usage

```
emoji_unicode_crosswalk
```

Format

A data frame with four columns:

emoji_name The emoji name / shortcode (e.g. "grinning").

unicode The emoji glyph.

emoji_category The Unicode category the emoji belongs to.

key Codepoint-normalised key (U+FE0F stripped) for robust joining.

Source

Derived from the `emojis` table of the **emoji** package; rebuilt by `data-raw/crosswalks.R`.

`register_emoji_lexicon` *Register a custom emoji lexicon*

Description

`register_emoji_lexicon()` adds a user-supplied lexicon to the in-session registry so it can be referenced by name in `emoji_score()`, `emoji_sentiment()` or `emoji_emotion()`. The lexicon is normalised through the package's codepoint key (`U+FE0F` stripped), so a lexicon keyed on unqualified glyphs still matches qualified text.

Usage

```
register_emoji_lexicon(name, tbl, by = "emoji")
```

Arguments

name	Name to register the lexicon under.
tbl	A data frame. Must contain a glyph column named by (default "emoji") and at least one score column.
by	Name of the column holding the emoji glyph. Default "emoji".

Details

Registration lasts for the session; it is not written to disk.

Value

Invisibly, the registered lexicon (with an added key column).

See Also

emoji_lexicons() to list lexicons; emoji_score() to use one.

Examples

```
my_lex <- data.frame(
  emoji = c("\U0001f600", "\U0001f621"),
  score = c(0.9, -0.8)
)
register_emoji_lexicon("mine", my_lex)
emoji_lexicons()
emoji_score(data.frame(text = "great \U0001f600"), text, lexicon = "mine")
```

---

text_to_emoji *Replace shortcodes with emoji (emojize)*

Description

text_to_emoji() returns a copy of data with its text column rewritten so that every :shortcode: token is replaced by the corresponding emoji glyph (the inverse of emoji_to_text() with format = "shortcode"). Shortcodes that do not match a known emoji are left unchanged.

Usage

```
text_to_emoji(data, text)
```

Arguments

data	A data frame or tibble containing a text column.
text	The text column to scan, supplied unquoted.

Details

A shortcode token is a colon, one or more of `A-Z`, `a-z`, `0-9`, `_`, `+` or `-`, and a closing colon. Restricting the token this way means colons used for other purposes – clock times, URLs, ratios, ordinary punctuation – cannot swallow a following shortcode: `"meet at 10:30 :wave:"` still emojizes the wave.

Value

`data`, as a tibble, with the text column rewritten in place. `NA` entries stay `NA`.

See Also

`emoji_to_text()`; `as_emoji()` for the vector helper.

Examples

```
df <- data.frame(text = "hi :grinning: bye :waving_hand:")
text_to_emoji(df, text)

# colons elsewhere in the text do not interfere
text_to_emoji(data.frame(text = "https://example.org at 10:30 :grinning:"),
              text)
```

`top_n_emojis` *The most frequent emoji in a text column*

Description

`top_n_emojis()` returns the `n` most frequent emoji. By default each emoji (unicode) appears on a single row; set `duplicated = TRUE` to list every name an emoji is known by, so glyphs that share several names occupy several rows.

Usage

```
top_n_emojis(
  data,
  text,
  n = 20,
  duplicated = FALSE,
  duplicated_unicode = lifecycle::deprecated()
)
```

Arguments

`data`	A data frame or tibble containing a text column.
`text`	The text column to scan, supplied unquoted.
`n`	Number of emoji to return. Default `20`.

duplicated If TRUE, emoji with several names occupy several rows. Default FALSE.

duplicated_unicode
[Deprecated] Use duplicated instead.

Value

A tibble with columns emoji_name, unicode, emoji_category and n.

See Also

emoji_frequency() for the full distribution.

Examples

```
df <- data.frame(text = c("\U0001f600\U0001f600\U0001f3c1", "\U0001f621"))
top_n_emojis(df, text, n = 2)
```

Index