

# Package ‘formatdown’

May 8, 2026

**Title** Formatting Numbers in 'rmarkdown' Documents

**Version** 0.2.0

**Language** en-US

**Description** Provides a small set of tools for formatting numbers in R-markdown documents. Convert a numerical vector to character strings in power-of-ten form, decimal form, or measurement-units form; convert symbols for chemical isotopes from hyphenated form to nuclear form; all are math-delimited for rendering as inline equations. Can also convert text into math-delimited text to match the font face and size of math-delimited numbers. Useful for rendering single numbers in inline R code chunks and for rendering columns in tables.

**Depends** R (>= 3.5.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** TRUE

**LazyDataCompression** bzip2

**RoxygenNote** 7.3.3

**Imports** checkmate, data.table, settings, units, wrapr

**Suggests** knitr, rmarkdown, tinytest

**VignetteBuilder** knitr

**URL** <https://github.com/graphdr/formatdown/>,  
<https://graphdr.github.io/formatdown/>,  
<https://CRAN.R-project.org/package=formatdown>

**BugReports** <https://github.com/graphdr/formatdown/issues>

**Config/Needs/website** rmarkdown

**NeedsCompilation** no

**Author** Richard Layton [aut, cre]

**Maintainer** Richard Layton <graphdoctor@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-05-02 23:10:03 UTC

## Contents

air_meas . . . . .	2
atmos . . . . .	3
element_set . . . . .	3
formatdown_options . . . . .	4
format_dcml . . . . .	7
format_engr . . . . .	9
format_nucl . . . . .	12
format_numbers . . . . .	14
format_sci . . . . .	17
format_text . . . . .	20
metals . . . . .	22
water . . . . .	22
<b>Index</b>	<b>24</b>

---

air_meas	<i>Air density measurements</i>
----------	---------------------------------

---

### Description

Table of air properties at room temperature and pressure, simulating multiple measurements at approximately steady state,

### Usage

```
data(air_meas, package = "formatdown")
```

### Format

Classes data.table and data.frame with 5 observations of 7 variables:

**date** "Date" class format "YYYY-MM-DD".

**trial** Character, label "a" through "e".

**humid** Factor, humidity, "low", "med", or "high."

**temp** Numeric, measured temperature (K).

**pres** Numeric, measured atmospheric pressure (Pa).

**sp\_gas** Numeric, specific gas constant in mass form  $R_{sp}$ , ideal gas reference value, ( $\text{J kg}^{-1}\text{K}^{-1}$ ).

**dens** Numeric, calculated air density  $\rho = pR_{sp}^{-1}T^{-1}$  ( $\text{kg m}^{-3}$ ).

---

atmos	<i>Properties of standard atmosphere</i>
-------	--

---

**Description**

Table of atmospheric properties as a function of altitude, sea level to 1000 km.

**Usage**

```
data(atmos, package = "formatdown")
```

**Format**

Classes data.table and data.frame with 20 observations of 5 variables:

**alt** Numeric, altitude (km)  
**temp** Numeric, air temperature (K)  
**pres** Numeric, atmospheric pressure (Pa)  
**dens** Numeric, air density ( $\text{kg m}^{-3}$ )  
**sound** Numeric, speed of sound (m/s)

**Source**

*Marks' Standard Handbook for Mechanical Engineers 9/e* (1987) E.A. Avallone and T. Baumeister (ed.), "Table 4.2.2 International Standard Atmosphere", pp. 4-38, McGraw-Hill, NY.

---

element_set	<i>Chemical elements</i>
-------------	--------------------------

---

**Description**

Table of 118 chemical elements with name, symbol, atomic number, and mass numbers. Elements with more than one possible mass number have separate rows for each isotope.

**Usage**

```
data(element_set, package = "formatdown")
```

**Format**

Classes data.table and data.frame with 354 observations of 4 variables:

**element** Character, element name, hydrogen (atomic number 1) through oganesson (atomic number 118).  
**symbol** Character, element symbol.  
**atomic\_number** Character, 1-118.  
**mass\_number** Character, 1-294.

**Details**

In "AZE" or "AZX" notation, "A" is the mass number, "Z" is the atomic number, and "E" or "X" is the element symbol.

---

formatdown\_options      *Get and set function arguments via options*

---

**Description**

Changes the default values of function arguments which affect the markup and appearance of formatdown results.

**Usage**

```
formatdown_options(..., reset = FALSE)
```

**Arguments**

...	One or more name = value pairs to set values; or one or more quoted option names to get values.
reset	Logical vector of length 1; if TRUE, reset all options to their default values.

**Details**

Global options are provided for arguments that users would likely prefer to set once in a document instead of repeating in every function call. For example, some users prefer a comma decimal marker (",") throughout a document.

Globally-set arguments can be overridden locally by assigning them in a function call.

The arguments that can be set with this function are as follows:

- **delim**: Character, length 1 or 2, to define the left and right math markup delimiters. The default setting, `delim = "$"`, produces left and right delimiters `$.$.` The alternate built-in setting, `delim = "\\( "`, produces left and right delimiters `\\( ... \\)`. Custom delimiters can be assigned in a vector of length 2 with left and right delimiter symbols, e.g., `c("\\[ ", "\\]")`. Special characters typically must be escaped.
- **size**: Character, length 1, to assign a font size. If not empty, adds a font size macro to the markup inside the math delimiters. Possible values are "scriptsize", "small", "normalsize", "large", and "huge". One may also assign the equivalent LaTeX-style markup itself, e.g., `"\\scriptsize"`, `"\\small"`, etc. Default is NULL.
- **decimal\_mark**: Character, length 1, to assign the decimal marker. Possible values are a period "." (default) or a comma ", ". Passed to `formatC(decimal.mark)`.
- **big\_mark**: Character, length 1, used as the mark between every `big_interval` number of digits to the left of the decimal marker to improve readability. Possible values are empty "" (default) or "thin" to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself `"\\ \\ , "`. Passed to `formatC(big.mark)`.

- `big_interval`: Integer, length 1, that defines the number of digits (default 3) in groups separated by `big_mark`. Passed to `formatC(big_interval)`.
- `small_mark`: Character, length 1, used as the mark between every `small_interval` number of digits to the right of the decimal marker to improve readability. Possible values are empty "" (default) or "thin" to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself "\\ ", ". Passed to `formatC(small_mark)`.
- `small_interval`: Integer, length 1, that defines the number of digits (default 5) in groups separated by `small_mark`. Passed to `formatC(small_interval)`.
- `whitespace`: Character, length 1, to define the LaTeX-style math-mode macro to preserve a horizontal space between words of text or between physical-unit abbreviations when formatting numbers of class "units". Default is "\\ ". Alternatives include "\\:" or "\\>".
- `multiply_mark`: Character, length 1, to define the multiplication symbol in power of ten notation. Possible values are "\\times" (default) or a half-high dot "\\cdot" which is often used when the decimal mark is a comma.
- `Z`: Logical, length 1, default FALSE, used to determine whether or not the atomic number subscript is used in formatting nuclear notation of isotopes.
- `warn`: Logical, length 1, default TRUE, used to determine whether or not a warning is issued when an input for a chemical element in hyphenated notation contains an error in the element symbol or the mass number of the isotope.

### Value

Nothing; used for its side-effect.

### Examples

```
# Show all options
formatdown_options()

# Store existing settings, including any changes made by the user
old_settings <- formatdown_options()

# View one option
formatdown_options()$delim

# View multiple options
formatdown_options("size", "delim")

# Change options
formatdown_options(size = "small", delim = "\\(")
formatdown_options("size", "delim")

# Reset to default values
formatdown_options(reset = TRUE)
formatdown_options("size", "delim")

# Reset options to those before this example was run
do.call(formatdown_options, old_settings)
```

```

# Option effects

# delim
x <- 101300
format_dcml(x)
# equivalent to
format_dcml(x, delim = c("$", "$"))
# built-in alternate
format_dcml(x, delim = "\\(")
# equivalent to
format_dcml(x, delim = c("\\(", "\\)"))

# size
format_dcml(x, size = "small")
# equivalent to
format_dcml(x, size = "\\small")
# other possible values
format_dcml(x, size = "scriptsize")
format_dcml(x, size = "large")
format_dcml(x, size = "huge")
# default NULL
format_dcml(x, size = NULL)
# renders equivalent to
format_dcml(x, size = "normalsize")

# decimal_mark
y <- 6.02214076E+10
format_sci(y, 5, decimal_mark = ".")
format_sci(y, 5, decimal_mark = ",")

# big_mark
format_dcml(y, 9)
format_dcml(y, 9, big_mark = "thin")
# equivalent to
format_dcml(y, 9, big_mark = "\\|",)

# big_interval
format_dcml(y, 9, big_mark = "thin", big_interval = 3)
format_dcml(y, 9, big_mark = "thin", big_interval = 5)

# small_mark
z <- 1.602176634e-8
format_sci(z, 10)
format_sci(z, 10, small_mark = "thin")
format_sci(z, 10, small_mark = "\\|",)
format_engr(z, 10, small_mark = "thin")

# small_interval
format_sci(z, 10, small_mark = "thin", small_interval = 3)
format_sci(z, 10, small_mark = "thin", small_interval = 5)
format_engr(z, 10, small_mark = "thin", small_interval = 5)

# whitespace in text

```

```

p <- "Hello world!"
format_text(p)
# equivalent to
format_text(p, whitespace = "\\ \\ ")
# alternates
format_text(p, whitespace = "\\ \\:")
format_text(p, whitespace = "\\ \\>")

# whitespace in physical units expression
x <- pi
units(x) <- "m/s"
format_dcml(x)
# equivalent to
format_dcml(x, whitespace = "\\ \\ ")

# multiply_mark
x <- 101300
format_engr(x, decimal_mark = ".", multiply_mark = "\\times")
format_engr(x, decimal_mark = ",", multiply_mark = "\\cdot")

```

---

format\_dcml

*Format numbers in decimal notation*


---

## Description

Convert a numeric vector to a character vector in which the numbers are formatted in decimal form and delimited for rendering as inline equations in an R markdown document.

## Usage

```

format_dcml(
  x,
  digits = 4,
  ...,
  delim = formatdown_options("delim"),
  size = formatdown_options("size"),
  decimal_mark = formatdown_options("decimal_mark"),
  big_mark = formatdown_options("big_mark"),
  big_interval = formatdown_options("big_interval"),
  small_mark = formatdown_options("small_mark"),
  small_interval = formatdown_options("small_interval"),
  whitespace = formatdown_options("whitespace")
)

```

## Arguments

**x** Number or numbers to be formatted. Can be a single number, a vector, or a column of a data frame.

<code>digits</code>	Integer from 1 through 20 that controls the number of significant digits in printed numeric values. Passed to <code>signif()</code> . Default is 4.
<code>...</code>	Not used for values; forces subsequent arguments to be referable only by name.
<code>delim</code>	Character, length 1 or 2, to define the left and right math markup delimiters. The default setting, <code>delim = "\$"</code> , produces left and right delimiters $$. . $. The alternate built-in setting, delim = "\\( ", produces left and right delimiters \\( . . \\). Custom delimiters can be assigned in a vector of length 2 with left and right delimiter symbols, e.g., c("\\[ ", "\\]"). Special characters typically must be escaped.$
<code>size</code>	Character, length 1, to assign a font size. If not empty, adds a font size macro to the markup inside the math delimiters. Possible values are <code>"scriptsize"</code> , <code>"small"</code> , <code>"normalsize"</code> , <code>"large"</code> , and <code>"huge"</code> . One may also assign the equivalent LaTeX-style markup itself, e.g., <code>"\\scriptsize"</code> , <code>"\\small"</code> , etc. Default is <code>NULL</code> .
<code>decimal_mark</code>	Character, length 1, to assign the decimal marker. Possible values are a period <code>."</code> (default) or a comma <code>","</code> . Passed to <code>formatC(decimal.mark)</code> .
<code>big_mark</code>	Character, length 1, used as the mark between every <code>big_interval</code> number of digits to the left of the decimal marker to improve readability. Possible values are empty <code>""</code> (default) or <code>"thin"</code> to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself <code>"\\ \\ ,"</code> . Passed to <code>formatC(big.mark)</code> .
<code>big_interval</code>	Integer, length 1, that defines the number of digits (default 3) in groups separated by <code>big_mark</code> . Passed to <code>formatC(big.interval)</code> .
<code>small_mark</code>	Character, length 1, used as the mark between every <code>small_interval</code> number of digits to the right of the decimal marker to improve readability. Possible values are empty <code>""</code> (default) or <code>"thin"</code> to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself <code>"\\ \\ ,"</code> . Passed to <code>formatC(small.mark)</code> .
<code>small_interval</code>	Integer, length 1, that defines the number of digits (default 5) in groups separated by <code>small_mark</code> . Passed to <code>formatC(small.interval)</code> .
<code>whitespace</code>	Character, length 1, to define the LaTeX-style math-mode macro to preserve a horizontal space between words of text or between physical-unit abbreviations when formatting numbers of class <code>"units"</code> . Default is <code>"\\ \\ "</code> . Alternatives include <code>"\\ \\ :"</code> or <code>"\\ \\ &gt;"</code> .

### Details

`format_dcml()` is a wrapper for the more general function `format_numbers()`. Where defaults are defined by `formatdown_options()`, users may reassign the arguments locally in the function call or globally using `formatdown_options()`.

Arguments after the dots (`...`) must be referred to by name.

### Value

A character vector in which numbers are formatted in decimal form and delimited for rendering as inline equations in an R markdown document.

**See Also**

Other format\_\*: [format\\_engr\(\)](#), [format\\_nucl\(\)](#), [format\\_numbers\(\)](#), [format\\_sci\(\)](#), [format\\_text\(\)](#)

**Examples**

```
# input: single number
x <- 103400
format_dcml(x)

# input: units class
x <- 103400
units(x) <- "N m2 C-2"
format_dcml(x)

# input: vector
data("metals", package = "formatdown")
x <- metals$thrm_cond
format_dcml(x)

# significant digits
x <- 155.77
format_dcml(x, 2)
format_dcml(x, 3)
format_dcml(x, 4)

# input: data frame
x <- metals[, c("dens", "thrm_cond")]
as.data.frame(apply(x, 2, format_dcml, digits = 3))

# format_dcml() same as format_numbers(..., format = "dcml")
x <- 103400
format_dcml(x)
format_numbers(x, format = "dcml")
```

---

format\_engr

*Format engineering notation*

---

**Description**

Convert a numeric vector to a character vector in which the numbers are formatted in power-of-ten notation in engineering form and delimited for rendering as inline equations in an R markdown document.

**Usage**

```
format_engr(
  x,
  digits = 4,
  ...,
```

```

omit_power = c(-1, 2),
set_power = NULL,
delim = formatdown_options("delim"),
size = formatdown_options("size"),
decimal_mark = formatdown_options("decimal_mark"),
small_mark = formatdown_options("small_mark"),
small_interval = formatdown_options("small_interval"),
whitespace = formatdown_options("whitespace"),
multiply_mark = formatdown_options("multiply_mark")
)

```

### Arguments

<code>x</code>	Number or numbers to be formatted. Can be a single number, a vector, or a column of a data frame.
<code>digits</code>	Integer from 1 through 20 that controls the number of significant digits in printed numeric values. Passed to <code>signif()</code> . Default is 4.
<code>...</code>	Not used for values; forces subsequent arguments to be referable only by name.
<code>omit_power</code>	Numeric vector <code>c(p, q)</code> with $p \leq q$ , specifying the range of exponents over which power-of-ten notation is omitted in either scientific or engineering format. Default is <code>c(-1, 2)</code> . If a single value is assigned, i.e., <code>omit_power = p</code> , the argument is interpreted as <code>c(p, p)</code> . If <code>NULL</code> or <code>NA</code> , all elements are formatted in power-of-ten notation. Argument is overridden by specifying <code>set_power</code> or decimal notation.
<code>set_power</code>	Integer, length 1. Formats all values in <code>x</code> with the same power-of-ten exponent. Default <code>NULL</code> . Overrides <code>format</code> and <code>omit_power</code> arguments.
<code>delim</code>	Character, length 1 or 2, to define the left and right math markup delimiters. The default setting, <code>delim = "\$"</code> , produces left and right delimiters <code>\$. . \$.</code> . The alternate built-in setting, <code>delim = "\\("</code> , produces left and right delimiters <code>\\( . . \\)</code> . Custom delimiters can be assigned in a vector of length 2 with left and right delimiter symbols, e.g., <code>c("\\[", "\\]")</code> . Special characters typically must be escaped.
<code>size</code>	Character, length 1, to assign a font size. If not empty, adds a font size macro to the markup inside the math delimiters. Possible values are <code>"scriptsize"</code> , <code>"small"</code> , <code>"normalsize"</code> , <code>"large"</code> , and <code>"huge"</code> . One may also assign the equivalent LaTeX-style markup itself, e.g., <code>"\\scriptsize"</code> , <code>"\\small"</code> , etc. Default is <code>NULL</code> .
<code>decimal_mark</code>	Character, length 1, to assign the decimal marker. Possible values are a period <code>."</code> (default) or a comma <code>","</code> . Passed to <code>formatC(decimal.mark)</code> .
<code>small_mark</code>	Character, length 1, used as the mark between every <code>small_interval</code> number of digits to the right of the decimal marker to improve readability. Possible values are empty <code>""</code> (default) or <code>"thin"</code> to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself <code>"\\, "</code> . Passed to <code>formatC(small.mark)</code> .
<code>small_interval</code>	Integer, length 1, that defines the number of digits (default 5) in groups separated by <code>small_mark</code> . Passed to <code>formatC(small.interval)</code> .

whitespace	Character, length 1, to define the LaTeX-style math-mode macro to preserve a horizontal space between words of text or between physical-unit abbreviations when formatting numbers of class "units". Default is "\\ \\ ". Alternatives include "\\ \\: " or "\\ \\>".
multiply_mark	Character, length 1, to define the multiplication symbol in power of ten notation. Possible values are "\\times" (default) or a half-high dot "\\cdot" which is often used when the decimal mark is a comma.

## Details

In engineering notation, all exponents are multiples of three. `format_engr()` is a wrapper for the more general function `format_numbers()`. Where defaults are defined by `formatdown_options()`, users may reassign the arguments locally in the function call or globally using `formatdown_options()`.

Arguments after the dots (...) must be referred to by name.

## Value

A character vector in which numbers are formatted in power-of-ten notation in engineering form and delimited for rendering as inline equations in an R markdown document.

## See Also

Other `format_*`: [format\\_dcml\(\)](#), [format\\_nucl\(\)](#), [format\\_numbers\(\)](#), [format\\_sci\(\)](#), [format\\_text\(\)](#)

## Examples

```
# input: single number
x <- 6.0221E+23
format_engr(x)

# input: units class
x <- 103400
units(x) <- "N m2 C-2"
format_engr(x)

# input: vector
data("metals", package = "formatdown")
x <- metals$dens
format_engr(x)

# significant digits
x <- 9.75358e+5
format_engr(x, 2)
format_engr(x, 3)
format_engr(x, 4)

# input: data frame
x <- metals[, c("thrm_exp", "thrm_cond")]
as.data.frame(apply(x, 2, format_engr, digits = 3))

# format_engr() same as format_numbers(..., format = "engr")
```

```

x <- 6.0221E+23
format_engr(x)
format_numbers(x, format = "engr")

# omit_power
x <- 103400
format_engr(x, omit_power = c(-1, 2)) # default
format_engr(x, omit_power = c(-1, 5))
format_engr(x, omit_power = 5) # equivalent to omit_power = c(5, 5)

# omit_power = NULL, power-of-ten notation for all elements
x <- c(1.2, 103400)
format_engr(x)
format_engr(x, omit_power = NULL)

# omit_power applies to native exponent (before engr formatting)
x <- 103400
format_sci(x) # native exponent is 5
format_engr(x, omit_power = 5)

# omit_power applies to exponent after engr formatting
x <- 103400
format_engr(x) # engr exponent is 3
format_engr(x, omit_power = 3)

# set_power overrides default engineering exponent
x <- 103400
format_engr(x)
format_engr(x, set_power = 4)

# set_power overrides omit_power
x <- 103400
format_engr(x)
format_engr(x, omit_power = 3)
format_engr(x, omit_power = 3, set_power = 3)

```

---

format\_nucl

*Format isotopes in nuclear notation*


---

## Description

Convert chemical element or isotope from from hyphenated notation to nuclear notation.

## Usage

```

format_nucl(
  x,
  face = "plain",
  ...,
  Z = formatdown_options("Z"),

```

```
warn = formatdown_options("warn"),
delim = formatdown_options("delim")
)
```

### Arguments

x	Character. Hyphenated form of chemical elements or isotopes. Can be a single character string or a vector of strings. Must include a single hyphen between the element symbol and the mass number. For example, carbon 12 in hyphenated notation is the character string "C-12".
face	Font face. Determines the font face macro inside the math delimiters. Possible values are "plain" (default), "italic", "bold", "sans", or "mono". One may assign instead the corresponding LaTeX-style markup itself, e.g., <code>\mathrm</code> , <code>\mathit</code> , <code>\mathbf</code> , <code>\mathsf</code> , or <code>\mathtt</code> .
...	Not used for values; forces subsequent arguments to be referable only by name.
Z	T/F add the atomic number. For details, see the help page for <code>formatdown_options()</code> .
warn	T/F issue warning for incorrect input x. For details, see the help page for <code>formatdown_options()</code> .
delim	The math-delimiter. For details, see the help page for <code>formatdown_options()</code> .

### Details

We start with a character string in hyphenated form  $X-A$  where  $X$  is the chemical symbol of an element and  $A$  is its mass number. For example, carbon 12 would be written as the string "C-12".

Given a character scalar, vector, or data frame column of isotopes in this form, `format_nucl()` constructs the form " $\mathit{xx}^{A}X$ " where `\mathit{xx}` determines the font face: plain type is set by `\mathrm`; italic by `\mathit`; bold by `\mathbf`; sans serif by `\mathsf`; and monospace (typewriter text) by `\mathtt`. The result includes markup delimiters `...` for rendering (in an R markdown or Quarto markdown document) as an inline code chunk.

Unlike other functions in `formatdown`, `format_nucl()` does not support a size argument.

### Value

A character vector of isotopes in nuclear notation with elements delimited as inline math markup in plain, italic, sans serif, bold, or monospace font face.

### See Also

Other `format_*`: `format_dcml()`, `format_engr()`, `format_numbers()`, `format_sci()`, `format_text()`

### Examples

```
# default arguments
x <- "C-12"
format_nucl(x)

# add atomic number
format_nucl(x, Z = TRUE)
```

```

# vector of isotopes
x <- c("C-12", "Fe-54", "U-238")
format_nucl(x, Z = TRUE)

# turn off the warning when using symbols
x <- "X-A"
format_nucl(x, Z = TRUE, warn = FALSE)

# change the typeface
x <- "C-12"
format_nucl(x, face = "bold")

# obtain all possible mass numbers for an element
element_set[symbol == "Ca"]

```

---

format\_numbers

*Format numbers*


---

## Description

Convert a numeric vector to a character vector in which the numbers are formatted in power-of-ten notation in scientific or engineering form and delimited for rendering as inline equations in an R markdown document. Decimal numbers can be similarly formatted, without the power-of-ten notation.

## Usage

```

format_numbers(
  x,
  digits = 4,
  format = "engr",
  ...,
  omit_power = c(-1, 2),
  set_power = NULL,
  delim = formatdown_options("delim"),
  size = formatdown_options("size"),
  decimal_mark = formatdown_options("decimal_mark"),
  big_mark = formatdown_options("big_mark"),
  big_interval = formatdown_options("big_interval"),
  small_mark = formatdown_options("small_mark"),
  small_interval = formatdown_options("small_interval"),
  whitespace = formatdown_options("whitespace"),
  multiply_mark = formatdown_options("multiply_mark")
)

```

**Arguments**

x	Number or numbers to be formatted. Can be a single number, a vector, or a column of a data frame.
digits	Integer from 1 through 20 that controls the number of significant digits in printed numeric values. Passed to signif(). Default is 4.
format	Character, length 1, defines the type of notation. Possible values are "engr" (default) for engineering power-of-ten notation, "sci" for scientific power-of-ten notation, and "dcm1" for decimal notation.
...	Not used for values; forces subsequent arguments to be referable only by name.
omit_power	Numeric vector c(p, q) with p <= q, specifying the range of exponents over which power-of-ten notation is omitted in either scientific or engineering format. Default is c(-1, 2). If a single value is assigned, i.e., omit_power = p, the argument is interpreted as c(p, p). If NULL or NA, all elements are formatted in power-of-ten notation. Argument is overridden by specifying set_power or decimal notation.
set_power	Integer, length 1. Formats all values in x with the same power-of-ten exponent. Default NULL. Overrides format and omit_power arguments.
delim	Character, length 1 or 2, to define the left and right math markup delimiters. The default setting, delim = "\$", produces left and right delimiters \$. . \$. The alternate built-in setting, delim = "\\(", produces left and right delimiters \\( . . \\). Custom delimiters can be assigned in a vector of length 2 with left and right delimiter symbols, e.g., c("\\[", "\\]"). Special characters typically must be escaped.
size	Character, length 1, to assign a font size. If not empty, adds a font size macro to the markup inside the math delimiters. Possible values are "scriptsize", "small", "normalsize", "large", and "huge". One may also assign the equivalent LaTeX-style markup itself, e.g., "\\scriptsize", "\\small", etc. Default is NULL.
decimal_mark	Character, length 1, to assign the decimal marker. Possible values are a period "." (default) or a comma ",". Passed to formatC(decimal.mark).
big_mark	Character, length 1, used as the mark between every big_interval number of digits to the left of the decimal marker to improve readability. Possible values are empty "" (default) or "thin" to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself "\\ \\ \\ \\, ". Passed to formatC(big.mark).
big_interval	Integer, length 1, that defines the number of digits (default 3) in groups separated by big_mark. Passed to formatC(big.interval).
small_mark	Character, length 1, used as the mark between every small_interval number of digits to the right of the decimal marker to improve readability. Possible values are empty "" (default) or "thin" to produce a LaTeX-style thin, horizontal space. One may also assign the thin-space markup itself "\\ \\ \\ \\, ". Passed to formatC(small.mark).
small_interval	Integer, length 1, that defines the number of digits (default 5) in groups separated by small_mark. Passed to formatC(small.interval).

whitespace	Character, length 1, to define the LaTeX-style math-mode macro to preserve a horizontal space between words of text or between physical-unit abbreviations when formatting numbers of class "units". Default is "\\ \\ ". Alternatives include "\\ \\:" or "\\ \\>".
multiply_mark	Character, length 1, to define the multiplication symbol in power of ten notation. Possible values are "\\times" (default) or a half-high dot "\\cdot" which is often used when the decimal mark is a comma.

### Details

Given a number, a numerical vector, or a numerical column from a data frame, `format_numbers()` converts the numbers to character strings of the form, " $a \times 10^n$ ", where  $a$  is the coefficient to a specified number of significant digits and  $n$  is the exponent. When used for decimal notation, `format_numbers()` converts numbers to character strings of the form " $a$ ".

Powers-of-ten notation is omitted over a range of exponents via `omit_power` such that numbers so specified are converted to decimal notation. For example, the default `omit_power = c(-1, 2)` formats numbers such as 0.123, 1.23, 12.3, and 123 in decimal form. To cancel these exceptions and convert all numbers to powers-of-ten notation, set the `omit_power` argument to `NULL` or `NA`.

Delimiters for inline math markup can be edited if necessary. If the default argument fails, try using "\\(" as an alternative. If using a custom delimiter to suit the markup environment, be sure to escape all special symbols.

When inputs are of class "units" (created with the units package), a math-text macro of the form `\mathrm{<units_string>}` is appended to the formatted numerical value inside the math delimiters.

Arguments after the dots (...) must be referred to by name.

### Value

A character vector in which numbers are formatted in power-of-ten or decimal notation and delimited for rendering as inline equations in an R markdown document.

### See Also

Other `format_*`: `format_dcml()`, `format_engr()`, `format_nucl()`, `format_sci()`, `format_text()`

### Examples

```
# input: single number
x <- 6.0221E+23
format_numbers(x)

# input: units class
x <- 103400
units(x) <- "N m2 C-2"
format_numbers(x)

# input: vector
data("metals", package = "formatdown")
x <- metals$dens
```

```
format_numbers(x)

# significant digits
x <- 9.75358e+5
format_numbers(x, 2)
format_numbers(x, 3)
format_numbers(x, 4)

# input: data frame
x <- metals[, c("thrm_exp", "thrm_cond")]
as.data.frame(apply(x, 2, format_sci, digits = 3))

# omit_power
x <- 103400
format_numbers(x, format = "sci", omit_power = c(-1, 2)) # default
format_numbers(x, format = "sci", omit_power = c(-1, 5))
format_numbers(x, format = "sci", omit_power = 5) # equivalent to omit_power = c(5, 5)

# omit_power = NULL, power-of-ten notation for all elements
x <- c(1.2, 103400)
format_numbers(x, format = "sci")
format_numbers(x, format = "sci", omit_power = NULL)

# set_power overrides default scientific exponent
x <- 103400
format_numbers(x, format = "sci")
format_numbers(x, format = "sci", set_power = 4)

# set_power overrides omit_power
x <- 103400
format_numbers(x, format = "sci")
format_numbers(x, format = "sci", omit_power = 5)
format_numbers(x, format = "sci", omit_power = 5, set_power = 4)

# decimal format overrides set_power
x <- 103400
format_numbers(x, format = "dcml")
format_numbers(x, format = "dcml", set_power = 3)
```

---

format\_sci

*Format numbers in scientific notation*

---

### Description

Convert a numeric vector to a character vector in which the numbers are formatted in power-of-ten notation in scientific form and delimited for rendering as inline equations in an R markdown document.

**Usage**

```
format_sci(
  x,
  digits = 4,
  ...,
  omit_power = c(-1, 2),
  set_power = NULL,
  delim = formatdown_options("delim"),
  size = formatdown_options("size"),
  decimal_mark = formatdown_options("decimal_mark"),
  small_mark = formatdown_options("small_mark"),
  small_interval = formatdown_options("small_interval"),
  whitespace = formatdown_options("whitespace"),
  multiply_mark = formatdown_options("multiply_mark")
)
```

**Arguments**

<code>x</code>	Number or numbers to be formatted. Can be a single number, a vector, or a column of a data frame.
<code>digits</code>	Integer from 1 through 20 that controls the number of significant digits in printed numeric values. Passed to <code>signif()</code> . Default is 4.
<code>...</code>	Not used for values; forces subsequent arguments to be referable only by name.
<code>omit_power</code>	Numeric vector <code>c(p, q)</code> with $p \leq q$ , specifying the range of exponents over which power-of-ten notation is omitted in either scientific or engineering format. Default is <code>c(-1, 2)</code> . If a single value is assigned, i.e., <code>omit_power = p</code> , the argument is interpreted as <code>c(p, p)</code> . If <code>NULL</code> or <code>NA</code> , all elements are formatted in power-of-ten notation. Argument is overridden by specifying <code>set_power</code> or decimal notation.
<code>set_power</code>	Integer, length 1. Formats all values in <code>x</code> with the same power-of-ten exponent. Default <code>NULL</code> . Overrides <code>format</code> and <code>omit_power</code> arguments.
<code>delim</code>	Character, length 1 or 2, to define the left and right math markup delimiters. The default setting, <code>delim = "\$"</code> , produces left and right delimiters <code>\$. . \$.</code> . The alternate built-in setting, <code>delim = "\\( "</code> , produces left and right delimiters <code>\\( . . \\)</code> . Custom delimiters can be assigned in a vector of length 2 with left and right delimiter symbols, e.g., <code>c("\\[ ", "\\]")</code> . Special characters typically must be escaped.
<code>size</code>	Character, length 1, to assign a font size. If not empty, adds a font size macro to the markup inside the math delimiters. Possible values are <code>"scriptsize"</code> , <code>"small"</code> , <code>"normalsize"</code> , <code>"large"</code> , and <code>"huge"</code> . One may also assign the equivalent LaTeX-style markup itself, e.g., <code>"\\scriptsize"</code> , <code>"\\small"</code> , etc. Default is <code>NULL</code> .
<code>decimal_mark</code>	Character, length 1, to assign the decimal marker. Possible values are a period <code>"."</code> (default) or a comma <code>","</code> . Passed to <code>formatC(decimal.mark)</code> .
<code>small_mark</code>	Character, length 1, used as the mark between every <code>small_interval</code> number of digits to the right of the decimal marker to improve readability. Possible values are empty <code>""</code> (default) or <code>"thin"</code> to produce a LaTeX-style thin, horizontal

	space. One may also assign the thin-space markup itself " <code>\\</code> ", ". Passed to <code>formatC(small.mark)</code> .
<code>small_interval</code>	Integer, length 1, that defines the number of digits (default 5) in groups separated by <code>small_mark</code> . Passed to <code>formatC(small.interval)</code> .
<code>whitespace</code>	Character, length 1, to define the LaTeX-style math-mode macro to preserve a horizontal space between words of text or between physical-unit abbreviations when formatting numbers of class "units". Default is " <code>\\</code> ". Alternatives include " <code>\\:</code> " or " <code>\\&gt;</code> ".
<code>multiply_mark</code>	Character, length 1, to define the multiplication symbol in power of ten notation. Possible values are " <code>\\times</code> " (default) or a half-high dot " <code>\\cdot</code> " which is often used when the decimal mark is a comma.

### Details

`format_sci()` is a wrapper for the more general function `format_numbers()`. Where defaults are defined by `formatdown_options()`, users may reassign the arguments locally in the function call or globally using `formatdown_options()`.

Arguments after the dots (`...`) must be referred to by name.

### Value

A character vector in which numbers are formatted in power-of-ten notation in scientific form and delimited for rendering as inline equations in an R markdown document.

### See Also

Other `format_*`: [format\\_dcml\(\)](#), [format\\_engr\(\)](#), [format\\_nucl\(\)](#), [format\\_numbers\(\)](#), [format\\_text\(\)](#)

### Examples

```
# input: single number
x <- 6.0221E+23
format_sci(x)

# input: units class
x <- 103400
units(x) <- "N m2 C-2"
format_sci(x)

# input: vector
data("metals", package = "formatdown")
x <- metals$dens
format_sci(x)

# significant digits
x <- 9.75358e+5
format_sci(x, 2)
format_sci(x, 3)
format_sci(x, 4)
```

```

# input: data frame
x <- metals[, c("thrm_exp", "thrm_cond")]
as.data.frame(apply(x, 2, format_sci, digits = 3))

# format_sci() same as format_numbers(..., format = "sci")
x <- 6.0221E+23
format_sci(x)
format_numbers(x, format = "sci")

# omit_power
x <- 103400
format_sci(x, omit_power = c(-1, 2)) # default
format_sci(x, omit_power = c(-1, 5))
format_sci(x, omit_power = 5) # equivalent to omit_power = c(5, 5)

# omit_power = NULL, power-of-ten notation for all elements
x <- c(1.2, 103400)
format_sci(x)
format_sci(x, omit_power = NULL)

# set_power overrides default scientific exponent
x <- 103400
format_sci(x)
format_sci(x, set_power = 4)

# set_power overrides omit_power
x <- 103400
format_sci(x)
format_sci(x, omit_power = 5)
format_sci(x, omit_power = 5, set_power = 4)

```

---

format\_text

*Format text*


---

## Description

Convert a character vector to "math text" delimited for rendering as inline equations in an R markdown document. Particularly useful for matching the font face of character columns to that of numerical columns in a table.

## Usage

```

format_text(
  x,
  face = "plain",
  ...,
  size = formatdown_options("size"),
  delim = formatdown_options("delim"),
  whitespace = formatdown_options("whitespace")
)

```

**Arguments**

x	Text to be formatted. Can be a single string, a vector, or a column of a data frame.
face	Font face. Determines the font face macro inside the math delimiters. Possible values are "plain" (default), "italic", "bold", "sans", or "mono". One may assign instead the corresponding LaTeX-style markup itself, e.g., <code>\mathrm</code> , <code>\mathit</code> , <code>\mathbf</code> , <code>\mathsf</code> , or <code>\mathtt</code> .
...	Not used for values; forces subsequent arguments to be referable only by name.
size, delim, whitespace	Used to format the math-delimited character strings. For details, see the help page for <code>formatdown_options()</code> .

**Details**

Given a scalar, vector, or data frame column, `format_text()` converts its argument to a character string of the form `" $\mathxx{a}$ "` where `a` is the element to be formatted and `\mathxx` determines the font face: plain type is set by `\mathrm`; italic by `\mathit`; bold by `\mathbf`; sans serif by `\mathsf`; and monospace (typewriter text) by `\mathtt`. All strings include markup delimiters `$. . $` for rendering (in an R markdown or Quarto markdown document) as an inline equation.

**Value**

A character vector with elements delimited as inline math markup in plain, italic, sans serif, bold, or monospace font face.

**See Also**

Other `format_*`: [format\\_dcml\(\)](#), [format\\_engr\(\)](#), [format\\_nucl\(\)](#), [format\\_numbers\(\)](#), [format\\_sci\(\)](#)

**Examples**

```
# Text vector

# default face = "plain"
x <- air_meas$humid
format_text(x)

# equivalently
format_text(x, face = "plain")

# input vector
x <- c("Hello world!", "Goodbye blues!")
format_text(x)

# argument coerced to character string if possible
format_text(c(1.2, 2.3, 3.4))
format_text(x = NA)
format_text(x = c(TRUE, FALSE, TRUE))

# numbers as strings are rendered as-is
```

```
format_text(x = c("1.2E-3", "3.4E+0", "5.6E+3"))

# other font faces
format_text(x, face = "italic")
format_text(x, face = "bold")
format_text(x, face = "sans")
format_text(x, face = "mono")
```

---

metals

*Properties of metals*


---

### Description

Table of mechanical and thermal properties of selected metals.

### Usage

```
data(metals, package = "formatdown")
```

### Format

Classes `data.table` and `data.frame` with 6 observations of 5 variables:

**metal** Character, name of material

**dens** Numeric, density ( $\text{kg m}^{-3}$ )

**thrm\_exp** Numeric, coefficient of thermal expansion ( $\text{m m}^{-1}\text{K}^{-1}$ )

**thrm\_cond** Numeric, thermal conductivity ( $\text{W m}^{-1}\text{K}^{-1}$ )

**elast\_mod** Numeric, modulus of elasticity (Pa)

### Source

*Marks' Standard Handbook for Mechanical Engineers 9/e* (1987) E.A. Avallone and T. Baumeister (ed.), "Basic Properties of Several Metals", pp. 6-11, McGraw-Hill, NY.

---

water

*Properties of water*


---

### Description

Table of water properties at atmospheric pressure as a function of temperature.

### Usage

```
data(water, package = "formatdown")
```

**Format**

Classes `data.table` and `data.frame` with 11 observations of 5 variables:

**temp** Numeric, temperature (K)

**dens** Numeric, density ( $\text{kg m}^{-3}$ )

**sp\_wt** Numeric, specific weight ( $\text{N m}^{-3}$ )

**visc** Numeric, dynamic viscosity (Pa s)

**bulk\_mod** Numeric, bulk modulus (Pa)

**Source**

E. Maurer E and I. Embry (2022) *hydraulics: Basic Pipe and Open Channel Hydraulics*, R package ver. 0.6.0, <https://edm44.github.io/hydraulics/>.

# Index

## \* datasets

- air\_meas, [2](#)
- atmos, [3](#)
- element\_set, [3](#)
- metals, [22](#)
- water, [22](#)

## \* format\_\*

- format\_dcml, [7](#)
- format\_engr, [9](#)
- format\_nucl, [12](#)
- format\_numbers, [14](#)
- format\_sci, [17](#)
- format\_text, [20](#)

air\_meas, [2](#)

atmos, [3](#)

element\_set, [3](#)

format\_dcml, [7](#), [11](#), [13](#), [16](#), [19](#), [21](#)

format\_engr, [9](#), [9](#), [13](#), [16](#), [19](#), [21](#)

format\_nucl, [9](#), [11](#), [12](#), [16](#), [19](#), [21](#)

format\_numbers, [9](#), [11](#), [13](#), [14](#), [19](#), [21](#)

format\_sci, [9](#), [11](#), [13](#), [16](#), [17](#), [21](#)

format\_text, [9](#), [11](#), [13](#), [16](#), [19](#), [20](#)

formatdown\_options, [4](#)

metals, [22](#)

water, [22](#)