

Package ‘schoolmath’

August 27, 2026

Type Package

Title Functions and Datasets for Math Used in School

Description Contains functions for math taught in school. A main focus is set to prime-calculation. The package also contains a dataset of all primes between 1 and 99,999,999.

Version 0.5.1

Date 2026-08-26

License GPL (>= 2)

Depends R (>= 3.5.0)

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

LinkingTo Rcpp

Imports Rcpp

NeedsCompilation yes

URL <https://github.com/produnis/schoolmath>

BugReports <https://github.com/produnis/schoolmath/issues>

Author Jörg große Schlarmann [aut, cre]

Maintainer Jörg große Schlarmann <schlarmann@produnis.de>

Repository CRAN

Date/Publication 2026-08-27 05:10:09 UTC

Contents

cancel.fraction	2
decimal2fraction	3
gcd	3

is.decimal	4
is.even	4
is.negative	5
is.odd	6
is.positive	6
is.prim	7
is.real.positive	8
is.whole	8
prime.factor	9
primes	9
primlist	10
scm	10
Index	12

cancel.fraction	<i>cancel a fraction to the smallest numbers</i>
-----------------	--

Description

cancel a fraction to the smallest numbers

Usage

cancel.fraction(numerator, denominator)

Arguments

- numerator the fraction's numerator
- denominator the fraction's denominator #'

Value

An integer vector of length 2 containing the reduced numerator and denominator.

Examples

cancel.fraction(40,15)
cancel.fraction(42, 56)

decimal2fraction	<i>convert a decimal-number into fraction</i>
------------------	---

Description

convert a decimal-number into fraction

Usage

```
decimal2fraction(decimal, period = 0)
```

Arguments

decimal	the decimal number to be converted, given without an repeating ending
period	if the decimal places have an repeating ending (period), set the period here. See examples. #'

Value

An integer vector of length 2 containing the numerator and denominator

Examples

```
## converting 23.4323
decimal2fraction(23.4323)

## converting a number with decimal period, e.g. 12.123444444444444444
decimal2fraction(12.123, 4)
```

gcd	<i>Greatest common divisor of two numbers</i>
-----	---

Description

Greatest common divisor of two numbers

Usage

```
gcd(x, y)
```

Arguments

x	first number
y	second number #'

Value

numeric greatest common divisor

Examples

```
gcd(42, 56)
```

is.decimal	<i>checks if a number is decimal or integer and returns TRUE for decimal</i>
------------	--

Description

checks if a number is decimal or integer and returns TRUE for decimal

Usage

```
is.decimal(x)
```

Arguments

x a vector with numbers to check #'

Value

true or false

Examples

```
is.decimal(40.15) # will return TRUE
is.decimal(4015)  # will return FALSE
```

is.even	<i>Checks whether numbers are even.</i>
---------	---

Description

Checks whether numbers are even.

Usage

```
is.even(x)
```

Arguments

x A numeric vector.

Value

A logical vector of the same length as 'x'.

Examples

```
is.even(45)  # will return FALSE
is.even(46)  # will return TRUE
is.even(46.2) # will return FALSE
```

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8.4)
is.even(x)
```

is.negative	<i>Check whether numbers are negative.</i>
-------------	--

Description

Check whether numbers are negative.

Usage

```
is.negative(x)
```

Arguments

x A numeric vector.

Value

A logical vector of the same length as 'x'.

Examples

```
is.negative(3) # FALSE
is.negative(0) # FALSE
is.negative(-2) # TRUE
```

```
x <- c(-1, -2, 3.02, 4, -5.2, 6, -7)
is.negative(x)
```

is.odd	<i>Checks whether numbers are odd.</i>
--------	--

Description

Checks whether numbers are odd.

Usage

```
is.odd(x)
```

Arguments

x	A numeric vector.
---	-------------------

Value

A logical vector of the same length as 'x'.

Examples

```
is.odd(45)      # will return TRUE
is.odd(46)      # will return FALSE
is.odd(45.123)  # will return FALSE

x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9.7)
is.odd(x)
```

is.positive	<i>Check whether numbers are positive.</i>
-------------	--

Description

Check whether numbers are positive.

Usage

```
is.positive(x)
```

Arguments

x	A numeric vector.
---	-------------------

Value

A logical vector of the same length as 'x'.

Examples

```
is.positive(3) # TRUE
is.positive(-2) # FALSE
is.positive(0) # FALSE

x <- c(-1, -2, 3.02, 4, -5.2, 6, -7, 0)
is.positive(x)
```

is.prim	<i>Check whether a vector contains prime numbers</i>
---------	--

Description

Check whether a vector contains prime numbers

Usage

```
is.prim(x)
```

Arguments

x	The number or vector to check.
---	--------------------------------

Value

A logical vector indicating whether values are prime.

Examples

```
is.prim(8)
is.prim(11)

x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
is.prim(x)
```

is.real.positive	<i>Check whether numbers are non-negative.</i>
------------------	--

Description

A real positive number is defined here as a number greater than or equal to zero.

Usage

```
is.real.positive(x)
```

Arguments

x	A numeric vector.
---	-------------------

Value

A logical vector of the same length as 'x'.

Examples

```
is.real.positive(-3) # FALSE
is.real.positive(0)  # TRUE

x <- c(0, -1, -2, 3.02, 4, -5.2, 6, -7)
is.real.positive(x)
```

is.whole	<i>checks if a number is decimal or integer and returns FALSE for decimal</i>
----------	---

Description

checks if a number is decimal or integer and returns FALSE for decimal

Usage

```
is.whole(x)
```

Arguments

x	a vector with numbers to check #'
---	-----------------------------------

Value

true or false

Examples

```
is.whole(40.15) # will return FALSE
is.whole(4015)  # will return TRUE
```

prime.factor	<i>This function calculates the prime-factors of a number</i>
--------------	---

Description

This function calculates the prime-factors of a number

Usage

```
prime.factor(n)
```

Arguments

n	the number to be checked #'
---	-----------------------------

Value

a vector with the prime factors

Examples

```
prime.factor(21)
prime.factor(100)
```

primes	<i>generate prime-numbers in a range from START to END</i>
--------	--

Description

generate prime-numbers in a range from START to END

Usage

```
primes(start = 1, end = 9999, live = FALSE)
```

Arguments

start	the number to start from
end	the number to end
live	should primes additionally be printed out to stdout when found? (default is FALSE)

Value

a vector of prime numbers

Examples

```
primes(12,150) # list prime-numbers between 12 and 150
```

primlist	<i>A vector containing primes between 1 and 99,999,999</i>
----------	--

Description

Contains primes between 1 and 99,999,999

Usage

```
data(primlist)
```

Format

A vector containing primes between 1 and 99,999,999

Details

Variables in the dataset:

- primlist. A vector containing primes between 1 and 99,999,999

scm	<i>calculating the smallest common multiple of two numbers</i>
-----	--

Description

calculating the smallest common multiple of two numbers

Usage

```
scm(x, y)
```

Arguments

x	first number
y	second number #'

Value

numeric least common multiple

Examples

```
scm(3528, 3780)
```

Index

* **datasets**

primlist, [10](#)

cancel.fraction, [2](#)

decimal2fraction, [3](#)

gcd, [3](#)

is.decimal, [4](#)

is.even, [4](#)

is.negative, [5](#)

is.odd, [6](#)

is.positive, [6](#)

is.prim, [7](#)

is.real.positive, [8](#)

is.whole, [8](#)

prime.factor, [9](#)

primes, [9](#)

primlist, [10](#)

scm, [10](#)