Package 'emoji'

October 28, 2024

Title Data and Function to Work with Emojis

Version 16.0.0

Description Contains data about emojis with relevant metadata, and functions to work with emojis when they are in strings.

License MIT + file LICENSE

URL https://emilhvitfeldt.github.io/emoji/, https://github.com/EmilHvitfeldt/emoji

BugReports https://github.com/EmilHvitfeldt/emoji/issues

Depends R (>= 3.5)

Imports glue, stringr, tibble

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre] (<https://orcid.org/0000-0002-0679-1945>), Hadley Wickham [ctb] (Data parsing code from hadley/emo), Romain François [ctb] (Data parsing code from hadley/emo)

Maintainer Emil Hvitfeldt <emilhhvitfeldt@gmail.com>

Repository CRAN

Date/Publication 2024-10-28 16:50:11 UTC

Contents

arrow 2
clock 3
emoji 3
emojis 4

emoji_count 5
emoji_detect 6
emoji_extract 7
emoji_find 8
emoji_fix 8
emoji_glue 9
emoji_keyword 10
emoji_locate 10
emoji_match 11
emoji_modifiers 12
emoji_modifier_extract 12
emoji_modifier_remove 13
emoji_name 14
emoji_p 15
emoji_replace 16
emoji_replace_name 16
emoji_rx 17
emoji_subset 18
flag 18
keycap 19
medal 20
moon 21
shape 21
zoo 22

Index **24**

`arrow` *Insert Arrow emojis*

Description

Insert Arrow emojis

Usage

```
arrow(direction)
```

Arguments

`direction`	Character denoting the direction of the arrow. Should be one of "up", "up-right", "right", "down-right", "down", "down-left", "left", "up-left", "up-down", or "left-right".

Details

This function is vectorized. Wrong input of `direction` will result in `NA`s.

#@return Character vector of emojis.

Examples

```
arrow("up-down")

arrow(c("up", "up", "down", "down", "left", "right", "left", "right"))
```

---

clock *emoji version of time*

Description

emoji version of time

Usage

```
clock(time)
```

Arguments

time	a POSIXct object

Details

This function is vectorized.

Value

Character vector of emojis showing the closest time.

Examples

```
times <- as.POSIXct("2021-09-17 14:33:21 PDT") + seq(1:30) * 3500
clock(times)
```

---

emoji *Find a single emoji*

Description

This function starts by looking for exact matches in emoji_name. If none is found in emoji_name then it looks in emoji_keyword. emoji_keyword can produce more then 1 matches, which will lead to one being returned at random.

Usage

```
emoji(keyword)
```

Arguments

keyword	Character, either name or keyword. If more than one emoji has the specified keyword, will pick one at random.

Details

This function isn't vectorized and will thus only work with 1 keyword at a time.

Examples

```
emoji("smile")
emoji("taco")

set.seed(1234)
replicate(24, emoji("clock"))
replicate(10, emoji("flag"))
```

---

emojis *Full List of Emojis*

Description

This data set is the heart of the emoji package. It contains various information regarding all the avaiable emojis as of v16.0.

Usage

```
emojis
```

Format

tibble with 19 columns and `nrow(emojis)` rows

emoji character representation of the emoji

name name

group group, e.g. "Smileys & People"

subgroup sub group, e.g. "face-positive"

version version where the emoji was introduced

points Decimal Code Point(s)

nrunes number of runes the emoji uses

runes vector of unicode runes, i.e. hexadecimal representations prefixed with "U+"

qualified Status of the emoji, can be one of 4 types; "component", "fully-qualified", "minimally-qualified", and "unqualified". See details for more.

vendor_* for apple ... windows logical indicating if the given vendor supports the emoji

keywords vector of keywords

keywords vector of aliases

Details

The levels of `qualified` have the following meaining

- **component**: an Emoji_Component, excluding Regional_Indicators, ASCII, and non-Emoji.
- **fully-qualified**: a fully-qualified emoji (see ED-18 in UTS #51), excluding Emoji_Component
- **minimally-qualified**: a minimally-qualified emoji (see ED-18a in UTS #51)
- **unqualified**: a unqualified emoji (See ED-19 in UTS #51)

Source

Unicode® Full Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/full-emoji-list.html

Unicode® Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/emoji-list.html

Unicode® Emoji Ordering, v16.0 - https://www.unicode.org/emoji/charts/emoji-ordering.txt

https://github.com/github/gemoji

https://github.com/muan/emojilib

See Also

emoji_name emoji_keyword

`emoji_count` *Count the number of emojis in a string*

Description

Vectorised over `string`

Usage

```
emoji_count(string)
```

Arguments

`string`	Input vector

Value

An integer vector

See Also

`stringr::str_count()`

Examples

```
string <- paste(c(letters[1:4], emoji_name[1:6]), collapse = " ")

emoji_count(string)

emoji_count(emoji_name[1:6])
```

`emoji_detect` *Detect the presence or absence of emojis in a string*

Description

Vectorised over `string`

Usage

```
emoji_detect(string, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`negate`	If `TRUE`, inverts the resulting boolean vector.

Value

A logical vector

See Also

`stringr::str_detect()`

Examples

```
string <- c(letters[1:4], emoji_name[1:6])
emoji_detect(string)
```

`emoji_extract` *Extract emojis from a string*

Description

vectorised over `string`

Usage

```
emoji_extract(string)

emoji_extract_all(string, simplify = FALSE)
```

Arguments

`string`	Input vector.
`simplify`	see `stringr::str_extract_all()`

Value

A character vector

See Also

`stringr::str_extract()` and `stringr::str_extract_all()`

Examples

```
chars <- c(letters[1:4], emoji_name[1:6])

set.seed(1234)

strings <- lapply(1:10, function(x) paste(sample(chars, x), collapse = ""))

extracts <- emoji_extract(strings)

all_extracts <- emoji_extract_all(strings)
```

emoji_find *List all emoji with a given keyword*

Description

This function will look in `emoji_keyword` to report back the given emojis.

Usage

```
emoji_find(keyword)
```

Arguments

`keyword`	Character, Emoji keyword.

Examples

```
emoji_find("happy")
emoji_find("cat")
emoji_find("family")
```

emoji_fix *Turn emojis into qualified emojis*

Description

Some emojis can be written in multiple different ways either as `fully-qualified`, `minimally-qualified`, or `unqualified`. `emoji_fix()` will take any emoji and return the `fully-qualified` version of that emoji.

Usage

```
emoji_fix(x)
```

Arguments

`x`	Characters, vector of emojis.

Details

This function is vectorized.

Value

vector of fully-qualified emojis

Examples

```
unqualified_ind <- which(emojis$qualified == "unqualified")[1:10]
unqualified <- emojis$emoji[unqualified_ind]

unqualified
emoji_fix(unqualified)
```

`emoji_glue` *Glue Interpolation for Emojis*

Description

Combine the power of `glue::glue` and `emoji()`.

Usage

```
emoji_glue(..., .envir = parent.frame())
```

Arguments

`...`	[`expressions`] Unnamed arguments are taken to be expression string(s) to format. Multiple inputs are concatenated together before formatting. Named arguments are taken to be temporary variables available for substitution. For `glue_data()`, elements in `...` override the values in `.x`.
`.envir`	[`environment`: `parent.frame()`] Environment to evaluate each expression in. Expressions are evaluated from left to right. If `.x` is an environment, the expressions are evaluated in that environment and `.envir` is ignored. If `NULL` is passed, it is equivalent to `emptyenv()`.

Details

`emoji_glue()` behaves in much the same way a lot of messaging apps work. Anything inside a pair of `:` will be interpolated into an emoji. You can think of `emoji_glue()` as being a shorthand for `glue("I love {emoji('taco')}s")`.

Block ending with `*` will be collapsed.

Value

a `glue::glue()` string.

Examples

```
emoji_glue("I love :taco:s")

emoji_glue("one :heart:")
emoji_glue("many :heart*:")
```

emoji_keyword *Emoji Keywords*

Description

This list contains information about which emojis are contained in which keywords.

Usage

```
emoji_keyword
```

Format

named list of characters with 7665 elements

Source

Unicode® Full Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/full-emoji-list.html

Unicode® Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/emoji-list.html

Unicode® Emoji Ordering, v16.0 - https://www.unicode.org/emoji/charts/emoji-ordering.txt#' @source https://github.com/github/gemoji

https://github.com/muan/emojilib

See Also

emojis emoji_name

emoji_locate *Locate the position of emojis in a string*

Description

Vectorised over `string`

Usage

```
emoji_locate(string)

emoji_locate_all(string)
```

Arguments

`string`	Input vector

Value

For `emoji_locate` an integer matrix, for `emoji_locate_all` a list of integer matrices

Examples

```
string <- paste(c(letters[1:4], emoji_name[1:6]), collapse = " ")

emoji_locate(string)

emoji_locate_all(string)
```

---

emoji_match *Extract matched emojis from a string*

Description

Vectorized over `string`

Usage

```
emoji_match(string)

emoji_match_all(string)
```

Arguments

`string`	Input vector

Value

see `stringr::str_match()`

See Also

stringr::str_match

Examples

```
chars <- c(letters[1:4], emoji_name[1:6])

set.seed(1234)

strings <- lapply(1:10, function(x) paste(sample(chars, x), collapse = ""))

extracts <- emoji_match(strings)


extracts <- emoji_match_all(strings)
```

emoji_modifiers *Emoji Modifiers*

Description

This data set contains all the emojis with modifiers, their unmodified version as well as a list of the the modifiers.

Usage

```
emoji_modifiers
```

Format

tibble with 3 columns and `nrow(emoji_modifiers)` rows

emoji_modifiers character representation of the emoji with modifiers

emoji character representation of the emoji without modifiers

modifiers list of modifiers

Source

Unicode® Full Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/full-emoji-list.html

Unicode® Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/emoji-list.html

Unicode® Emoji Ordering, v16.0 - https://www.unicode.org/emoji/charts/emoji-ordering.txt#' @source https://github.com/github/gemoji

https://github.com/muan/emojilib

See Also

emojis emoji_name

emoji_modifier_extract *Extract Modifiers from Emojis*

Description

Extract Modifiers from Emojis

Usage

```
emoji_modifier_extract(x)
```

Arguments

x	Characters, vector of emojis.

Details

This function is vectorized. See emoji_modifiers for full list of modified emojis and their unmodified state.

Value

list of character vectors.

Examples

```
waving_hands <- emojis$emoji[grepl("waving hand", emojis$name)]
waving_hands

emoji_modifier_extract(waving_hands)

set.seed(1234)
emoji_sample <- sample(emojis$emoji, 10)
emoji_sample

emoji_modifier_extract(emoji_sample)
```

---

emoji_modifier_remove *Remove Modifiers from Emojis*

Description

Remove Modifiers from Emojis

Usage

```
emoji_modifier_remove(x)
```

Arguments

x	Characters, vector of emojis.

Details

This function is vectorized. See emoji_modifiers for full list of modified emojis and their unmodified state.

Value

character vector, single emojis will be replaced with un-modified if possible.

Examples

```
waving_hands <- emojis$emoji[grepl("waving hand", emojis$name)]
waving_hands

emoji_modifier_remove(waving_hands)

set.seed(1234)
emoji_sample <- sample(emojis$emoji, 10)
emoji_sample

emoji_modifier_remove(emoji_sample)
```

---

emoji_name *Emoji Names*

Description

This vector is a named vector of emojis, where then names are unique descriptive identifiers for the emojis. This vector is well suited to be used as a tool to replace emojis with natural language descriptions.

Usage

```
emoji_name
```

Format

named character vector with 4698 elements

Details

Some emojis will appear multiple times since they have multiple names associated with them. Such as "grinning" and "grinning_face" leading to the same emoji.

Source

Unicode® Full Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/full-emoji-list.html

Unicode® Emoji Charts v16.0 - https://www.unicode.org/emoji/charts-16.0/emoji-list.html

Unicode® Emoji Ordering, v16.0 - https://www.unicode.org/emoji/charts/emoji-ordering.txt#' @source https://github.com/github/gemoji

https://github.com/muan/emojilib

See Also

emojis emoji_keyword

emoji_p *Summarise your p-values with emoji*

Description

Summarise your p-values with emoji

Usage

```
emoji_p(
  x,
  names = c("laughing", "joy", "grin", "smile", "thinking", "poop"),
  cutpoints = c(1e-05, 0.001, 0.01, 0.05, 0.1),
  legend = FALSE
)
```

Arguments

x	A vector of p-values.
names	A character vector, for each of the p-value cutoff points. The names are being passed to `emoji()`.
cutpoints	A numeric vector of cutpoints between emojis.
legend	Logical, denotes if the result should be returned with a legend.

Details

This function is vectorized. The input `cutpoints` must be 1 shorter than the `names` input. The input `cutpoints` should not include 0 or 1 and be in accending order.

Examples

```
set.seed(1234)
emoji_p(1)
emoji_p(0.1)
emoji_p(0.05)
emoji_p(0.01)
emoji_p(1e-6)

emoji_p(0.01, legend = TRUE)

emoji_p(rbeta(50, 2, 5))

emoji_p(
  runif(100, 0, 0.1),
  names = c("biceps", "hundred", "thumbs_down", "thumbs_up"),
  cutpoints = c(0.001, 0.01, 0.05)
)
```

emoji_replace *Replace emojis in a string*

Description

Vectorised over `string` and `replacement`

Usage

```
emoji_replace(string, replacement)

emoji_replace_all(string, replacement)
```

Arguments

`string`	Input vector
`replacement`	A character vector of replacements. Should either be of length 1 or the same length as `string`. See `stringr::str_replace()` for details

Value

A character vector

Examples

```
emoji_replace(emoji_name[1], "_emoji_")

string <- paste(c(letters[1:4], emoji_name[1:6]), collapse = " ")

emoji_replace_all(emoji_name[1:6], "_emoji_")
```

emoji_replace_name *Replace emojis in a string with name*

Description

Vectorised over `string`

Usage

```
emoji_replace_name(string)
```

Arguments

`string`	Input vector

Details

Each emoji is replaced with human readable string in the form `:name_of_emoji:`.

Value

A character vector

Examples

```
example <- c(
  paste0("This is an emoji; ", emoji("person_facepalming")),
  paste0("You can write slides in ", emoji("key"), emoji("musical_note"))
)

example

emoji_replace_name(example)
```

---

emoji_rx *A regular expression to catch all emojis*

Description

This regex will capture all `fully-qualified` and `minimally-qualified` emojis.

Usage

```
emoji_rx
```

Format

character vector

Source

https://www.unicode.org/reports/tr51/#emoji_data

emoji_subset *Keep strings containing an emoji, or find positions*

Description

Keep strings containing an emoji, or find positions

Usage

```
emoji_subset(string, negate = FALSE)

emoji_which(string, negate = FALSE)
```

Arguments

string	input vector
negate	If TRUE, inverts the resulting boolean vector.

Value

A character vector

See Also

stringr::str_subset()

Examples

```
string <- c(letters[1:4], emoji_name[1:6])

emoji_subset(string) == emoji_name[1:6]
emoji_subset(string, negate = TRUE)

emoji_which(string)
emoji_which(string, negate = TRUE)
```

flag *Insert Flag Emojis*

Description

Insert Flag Emojis

Usage

```
flag(name, return_key = FALSE)
```

Arguments

name	Character denoting the place of the flag. Set return_key = TRUE to get full list of allowed names.
return_key	Logical, set to TRUE to get full list of allowed names.

Details

This function is vectorized. The input is being normalized before matching which will hopefully lead to lower friction and easier matching. Punctuation is being removed and case is not taken into consideration when matching. You can run flag(return_key = TRUE) to get full list of allowed names.

Value

Character vector of emojis.

Examples

```
flag(c("Vietnam", "Greenland", "Estonia", "Denmark", "united states"))

flag(c("US Virgin Islands", "U.S. Virgin Islands", "u.s. virgin islands"))
```

---

keycap *Keycap emoji sequence*

Description

Keycap emoji sequence

Usage

```
keycap(x)
```

Arguments

x	character, must be a number between 0 and 10, "#", or "*".

Details

This function is vectorized.

Value

a keycap version of x

Examples

```
keycap(6)
keycap('#')

keycap(1:10)
```

`medal` *Insert medal emojis*

Description

Insert medal emojis

Usage

```
medal(place)
```

Arguments

`place`	Character denoting the place of the medal. See details for allowed names.

Details

This function is vectorized. There are a 1st, 2nd and 3rd place medals and allowed names are listed below. Note that matches are made without case.

- 1st place medal: "1", "1st", or "gold"
- 2nd place medal: "2", "2nd", or "silver"
- 3rd place medal: "3", "3rd", or "bronze"

Value

Character vector of emojis.

Examples

```
medal(1:3)

medal("gold")
medal("Gold")
```

moon *Insert Moon Phase Emoji*

Description

Insert Moon Phase Emoji

Usage

```
moon(date, day = day_in_synodic_cycle(date))
```

Arguments

date	a date
day	number of days since new moon

Details

This function is vectorized. If not supplied, day is calculated using the approximation of day_in_synodic_cycle, i.e the number of days since a known new moon modulo `29.530588853` days.

Value

a moon emoji

Examples

```
moon(Sys.Date())

january <- as.Date("2021-01-01") + 0:30
moon(january)
```

shape *Insert Arrow emojis*

Description

Insert Arrow emojis

Usage

```
shape(color, type)
```

Arguments

color	Character, denoting the color of the shape. Must be one of "red", "orange", "yellow", "green", "blue", "purple", "brown", "black", "white".
type	Character, denoting the type of shape. Must be one of "heart", "circle", or "square.

Details

This function is vectorized.

#@return Character vector of emojis.

Examples

```
shape("yellow", "heart")

shape("yellow", c("heart", "circle", "square"))

shape(color = c("red", "orange", "yellow", "green", "blue",
                "purple", "brown", "black", "white"),
      type = "circle")

outer(
  c("red", "orange", "yellow", "green", "blue",
    "purple", "brown", "black", "white"),
  c("heart", "circle", "square"),
  shape
)
```

---

zoo *Random Animals*

Description

This function returns random animals emojis.

Usage

```
zoo(size, replace = FALSE)
```

Arguments

size	a non-negative integer giving the number of items to choose.
replace	should sampling be with replacement? Defaults to FALSE.

Value

Character vector of animal emojis.

Examples

```
set.seed(1234)

zoo(1)

zoo(10)
```

Index

∗ **datasets**
    emoji_keyword, 10
    emoji_modifiers, 12
    emoji_name, 14
    emoji_rx, 17
    emojis, 4

arrow, 2

clock, 3

day_in_synodic_cycle, *21*

emoji, 3
emoji_count, 5
emoji_detect, 6
emoji_extract, 7
emoji_extract_all *(*emoji_extract*)*, 7
emoji_find, 8
emoji_fix, 8
emoji_glue, 9
emoji_keyword, 10
emoji_locate, 10
emoji_locate_all *(*emoji_locate*)*, 10
emoji_match, 11
emoji_match_all *(*emoji_match*)*, 11
emoji_modifier_extract, 12
emoji_modifier_remove, 13
emoji_modifiers, 12, *13*
emoji_name, 14
emoji_p, 15
emoji_replace, 16
emoji_replace_all *(*emoji_replace*)*, 16
emoji_replace_name, 16
emoji_rx, 17
emoji_subset, 18
emoji_which *(*emoji_subset*)*, 18
emojis, 4
emptyenv(), *9*

flag, 18

keycap, 19

medal, 20
moon, 21

shape, 21
stringr::str_count(), *5*
stringr::str_detect(), *6*
stringr::str_extract(), *7*
stringr::str_extract_all(), *7*
stringr::str_match, *11*
stringr::str_match(), *11*
stringr::str_replace(), *16*
stringr::str_subset(), *18*

zoo, 22