Package ‘primefactr’

October 14, 2022

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

NeedsCompilation no

Author Florian Privé [aut, cre]

Maintainer Florian Privé <florian.prive.21@gmail.com>

Repository CRAN

Date/Publication 2018-05-19 09:19:16 UTC

R topics documented:

`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` *Compute the ratio of factorials.*

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

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

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 — *Is a prime number?*

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(59999999) # TRUE
```

ReducePrime — *Get the Prime Factorization.*

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(length(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