

Package ‘schoolmath’

July 23, 2025

Type Package

Title Functions and Datasets for Math Used in School

Description Contains functions and datasets for math taught in school. A main focus is set to prime-calculation.

Version 0.4.2

Date 2023-08-21

License GPL (≥ 2)

Depends R (≥ 2.10)

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation no

RoxygenNote 7.2.3

Author Jörg große Schlarmann [aut, cre],
Josef Wienand [ctb]

Maintainer Jörg große Schlarmann <schlarmann@produnis.de>

Repository CRAN

Date/Publication 2023-08-21 14:20:02 UTC

Contents

cancel.fraction	2
decimal2fraction	2
gcd	3
is.decimal	4
is.even	4
is.negative	5
is.odd	5
is.positive	6
is.prim	7
is.real.positive	7

is.whole	8
prime.factor	9
primes	9
primlist	10
scm	10
Index	11

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

Description

returns a frequency table with absolute and relative frequencies and cumulated frequencies

Usage

cancel.fraction(numerator, denominator)

Arguments

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

Value

Character string

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

a character string with the fraction.

Examples

```
## converting 23.4323
decimal2fraction(23.4323)

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

gcd

Greatest common divisor of two numbers

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</i>
------------	---

Description

checks if a number is decimal or integer

Usage

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

Arguments

x	the number to check #'
---	------------------------

Value

true or false

Examples

```
is.decimal(40.15)  
is.decimal(4015)
```

is.even	<i>checks if a number or vector is even</i>
---------	---

Description

checks if a number or vector is even

Usage

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

Arguments

x	the number to check #'
---	------------------------

Value

true or false

Examples

```
is.even(45)
is.even(46)
x <- c(1,2,3,4,5, 6, 7)
is.even(x)
```

is.negative	<i>check whether numbers of a vector are negative</i>
-------------	---

Description

check whether numbers of a vector are negative

Usage

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

Arguments

x	the number or vector to check #'
---	----------------------------------

Value

true or false

Examples

```
is.negative(3) # this will return FALSE
is.negative(-2) # this will return TRUE

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

is.odd	<i>checks if a number or vector is odd</i>
--------	--

Description

checks if a number or vector is odd

Usage

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

Arguments

x the number or vector to check #'

Value

true or false

Examples

```
is.odd(45)
is.odd(46)
x <- c(1,2,3,4,5, 6, 7)
is.odd(x)
```

is.positive	<i>check whether numbers of a vector are positive</i>
-------------	---

Description

check whether numbers of a vector are positive

Usage

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

Arguments

x the number or vector to check #'

Value

true or false

Examples

```
is.positive(-3) # this will return FALSE
is.positive(2)  # this will return TRUE

x <- c(-1, -2, 3.02, 4, -5.2, 6, -7)
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(y)
```

Arguments

y the number or vector to check

Value

true or false

Examples

```
is.prim(8) # this will return FALSE
is.prim(11) # this will return TRUE

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

is.real.positive	<i>check whether numbers of a vector are real positive. Real positive means, that zero is included as a positive number.</i>
------------------	--

Description

check whether numbers of a vector are real positive. Real positive means, that zero is included as a positive number.

Usage

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

Arguments

x the number or vector to check #'

Value

true or false

Examples

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

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

is.whole

check whether a vector contains numbers with decimal places

Description

check whether a vector contains numbers with decimal places

Usage

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

Arguments

x the number or vector to check #'

Value

true or false

Examples

```
is.whole(3.12)  # this will return FALSE
is.whole(2)     # this will return TRUE

x <- c(1, 2, 3, 4, 5.5, 6.03, 23.07)
is.whole(x)
```

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 = 12, end = 9999)
```

Arguments

start	the number to start from
end	the number to end #'

Value

a vector of prime numbers

Examples

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

primlist	<i>A vector containing primes from 0 to 9999999</i>
----------	---

Description

Contains primes from 0 to 9999999

Usage

```
data(primlist)
```

Format

A vector containing primes from 0 to 9999999

Details

Variables in the dataset:

- primlist. A vector containing primes from 0 to 9999999

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, [2](#)

gcd, [3](#)

is.decimal, [4](#)

is.even, [4](#)

is.negative, [5](#)

is.odd, [5](#)

is.positive, [6](#)

is.prim, [7](#)

is.real.positive, [7](#)

is.whole, [8](#)

prime.factor, [9](#)

primes, [9](#)

primlist, [10](#)

scm, [10](#)