# Package 'fractional'

October 13, 2022

**Type** Package

**Title** Vulgar Fractions in R

**Version** 0.1.3

**Author** Bill Venables

**Maintainer** Bill Venables <bill.venables@gmail.com>

**Description** The main function of this package allows numerical vector objects to
be displayed with their values in vulgar fractional form. This is convenient if
patterns can then be more easily detected. In some cases replacing the components
of a numeric vector by a rational approximation can also be expected to remove
some component of round-off error. The main functions form a re-implementation
of the functions 'fractions' and 'rational' of the MASS package, but using a
radically improved programming strategy.

**License** GPL (>= 2)

**Suggests** stats, MASS, knitr, rmarkdown, ggplot2, dplyr

**Imports** Rcpp

**LazyData** TRUE

**LinkingTo** Rcpp

**RoxygenNote** 5.0.1

**NeedsCompilation** yes

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2016-02-15 16:03:13

## R topics documented:

**Index**                                                                                        **10**

---

fractional                        *Representation of a numeric vector in vulgar fractional form*

---

### Description

The object is flagged so that if it is coerced to `character`, or printed, the numerical quantities are represented by a rational approximation. In other respects the numerical object behaves as normally.

### Usage

```
fractional(x, eps = 1e-06, maxConv = 20, sync = FALSE)

## S3 method for class 'fractional'
as.character(x, eps = attr(x, "eps"), maxConv = attr(x,
  "maxConv"), ...)

## S3 method for class 'charFrac'
print(x, ...)

## S3 method for class 'fractional'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A numeric object |
| eps | An absolute error tolerance |
| maxConv | An upper limit on the number of convergents to use in the continued fractions. |
| sync | A logical value. Should the numerical value be changed to match the rational approximation, as closely as possible with floating point, (TRUE)? Or, should it be left and used in its original state (FALSE)? |
| ... | Currently ignored. |

### Value

A numeric object of class `"fractional"`.

### Methods (by generic)

- `as.character`: S3 method for coercion to character, producing an object inheriting from class `"charFrac"`
- `print`: Print method for class `"charFrac"` objects, unquoted.
- `print`: Print method for `"fractional"` objects

## See Also

[fractions](#) for a similar functionality.

## Examples

```
(M <- solve(cbind(1, contr.helmert(5))))
(Mf <- fractional(M))      ## print method right justifies
(Mc <- as.character(Mf))  ## print method left justifies
(Mn <- numerical(Mc))
set.seed(123)
u <- matrix(runif(10), 2, 5)
(uf <- fractional(u))
(us <- fractional(u, sync = TRUE))  ## may look different!
unfractional(uf) - unfractional(us)  ## rational approximation errors
```

---

| Math.fractional | *Method for the group generic function for the elementary mathematical functions* |
|---|---|

---

## Description

Allows graceful operations with mathematical functions.

## Usage

```
## S3 method for class 'fractional'
Math(x, ...)
```

## Arguments

x               A numerical object flagged as `fractional`

...             Passed on to further methods (but usually not required)

## Value

A numeric object with the results of the computations, but NOT flagged as of class `"fractional"`.

## Examples

```
(M <- fractional(solve(cbind(1, contr.helmert(5)))))
(M0 <- abs(M)*sign(M))  ## fractional flag lost
```

---

numerators                        *Extract the parts of a fractional object*

---

**Description**

Generic function for extracting numerators with methods for "fractional" or "charFrac" objects

Generic function for extracting denominators with methods for "fractional" or "charFrac" objects

**Usage**

```
numerators(x)

## S3 method for class 'charFrac'
numerators(x)

## S3 method for class 'fractional'
numerators(x)

## Default S3 method:
numerators(x)

denominators(x)

## S3 method for class 'charFrac'
denominators(x)

## S3 method for class 'fractional'
denominators(x)

## Default S3 method:
denominators(x)
```

**Arguments**

x                 An object of class "fractional" or "charFrac"

**Value**

An integer vector of numerators

An integer vector of denominators

**Methods (by class)**

- charFrac: numerators method function for "charFrac" objects
- fractional: numerators method function for "fractional" objects

- default: Default numerators method for numeric objects

- charFrac: denominators method function for "charFrac" objects

- fractional: denominators method function for "fractional" objects

- default: Default denominators method for numeric objects

### Examples

```
(pi_approx <- vfractional(base::pi, eps = 0, maxConv = 1:10))
numerators(pi_approx)
denominators(pi_approx)
```

---

numerical                          *Convert a fractional object to the equivalent numeric object*

---

### Description

Convert an object of class "fractional" or "charFrac" to a purely numeric object. This is effectively a method function for the .Primitive generic function as.numeric but written as a separate function for purely technical reasons.

### Usage

```
numerical(vulgar)

## S3 method for class 'fractional'
numerical(vulgar)

## S3 method for class 'charFrac'
numerical(vulgar)

## Default S3 method:
numerical(vulgar)
```

### Arguments

vulgar            character string form of a class 'fractional' object.

### Value

A numeric object as represented by its (usually fractional) display.

### Methods (by class)

- fractional: Method for "fractional" objects

- charFrac: Method for "charFrac" objects

- default: Default method for numerical generic

## Examples

```
suppressPackageStartupMessages(library(dplyr))
m <- 2*diag(5)
m[abs(row(m) - col(m)) == 1] <- -1
m  ## How much roundoff error does inverting entail?
(mi  <- solve(m) %>% fractional) ## patterned inverse
mi * max(denominators(mi)) ## clearer pattern
m1 <- solve(mi)
range(m1 - m)            ## roundoff still present
m2 <- m1 %>% numerical  ## remove roundoff error - hopefully!
identical(m2, m)        ## no roundoff
```

---

Ops.fractional             *Method for the group generic function for the arithmetic operators*

---

## Description

Provides arithmetic operations for numeric objects or of class "fractional".

## Usage

```
## S3 method for class 'fractional'
Ops(e1, e2)
```

## Arguments

e1              A numeric objet, possibly of class "fractional"

e2              A numeric objet, possibly of class "fractional"

## Value

The result of the arithmetic operation, flagged as class "fractional"

## Examples

```
(M <- fractional(1:10/7)
M + 1
1 + M + M^2
```

---

rat                          *Calculate Rational Approximation Using Continued Fraction Methods*

---

### Description

This is a behind-the-scenes function not likely to be used other than internally within the package. It computes the rational approximations for each value in the principal argument.

### Usage

```
rat(x, eps = 1e-06, maxConv = 20L)

.ratr(x, eps = 1e-06, maxConv = 20)

ratr(x, eps = 1e-06, maxConv = 20)
```

### Arguments

| | |
|---|---|
| x | A numeric vector for which rational approximations are required. |
| eps | An absolute error tolerance on the approximation |
| maxConv | An upper limit on the number of convergents that the continued fraction expansion may employ. The fraction is terminated once the desired accuracy is met (or the upper limit is about to be exceeded). |

### Value

A 3 column matrix giving, respectively, the numerators, denominators and number of convergents needed to achieve the error tolerance, in the columns

### Functions

- `rat`: C++ version of the same function used for speed
- `.ratr`: Workhorse function for a single value

### See Also

`rat` which has the same functionality, but is coded in C++.

### Examples

```
fractional(base::pi)
ratr(base::pi)

set.seed(123)
(u <- matrix(runif(10), 2, 5))
(ru <- ratr(u, eps = 1.0e-3, maxConv = 6))
(abs_error <- matrix(abs(u - ru[, 1]/ru[, 2]), 2, 5))
```

---

unfractional            *Demote a fractional object back to a numeric one*

---

### Description

Given an object of class `"fractional"` this simple function removes the attributes that signal that it is to be treated as a fractional object, thus returning it to its original numeric status alone

### Usage

```
unfractional(x)
```

### Arguments

x            A `"fractional"` object

### Value

A simple numeric object like x

### Examples

```
(tst <- fractional(matrix(0:9/10, 2, 5)))
(tst <- unfractional(tst))
```

---

vfractional            *Vectorized form for fractional*

---

### Description

A function which allows any or all of the first three arguments of `fractional` to be vectors, with short vectors recycled in the usual way. Note that the return value is a *character string* vector and may not be used in arithmetic operations

### Usage

```
vfractional(x, eps = 1e-06, maxConv = 20)
```

### Arguments

x            as for `fractional`

eps          as for `fractional`, but may be a vector

maxConv     as for `fractional` but may be a vector

## Value

A character string vector of class "charFrac"

## Examples

```
oldOpt <- options(scipen = 15)
pi_approx <- vfractional(base::pi, eps = 0, maxConv = 1:10)
within(data.frame(pi_approx, stringsAsFactors = FALSE), {
  value = numerical(pi_approx)
  error = signif(base::pi - value, 3)
  n = seq_along(value) - 1
})[, c("n", "pi_approx", "value", "error")]
options(oldOpt)
```

# Index