

# Package ‘fracture’

May 21, 2022

**Title** Convert Decimals to Fractions

**Version** 0.2.1

**Description** Provides functions for converting decimals to a matrix of numerators and denominators or a character vector of fractions. Supports mixed or improper fractions, finding common denominators for vectors of fractions, limiting denominators to powers of ten, and limiting denominators to a maximum value. Also includes helper functions for finding the least common multiple and greatest common divisor for a vector of integers. Implemented using C++ for maximum speed.

**License** MIT + file LICENSE

**URL** <https://fracture.rossellhayes.com/>,  
<https://github.com/rossellhayes/fracture>

**BugReports** <https://github.com/rossellhayes/fracture/issues>

**Depends** R (>= 2.10)

**Imports** Rcpp

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

**LinkingTo** Rcpp

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**SystemRequirements** C++11

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Alexander Rossell Hayes [aut, cre, cph]  
(<https://orcid.org/0000-0001-9412-0457>)

**Maintainer** Alexander Rossell Hayes <[alexander@rossellhayes.com](mailto:alexander@rossellhayes.com)>

**Repository** CRAN

**Date/Publication** 2022-05-21 07:20:09 UTC

## R topics documented:

fracture . . . . .	2
frac_lcm . . . . .	3
frac_mat . . . . .	4
frac_style . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

fracture	<i>Convert decimals to a character vector of fractions</i>
----------	--

---

### Description

Convert decimals to a character vector of fractions

### Usage

```
fracture(
  x,
  ...,
  denom = NULL,
  base_10 = FALSE,
  common_denom = FALSE,
  mixed = FALSE,
  max_denom = 1e+07
)
```

as.fracture(x)

is.fracture(x)

### Arguments

x	A vector of decimals or, for as.fracture(), a matrix created by <a href="#">frac_mat()</a>
...	These dots are for future extensions and must be empty.
denom	If denom is not <code>NULL</code> , all fractions will have a denominator of denom. This will ignore all other arguments that affect the denominator.
base_10	If TRUE, all denominators will be a power of 10.
common_denom	If TRUE, all fractions will have the same denominator. If the least common denominator is greater than max_denom, max_denom is used.
mixed	If TRUE, integer components will be displayed separately from fractional components for x values greater than 1. If FALSE, improper fractions will be used for x values greater than 1.

`max_denom` All denominators will be less than or equal to `max_denom`.  
 If `base_10` is TRUE, the maximum denominator will be the largest power of 10 less than `max_denom`.  
 A `max_denom` greater than the inverse square root of [machine double epsilon](#) will produce a warning because floating point rounding errors can occur when denominators grow too large.

**Value**

A character vector.

**See Also**

[frac\\_mat\(\)](#) to return a matrix of numerators and denominators.

**Examples**

```
x <- (6:1) / (1:6)

fracture(x)
fracture(x, common_denom = TRUE)

fracture(x, base_10 = TRUE)
fracture(x, base_10 = TRUE, max_denom = 100)
fracture(x, base_10 = TRUE, common_denom = TRUE)
fracture(x, base_10 = TRUE, common_denom = TRUE, max_denom = 100)

fracture(x, mixed = TRUE)
fracture(x, mixed = TRUE, common_denom = TRUE)
fracture(x, mixed = TRUE, base_10 = TRUE)
fracture(x, mixed = TRUE, base_10 = TRUE, max_denom = 100)
fracture(x, mixed = TRUE, base_10 = TRUE, common_denom = TRUE)
fracture(x, mixed = TRUE, base_10 = TRUE, common_denom = TRUE, max_denom = 100)
```

---

 frac\_lcm

---

*Least common multiple and greatest common divisor*


---

**Description**

Least common multiple and greatest common divisor

**Usage**

```
frac_lcm(..., max = 1e+07)

frac_gcd(...)
```

**Arguments**

... Integer vectors or vectors that can be coerced to integer.  
max If the least common multiple is greater than max, max is returned instead.

**Value**

An integer.

**Examples**

```
frac_lcm(1, 2, 3, 4, 5, 6)
x <- 1:6
frac_lcm(x)
frac_lcm(x, 7)

frac_gcd(12, 42, 60)
y <- c(12, 42, 60)
frac_gcd(y)
frac_gcd(y, 39)
```

---

frac\_mat

*Convert decimals to a matrix of numerators and denominators*

---

**Description**

Convert decimals to a matrix of numerators and denominators

**Usage**

```
frac_mat(
  x,
  ...,
  denom = NULL,
  base_10 = FALSE,
  common_denom = FALSE,
  mixed = FALSE,
  max_denom = 1e+07
)

as.frac_mat(x)

is.frac_mat(x)
```

**Arguments**

x	A vector of decimals or, for <code>as.frac_mat()</code> , a character vector created by <code>fracture()</code>
...	These dots are for future extensions and must be empty.
denom	If <code>denom</code> is not <code>NULL</code> , all fractions will have a denominator of <code>denom</code> . This will ignore all other arguments that affect the denominator.
base_10	If <code>TRUE</code> , all denominators will be a power of 10.
common_denom	If <code>TRUE</code> , all fractions will have the same denominator. If the least common denominator is greater than <code>max_denom</code> , <code>max_denom</code> is used.
mixed	If <code>TRUE</code> , integer components will be displayed separately from fractional components for <code>x</code> values greater than 1. If <code>FALSE</code> , improper fractions will be used for <code>x</code> values greater than 1.
max_denom	All denominators will be less than or equal to <code>max_denom</code> . If <code>base_10</code> is <code>TRUE</code> , the maximum denominator will be the largest power of 10 less than <code>max_denom</code> . A <code>max_denom</code> greater than the inverse square root of <code>machine double epsilon</code> will produce a warning because floating point rounding errors can occur when denominators grow too large.

**Value**

A matrix with the same number of columns as the length of `x` and rows for integers (if `mixed` is `TRUE`), numerators, and denominators.

**See Also**

`fracture()` to return a character vector of fractions.

**Examples**

```
x <- (6:1) / (1:6)

frac_mat(x)
frac_mat(x, common_denom = TRUE)

frac_mat(x, base_10 = TRUE)
frac_mat(x, base_10 = TRUE, max_denom = 100)
frac_mat(x, base_10 = TRUE, common_denom = TRUE)
frac_mat(x, base_10 = TRUE, common_denom = TRUE, max_denom = 100)

frac_mat(x, mixed = TRUE)
frac_mat(x, mixed = TRUE, common_denom = TRUE)
frac_mat(x, mixed = TRUE, base_10 = TRUE)
frac_mat(x, mixed = TRUE, base_10 = TRUE, max_denom = 100)
frac_mat(x, mixed = TRUE, base_10 = TRUE, common_denom = TRUE)
frac_mat(x, mixed = TRUE, base_10 = TRUE, common_denom = TRUE, max_denom = 100)
```

---

frac_style	<i>Style a fracture with superscripts and subscripts</i>
------------	--

---

**Description**

Uses Unicode superscripts and subscripts to format a fracture.

**Usage**

```
frac_style(fracture, ...)
```

**Arguments**

fracture	A <a href="#">fracture</a> or a vector to be passed to <a href="#">fracture()</a> .
...	Additional arguments passed to <a href="#">fracture()</a> .

**Value**

fracture with numerators formatted with Unicode superscripts and denominators formatted with Unicode subscripts.

**Examples**

```
frac_style(fracture(0.5))  
frac_style(fracture(c(0.5, 1.5), mixed = TRUE))
```

# Index

as.frac\_mat (frac\_mat), 4  
as.fracture (fracture), 2

frac\_gcd (frac\_lcm), 3  
frac\_lcm, 3  
frac\_mat, 4  
frac\_mat(), 2, 3  
frac\_style, 6  
fracture, 2, 6  
fracture(), 5, 6

is.frac\_mat (frac\_mat), 4  
is.fracture (fracture), 2

machine double epsilon, 3, 5

NULL, 2, 5