Package ‘tropicalSparse’

July 1, 2018

Type Package

Title Sparse Tropical Algebra

Version 0.1.0

Date 2018-06-25

Description Some of the basic tropical algebra functionality is provided for sparse matrices by applying sparse matrix storage techniques. Some of these are addition and multiplication of vectors and matrices, dot product of the vectors in tropical form and some general equations are also solved using tropical algebra.

License GPL (>= 3)

Depends R (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

URL https://math.berkeley.edu/~bernd/mathmag.pdf

NeedsCompilation no

Author Hamza Farooq [aut],
Hafiz Zain Ul Haq [aut, cre],
Muhammad Kashif Hanif [aut],
Shamsa Javaid [aut],
Karl-Heinz Zimmermann [aut]

Maintainer Hafiz Zain Ul Haq <zainalibajwa4u@yahoo.com>

Repository CRAN

Date/Publication 2018-07-01 13:30:03 UTC

R topics documented:

`check.infinityM` *Check Infinity in Matrix*

Description

`check.infinityM` checks infinite value in a matrix based on `algebraType` input.

Usage

```
check.infinityM(M, algebraType)
```

Arguments

`M`	is matrix.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of this function is a matrix and type of tropical algebra. A matrix may contain infinite values that can be positive or negative. Both the positive and negative infinite values works differently on each algebra type. Due to the difference between `minplus` and `maxplus` tropical algebra, it is important to manage them so they can work in their own bounderies. In `minplus` -Inf cannot be used while in `maxplus` Inf cannot be used. So the main purpose of this funnction is to check such possibilities that can cause errors. If this function finds a -Inf in the matrix and the type of algebra is `minplus` then the function generates an error. Similarly, if the function finds a Inf in the matrix and the type of algebra is `maxplus` then the function also generates an error.

Value

Returns nothing but generates an error if specific conditions met.

See Also

`check.infinityV`

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)
check.infinityM(a, 'minplus')
```

`check.infinityV` *Check Infinity in Vector*

Description

`check.infinityV` checks infinite value in a vector based on `algebraType` input.

Usage

```
check.infinityV(V, algebraType)
```

Arguments

`V`	is vector.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of this function is a vector and type of tropical algebra. A vector may contain infinite values that can be positive or negative. Both the positive and negative infinite values works differently on each algebra type. Due to the difference between `minplus` and `maxplus` tropical algebra, it is important to manage them so they can work in their own bounderies. In `minplus` -Inf cannot be used while in `maxplus` Inf cannot be used. So the main purpose of this funnction is to check such possibilities that can cause errors. If this function finds a -Inf in the vector and the type of algebra is `minplus` then the function generates an error. Similarly, if the function finds a Inf in the vector and the type of algebra is `maxplus` then the function also generates an error.

Value

Returns nothing but generates an error if specific conditions met.

See Also

`check.infinityM`

Examples

```
a <- c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf)
check.infinityV(a, 'minplus')
```

counter *Count non-infinit values*

Description

`counter` is used to get total number of non-infinit values in a matrix.

Usage

```
counter(M)
```

Arguments

`M`	is a matrix.

Details

The input of this function is a matrix. This function returns the total number of non-infinite values in the given matrix. In order to work properly, `M` must be a `matrix` otherwise this method generates an error.

Value

Returns total number of non-infinit values.

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf), nrow = 3, ncol = 3, byrow = TRUE)

counter(a)

# [1] 3
```

row.col.Number *Row/Column Number of a Value*

Description

`row.col.Number` method is used to get the row or column number of a specific value in the matrix.

Usage

```
row.col.Number(i, x, arr)
```

Arguments

`i`	is an index of array containing non-infinite values of the matrix.
`x`	is total number of rows or columns of the matrix.
`arr`	is an array containing row or column pointer of the matrix.

Details

The function `row.col.Number` recieves three parameters `i`, `x` and `arr`. As mentioned above `i` is an index of array containing non-infinite values of the matrix. This array can only be obtained in the `CSR` and `CSC` storage techniques and has zero sparsity. `x` is total number of rows in case of `CSR` or total number of columns in case of `CSC` of the matrix. `arr` is an array containing row pointer in case of `CSR` or column pointer in case of `CSC` of the matrix. From these inputs `row.col.Number` finds row or column number of a specific value in the matrix. This function is used especially for `CSR` and `CSC` storage techniques.

Value

Returns the row or column number of a specific value if succeded, otherwise `NA`.

See Also

`tropicalsparse.add`, `tropicalsparse.mul`.

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

List = tropicalsparse.storage(a,'csr','minplus')
i = 2
row.col.Number(i, nrow(a), List[[1]])
# [1] 2
```

`tropicalsparse.add` *Addition With or Without Storage Techniques*

Description

`tropicalsparse.add` function adds the provided inputs in Tropical Algebra based on type of Tropical Algebra.

Usage

```
tropicalsparse.add(A, B, store = NULL, algebraType)
```

Arguments

`A`	is matrix or vector.
`B`	is matrix or vector.
`store`	is storage technique.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The compulsory inputs of the function `tropicalsparse.add` are `A`, `B` and `algebraType` while the remaining input is optional that is `store`. The inputs `A` and `B` can be matrix/matrix, matrix/vector, vector/matrix and vector/vector otherwise the function generates an error. For `A` and `B`, the order of the input does not matter. It can be in any of the following way: the first input of the function is matrix and second input is a vector. Similarly, vise versa. `store` can be `coo`, `csc` and `csr` for applying following storage techniques respectively: Coordinate-Wise, Compressed Sparse Row, Compressed Sparse Column. This input is case sensitive. If the `store` input is other than the specified storage techniques then the function generates an error. The input `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. `tropicalsparse.storage` function is used to apply storage technique depends upon the input given. If `store` input is not specified then the functionality will be performed without using any storage technique.

Value

Addition of `A` and `B` in Tropical Algebra.

See Also

`tropicalsparse.mul`, `tropicalsparse.storage`

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- matrix(data = c(Inf, Inf, 4, Inf, -0.3, Inf, Inf, 2, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.add(a, b, 'csr', 'minplus')

#     [,1] [,2]  [,3]
# [1,]    2  Inf    4
# [2,]    0 -0.3  Inf
# [3,]  Inf  2.0  Inf

# also

a <- matrix(data = c(5, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, 10, 2),
nrow = 3, ncol = 3, byrow = TRUE)
```

```
b <- matrix(data = c(-Inf, -Inf, 3, -Inf, -0.5, -Inf, 1.1, -Inf, -Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.add(a, b, 'coo', 'maxplus')

#      [,1] [,2] [,3]
# [1,]  5.0 -Inf    3
# [2,] -Inf -0.5 -Inf
# [3,]  1.1 10.0    2

# also

a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 2, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- c(Inf, 0, 10)

tropicalsparse.add(a, b, algebraType = 'minplus')

#      [,1] [,2] [,3]
# [1,]    2  Inf  Inf
# [2,]    0    0    0
# [3,]   10    2   10
```

`tropicalsparse.axpyi` *tropicalsparse.axpyi()*

Description

`tropicalsparse.axpyi` function multiplies the vector `x` by the constant `alpha` and adds the result to the vector `y`.

Usage

```
tropicalsparse.axpyi(y, alpha, x, algebraType)
```

Arguments

`y`	is a vector.
`alpha`	is a constant.
`x`	is a vector.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of the function `tropicalsparse.axpyi` is two vectors, a constant and type of Tropical Algebra. `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. All inputs of this method are compulsory. The operation is expressed as: y = y + alpha * x where `x` and `y` must be a vector while `alpha` must be a constant.

Value

Returns a vector.

See Also

`tropicalsparse.doti`

Examples

```
a <- c(2, Inf, 5, 0, Inf, Inf, Inf, 10, Inf)
b <- c(0, 5, Inf, Inf, 12, 2, Inf, Inf, 3)
alpha <- 5

tropicalsparse.axpyi(a, alpha, b, 'minplus')

# [1]   2  10   5   0  17   7 Inf  10   8
```

`tropicalsparse.doti` *tropicalsparse.doti()*

Description

`tropicalsparse.doti` function multiplies the vector `y` with the vector `x`.

Usage

```
tropicalsparse.doti(x, y, algebraType)
```

Arguments

`x`	is a vector.
`y`	is a vector.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of this function is `x`, `y` and `algebraType`. If any of the input is missing then the function generates an error. The operation is expressed as: result = yx where `x` and `y` must be a vector. `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`.

Value

Returns a vector.

See Also

`tropicalsparse.axpyi`

Examples

```
a <- c(2, Inf, 5, 0, Inf, Inf, Inf, 10, Inf)
b <- c(0, 5, Inf, Inf, 12, 2, Inf, Inf, 3)

tropicalsparse.doti(a, b, 'minplus')

# [1]   2 Inf Inf Inf Inf Inf Inf Inf Inf
```

`tropicalsparse.mm` *tropicalsparse.mm()*

Description

`tropicalsparse.mm` function performs the matrix-matrix operation on the equation: y = alpha * op(A) * op(B) + beta * op(C).

Usage

```
tropicalsparse.mm(alpha = NULL, A, opA = FALSE, B, opB = FALSE,
  beta = NULL, C, opC = FALSE, store = NULL, algebraType)
```

Arguments

`alpha`	is a single real value.
`A`	is a matrix.
`opA`	is transpose of `A`.
`B`	is a matrix.
`opB`	is transpose of `B`.
`beta`	is a single real value.
`C`	is a matrix.
`opC`	is transpose of `C`.
`store`	is storage technique.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of this function is three matrices, two constants, operation on these matrices, storage technique and type of Tropical Algebra. Matrices and `algebraType` inputs are compulsory while all other inputs are optional. `A`, `B` and `C` must be the matrix of same dimensions and these matrices must be sparse otherwise error occured. `alpha` and `beta` must be the single real value. `opA`, `opB` and `opC` can be set to TRUE to take transpose of `A`, `B` and `C` matrices repectively. `store` input can be `coo`, `csc` and `csr` for applying following storage techniques respectively: Coordinate-Wise, Compressed Sparse Row, Compressed Sparse Column. If `store` is not specified then functionality is performed without using any storage technique. `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. First of all `A` is multiplied with `B` and if `alpha` is given then the product of `A` and `B` will be multipied with `alpha` otherwise it remais the same. After this, `beta` is multiplied with `C` only if `beta` is given. Finally, both result are added to each other and the resultant matrix is obtained. Same functionailty is applied if any of the `store` technique is specified.

Value

Returns the resultant matrix.

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- matrix(data = c(Inf, Inf, 4, Inf, -0.3, Inf, Inf, 2, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

c <- matrix(data = c(1, Inf, Inf, Inf, 0, 6, Inf, Inf, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.mm(A = a, alpha = 5, opB = TRUE, B = b, C = c,
store = 'csr', algebraType = 'minplus')

#      [,1] [,2] [,3]
# [1,]    1  Inf  Inf
# [2,]  Inf  0.0    6
# [3,]  Inf 14.7   17
```

`tropicalsparse.mul` *Multiplication With or Without Storage Techniques*

Description

`tropicalsparse.mul` function multiplies the provided inputs in Tropical Algebra based on type of Tropical Algebra.

Usage

```
tropicalsparse.mul(A, B, store = NULL, algebraType)
```

Arguments

`A`	is matrix or vector.
`B`	is matrix or vector.
`store`	is storage technique.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The compulsory inputs of the function `tropicalsparse.mul` are `A`, `B` and `algebraType` while the remaining input is optional that is `store`. The inputs `A` and `B` can be matrix/matrix, matrix/vector, vector/matrix and vector/vector otherwise the function generates an error. For `A` and `B`, the order of the input does not matter. It can be in any of the following way: the first input of the function is matrix and second input is a vector. Similarly, vise versa. `store` can be `coo`, `csc` and `csr` for applying following storage techniques respectively: Coordinate-Wise, Compressed Sparse Row, Compressed Sparse Column. This input is case sensitive. If the `store` input is other than the specified storage techniques then the function generates an error. The input `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. `tropicalsparse.storage` function is used to apply storage technique depends upon the input given. If `store` input is not specified then the functionality will be performed without using any storage technique.

Value

Multiplication of `A` and `B` in Tropical Algebra.

See Also

`tropicalsparse.add`, `tropicalsparse.storage`

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- matrix(data = c(Inf, Inf, 4, Inf, -0.3, Inf, Inf, 2, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.mul(a, b, 'csr', 'minplus')

#      [,1] [,2] [,3]
# [1,]  Inf  Inf    6
# [2,]  Inf  Inf    4
# [3,]  Inf  9.7  Inf

# also
```

```
a <- matrix(data = c(5, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, 10, 2),
nrow = 3, ncol = 3, byrow = TRUE)

b <- matrix(data = c(-Inf, -Inf, 3, -Inf, 0.5, -Inf, 1.1, -Inf, -Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.mul(a, b, 'coo', 'maxplus')

#      [,1] [,2] [,3]
# [1,] -Inf -Inf    8
# [2,] -Inf -Inf -Inf
# [3,]  3.1 10.5 -Inf

# also

a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 2, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- c(Inf, 0, 10)

tropicalsparse.mul(a, b, algebraType = 'minplus')

#      [,1] [,2] [,3]
# [1,]  Inf  Inf  Inf
# [2,]    0  Inf  Inf
# [3,]  Inf   12  Inf
```

`tropicalsparse.mv` *tropicalsparse.mv()*

Description

`tropicalsparse.mv` function performs the matrix-vector operation on the equation: y = alpha * op(A) * x + beta * y.

Usage

```
tropicalsparse.mv(alpha = NULL, A, opA = FALSE, x, beta = NULL, y,
  store = NULL, algebraType)
```

Arguments

`alpha`	is a single real value.
`A`	is a matrix.
`opA`	is transpose of `A`.
`x`	is a vector.

`beta`	is a single real value.
`y`	is a vector.
`store`	is storage technique.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The input of this function is one matrix, transpose of that matrix, two vectors, two constants, storage technique and type of Tropical Algebra. The inputs of the matrix, vectors and `algebraType` are compulsory while all other inputs are optional. The matrix must be sparse otherwise error occured. `alpha` and `beta` must be a single real value. `opA` can be set to TRUE to take transpose of `A`. `store` input can be `coo`, `csc` and `csr` for applying following storage techniques respectively: Coordinate-Wise, Compressed Sparse Row, Compressed Sparse Column. If `store` is not specified then functionality is performed without using any storage technique. `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. First of all `A` is multiplied with `x` and if `alpha` is given then the product of `A` and `x` will be multipied with `alpha` otherwise it remais the same. After this, `beta` is multiplied with `y` only if `beta` is given. Finally, both result are added to each other and the resultant matrix is obtained. Same functionailty is applied if any of the `store` technique is specified.

Value

Returns the resultant matrix.

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

b <- c(2, Inf, 5)

c <- c(Inf, 9, Inf)

tropicalsparse.mv(A = a, alpha = 5, opA = TRUE, x = b, y = c,
store = 'csr', algebraType = 'minplus')

#      [,1] [,2] [,3]
# [1,]    9  Inf  Inf
# [2,]    9    9    9
# [3,]  Inf   20  Inf
```

tropicalsparse.storage

Storage Techniques

Description

`tropicalsparse.storage` function is used to apply `coo`, `csr` and `csc` storage techniques on the sparse matrix in Tropical Algebra.

Usage

```
tropicalsparse.storage(M, store, algebraType)
```

Arguments

`M`	is Matrix
`store`	is storage technique.
`algebraType`	is string input that can be `minplus` or `maxplus`.

Details

The function `tropicalsparse.storage` recieves a matrix as first input, storage technique as second input and the type of Tropical Algebra as third input. All the inputs are compulsory. `store` can be `coo`, `csr` and `csc`. `algebraType` is used to specify type of Tropical Algebra. This can be `minplus` or `maxplus`. For more details about `algebraType`, see `detail` section of `check.infinityM` or `check.infinityV`. If store is equal to `coo` then the function returns a list containing three arrays that are `row_Indices_COO`, `col_Indices_COO` and `values_COO`. If store is equal to `csc` then the function returns a list containing three arrays that are `col_Pointer_CSC`, `row_Indices_CSC` and `values_CSC`. If store is equal to `csr` then the function returns a list containing three arrays that are `row_Pointer_CSR`, `col_Indices_CSR` and `values_CSR`. These storage techniques are especially designed for sparse matrices and are very helpful and time saving.

Value

Returns a list `result` that contains three arrays depends upon the `store` input.

See Also

`tropicalsparse.add`, `tropicalsparse.mv`.

Examples

```
a <- matrix(data = c(2, Inf, Inf, 0, Inf, Inf, Inf, 10, Inf),
nrow = 3, ncol = 3, byrow = TRUE)

tropicalsparse.storage(a, 'coo', 'minplus')

# $row_Indices_COO
# [1] 1 2 3

# $col_Indices_COO
# [1] 1 1 2

# $values_COO
# [1]  2  0 10
```

Index