Package: trotter (via r-universe)

August 25, 2024

Type Package

Title Pseudo-Vectors Containing All Permutations, Combinations and Subsets of Objects Taken from a Vector.

Version 0.6

Date 2014-09-05

Author Richard Ambler

Maintainer Richard Ambler <rambler@ibwya.net>

Description Class definitions and constructors for pseudo-vectors containing all permutations, combinations and subsets of objects taken from a vector. Simplifies working with structures commonly encountered in combinatorics.

Depends methods

License GPL-3

NeedsCompilation no

Date/Publication 2014-09-21 11:06:59

Repository https://ram6ler.r-universe.dev

RemoteUrl https://github.com/cran/trotter

RemoteRef HEAD

RemoteSha 4456cd442ebe0895eaf85a7474388fce3a307ebf

Contents

---

apv *Amalgams Pseudo-Vector Constructor*

Description

The APV class defines a pseudo-vector containing all the arranged k-amalgams (permutations with replacement) of the objects stored in items. The function apv is a constructor for this class.

Usage

```
apv(k, items)
```

Arguments

k	the number of objects taken at a time.
items	a vector of objects to be amalgamated.

Details

The amalgams are arranged according to the order in which the objects appear in items. The arrangement is very similar to that used by the PPV class (see ppv) except that objects are replaced during permutation creation.

Value

an instance of APV.

Author(s)

Richard Ambler

References

Steinhaus-Johnson-Trotter algorithm. (2014, April 29). In *Wikipedia, The Free Encyclopedia*. Retrieved 13:24, September 5, 2014

See Also

Permutations Pseudo-Vector `ppv`

Combinations Pseudo-Vector `cpv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

Examples

```
# create a pseudo-vector of 10-amalgams from the first 15 letters
a <- apv(10, letters[1:15])
# generate a description
print(a)
# compatable with length
length(a)
# inspect a few of the combinations "stored" in a
a[1]
a[1000000]
a[576650390625]
```

`cpv` *Combinations Pseudo-Vector Constructor*

Description

The `CPV` class defines a pseudo-vector containing all the arranged `k`-combinations of the objects stored in `items`. The function `cpv` is a constructor for this class.

Usage

```
cpv(k, items)
```

Arguments

`k`	the number of objects taken at a time.
`items`	a vector of objects to be combined.

Details

The combinations are arranged according to the order in which the objects appear in `items`. Combinations containing the first object in `items` are followed by combinations that contain the second object but not the first, which are followed by combinations that contain the third but neither the first or the second, etc.

Value

an instance of `CPV`.

Author(s)

Richard Ambler

References

Steinhaus-Johnson-Trotter algorithm. (2014, April 29). In *Wikipedia, The Free Encyclopedia*. Retrieved 13:24, September 5, 2014

See Also

Permutations Pseudo-Vector `ppv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

Examples

```
# create a pseudo-vector of 10-combinations from the first 15 letters
c <- cpv(10, letters[1:15])
# generate a description
print(c)
# compatable with length
length(c)
# inspect a few of the combinations "stored" in c
c[1]
c[1000]
c[3003]
```

`length,APV-method` *Amalgams Pseudo-Vector Length*

Description

Get the length of an `APV` instance.

Usage

```
## S4 method for signature 'APV'
length(x)
```

Arguments

`x`	an instance of `APV`

Details

Since `x` contains all the `k`-amalgams of objects in vector `items`, `length(x)` will return `length(items) ^ k)`.

Value

the number of amalgams (permutations with replacement) in pseudo-vector `x`

See Also

Permutations Pseudo-Vector `ppv`

Combinations Pseudo-Vector `cpv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

`length,CPV-method` *Combinations Pseudo-Vector Length*

Description

Get the length of a `CPV` instance.

Usage

```
## S4 method for signature 'CPV'
length(x)
```

Arguments

`x`	an instance of `CPV`

Details

Since `x` contains all the `k`-combinations of objects in vector `items`, `length(x)` will return `choose(length(items), k)`.

Value

the number of combinations in pseudo-vector `x`

See Also

Permutations Pseudo-Vector `ppv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

`length,PPV-method` *Permutations Pseudo-Vector Length*

Description

Get the length of a `PPV` instance.

Usage

```
## S4 method for signature 'PPV'
length(x)
```

Arguments

`x`	an instance of `PPV`

Details

Since `x` contains all the `k`-permutations of objects in vector `items`, `length(x)` will return `choose(length(items), k) * factorial(k)`.

Value

the number of permutations in pseudo-vector `x`

See Also

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

`length,SPV-method` *Selections Pseudo-Vector Length*

Description

Get the length of an `SPV` instance.

Usage

```
## S4 method for signature 'SPV'
length(x)
```

Arguments

`x`	an instance of `SPV`

Details

Since `x` contains all the `k`-selections of objects in vector `items`, `length(x)` will return `choose(length(items) + k - 1, k)`.

Value

the number of selections (combinations with replacement) in pseudo-vector `x`

See Also

Permutations Pseudo-Vector `ppv`

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Subsets Pseudo-Vector `sspv`

`length,SSPV-method` *Subsets Pseudo-Vector Length*

Description

Get the length of an `SSPV` instance.

Usage

```
## S4 method for signature 'SSPV'
length(x)
```

Arguments

`x`	an instance of `SSPV`

Details

Since `x` contains all the subsets of objects in vector `items`, `length(x)` will return `2 ^ length(items)`.

Value

the number of subsets in pseudo-vector `x`

See Also

Permutations Pseudo-Vector `ppv`

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

`ppv` *Permutations Pseudo-Vector Constructor*

Description

The `PPV` class defines a pseudo-vector containing all the `k`-permutations of the objects stored in `items`. The function `ppv` is a constructor for this class.

Usage

```
ppv(k, items)
```

Arguments

`k`	the number of objects taken at a time.
`items`	a vector of objects to be permuted.

Details

The arrangement of permutations is similar, but in many cases not identical, to that obtained from the Steinhaus-Johnson-Trotter algorithm (see references).

Value

an instance of `PPV`.

Author(s)

Richard Ambler

References

Steinhaus-Johnson-Trotter algorithm. (2014, April 29). In *Wikipedia, The Free Encyclopedia*. Retrieved 13:24, September 5, 2014

See Also

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

Examples

```
# create a pseudo-vector of 5-permutations from the first 10 letters
p <- ppv(5, letters[1:10])
# generate a description
print(p)
# compatable with length
length(p)
# inspect a few of the permutations "stored" in p
p[1]
p[1000]
p[30240]
```

---

spv *Selections Pseudo-Vector Constructor*

Description

The `SPV` class defines a pseudo-vector containing all the arranged `k`-selections (combinations with replacement) of the objects stored in `items`. The function `spv` is a constructor for this class.

Usage

```
spv(k, items)
```

Arguments

`k`	the number of objects taken at a time.
`items`	a vector of objects to be selected.

Details

The selections are arranged according to the order in which the objects appear in `items`. The arrangement is very similar to the arrangement of combinations (see cpv) except that objects may be repeatedly selected.

Value

an instance of `SPV`.

Author(s)

Richard Ambler

References

Steinhaus-Johnson-Trotter algorithm. (2014, April 29). In *Wikipedia, The Free Encyclopedia*. Retrieved 13:24, September 5, 2014

See Also

Permutations Pseudo-Vector ppv

Combinations Pseudo-Vector cpv

Amalgams Pseudo-Vector apv

Subsets Pseudo-Vector sspv

Examples

```
# create a pseudo-vector of 10-selections from the first 15 letters
s <- spv(10, letters[1:15])
# generate a description
print(s)
# compatable with length
length(s)
# inspect a few of the combinations "stored" in s
s[1]
s[1000]
s[1961256]
```

---

sspv *Subsets Pseudo-Vector Constructor*

Description

The `SSPV` class defines a pseudo-vector containing all the arranged subsets of the objects stored in `items`. The function `sspv` is a constructor for this class.

Usage

```
sspv(items)
```

Arguments

`items`	a vector of objects to be subsetted.

Details

The subsets are arranged according to the order in which the objects appear in `items`. The first subset, containing none of the objects, is `NULL`.

Value

an instance of `SSPV`.

Author(s)

Richard Ambler

See Also

Permutations Pseudo-Vector ppv

Combinations Pseudo-Vector cpv

Amalgams Pseudo-Vector apv

Selections Pseudo-Vector spv

Examples

```
# create a pseudo-vector of subsets from the first 15 letters
ss <- sspv(letters[1:15])
# generate a description
print(ss)
# compatable with length
length(ss)
# inspect a few of the combinations "stored" in ss
ss[1]
ss[1000]
ss[32768]
```

`[,APV-method` *Retrieve an Amalgam by Index*

Description

Access an amalgam (permutation with replacement) stored in an `APV` instance by index.

Usage

```
## S4 method for signature 'APV'
x[i, j, drop]
```

Arguments

`x`	an instance of `APV`.
`i`	an index specifying the position of the sought amalgam
`j`	not used.
`drop`	not used.

Details

The amalgam at index `i` of pseudo-vector `x` is not actually stored in memory but calculated as needed. The extract method is used solely for interpretation.

Value

the amalgam located at position `i` in pseudo-vector `x`

See Also

Permutations Pseudo-Vector ppv

Combinations Pseudo-Vector cpv

Selections Pseudo-Vector spv

Subsets Pseudo-Vector sspv

[,CPV-method *Retrieve a Combination by Index*

Description

Access a combination stored in a CPV instance by index.

Usage

```
## S4 method for signature 'CPV'
x[i, j, drop]
```

Arguments

x	an instance of CPV.
i	an index specifying the position of the sought combination.
j	not used.
drop	not used.

Details

The combination at index i of pseudo-vector x is not actually stored in memory but calculated as needed. The extract method is used solely for interpretation.

Value

the combination located at position i in pseudo-vector x

See Also

Permutations Pseudo-Vector ppv

Amalgams Pseudo-Vector apv

Selections Pseudo-Vector spv

Subsets Pseudo-Vector sspv

`[,PPV-method` *Retrieve a Permutation by Index*

Description

Access a permutation stored in a `PPV` instance by index.

Usage

```
## S4 method for signature 'PPV'
x[i, j, drop]
```

Arguments

`x`	an instance of `PPV`.
`i`	an index specifying the position of the sought permutation.
`j`	not used.
`drop`	not used.

Details

The permutation at index `i` of pseudo-vector `x` is not actually stored in memory but calculated as needed. The extract method is used solely for interpretation.

Value

the permutation located at position `i` in pseudo-vector `x`

See Also

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Subsets Pseudo-Vector `sspv`

`[,SPV-method` *Retrieve a Selection by Index*

Description

Access a selection (combination with replacement) stored in an SPV instance by index.

Usage

```
## S4 method for signature 'SPV'
x[i, j, drop]
```

Arguments

x	an instance of SPV.
i	an index specifying the position of the sought selection.
j	not used.
drop	not used.

Details

The selection at index i of pseudo-vector x is not actually stored in memory but calculated as needed. The extract method is used solely for interpretation.

Value

the selection located at position i in pseudo-vector x

See Also

Permutations Pseudo-Vector ppv

Combinations Pseudo-Vector cpv

Amalgams Pseudo-Vector apv

Subsets Pseudo-Vector sspv

`[,SSPV-method` *Retrieve a Subset by Index*

Description

Access asubset stored in an `SSPV` instance by index.

Usage

```
## S4 method for signature 'SSPV'
x[i, j, drop]
```

Arguments

`x`	an instance of `SSPV`.
`i`	an index specifying the position of the sought amalgam
`j`	not used.
`drop`	not used.

Details

The subset at index `i` of pseudo-vector `x` is not actually stored in memory but calculated as needed. The extract method is used solely for interpretation.

Value

the subset located at position `i` in pseudo-vector `x`

See Also

Permutations Pseudo-Vector `ppv`

Combinations Pseudo-Vector `cpv`

Amalgams Pseudo-Vector `apv`

Selections Pseudo-Vector `spv`

Index