

Package ‘sparta’

July 23, 2025

Type Package

Title Sparse Tables

Version 1.0.1

Date 2024-11-22

Description Fast Multiplication and Marginalization of Sparse Tables <doi:10.18637/jss.v111.i02>.

Encoding UTF-8

License MIT + file LICENSE

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp, RcppArmadillo

ByteCompile Yes

NeedsCompilation yes

RoxygenNote 7.3.1

Suggests rmarkdown, knitr, tinytest

URL <https://github.com/mlindsk/sparta>

BugReports <https://github.com/mlindsk/sparta/issues>

VignetteBuilder knitr

Author Mads Lindskou [aut, cre]

Maintainer Mads Lindskou <madslindskou@gmail.com>

Repository CRAN

Date/Publication 2024-11-22 14:20:01 UTC

Contents

sparta-package	2
allowed_class_to_sparta	2
as_array	3
as_cpt	4
as_df	5
as_sparta	6

equiv	7
get_val	8
marg	9
mult	10
normalize	12
print.sparta	13
slice	13
sparsity	14
sparta_ones	15
sparta_struct	16
sparta_unity_struct	17
sum.sparta	17
table_size	18
vals	19
Index	21

sparta-package	<i>sparta: Sparse Tables</i>
----------------	------------------------------

Description

Fast Multiplication and Marginalization of Sparse Tables.

Author(s)

Maintainer: Mads Lindskou <madslindskou@gmail.com>

See Also

- Useful links:
- <https://github.com/mlindsk/sparta>
 - Report bugs at <https://github.com/mlindsk/sparta/issues>

allowed_class_to_sparta	<i>Classes that can be converted to sparta</i>
-------------------------	--

Description

A non-argument function, that outputs the classes that can be converted to a sparta object

Usage

allowed_class_to_sparta()

`as_array`*As array*

Description

Turn a sparse table into an array

Usage

```
as_array(x)
```

```
## S3 method for class 'sparta'  
as_array(x)
```

```
## S3 method for class 'sparta_unity'  
as_array(x)
```

Arguments

`x` sparta object

Value

An array

See Also

[as_array](#)

Examples

```
x <- array(  
  c(1,0,0,2,3,4,0,0),  
  dim = c(2,2,2),  
  dimnames = list(  
    a = c("a1", "a2"),  
    b = c("b1", "b2"),  
    c = c("c1", "c2")  
  )  
)  
  
as_array(as_sparta(x))
```

`as_cpt`*As cpt*

Description

Turn a sparta into a conditional probability table

Usage

```
as_cpt(x, y)

## S3 method for class 'sparta'
as_cpt(x, y)
```

Arguments

<code>x</code>	sparta object
<code>y</code>	the conditioning variables

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)

# A joint probability table p(a, b, c)
as_cpt(sx, character(0))
# the same as normalize
normalize(sx)

# A conditional probability table p(a, c | b)
pacb <- as_cpt(sx, "b")

# The probability distribution when b = b1
slice(pacb, c(b = "b1"))
```

as_df

As data frame

Description

Turn a sparse table into a data frame

Usage

```
as_df(x, dense = FALSE)

## S3 method for class 'sparta'
as_df(x, dense = FALSE)
```

Arguments

x	sparta object
dense	Logical indicating if zero cells should be present or not

Value

A data frame

See Also

[as_array](#)

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

as_df(as_sparta(x))
```

`as_sparta`*As sparse table*

Description

Turn an array-like object or a data.frame into a sparse representation

Usage

```
as_sparta(x)
```

```
## S3 method for class 'array'
```

```
as_sparta(x)
```

```
## S3 method for class 'matrix'
```

```
as_sparta(x)
```

```
## S3 method for class 'table'
```

```
as_sparta(x)
```

```
## S3 method for class 'sparta'
```

```
as_sparta(x)
```

```
## S3 method for class 'data.frame'
```

```
as_sparta(x)
```

Arguments

`x` array-like object or a data.frame

Value

A sparta object

References

<doi:10.18637/jss.v111.i02>

See Also

[as_array](#)

Examples

```
# -----  
# Example 1)  
# -----
```

```

x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

as_sparta(x)

# -----
# Example 2)
# -----

y <- mtcars[, c("gear", "carb")]
y[] <- lapply(y, as.character)
as_sparta(y)

```

equiv

*Equiv***Description**

Determine if two sparta objects are equivalent

Usage

```

equiv(x, y)

## S3 method for class 'sparta'
equiv(x, y)

```

Arguments

x	sparta object
y	sparta object

Value

Logical. TRUE if x and y are equivalent

Examples

```

x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

y <- array(
  c(2,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
sy <- as_sparta(y)

equiv(sx, sy)
equiv(sx, sx)

```

get_val

Get value or cell name

Description

Find the value or the name of a cell

Usage

```

get_val(x, y)

## S3 method for class 'sparta'
get_val(x, y)

get_cell_name(x, y)

## S3 method for class 'sparta'
get_cell_name(x, y)

```

Arguments

x	sparta
y	named character vector or vector of cell indices

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
get_val(sx, c(a = "a2", b = "b1", c = "c2"))
get_cell_name(sx, sx[, 4])
```

marg

Marginalization of sparse tables

Description

Marginalize a sparse table given a vector of variables to marginalize out

Usage

```
marg(x, y, flow = "sum")

## S3 method for class 'sparta'
marg(x, y, flow = "sum")

## S3 method for class 'numeric'
marg(x, y, flow = "sum")
```

Arguments

x	sparta object or a numeric. If numeric, the value is just returned.
y	character vector of the variables to marginalize out
flow	either "sum" or "max"

Value

A sparta object (or scalar if all variables are summed out)

Examples

```

x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
marg(sx, c("c"))

su <- sparta_unity_struct(dim_names(sx), rank = 3.14)
marg(su, c("a", "b"))

```

mult

Multiplication and division of sparse tables

Description

Multiplication and division of sparse tables

Usage

```

mult(x, y)

## S3 method for class 'sparta'
mult(x, y)

## S3 method for class 'numeric'
mult(x, y)

div(x, y)

## S3 method for class 'sparta'
div(x, y)

## S3 method for class 'numeric'
div(x, y)

```

Arguments

x	sparta object or scalar
y	sparta object or scalar

Value

A sparta object or a scalar

References

<doi:10.18637/jss.v111.i02>

Examples

```
# -----
# Example 1)
# -----

x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

y <- array(
  c(1,3,0,2,4,2,7,0,
    1,8,0,1,6,2,1,0,
    1,5,0,3,2,9,1,0),
  dim = c(2,2,2, 3),
  dimnames = list(
    b = c("b1", "b2"),
    d = c("d1", "d2"),
    a = c("a1", "a2"),
    e = c("e1", "e2", "e3")
  )
)

sx <- as_sparta(x)
sy <- as_sparta(y)

sparsity(sx)
table_size(sx)
dim_names(sx)
names(sx)

mult(sx, sy)
div(sy, sx)

# -----
# Example 2)
# -----

d1 <- mtcars[, c("cyl", "vs", "am")]
```

```

d1[] <- lapply(d1, as.character)
d2  <- mtcars[, c("am", "gear", "carb")]
d2[] <- lapply(d2, as.character)
ds1  <- as_sparta(d1)
ds2  <- as_sparta(d2)

mult(ds1, ds2)

# -----
# Example 3)
# -----

su <- sparta_unity_struct(dim_names(sy), rank = 3.1415)
sparta_rank(su)
sum(su)
sun <- normalize(su)
sun
sum(sun)

mult(sx, sun)

# -----
# Example 4)
# -----
so <- sparta_ones(dim_names(sx))
mult(so, 2)

```

normalize

Normalize

Description

Normalize

Usage

```

normalize(x)

## S3 method for class 'sparta'
normalize(x)

## S3 method for class 'numeric'
normalize(x)

```

Arguments

x sparta

Value

A sparta object

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
normalize(sx)
```

print.sparta	<i>Print</i>
--------------	--------------

Description

Print method for sparta objects

Usage

```
## S3 method for class 'sparta'
print(x, ...)
```

Arguments

- x sparta object
- ... For S3 compatability. Not used.

slice	<i>Slice</i>
-------	--------------

Description

Find the slice of a sparse table

Usage

```
slice(x, s, drop = FALSE)

## S3 method for class 'sparta'
slice(x, s, drop = FALSE)
```

Arguments

x	sparta object
s	a slice in form of a named character vector
drop	Logical. If TRUE, the variables in s are removed

Value

A sparta object

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)

# conditional probability table p(b,c|a)
sx <- as_cpt(sx, "a")

# the probability distriubtion when 'a' is 'a2'
sxa2 <- slice(sx, c(a = "a2"))
get_val(sxa2, c(a = "a2", b = "b1", c = "c2"))

sxa2_drop <- slice(sx, c(a = "a2"), drop = TRUE)
get_val(sxa2_drop, c(b = "b1", c = "c2"))

u <- sparta_unity_struct(dim_names(sx))
slice(u, c(a = "a1"), drop = TRUE)
```

sparsity	<i>Sparsity</i>
----------	-----------------

Description

Sparsity

Usage

```
sparsity(x)

## S3 method for class 'sparta'
sparsity(x)

## S3 method for class 'sparta_unity'
sparsity(x)
```

Arguments

x sparta

Value

The ratio of ncol(x) and the total statespace of x

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
sparsity(sx)
```

sparta_ones	<i>Sparta Ones</i>
-------------	--------------------

Description

Construct a sparta object filled with ones

Usage

```
sparta_ones(dim_names)
```

Arguments

dim_names A named list of discrete levels

Value

A sparta object

Examples

```
sparta_ones(list(a = c("a1", "a2"), b = c("b1", "b2")))
```

sparta_struct	<i>Construct sparta object</i>
---------------	--------------------------------

Description

Helper function to construct a sparta object with given values and dim names

Usage

```
sparta_struct(x, vals, dim_names)
```

Arguments

x	matrix where columns represents cells in an array-like object
vals	vector of values corresponding to x
dim_names	a named list

Value

A sparta object

Examples

```
x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
sparta_struct(unclass(sx), vals(sx), dim_names(sx))
```

sparta_unity_struct	<i>Sparse unity table</i>
---------------------	---------------------------

Description

Construct a sparse table of ones

Usage

```
sparta_unity_struct(dim_names, rank = 1L)
```

Arguments

dim_names	A named list of discrete levels
rank	The value of each element. Default is 1.

Value

A sparta object

Examples

```
s <- sparta_unity_struct(list(a = c("a1", "a2"), b = c("b1", "b2")), rank = 1)
mult(s, 2)
```

sum.sparta	<i>Vector-like operations on sparta objects</i>
------------	---

Description

Vector-like operations on sparta objects

Usage

```
## S3 method for class 'sparta'
sum(x, ...)

## S3 method for class 'sparta'
max(x, ...)

## S3 method for class 'sparta'
min(x, ...)

which_min_cell(x)

## S3 method for class 'sparta'
```

```
which_min_cell(x)

which_min_idx(x)

## S3 method for class 'sparta'
which_min_idx(x)

which_max_cell(x)

## S3 method for class 'sparta'
which_max_cell(x)

which_max_idx(x)

## S3 method for class 'sparta'
which_max_idx(x)
```

Arguments

```
x          sparta
...        For S3 compatability.
```

table_size	<i>Number of elements in a table</i>
------------	--------------------------------------

Description

Number of elements in a table

Usage

```
table_size(x)

## S3 method for class 'sparta'
table_size(x)
```

Arguments

```
x          sparta
```

Value

The size of the sparta table x

Examples

```

x <- array(
  c(1,0,0,2,3,4,0,0),
  dim = c(2,2,2),
  dimnames = list(
    a = c("a1", "a2"),
    b = c("b1", "b2"),
    c = c("c1", "c2")
  )
)

sx <- as_sparta(x)
table_size(sx)

```

vals

*Sparta getters***Description**

Getter methods for sparta objects

Usage

```

vals(x)

## S3 method for class 'sparta'
vals(x)

get_values(x)

## S3 method for class 'sparta'
get_values(x)

dim_names(x)

## S3 method for class 'sparta'
dim_names(x)

## S3 method for class 'sparta'
names(x)

sparta_rank(x)

## S3 method for class 'sparta_unity'
sparta_rank(x)

```

Arguments

x sparta object

Index

`allowed_class_to_sparta`, [2](#)
`as_array`, [3](#), [3](#), [5](#), [6](#)
`as_cpt`, [4](#)
`as_df`, [5](#)
`as_sparta`, [6](#)

`dim_names (vals)`, [19](#)
`div (mult)`, [10](#)

`equiv`, [7](#)

`get_cell_name (get_val)`, [8](#)
`get_val`, [8](#)
`get_values (vals)`, [19](#)

`marg`, [9](#)
`max.sparta (sum.sparta)`, [17](#)
`min.sparta (sum.sparta)`, [17](#)
`mult`, [10](#)

`names.sparta (vals)`, [19](#)
`normalize`, [12](#)

`print.sparta`, [13](#)

`slice`, [13](#)
`sparsity`, [14](#)
`sparta (sparta-package)`, [2](#)
`sparta-package`, [2](#)
`sparta_ones`, [15](#)
`sparta_rank (vals)`, [19](#)
`sparta_struct`, [16](#)
`sparta_unity_struct`, [17](#)
`sum.sparta`, [17](#)

`table_size`, [18](#)

`vals`, [19](#)

`which_max_cell (sum.sparta)`, [17](#)
`which_max_idx (sum.sparta)`, [17](#)
`which_min_cell (sum.sparta)`, [17](#)
`which_min_idx (sum.sparta)`, [17](#)