Package ‘emoji’

September 18, 2021

Title Data and Function to Work with Emojis
Version 0.2.0
Description Contains data about emojis with relevant metadata, and functions to work with emojis when they are in strings.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Depends R (>= 2.10)
BugReports https://github.com/EmilHvitfeldt/emoji/issues
Imports tibble, stringr, glue
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
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 2021-09-18 06:10:02 UTC

R topics documented:

arrow ................................................................. 2
clock ............................................................... 3
emoji ............................................................... 3
emojis .............................................................. 4
**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 NAs.

# @return Character vector of emojis.

**Examples**

arrow("up-down")

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

Description
Find a single emoji

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

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

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

Description

This data set is the heart of the emoji package. It contains various information regarding all the available emojis as of v13.1.

Usage

```r
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 meaning

- **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 v13.1
Unicode® Emoji Charts v13.1
Unicode® Emoji Ordering, v13.1
https://github.com/github/gemoji
https://github.com/muan/emojilib

See Also

emoji_name emoji_keyboard

---

**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**

```r
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**

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

**Arguments**

- `string` Input vector. Either a character vector, or something coercible to one.
- `negate` If TRUE, return non-matching elements.

**Value**

A logical vector

**See Also**

`stringr::str_detect()`

**Examples**

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

emoji_detect(string)
```
Extract emojis from a string

Description

vectorised over string

Usage

emoji_extract(string)

emoji_extract_all(string, simplify = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Input vector.</td>
</tr>
<tr>
<td>simplify</td>
<td>see stringr::str_extract_all()</td>
</tr>
</tbody>
</table>

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**
```r
call(emoji_find(keyword))
```

**Arguments**
- **keyword**: Character, Emoji keyword.

**Examples**
```r
call(emoji_find("happy"))
call(emoji_find("cat"))
call(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**
```r
call(emoji_fix(x))
```

**Arguments**
- **x**: Characters, vector of emojis.

**Details**
This function is vectorized.

**Value**
vector of fully-qualified emojis
Examples

```r
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 expressions string(s) to format. Multiple inputs are concatenated together before formatting. Named arguments are taken to be temporary variables available for substitution.

`.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

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

### Usage
```r
emoji_keyword
```

### Format
named list of characters with 6825 elements

### Source
- Unicode® Full Emoji Charts v13.1
- Unicode® Emoji Charts v13.1
- Unicode® Emoji Ordering, v13.1
- [https://github.com/github/gemoji](https://github.com/github/gemoji)
- [https://github.com/muan/emojilib](https://github.com/muan/emojilib)

### See Also
- emojis emoji_name

## emoji_locate

### Description
Locate the position of emojis in a string

### Usage
```r
emoji_locate(string) 
emoji_locate_all(string)
```

### Arguments
- **string** Input vector
emoji_match

Value

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

Examples

```r
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

```r
emoji_match(string)
emoji_match_all(string)
```

Arguments

- `string` Input vector

Value

see stringr::str_match()

See Also

stringr::str_match

Examples

```r
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)
```
### 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

```r
emoji_name
```

### Format

named character vector with 4538 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 v13.1
- Unicode® Emoji Charts v13.1
- Unicode® Emoji Ordering, v13.1
- https://github.com/github/gemoji
- https://github.com/muan/emojilib

### See Also

- emojis
- emoji_keyboard

---

### Description

Summarise your p-values with emoji
Usage

```r
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 ascending order.

Examples

```r
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("flexed biceps", "hundred points", "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

```r
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

```r
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_rx`  

A regular expression to catch all emojis

Description

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

Usage

```r
emoji_rx
```

Format

character vector

Source

[https://www.unicode.org/reports/tr51/#emoji_data](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, return non-matching elements.

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

<table>
<thead>
<tr>
<th>Keycap emoji sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>keycap</td>
</tr>
</tbody>
</table>

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

```r
table(6)
table('#')
table(1:10)
```

**medal**  
*Insert medal emojis*

Description

Insert medal emojis

Usage

```r
medal(place)
```

Arguments

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

Details

This function is vectorized. There are 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"

`#`return Character vector of emojis.

Examples

```r
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.

```r
@return Character vector of emojis.
```

**Examples**

```r
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
)`
Index

* datasets
  emoji_keyword, 10
  emoji_name, 12
  emoji_rx, 14
  emojis, 4

arrow, 2

clock, 3

day_in_synodic_cycle, 18

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_name, 12
emoji_p, 12
emoji_replace, 13
emoji_replace_all(emoji_replace), 13
emoji_rx, 14
emoji_subset, 15
emoji_which(emoji_subset), 15
emojis, 4
emptyenv(), 9

flag, 15

keycap, 16

medal, 17

moon, 18

shape, 18

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(), 14
stringr::str_subset(), 15