Package 'Rlof'

September 17, 2015

Version 1.1.1

Date 2015-09-16

Title R Parallel Implementation of Local Outlier Factor(LOF)

Author Yingsong Hu, Wayne Murray and Yin Shan, Australia.

Maintainer Yingsong Hu <yingsonghu@hotmail.com>

Depends R (>= 2.14.0), doParallel, foreach

Description R parallel implementation of Local Outlier Factor(LOF) which uses multiple CPUs to significantly speed up the LOF computation for large datasets. (Note: The overall performance depends on the computers especially the number of the cores).It also supports multiple k values to be calculated in parallel, as well as various distance measures in addition to the default Euclidean distance.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-09-17 07:51:00

R topics documented:

`Rlof-package` *R Parallel Implementation of Local Outlier Factor(LOF)*

Description

R parallel implementation of Local Outlier Factor(LOF) which uses multiple CPUs to significantly speed up the LOF computation for large datasets. (Note: The overall performance depends on the computers especially the number of the cores).It also supports multiple k values to be calculated in parallel, as well as various distance measures in addition to the default Euclidean distance.

Details

Package:	Rlof
Version:	1.1.0
Date:	2015-09-10
Depends:	R (>= 2.14.0), doParallel, foreach
License:	GPL (>= 2)
URL:	http://CRAN.R-project.org/package=Rlof
What's new:	i) a new optional parameter provided to specify the number of cores used for parallel computing ii) support of Windows system

Author(s)

Yingsong Hu <yingsonghu@hotmail.com>, Wayne Murray and Yin Shan, Australia.

Maintainer: Yingsong Hu <yingsonghu@hotmail.com>

`distmc` *Distance Matrix Computation with multi-threads*

Description

This function is similar to `dist()` in **stats**, with additional support of multi-threading.

Usage

```
distmc(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

Arguments

`x`	a numeric matrix, data frame or `"dist"` object.
`method`	the distance measure to be used. This must be one of `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"` or `"minkowski"`. Any unambiguous substring can be given.
`diag`	logical value indicating whether the diagonal of the distance matrix should be printed by `print.dist`.
`upper`	logical value indicating whether the upper triangle of the distance matrix should be printed by `print.dist`.
`p`	The power of the Minkowski distance.

Details

Available distance measures are (written for two vectors x and y):

`euclidean`: Usual square distance between the two vectors (2 norm).

`maximum`: Maximum distance between two components of x and y (supremum norm)

`manhattan`: Absolute distance between the two vectors (1 norm).

`canberra`: $\sum_i |x_i - y_i| / |x_i + y_i|$. Terms with zero numerator and denominator are omitted from the sum and treated as if the values were missing.

This is intended for non-negative values (e.g. counts): taking the absolute value of the denominator is a 1998 R modification to avoid negative distances.

`binary`: (aka *asymmetric binary*): The vectors are regarded as binary bits, so non-zero elements are 'on' and zero elements are 'off'. The distance is the *proportion* of bits in which only one is on amongst those in which at least one is on.

`minkowski`: The p norm, the pth root of the sum of the pth powers of the differences of the components.

Missing values are allowed, and are excluded from all computations involving the rows within which they occur. Further, when `Inf` values are involved, all pairs of values are excluded when their contribution to the distance gave `NaN` or `NA`.
If some columns are excluded in calculating a Euclidean, Manhattan, Canberra or Minkowski distance, the sum is scaled up proportionally to the number of columns used. If all pairs are excluded when calculating a particular distance, the value is `NA`.

The `"distmc"` method of `as.matrix()` and `as.dist()` can be used for conversion between objects of class `"dist"` and conventional distance matrices.

`as.dist()` is a generic function. Its default method handles objects inheriting from class `"dist"`, or coercible to matrices using `as.matrix()`. Support for classes representing distances (also known as dissimilarities) can be added by providing an `as.matrix()` or, more directly, an `as.dist` method for such a class.

Value

`distmc` returns an object of class `"dist"`.

The lower triangle of the distance matrix stored by columns in a vector, say `do`. If `n` is the number of observations, i.e., `n <- attr(do, "Size")`, then for $i < j \le n$, the dissimilarity between (row) `i` and `j` is `do[n*(i-1) - i*(i-1)/2 + j-i]`. The length of the vector is $n * (n-1)/2$, i.e., of order n^2.

The object has the following attributes (besides `"class"` equal to `"dist"`):

`Size`	integer, the number of observations in the dataset.
`Labels`	optionally, contains the labels, if any, of the observations of the dataset.
`Diag, Upper`	logic, corresponding to the arguments `diag` and `upper` above, specifying how the object should be printed.
`call`	optional, the `call` used to create the object.
`method`	optional, the distance measure used; resulting from `distmc()`, the (`match.arg()`ed) `method` argument.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979) *Multivariate Analysis.* Academic Press.

Borg, I. and Groenen, P. (1997) *Modern Multidimensional Scaling. Theory and Applications.* Springer.

See Also

`dist()` in the **stats** package

Examples

```
data(iris)
df<-iris[-5]
dist.data<-distmc(df,'manhattan')
```

---

lof *Local Outlier Factor*

Description

A function that finds the local outlier factor (Breunig et al.,2000) of the matrix "data" using k neighbours. The local outlier factor (LOF) is a measure of outlierness that is calculated for each observation. The user decides whether or not an observation will be considered an outlier based on this measure. The LOF takes into consideration the density of the neighbourhood around the observation to determine its outlierness. This is a faster implementation of LOF by using a different data structure and distance calculation function compared to `lofactor()` function available in **dprep** package. It also supports multiple k values to be calculated in parallel, as well as various distance measures besides the default Euclidean distance.

Usage

```
lof(data, k, cores = NULL, ...)
```

Arguments

`data`	The data set to be explored, which can be a data.frame or matrix
`k`	The kth-distance to be used to calculate LOFs. k can be a vector which contains multiple k values based on which LOFs need to be calculated.
`cores`	optional, The number of cores to be used for parallel computing. If not provided, the maximum number of cores available is used by default.
`...`	The parameters to be passed to `distmc()` function, specifying the distance measure.

Details

The LOFs are calculated over multiple k values in parallel, and the maximum number of the cpus will be utilised to achieve the best performance.

Value

`lof`	A matrix with the local outlier factor of each observation as rows and each k value as columns

Author(s)

Yingsong Hu, Wayne Murray and Yin Shan, Australia

References

Breuning, M., Kriegel, H., Ng, R.T, and Sander. J. (2000). LOF: Identifying density-based local outliers. In Proceedings of the ACM SIGMOD International Conference on Management of Data.

Examples

```
## Not run: ---- Detecting the top outliers using the LOF algorithm
## Not run: ---- with k = 5,6,7,8,9 and 10, respectively----
data(iris)
df<-iris[-5]
df.lof<-lof(df,c(5:10),cores=2)
```

Index