

# Package ‘primes’

October 14, 2022

**Type** Package

**Title** Fast Functions for Prime Numbers

**Version** 1.4.0

**Date** 2022-09-10

**Description** Fast functions for dealing with prime numbers, such as testing whether a number is prime and generating a sequence prime numbers. Additional functions include finding prime factors and Ruth-Aaron pairs, finding next and previous prime numbers in the series, finding or estimating the nth prime, estimating the number of primes less than or equal to an arbitrary number, computing primorials, prime k-tuples (e.g., twin primes), finding the greatest common divisor and smallest (least) common multiple, testing whether two numbers are coprime, and computing Euler's totient function. Most functions are vectorized for speed and convenience.

**License** MIT + file LICENSE

**Depends** R (>= 2.10)

**Imports** Rcpp

**LinkingTo** Rcpp

**Suggests** testthat

**SystemRequirements** C++11

**URL** <https://github.com/ironholds/primes>

**BugReports** <https://github.com/ironholds/primes/issues>

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** yes

**Author** Os Keyes [aut, cre],  
Paul Egeler [aut] (<<https://orcid.org/0000-0001-6948-9498>>)

**Maintainer** Os Keyes <[ironholds@gmail.com](mailto:ironholds@gmail.com)>

**Repository** CRAN

**Date/Publication** 2022-09-12 11:50:02 UTC

**R topics documented:**

gcd . . . . .	2
generate_n_primes . . . . .	3
is_prime . . . . .	4
k_tuple . . . . .	5
next_prime . . . . .	6
nth_prime . . . . .	7
primes . . . . .	8
prime_count . . . . .	8
primorial . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

gcd	<i>Find the Greatest Common Divisor, Smallest Common Multiple, or Coprimality</i>
-----	---

---

**Description**

These functions provide vectorized computations for the greatest common divisor (gcd), smallest common multiple (scm), and coprimality. Coprime numbers are also called *mutually prime* or *relatively prime* numbers. The smallest common multiple is often called the *least common multiple*.

**Usage**

gcd(m, n)

scm(m, n)

coprime(m, n)

Rgcd(...)

Rscm(...)

**Arguments**

m, n, ... integer vectors.

**Details**

The greatest common divisor uses Euclid's algorithm, a fast and widely used method. The smallest common multiple and coprimality are computed using the gcd, where  $scm = \frac{a}{gcd(a,b)} \times b$  and two numbers are coprime when  $gcd = 1$ .

The gcd, scm, and coprime functions perform element-wise computation. The Rgcd and Rscm functions perform gcd and scm over multiple values using reduction. That is, they compute the greatest

common divisor and least common multiple for an arbitrary number of integers based on the properties  $gcd(a_1, a_2, \dots, a_n) = gcd(gcd(a_1, a_2, \dots), a_n)$  and  $scm(a_1, a_2, \dots, a_n) = scm(scm(a_1, a_2, \dots), a_n)$ . The binary operation is applied to two elements; then the result is used as the first operand in a call with the next element. This is done iteratively until all elements are used. It is idiomatically equivalent to `Reduce(gcd, x)` or `Reduce(scm, x)`, where `x` is a vector of integers, but much faster.

### Value

The functions `gcd`, `scm`, and `coprime` return a vector of the length of longest input vector. If one vector is shorter, it will be recycled. The `gcd` and `scm` functions return an integer vector while `coprime` returns a logical vector. The reduction functions `Rgcd` and `Rscm` return a single integer.

### Author(s)

Paul Egeler, MS

### Examples

```
gcd(c(18, 22, 49, 13), 42)
## [1] 6 2 7 1
```

```
Rgcd(18, 24, 36, 12)
## [1] 6
```

```
scm(60, 90)
## [1] 180
```

```
Rscm(1:10)
## [1] 2520
```

```
coprime(60, c(77, 90))
## [1] TRUE FALSE
```

---

generate_n_primes	<i>Generate a Sequence of Prime Numbers</i>
-------------------	---

---

### Description

Generate a sequence of prime numbers from `min` to `max` or generate a vector of the first `n` primes. Both functions use a fast implementation of the Sieve of Eratosthenes.

### Usage

```
generate_n_primes(n)
```

```
generate_primes(min = 2L, max)
```

**Arguments**

n                    the number of primes to generate.  
min                  the lower bound of the sequence.  
max                  the upper bound of the sequence.

**Value**

An integer vector of prime numbers.

**Author(s)**

Paul Egeler, MS

**Examples**

```
generate_primes(max = 12)
## [1] 2 3 5 7 11

generate_n_primes(5)
## [1] 2 3 5 7 11
```

---

is\_prime

*Test for Prime Numbers*

---

**Description**

Test whether a vector of numbers is prime or composite.

**Usage**

```
is_prime(x)
```

**Arguments**

x                    an integer vector containing elements to be tested for primality.

**Value**

A logical vector.

**Author(s)**

Os Keyes and Paul Egeler, MS

**Examples**

```
is_prime(4:7)
## [1] FALSE TRUE FALSE TRUE
```

```
is_prime(1299827)
## [1] TRUE
```

---

k_tuple	<i>Prime k-tuples</i>
---------	-----------------------

---

**Description**

Use prime  $k$ -tuples to create lists of twin primes, cousin primes, prime triplets, and so forth.

**Usage**

```
k_tuple(min, max, tuple)
sexy_prime_triplets(min, max)
twin_primes(min, max)
cousin_primes(min, max)
sexy_primes(min, max)
third_cousin_primes(min, max)
```

**Arguments**

min	the lower bound of the sequence.
max	the upper bound of the sequence.
tuple	an integer vector representing the target $k$ -tuple pattern.

**Details**

You can construct your own tuples and generate series of primes using `k_tuple`; however, there are functions that exist for some of the named relationships. They are listed below.

- `twin_primes`: represents  $c(0, 2)$ .
- `cousin_primes`: represents  $c(0, 4)$ .
- `third_cousin_primes`: represents  $c(0, 8)$ .
- `sexy_primes`: represents  $c(0, 6)$ .
- `sexy_prime_triplets`: represents  $c(0, 6, 12)$ . (This relationship is unique in that  $p + 18$  is guaranteed to be composite.)

The term "third cousin primes" is of the author's coinage. There is no canonical name for that relationship to the author's knowledge.

**Value**

A list of vectors of prime numbers satisfying the condition of tuple.

**Author(s)**

Paul Egeler, MS

**Examples**

```
# All twin primes up to 13
twin_primes(2, 13) # Identical to `k_tuple(2, 13, c(0,2))`
## [[1]]
## [1] 3 5
##
## [[2]]
## [1] 5 7
##
## [[3]]
## [1] 11 13

# Some prime triplets
k_tuple(2, 19, c(0,4,6))
## [[1]]
## [1] 7 11 13
##
## [[2]]
## [1] 13 17 19
```

---

next\_prime

*Find the Next and Previous Prime Numbers*

---

**Description**

Find the next prime numbers or previous prime numbers over a vector.

**Usage**

```
next_prime(x)
```

```
prev_prime(x)
```

**Arguments**

`x` a vector of integers from which to start the search.

**Details**

For `prev_prime`, if a value is less than or equal to 2, the function will return NA.

**Value**

An integer vector of prime numbers.

**Author(s)**

Paul Egeler, MS

**Examples**

```
next_prime(5)
## [1] 7

prev_prime(5:7)
## [1] 3 5 5
```

---

*nth\_prime*

*Get the n-th Prime from the Sequence of Primes.*

---

**Description**

Get the n-th prime,  $p_n$ , in the sequence of primes.

**Usage**

```
nth_prime(x)
```

**Arguments**

x                    an integer vector.

**Value**

An integer vector.

**Author(s)**

Paul Egeler, MS

**Examples**

```
nth_prime(5)
## [1] 11

nth_prime(c(1:3, 7))
## [1] 2 3 5 17
```

---

primes	<i>Pre-computed Prime Numbers</i>
--------	-----------------------------------

---

**Description**

The first one thousand prime numbers.

**Usage**

primes

**Format**

An integer vector containing the first one thousand prime numbers.

**See Also**

[generate\\_primes](#), [generate\\_n\\_primes](#)

---

prime_count	<i>Prime-counting Functions and Estimating the Value of the n-th Prime</i>
-------------	--

---

**Description**

Functions for estimating  $\pi(n)$ —the number of primes less than or equal to  $n$ —and for estimating the value of  $p_n$ , the  $n$ -th prime number.

**Usage**

prime\_count(n, upper\_bound)

nth\_prime\_estimate(n, upper\_bound)

**Arguments**

`n` an integer. See *Details* for more information.

`upper_bound` a logical indicating whether to estimate the lower- or upper bound.

**Details**

The `prime_count` function estimates the number of primes  $\leq n$ . When `upper_bound = FALSE`, it is guaranteed to under-estimate for all  $n \geq 17$ . When `upper_bound = TRUE`, it holds for all positive  $n$ .

The `nth_prime_estimate` function brackets upper and lower bound values of the  $n$ th prime. It is valid for  $n \geq 6$ .

The methods of estimation used here are a few of many alternatives. For further information, the reader is directed to the *References* section.



**Author(s)**

Paul Egeler, MS

**References**

"Prime-counting function" (2020) *Wikipedia*. [https://en.wikipedia.org/wiki/Prime-counting\\_function#Inequalities](https://en.wikipedia.org/wiki/Prime-counting_function#Inequalities) (Accessed 26 Jul 2020).

---

primorial

*Compute the Primorial*

---

**Description**

Computes the primorial for prime numbers and natural numbers.

**Usage**

primorial\_n(n)

primorial\_p(n)

**Arguments**

n                    an integer indicating the numbers to be used in the computation. See *Details* for more information.

**Details**

The primorial\_p function computes the primorial with respect the the first n *prime* numbers; while the primorial\_n function computes the primorial with respect the the first n *natural* numbers.

**Value**

A numeric vector of length 1.

**Author(s)**

Paul Egeler, MS

# Index

## \* datasets

primes, 8

`coprime (gcd)`, 2

`cousin_primes (k_tuple)`, 5

`gcd`, 2

`generate_n_primes`, 3, 8

`generate_primes`, 8

`generate_primes (generate_n_primes)`, 3

`is_prime`, 4

`k_tuple`, 5

`lcm (gcd)`, 2

`next_prime`, 6

`nth_prime`, 7

`nth_prime_estimate (prime_count)`, 8

`prev_prime (next_prime)`, 6

`prime_count`, 8

`primes`, 8

`primorial`, 9

`primorial_n (primorial)`, 9

`primorial_p (primorial)`, 9

`Rgcd (gcd)`, 2

`Rscm (gcd)`, 2

`scm (gcd)`, 2

`sexy_prime_triplets (k_tuple)`, 5

`sexy_primes (k_tuple)`, 5

`third_cousin_primes (k_tuple)`, 5

`twin_primes (k_tuple)`, 5