Package 'arrApply'

November 10, 2016

Type Package

Title Apply a Function to a Margin of an Array

Version 2.0.1

Date 2016-11-10

Author Serguei Sokol

Maintainer Serguei Sokol <sokol@insa-toulouse.fr>

Description High performance variant of apply() for a fixed set of functions.
Considerable speedup is a trade-off for universality, user defined
functions cannot be used with this package. However, 20 most currently employed
functions are available for usage. They can be divided in three types:
reducing functions (like mean(), sum() etc., giving a scalar when applied to a vector),
mapping function (like normalise(), cumsum() etc., giving a vector of the same length
as the input vector) and finally, vector reducing function (like diff() which produces
result vector of a length different from the length of input vector).
Optional or mandatory additional arguments required by some functions
(e.g. norm type for norm() or normalise() functions) can be
passed as named arguments in '...'.

License GPL (>= 2)

Imports Rcpp (>= 0.12.0),

LinkingTo Rcpp, RcppArmadillo

Suggests testthat

RoxygenNote 5.0.1

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-11-10 17:09:31

R topics documented:

arrApply *High Performance Variant of apply()*

Description

High performance variant of apply() for a fixed set of functions. Considerable speedup is a trade-off for universality, user defined functions cannot be used with arrApply. However, 20 most currently employed functions are available for usage. They can be divided in three types: reducing functions (like mean(), sum() etc., giving a scalar when applied to a vector), mapping function (like normalise(), cumsum() etc., giving a vector of the same length as the input vector) and finally, vector reducing function (like diff() which produces result vector of a length different from the length of input vector). Optional or mandatory additional arguments required by some functions (e.g. norm type for norm() or normalise() functions) can be passed as named arguments in '...'.

Usage

```
arrApply(arr, idim = 1L, fun = "sum", ...)
```

Arguments

arr	numeric array of arbitrary dimension
idim	integer, dimension number along which a function must be applied
fun	character string, function name to be applied
...	additional named parameters. Optional parameters can be helpful for the following functions: sd(), var() [norm_type: 0 normalisation using N-1 entries (default); 1 normalisation using N entries]; norm() [p: integer >= 1 (default=2) or one of "-inf", "inf", "fro".] normalise() [p: integer >= 1, default=2] diff() [k: integer >= 1 (default=1) number of recursive application of diff(). The size of idim-th dimension will be reduced by k.] trapz() [x: numerical vector of the same length as idim-th size of arr] Mandatory parameter: multv(), divv(), addv(), subv() [v: numerical vector of the same length as idim-th size of arr]

Details

The following functions can be used as argument 'fun' (brackets [] indicate additional parameters that can be passed in '...'): - reducing functions: sum(), prod(), all(), any(), min(), max(), mean(), median(), sd() [norm_type], var() [norm_type], norm() [p], trapz() [x] (trapezoidal integration with respect to spacing in x, if x is provided, otherwise unit spacing is used); - mapping functions: normalise() [p], cumsum(), cumprod(), multv() [v] (multiply a given dimension by a vector v, term by term), divv() [v] (divide by a vector v), addv() [v] (add a vector v), subv() [v] (subtract a vector v); - vector reducing function: diff() [k].

RcppArmadillo is used to do the job in very fast way but it comes at price of not allowing NA in the input numeric array. Vectors are allowed at input. They are considered as arrays of dimension 1. So in this case, idim can only be 1.

Value

output array of dimension cut by 1 (the idim-th dimension will disappear for reducing functions) or of the same dimension as the input arr for mapping and vector reducing functions. For vector reducing functions, the idim-th dimension will be different from idim-th dimension of arr. The type of result (numeric or logical) depends on the function applied, logical for all() and any(), numerical – for all other functions.

Author(s)

Serguei Sokol <sokol at insa-toulouse.fr>

Examples

```
arr=matrix(1:12, 3, 4)
v1=arrApply(arr, 2, "mean")
v2=rowMeans(arr)
stopifnot(all(v1==v2))

arr=array(1:24, dim=2:4) # dim(arr)=c(2, 3, 4)
mat=arrApply(arr, 2, "prod") # dim(mat)=c(2, 4), the second dimension is cut out
stopifnot(all(mat==apply(arr, c(1, 3), prod)))
```

Index