

# Package: prettyunits (via r-universe)

June 15, 2024

**Title** Pretty, Human Readable Formatting of Quantities

**Version** 1.2.0.9000

**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>,  
<http://r-lib.github.io/prettyunits/>

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

**Depends** R(>= 2.10)

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

**Config/Needs/website** tidyverse/tidytemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

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

**RemoteUrl** <https://github.com/r-lib/prettyunits>

**RemoteRef** HEAD

**RemoteSha** 93bee517697590b8d39dbd955c1263ef114fe713

## Contents

pretty_bytes . . . . .	2
pretty_color . . . . .	3
pretty_dt . . . . .	3

pretty_ms . . . . .	4
pretty_num . . . . .	5
pretty_p_value . . . . .	6
pretty_round . . . . .	6
pretty_sec . . . . .	7
pretty_signif . . . . .	8
time_ago . . . . .	8
vague_dt . . . . .	9

## Index 11

---

pretty_bytes	<i>Bytes in a human readable string</i>
--------------	---

---

### 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

<code>bytes</code>	Numeric vector, number of bytes.
<code>style</code>	Formatting style: <ul style="list-style-type: none"> <li>• "default" is the original <code>pretty_bytes</code> formatting, and it always pads the output, so that all vector elements are of the same width,</li> <li>• "nopad" is similar, but does not pad the output,</li> <li>• "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.</li> </ul>
<code>smallest_unit</code>	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	<i>Color definition (like RGB) to a name</i>
--------------	--

---

**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 <code>col2rgb()</code> (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	<i>Pretty formatting of time intervals (difftime objects)</i>
-----------	---

---

**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\(\)](#)

## Examples

```
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

```
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

```
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

```
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

```
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^" 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	<i>Round a value to a defined number of significant digits printing out trailing zeros, if applicable</i>
---------------	---

---

**Description**

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

**Usage**

```
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 $\text{abs}(\log_{10}(x)) > \text{sci\_range}$ .
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	<i>Human readable format of the time interval since a time point</i>
----------	--

---

**Description**

It calls [vague\\_dt](#) to do the actual formatting.

**Usage**

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



**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"))
```

**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, [3](#)
- pretty\_ms, [4](#)
- pretty\_sec, [7](#)

- compute\_bytes (pretty\_bytes), [2](#)
- compute\_num (pretty\_num), [5](#)

- pretty\_bytes, [2](#)
- pretty\_color, [3](#)
- pretty\_colour (pretty\_color), [3](#)
- pretty\_dt, [3](#), [4](#), [7](#)
- pretty\_ms, [3](#), [4](#), [7](#)
- pretty\_num, [5](#)
- pretty\_p\_value, [6](#)
- pretty\_round, [6](#)
- pretty\_round(), [8](#)
- pretty\_sec, [3](#), [4](#), [7](#)
- pretty\_signif, [8](#)
- pretty\_signif(), [6](#), [7](#)

- round(), [7](#)

- signif(), [8](#)

- time\_ago, [8](#)

- vague\_dt, [8](#), [9](#)