

# Package ‘mapReduce’

January 2, 2012

**Type** Package

**Title** mapReduce - flexible mapReduce algorithm for parallel computation

**Version** 1.2.3

**Date** 2011-03-05

**Author** Christopher Brown

**Maintainer** Christopher Brown <chris.brown@decisionpatterns.com>

**Depends** R (>= 2.6.0)

**Suggests** multicore, papply, testthat

**License** GPL (>= 2)

**Description** mapReduce is an algorithm provides a simple framework for parallel computations. This implementation provides (a) a pure R implementation (b) a syntax following the mapReduce paper and (c) flexible and parallelizable back end.

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2011-03-06 09:47:29

## R topics documented:

mapReduce-package . . . . .	2
mapReduce . . . . .	2

<b>Index</b>	<b>5</b>
--------------	----------

mapReduce-package      *mapReduce - flexible mapReduce algorithm for parallel computation*

---

### Description

mapReduce is an algorithm provides a simple framework for parallel computations. This implementation provides the following features: \* pure R \* simple R-style syntax \* agnostic to parallelization backend

mapReduce is also a convenient replacement for [by](#) and [aggregate](#).

### Details

Package:      mapReduce  
Type:          Package  
Version:      1.2.2  
Date:          2011-03-02  
License:      LGPL (>=2 )  
LazyLoad:    yes

All examples are in [mapReduce](#)

### Author(s)

Christopher Brown

Maintainer: Christopher Brown <cbrown -at- decisionpatterns.com>

### References

Dean and Gemawatt, (2004) MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation.

also: <http://labs.google.com/papers/mapreduce.html>

---

mapReduce      *mapReduce - mapReduce algorithm for parallel computation*

---

### Description

mapReduce is an algorithm provides a simple framework for parallel computations. See references for details. This implementation provides the following features: \* pure R \* simple R-style syntax \* agnostic to parallelization backend

mapReduce is also a convenient replacement for [by](#) and [aggregate](#).

**Usage**

```
mapReduce( map, ..., data, apply = sapply)
```

**Arguments**

map	An expression to be evaluated on data which yielding a vector that is subsequently used to split the data into parts that can be operated on independently.
...	The reduce step. One or more expressions that are evaluated for each of the partitions made
data	A R data structure such as a matrix, list or data.frame.
apply	The functions used for parallelization (default: <a href="#">sapply</a> ) See Details for how to use another parallelization backend.

**Details**

The mapReduce package provides a divide-and-conquer approach to parallel computations closely following the framework and nomenclature proposed by Dean and Gemawatt. The approach is not different from the parallelization approach used internally by R's [apply](#) function. In fact, mapReduce is nothing more than:

```
apply( map(data), reduce )
```

The novelty of both this package and the Dean and Gemawatt paper is the extension beyond a single-process to modern architectures: multiple cores, processes, machines, clusters, data centers, or clouds.

Because there is no standard "out-of-process" parallelization function in R, by default, mapReduce runs "in-process" using [sapply](#). Here, mapReduce can be thought of as a replacement of apply type function such as [by](#) and [aggregate](#).

This package was designed to make "out-of-process" parallelization easy and seamless across all parallelization infrastructure methods and techniques. The user need only supply his own parallelization function to the apply argument.

**Value**

The value returned depends on the reduce step. Commonly, this is a simple R data structure such as a data.frame.

**Note**

Special Thanks to Collin Bennett and Robert Grossman of Open Data group for advice and feedback.

**Author(s)**

Christopher Brown <[cbrown-at-opendatagroup.com](mailto:cbrown-at-opendatagroup.com)>

## References

Dean and Gemawatt, (2004) MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation.

also: <http://labs.google.com/papers/mapreduce.html>

The Open Data Group at <http://www.opendatagroup.com>

## See Also

[apply](#), [sapply](#) - [by](#), [aggregate](#) ,

Parallelization Backends: [papply](#), [multicore](#), [snow](#).

## Examples

```
mapReduce(  
  map=Species,  
  mean.sepal.length=mean(Sepal.Length),  
  max.sepal.length=max(Sepal.Length) ,  
  data = iris  
)
```

```
mapReduce(  
  substr(Species,1,3),  
  mean.sepal.length=mean(Sepal.Length),  
  max.sepal.length=max(Sepal.Length),  
  data=iris  
)
```

```
mapReduce( cyl, mean(mpg), avg.hp=mean(hp), data=mtcars )
```

# Index

\*Topic **iteration**

mapReduce, [2](#)

\*Topic **package**

mapReduce-package, [2](#)

aggregate, [2-4](#)

apply, [3, 4](#)

by, [2-4](#)

mapReduce, [2, 2](#)

mapReduce-package, [2](#)

sapply, [3, 4](#)