Package 'multiApply'

February 5, 2018

Title Apply Functions to Multiple Multidimensional Arguments

Version 1.0.0

Description The base apply function and its variants, as well as the related functions in the 'plyr' package, typically apply user-defined functions to a single argument (or a list of vectorized arguments in the case of mapply). The 'multiApply' package extends this paradigm to functions taking a list of multiple unidimensional or multidimensional arguments (or combinations thereof) as input, which can have different numbers of dimensions as well as different dimension lengths.

Depends R (>= 3.2.0)

Imports abind, doParallel, foreach, plyr

License LGPL-3

URL https://earth.bsc.es/gitlab/ces/multiApply

BugReports https://earth.bsc.es/gitlab/ces/multiApply/issues

Encoding UTF-8

LazyData true

RoxygenNote 5.0.0

NeedsCompilation no

Author BSC-CNS [aut, cph],
Alasdair Hunter [aut, cre],
Nicolau Manubens [aut]

Maintainer Alasdair Hunter <alasdair.hunter@bsc.es>

R topics documented:

Apply *Wrapper for Applying Atomic Functions to Arrays.*

Description

This wrapper applies a given function, which takes N [multi-dimensional] arrays as inputs (which may have different numbers of dimensions and dimension lengths), and applies it to a list of N [multi-dimensional] arrays with at least as many dimensions as expected by the given function. The user can specify which dimensions of each array (or matrix) the function is to be applied over with the `margins` or `target_dims` option. A user can apply a function that receives (in addition to other helper parameters) 1 or more arrays as input, each with a different number of dimensions, and returns any number of multidimensional arrays. The target dimensions can be specified by their names. It is recommended to use this wrapper with multidimensional arrays with named dimensions.

Usage

```
Apply(data, target_dims = NULL, AtomicFun, ..., output_dims = NULL,
  margins = NULL, ncores = NULL)
```

Arguments

`data`	A single object (vector, matrix or array) or a list of objects. They must be in the same order as expected by AtomicFun.
`target_dims`	List of vectors containing the dimensions to be input into AtomicFun for each of the objects in the data. These vectors can contain either integers specifying the dimension position, or characters corresponding to the dimension names. This parameter is mandatory if margins is not specified. If both margins and target_dims are specified, margins takes priority over target_dims.
`AtomicFun`	Function to be applied to the arrays.
`...`	Additional arguments to be used in the AtomicFun.
`output_dims`	Optional list of vectors containing the names of the dimensions to be output from the AtomicFun for each of the objects it returns (or a single vector if the function has only one output).
`margins`	List of vectors containing the margins for the input objects to be split by. Or, if there is a single vector of margins specified and a list of objects in data, then the single set of margins is applied over all objects. These vectors can contain either integers specifying the dimension position, or characters corresponding to the dimension names. If both margins and target_dims are specified, margins takes priority over target_dims.
`ncores`	The number of multicore threads to use for parallel computation.

Details

When using a single object as input, Apply is almost identical to the apply function. For multiple input objects, the output array will have dimensions equal to the dimensions specified in 'margins'.

Value

List of arrays or matrices or vectors resulting from applying AtomicFun to data.

References

Wickham, H (2011), The Split-Apply-Combine Strategy for Data Analysis, Journal of Statistical Software.

Examples

```
#Change in the rate of exceedance for two arrays, with different
#dimensions, for some matrix of exceedances.
data = list(array(rnorm(2000), c(10,10,20)), array(rnorm(1000), c(10,10,10)),
            array(rnorm(100), c(10, 10)))
test_fun <- function(x, y, z) {((sum(x > z) / (length(x))) /
                                (sum(y > z) / (length(y)))) * 100}
margins = list(c(1, 2), c(1, 2), c(1,2))
test <- Apply(data, margins = margins, AtomicFun = "test_fun")
```

Index