

Package: primefactr (via r-universe)

September 17, 2024

Encoding UTF-8

Type Package

Title Use Prime Factorization for Computations

Version 0.1.1

Date 2018-05-17

Description Use Prime Factorization for simplifying computations, for instance for ratios of large factorials.

License GPL-3

LazyData TRUE

Depends R (>= 3.2.3)

RoxygenNote 6.0.1

Suggests testthat, covr

URL <https://github.com/privefl/primefactr>

BugReports <https://github.com/privefl/primefactr/issues>

Roxygen list(markdown = TRUE)

Repository <https://privefl.r-universe.dev>

RemoteUrl <https://github.com/privefl/primefactr>

RemoteRef HEAD

RemoteSha aa70fa5cc7b8086924ec6cb4be0f5bb91931a0f2

Contents

primefactr-package	2
AllPrimesUpTo	2
ComputeDivFact	3
IsPrime	4
ReducePrime	4

Index	6
--------------	----------

primefactr-package *R package that uses Prime Factorization for computations.*

Description

TODO

Arguments

n A positive integer.

AllPrimesUpTo *Get all prime numbers.*

Description

Get all prime numbers up to n.

Usage

```
AllPrimesUpTo(n)
```

Arguments

n A positive integer.

Value

A integer vector of all prime numbers up to n.

Examples

```
AllPrimesUpTo(10)
AllPrimesUpTo(100)
AllPrimesUpTo(1e6)
```

ComputeDivFact	<i>Compute the ratio of factorials.</i>
----------------	---

Description

Compute the ratio of factorials using Prime Factorization. For example, `ComputeDivFact(c(a, b), c(d, e, f))` computes $\frac{a!b!}{d!e!f!}$.

Usage

```
ComputeDivFact(num, deno = NULL, out.log = FALSE)
```

Arguments

<code>num</code>	The vector of all numbers which have their factorials in the numerator.
<code>deno</code>	The vector of all numbers which have their factorials in the denominator. Default is <code>NULL</code> , there is only a numerator.
<code>out.log</code>	Is the logarithm of the result returned instead? Default is <code>FALSE</code> .

Value

The result of the ratio or its logarithm if `out.log = TRUE`.

See Also

`choose`

Examples

```
choose(100, 20)
ComputeDivFact(100, c(20, 80))
lchoose(100, 20)
ComputeDivFact(100, c(20, 80), out.log = TRUE)
```

```
factorial(100)
ComputeDivFact(100)
lfactorial(100)
ComputeDivFact(100, out.log = TRUE)
```

IsPrime	<i>Is a prime number?</i>
---------	---------------------------

Description

Is n a prime number? You can see what is a prime number [there](#).

Usage

```
IsPrime(n)
```

Arguments

n	A positive integer.
---	---------------------

Value

A boolean.

Examples

```
IsPrime(1)      # FALSE
IsPrime(5)      # TRUE
IsPrime(5999999) # TRUE
```

ReducePrime	<i>Get the Prime Factorization.</i>
-------------	-------------------------------------

Description

Get the Prime Factorization for a number with a particular coding.

Usage

```
ReducePrime(code, out.summary = FALSE, primes.div = NULL)
```

Arguments

code	A vector representing a number. See details.
out.summary	Is the result to be summarized? For example, (2, 3, 0, 0, 1) can be summarized as (2, 5; 3, 1). Default is FALSE.
primes.div	The vector of all prime numbers up to $\sqrt{\text{length}(\text{code})}$. Default get them for you.

Details

A code is the coding of a number as follows,

$$number = \prod i^{code[i]},$$

or, which is equivalent,

$$\log(number) = \sum code[i] * \log(i).$$

For example,

- 5 is coded as (0, 0, 0, 0, 1),
- 5! is coded as (1, 1, 1, 1, 1),
- 8! is coded as (1, 1, 1, 1, 1, 1, 1, 1),
- 8! / 5! is therefore coded as (0, 0, 0, 0, 0, 1, 1, 1),
- 5! = 5 * 3 * 2^3 can be reduced to (0, 3, 1, 0, 1).

Note that the first element of a code has no effect.

Value

Two rows representing prime numbers

Examples

```
code100 <- c(rep(0, 99), 1)
ReducePrime(c(rep(0, 99), 1), out.summary = TRUE)
primes.div <- AllPrimesUpTo(floor(sqrt(length(code100))))
ReducePrime(c(rep(0, 99), 1), primes.div = primes.div)
```

Index

* **package**

primefactr-package, [2](#)

AllPrimesUpTo, [2](#)

ComputeDivFact, [3](#)

IsPrime, [4](#)

primefactr (primefactr-package), [2](#)

primefactr-package, [2](#)

ReducePrime, [4](#)