Package ‘prettyunits’

September 24, 2023

Title Pretty, Human Readable Formatting of Quantities

Version 1.2.0

Description Pretty, human readable formatting of quantities.
   Time intervals: '1337000' -> '15d 11h 23m 20s'.
   Vague time intervals: '2674000' -> 'about a month ago'.
   Bytes: '1337' -> '1.34 kB'.
   Rounding: '99' with 3 significant digits -> '99.0'
   p-values: '0.00001' -> '<0.0001'.
   Colors:,'#FF0000' -> 'red'.
   Quantities: '1239437' -> '1.24 M'.

License MIT + file LICENSE

URL https://github.com/r-lib/prettyunits

BugReports https://github.com/r-lib/prettyunits/issues

Depends R(>= 2.10)

Suggests codetools, covr, testthat

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Gabor Csardi [aut, cre],
   Bill Denney [ctb] (<https://orcid.org/0000-0002-5759-428X>),
   Christophe Regouby [ctb]

Maintainer Gabor Csardi <csardi.gabor@gmail.com>

Repository CRAN

Date/Publication 2023-09-24 21:10:02 UTC

R topics documented:

 prettyunits .................................................. 2
 pretty_bytes .................................................. 3
 pretty_color .................................................. 4
Description

Render quantities with a pretty, human-readable formatting.

- Time intervals: '1337000' -> '15d 11h 23m 20s'.
- Vague time intervals: '2674000' -> 'about a month ago'.
- Bytes: '1337' -> '1.34 kB'.
- p-values: '0.00001' -> '<0.0001'.
- Colors: '#FF0000' -> 'red'.
- Quantities: '1239437' -> '1.24 M'.

Author(s)

Maintainer: Gabor Csardi <csardi.gabor@gmail.com>

Other contributors:

- Bill Denney <wenney@humanpredictions.com> (ORCID) [contributor]
- Christophe Regouby <christophe.regouby@free.fr> [contributor]

See Also

Useful links:

- https://github.com/r-lib/prettyunits
- Report bugs at https://github.com/r-lib/prettyunits/issues
pretty_bytes

Bytes in a human readable string

Description

Use pretty_bytes() to format bytes. compute_bytes() is the underlying engine that may be useful for custom formatting.

Usage

pretty_bytes(bytes, style = c("default", "nopad", "6"))
compute_bytes(bytes, smallest_unit = "B")

Arguments

bytes Numeric vector, number of bytes.
style Formatting style:
  • "default" is the original pretty_bytes formatting, and it always pads the output, so that all vector elements are of the same width,
  • "nopad" is similar, but does not pad the output,
  • "6" always uses 6 characters, The "6" style is useful if it is important that the output always has the same width (number of characters), e.g. in progress bars. See some examples below.
smallest_unit A character scalar, the smallest unit to use.

Value

Character vector, the formatted sizes. For compute_bytes, a data frame with columns amount, unit, negative.

Examples

bytes <- c(1337, 133337, 13333337, 1333333337, 133333333337)
pretty_bytes(bytes)
pretty_bytes(bytes, style = "nopad")
pretty_bytes(bytes, style = "6")
pretty_color

*Color definition (like RGB) to a name*

**Description**

Color definition (like RGB) to a name

**Usage**

```
pretty_color(color)
pretty_colour(color)
```

**Arguments**

- `color` A scalar color that is usable as an input to `col2rgb()` (assumed to be in the sRGB color space).

**Value**

A character string that is the closest named colors to the input color. The output will have an attribute of alternate color names (named "alt").

---

pretty_dt

*Pretty formatting of time intervals (difftime objects)*

**Description**

Pretty formatting of time intervals (difftime objects)

**Usage**

```
pretty_dt(dt, compact = FALSE)
```

**Arguments**

- `dt` A `difftime` object, a vector of time differences.
- `compact` If true, then only the first non-zero unit is used. See examples below.

**Value**

Character vector of formatted time intervals.

**See Also**

Other time: `pretty_ms()`, `pretty_sec()`
pretty_ms

Examples

```r
pretty_dt(as.difftime(1000, units = "secs"))
pretty_dt(as.difftime(0, units = "secs"))
```

---

**pretty_ms**  
Pretty formatting of milliseconds

**Description**

Pretty formatting of milliseconds

**Usage**

```r
pretty_ms(ms, compact = FALSE)
```

**Arguments**

- `ms` Numeric vector of milliseconds
- `compact` If true, then only the first non-zero unit is used. See examples below.

**Value**

Character vector of formatted time intervals.

**See Also**

Other time: `pretty_dt()`, `pretty_sec()`

**Examples**

```r
pretty_ms(c(1337, 13370, 133700, 1337000, 1337000000))
pretty_ms(c(1337, 13370, 133700, 1337000, 1337000000), compact = TRUE)
```
**pretty_num**

**Linear quantities in a human readable string**

**Description**

Use `pretty_num()` to format numbers. `compute_num()` is the underlying engine that may be useful for custom formatting.

**Usage**

```r
pretty_num(number, style = c("default", "nopad", "6"))
compute_num(number, smallest_prefix = "y")
```

**Arguments**

- **number**: Numeric vector, number related to a linear quantity.
- **style**: Formatting style:
  - "default" is the original `pretty_num` formatting, and it always pads the output, so that all vector elements are of the same width,
  - "nopad" is similar, but does not pad the output,
  - "6" always uses 6 characters. The "6" style is useful if it is important that the output always has the same width (number of characters), e.g. in progress bars. See some examples below.
- **smallest_prefix**: A character scalar, the smallest prefix to use.

**Value**

Character vector, the formatted sizes. For `compute_num`, a data frame with columns `amount`, `prefix`, `negative`.

**Examples**

```r
numbers <- c(1337, 1.3333e-5, 13333337, 1333333337, 133333333337)
pretty_num(numbers)
pretty_num(numbers, style = "nopad")
pretty_num(numbers, style = "6")
```
pretty_p_value  

p-values in a human-readable string

Description

p-values in a human-readable string

Usage

pretty_p_value(x, minval = 1e-04)

Arguments

x  
A numeric vector.

minval  
The minimum p-value to show (lower values will show as paste0("<", minval)).

Value

A character vector of p-value representations.

Examples

pretty_p_value(c(1, 0, NA, 0.01, 0.0000001))
pretty_p_value(c(1, 0, NA, 0.01, 0.0000001), minval = 0.05)

pretty_round  

Round a value to a defined number of digits printing out trailing zeros, if applicable

Description

Round a value to a defined number of digits printing out trailing zeros, if applicable

Usage

pretty_round(x, digits = 0, sci_range = Inf, sci_sep = "e")

Arguments

x  
The number to round.

digits  
integer indicating the number of decimal places.

sci_range  
See help for pretty_signif() (and you likely want to round with pretty_signif() if you want to use this argument).

sci_sep  
The separator to use for scientific notation strings (typically this will be either "e" or "x10^n" for computer- or human-readable output).
Details

Values that are not standard numbers like Inf, NA, and NaN are returned as "Inf", "NA", and "NaN".

Value

A string with the value.

See Also

round(), pretty_signif().

pretty_sec

Pretty formatting of seconds

Description

Pretty formatting of seconds

Usage

pretty_sec(sec, compact = FALSE)

Arguments

sec Numeric vector of seconds.
compact If true, then only the first non-zero unit is used. See examples below.

Value

Character vector of formatted time intervals.

See Also

Other time: pretty_dt(), pretty_ms()

Examples

pretty_sec(c(1337, 13370, 133700, 1337000, 13370000))

pretty_sec(c(1337, 13370, 133700, 1337000, 13370000), compact = TRUE)
pretty_signif

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pretty_signif</td>
<td>Round a value to a defined number of significant digits printing out trailing zeros, if applicable</td>
</tr>
</tbody>
</table>

**Description**

Round a value to a defined number of significant digits printing out trailing zeros, if applicable.

**Usage**

```r
pretty_signif(x, digits = 6, sci_range = 6, sci_sep = "e")
```

**Arguments**

- `x` The number to round.
- `digits` integer indicating the number of significant digits.
- `sci_range` integer (or `Inf`) indicating when to switch to scientific notation instead of floating point. Zero indicates always use scientific; `Inf` indicates to never use scientific notation; otherwise, scientific notation is used when `abs(log10(x)) > sci_range`.
- `sci_sep` The separator to use for scientific notation strings (typically this will be either "e" or "x10^" for computer- or human-readable output).

**Details**

Values that are not standard numbers like `Inf`, `NA`, and `NaN` are returned as "Inf", "NA", and NaN.

**Value**

A string with the value.

**See Also**

- `signif()`, `pretty_round()`.

---

time_ago

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_ago</td>
<td>Human readable format of the time interval since a time point</td>
</tr>
</tbody>
</table>

**Description**

It calls `vague_dt` to do the actual formatting.

**Usage**

```r
time_ago(date, format = c("default", "short", "terse"))
```
vague_dt

Arguments

date  Date(s), as.POSIXct will be called on them.
format  Format, currently available formats are: ‘default’, ‘short’, ‘terse’. See examples below.

Value

Character vector of the formatted time intervals.

Examples

now <- Sys.time()

    time_ago(now)
    time_ago(now - as.difftime(30, units = "secs"))
    time_ago(now - as.difftime(14, units = "mins"))
    time_ago(now - as.difftime(5, units = "hours"))
    time_ago(now - as.difftime(25, units = "hours"))
    time_ago(now - as.difftime(5, units = "days"))
    time_ago(now - as.difftime(30, units = "days"))
    time_ago(now - as.difftime(365, units = "days"))
    time_ago(now - as.difftime(365 * 10, units = "days"))

## Short format
    time_ago(format = "short", now)
    time_ago(format = "short", now - as.difftime(30, units = "secs"))
    time_ago(format = "short", now - as.difftime(14, units = "mins"))
    time_ago(format = "short", now - as.difftime(5, units = "hours"))
    time_ago(format = "short", now - as.difftime(25, units = "hours"))
    time_ago(format = "short", now - as.difftime(5, units = "days"))
    time_ago(format = "short", now - as.difftime(30, units = "days"))
    time_ago(format = "short", now - as.difftime(365, units = "days"))
    time_ago(format = "short", now - as.difftime(365 * 10, units = "days"))

## Even shorter, terse format, (almost always) exactly 3 characters wide
    time_ago(format = "terse", now)
    time_ago(format = "terse", now - as.difftime(30, units = "secs"))
    time_ago(format = "terse", now - as.difftime(14, units = "mins"))
    time_ago(format = "terse", now - as.difftime(5, units = "hours"))
    time_ago(format = "terse", now - as.difftime(25, units = "hours"))
    time_ago(format = "terse", now - as.difftime(5, units = "days"))
    time_ago(format = "terse", now - as.difftime(30, units = "days"))
    time_ago(format = "terse", now - as.difftime(365, units = "days"))
    time_ago(format = "terse", now - as.difftime(365 * 10, units = "days"))

vague_dt  

Human readable format of a time interval
vague_dt

Description

Human readable format of a time interval

Usage

vague_dt(dt, format = c("default", "short", "terse"))

Arguments

dt
A difftime object, the time interval(s).

format
Format, currently available formats are: ‘default’, ‘short’, ‘terse’. See examples below.

Value

Character vector of the formatted time intervals.

Examples

vague_dt(as.difftime(30, units = "secs"))
vague_dt(as.difftime(14, units = "mins"))
vague_dt(as.difftime(5, units = "hours"))
vague_dt(as.difftime(25, units = "hours"))
vague_dt(as.difftime(5, units = "days"))
vague_dt(as.difftime(30, units = "days"))
vague_dt(as.difftime(365, units = "days"))
vague_dt(as.difftime(365 * 10, units = "days"))

## Short format
vague_dt(format = "short", as.difftime(30, units = "secs"))
vague_dt(format = "short", as.difftime(14, units = "mins"))
vague_dt(format = "short", as.difftime(5, units = "hours"))
vague_dt(format = "short", as.difftime(25, units = "hours"))
vague_dt(format = "short", as.difftime(5, units = "days"))
vague_dt(format = "short", as.difftime(30, units = "days"))
vague_dt(format = "short", as.difftime(365, units = "days"))
vague_dt(format = "short", as.difftime(365 * 10, units = "days"))

## Even shorter, terse format, (almost always) exactly 3 characters wide
vague_dt(format = "terse", as.difftime(30, units = "secs"))
vague_dt(format = "terse", as.difftime(14, units = "mins"))
vague_dt(format = "terse", as.difftime(5, units = "hours"))
vague_dt(format = "terse", as.difftime(25, units = "hours"))
vague_dt(format = "terse", as.difftime(5, units = "days"))
vague_dt(format = "terse", as.difftime(30, units = "days"))
vague_dt(format = "terse", as.difftime(365, units = "days"))
vague_dt(format = "terse", as.difftime(365 * 10, units = "days"))
Index

* time
  pretty_dt, 4
  pretty_ms, 5
  pretty_sec, 8
  compute_bytes (pretty_bytes), 3
  compute_num (pretty_num), 6

  pretty_bytes, 3
  pretty_color, 4
  prettyColour (pretty_color), 4
  pretty_dt, 4, 5, 8
  pretty_ms, 4, 5, 8
  pretty_num, 6
  pretty_p_value, 7
  pretty_round, 7
  pretty_round(), 9
  pretty_sec, 4, 5, 8
  pretty_signif, 9
  pretty_signif(), 7, 8
  prettyunits, 2
  prettyunits-package (prettyunits), 2

  round(), 8
  signif(), 9
  time_ago, 9
  vague_dt, 9, 10