

# Package: emoji (via r-universe)

September 23, 2024

**Title** Data and Function to Work with Emojis

**Version** 15.0.0.9000

**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

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.1.9000

**Depends** R (>= 2.10)

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

**BugReports** <https://github.com/EmilHvitfeldt/emoji/issues>

**Imports** tibble, stringr, glue

**Suggests** covr, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Repository** <https://emilhvitfeldt.r-universe.dev>

**RemoteUrl** <https://github.com/emilhvitfeldt/emoji>

**RemoteRef** HEAD

**RemoteSha** 7456f007974b4e870afe27c1f74a741ba57a8952

## 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_rx . . . . .	16
emoji_subset . . . . .	17
flag . . . . .	18
keycap . . . . .	18
medal . . . . .	19
moon . . . . .	20
shape . . . . .	20
zoo . . . . .	21

## **Index** **23**

---

arrow	<i>Insert Arrow emojis</i>
-------	----------------------------

---

### **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"))
```

---

clock	<i>emoji version of time</i>
-------	------------------------------

---

**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	<i>Find a single emoji</i>
-------	----------------------------

---

**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 than 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 available emojis as of v15.0.

**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 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 v15.0](#)

[Unicode® Emoji Charts v15.0](#)

[Unicode® Emoji Ordering, v15.0](#)

<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	<i>Detect the presence or absence of emojis in a string</i>
--------------	---

---

**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, return non-matching elements.

**Value**

A logical vector

**See Also**

[stringr::str\\_detect\(\)](#)

**Examples**

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

---

emoji_extract	<i>Extract emojis from a string</i>
---------------	-------------------------------------

---

## Description

vectorised over string

## Usage

```
emoji_extract(string)
emoji_extract_all(string, simplify = FALSE)
```

## Arguments

string	Input vector.
simplify	see <a href="#">stringr::str_extract_all()</a>

## 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	<i>List all emoji with a given keyword</i>
------------	--

---

**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	<i>Turn emojis into qualified emojis</i>
-----------	--

---

**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**

<code>...</code>	[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.
<code>.envir</code>	[environment: <code>parent.frame()</code> ] Environment to evaluate each expression in. Expressions are evaluated from left to right. If <code>.x</code> is an environment, the expressions are evaluated in that environment and <code>.envir</code> is ignored. If <code>NULL</code> is passed, it is equivalent to <code>emptyenv()</code> .

**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	<i>Emoji Keywords</i>
---------------	-----------------------

---

**Description**

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

**Usage**

```
emoji_keyword
```

**Format**

named list of characters with 6878 elements

**Source**

[Unicode® Full Emoji Charts v15.0](#)

[Unicode® Emoji Charts v15.0](#)

[Unicode® Emoji Ordering, v15.0](#)

<https://github.com/github/gemoji>

<https://github.com/muan/emojilib>

**See Also**

emojis emoji\_name

---

emoji_locate	<i>Locate the position of emojis in a string</i>
--------------	--

---

**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	<i>Extract matched emojis from a string</i>
-------------	---

---

**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	<i>Emoji Modifiers</i>
-----------------	------------------------

---

### 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 v15.0

Unicode® Emoji Charts v15.0

Unicode® Emoji Ordering, v15.0

<https://github.com/github/gemoji>

<https://github.com/muan/emojilib>

### See Also

emojis emoji\_name

---

emoji_modifier_extract	<i>Extract Modifiers from Emojis</i>
------------------------	--------------------------------------

---

### 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$emoji[grep("waving hand", emojis$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	<i>Emoji Names</i>
------------	--------------------

---

### Description

This vector is a named vector of emojis, where the 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 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 v15.0](#)  
[Unicode® Emoji Charts v15.0](#)  
[Unicode® Emoji Ordering, v15.0](#)  
<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**

<code>x</code>	A vector of p-values.
<code>names</code>	A character vector, for each of the p-value cutoff points. The names are being passed to <code>emoji()</code> .
<code>cutpoints</code>	A numeric vector of cutpoints between emojis.
<code>legend</code>	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**

```
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	<i>Replace emojis in a string</i>
---------------	-----------------------------------

---

**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 <a href="#">stringr::str_replace()</a> 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_rx	<i>A regular expression to catch all emojis</i>
----------	---

---

**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](https://www.unicode.org/reports/tr51/#emoji_data)

---

emoji_subset	<i>Keep strings containing an emoji, or find positions</i>
--------------	--

---

**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	<i>Insert Flag Emojis</i>
------	---------------------------

---

**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	<i>Keycap emoji sequence</i>
--------	------------------------------

---

**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"

#@return Character vector of emojis.

**Examples**

```
medal(1:3)

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

---

moon	<i>Insert Moon Phase Emoji</i>
------	--------------------------------

---

**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	<i>Insert Arrow emojis</i>
-------	----------------------------

---

**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, 16
- emojis, 4

arrow, 2

clock, 3

day\_in\_synodic\_cycle, 20

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\_rx, 16

emoji\_subset, 17

emoji\_which (emoji\_subset), 17

emojis, 4

emptyenv(), 9

flag, 18

keycap, 18

medal, 19

moon, 20

shape, 20

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(), 17

zoo, 21