Package 'iterpc'

April 24, 2018

Type Package

Title Efficient Iterator for Permutations and Combinations

Version 0.4.0

Date 2018-04-14

Author Randy Lai [aut, cre]

Maintainer Randy Lai <randy.cs.lai@gmail.com>

Description Iterator for generating permutations and combinations. They can be either drawn with or without replacement, or with distinct/ non-distinct items (multiset). The generated sequences are in lexicographical order (dictionary order). The algorithms to generate permutations and combinations are memory efficient. These iterative algorithms enable users to process all sequences without putting all results in the memory at the same time.
The algorithms are written in C/C++ for faster performance. Note: 'iterpc' is no longer being maintained. Users are recommended to switch to 'arrangements'.

URL https://randy3k.github.io/iterpc

License GPL-2

Depends R (>= 3.0.0)

Imports iterators, gmp (>= 0.5-12), arrangements (>= 1.0.0)

Suggests foreach, testthat, knitr, rmarkdown

ByteCompile yes

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-24 05:12:23 UTC

R topics documented:

getall 2
getcurrent 2

getlength 3
getnext 3
iterpc 4
iter_wrapper 5
multichoose 5
nc_multiset 6
np_multiset 7

Index **8**

`getall` *Get all permutations/combinations for a iterator*

Description

Get all permutations/combinations for a iterator

Usage

```
getall(I)
```

Arguments

`I`	a permutation/combination iterator

Value

next permutation/combination sequence for the iterator `I`

`getcurrent` *Get the current element of a iterator*

Description

Get the current element of a iterator

Usage

```
getcurrent(I)
```

Arguments

`I`	a permutation/combination iterator

Value

current element of a iterator

getlength *Get the length for a iterator*

Description

Get the length for a iterator

Usage

```
getlength(I, bigz = FALSE)
```

Arguments

I	a permutations/combinations iterator
bigz	use gmp's Big Interger

Value

an integer

getnext *Get the next permutation(s)/combination(s) for a iterator*

Description

Get the next permutation(s)/combination(s) for a iterator

Usage

```
getnext(I, d = 1, drop = TRUE)
```

Arguments

I	a permutation/combination iterator
d	number of permutation(s)/combination(s) wanted, default to 1
drop	if d is 1, drop simplify to vector if possible, default to TRUE.

Value

next d permutation(s)/combination(s) sequence for the iterator I

iterpc *Efficient Iterator for Permutations and Combinations*

Description

Initialize a iterator for permutations or combinations

Usage

```
iterpc(n, r = NULL, labels = NULL, ordered = FALSE, replace = FALSE)
```

Arguments

n	the length of the input sequence or a vector of frequencies for a multiset.
r	the length of the output sequence. If missing, equals to sum(n).
labels	if missing, natural numbers are used unless n is a table object. In that case, the names of n are used.
ordered	TRUE corresponds to permutation and FALSE corresponds to combinations.
replace	with/without replacement. Default is FALSE.

Value

a permutation/combination iterator

Examples

```
#1) all combinations of drawing 2 items from {1, 2, 3}
I <- iterpc(5, 2)
getall(I)

#2) continuing 1), get combination by combination
I <- iterpc(5, 2)
getnext(I) # return 1,2
getnext(I) # return 1,3
getnext(I, 2) # return next 2 results

#3) 3) all permutations of {1, 2, 3} and use of labels
I <- iterpc(3, labels=c("a", "b", "c"), ordered=TRUE)
getall(I)

#4) permutations of multiset and
I <- iterpc(c(2, 1, 1), labels=c("a", "b", "c"), ordered=TRUE)
getall(I)

#5) combinations with replacement and the use of table as input
x <- c("a","a","b","c")
I <- iterpc(table(x), 3, replace=TRUE)
getall(I)
```

iter_wrapper *Wrap iterpc objects by iterators::iter*

Description

Wrap iterpc objects by iterators::iter

Usage

```
iter_wrapper(I, d = 1)
```

Arguments

I	the iterpc object
d	number of permutation(s)/combination(s) wanted in each iteration, default to 1

Value

a iter object compatible with iterators package

Examples

```
library(iterators)
I <- iterpc(5, 2)
it <- iter_wrapper(I)
nextElem(it)
nextElem(it)

library(foreach)
I <- iterpc(5, 2)
it <- iter_wrapper(I)
foreach(x=it, .combine=c) %do% { sum(x) }
```

multichoose *Calculate multinomial coefficient*

Description

This function calculates the multinomial coefficient

$$\frac{(\sum n_j)!}{\prod n_j!}.$$

where n_j's are the number of multiplicities in the multiset.

Usage

```
multichoose(n, bigz = FALSE)
```

Arguments

n	a vector of group sizes
bigz	use gmp's Big Interger

Value

multinomial coefficient

Examples

```
# (3+1+1)!/ (3! 1! 1!) = 20
multichoose(c(3,1,1))
```

---

nc_multiset *Calculate the number of r-combinations of a multiset*

Description

Calculate the number of r-combinations of a multiset

Usage

```
nc_multiset(f, r, bigz = FALSE)
```

Arguments

f	the frequencies of the mutliset
r	the number of object drawn from the multiset
bigz	use gmp's Big Interger

Value

the number of combinations (Big Integer from gmp)

Examples

```
x <- c("a","a","b")
# possible combinations of size 2 are "aa" and "ab".
nc_multiset(table(x), 2) # <- 2
```

`np_multiset` *Calculate the number of r-permutations of a multiset*

Description

Calculate the number of r-permutations of a multiset

Usage

```
np_multiset(f, r, bigz = FALSE)
```

Arguments

`f`	the frequencies of the mutliset
`r`	the number of object drawn from the multiset
`bigz`	use gmp's Big Interger

Value

the number of r-permutations (Big Integer from gmp)

Examples

```
x = c("a","a","b")
# possible permutations of size 2 are "aa", "ab" and "ba".
np_multiset(table(x), 2) # = 3
```

Index

getall, 2
getcurrent, 2
getlength, 3
getnext, 3

iter_wrapper, 5
iterpc, 4
iterpc-package (iterpc), 4

multichoose, 5

nc_multiset, 6
np_multiset, 7