Package 'sprint'

February 20, 2015

Title Simple Parallel R INTerface

Version 1.0.7

Date 2014-09-24

Author University of Edinburgh SPRINT Team <sprint@ed.ac.uk>

Copyright University of Edinburgh

Maintainer Eilidh Troup <e.troup@epcc.ed.ac.uk>

Contact University of Edinburgh SPRINT Team <sprint@ed.ac.uk>

Depends R (>= 2.9.2), rlecuyer, ff (>= 2.1-1), randomForest

Suggests cluster, stringdist, RUnit, Matrix, SparseM, multtest, Biostrings, ShortRead, golubEsets, RankProd

Imports boot, e1071

SystemRequirements MPI(>= 2.0)

Description SPRINT (Simple Parallel R INTerface) is a parallel framework for R. It provides a High Performance Computing (HPC) harness which allows R scripts to run on HPC clusters. SPRINT contains a library of selected R functions that have been parallelized. Functions are named after the original R function with the added prefix 'p', i.e. the parallel version of cor() in SPRINT is called pcor(). Call to the parallel R functions are included directly in standard R scripts.
SPRINT contains functions for correlation (pcor), partitioning around medoids (ppam), apply (papply), permutation testing (pmaxT), bootstrapping (pboot), random forest (prandomForest),
rank product (pRP) and hamming distance (pstringdistmatrix).

License GPL (>= 3)

URL http://www.r-sprint.org

Repository CRAN

Date/Publication 2014-09-29 18:03:02

NeedsCompilation yes

OS_type unix

R topics documented:

`About SPRINT` *Overview of SPRINT*

Description

SPRINT (Simple Parallel R INTerface) is a parallel framework for R. It provides a High Performance Computing (HPC) harness which allow R scripts to run on HPC clusters. SPRINT contains a library of selected R functions that have been parallelized. Functions are named after the original R function with the added prefix 'p', i.e. the parallel version of cor() in SPRINT is called pcor(). These parallelized functions are written in C and MPI. Call to these functions are included directly in standard R scripts.

The following functions are implemented in SPRINT 1.5.0: - papply - pboot - pcor - pmaxT - ppam - prandomForest - pRP - pstringdistmatrix - pterminate - ptest

See the User Guide and Release Notes in the sprint folder or the SPRINT web page at http://www.r-sprint.org for more information.

Details

To make use of SPRINT it is necessary to include the library first. Then include calls to the SPRINT functions you want to use. It is also necessary to exit SPRINT using the pterminate function which shutdown MPI as well as SPRINT.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

Examples

```
library("sprint")
ptest()
pterminate()
quit()
```

`init.rng` *init.rng*

Description

Internal utility function.

`papply` *Parallel Apply*

Description

Parallel apply function is used to perform the same operation over all the elements of data objects like matrices, or lists. This function provides a parallel implementation of both the apply() and lapply() functions from the core of the R programming language. The papply() function only accepts matrices or lists of matrices as input data objects.

Usage

```
papply(data, fun, margin = 1, out_filename = NULL)
```

Arguments

`data`	input data matrix or list of matrices
`fun`	function to be applied
`margin`	vector indicating which elements of the matrix the function will be applied to. The default value is 1 and indicates the rows, 2 indicates the columns and the parameter is ignored if data is a list.
`out_filename`	string, not used at present.

Details

The function to be applied can be supplied to papply() either as a function name or as a function definition. When only the function name is provided, the package implementing the function has to be loaded before the SPRINT library is initialised in order to ensure that the name is recognised by all the processes involved in the computation.

Note that papply() does not fully implement lapply functionality. It will only accept lists of matrices, and not lists made up of other data types.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

apply lapply ff SPRINT

---

pboot *Parallel Bootstrapping*

Description

pboot() generates R bootstrap replicates of a statistic applied to data. It implements a parallel version of the bootstrapping method boot() from the boot R package.

Arguments

data	array of data, if a 2D array then each row is considered as one multivariate observation
statistic	function, when sim is set to parametric, the first argument to statistic must be the data. For each replicate a simulated dataset returned by ran.gen will be passed. In all other cases, statistic must take at least two arguments. The first argument passed will always be the original data. The second will be a vector of indices, frequencies or weights which define the bootstrap sample.
R	number of bootstrap replicates
sim	string, indicates the type of simulation. The default value is "ordinary". Other possible values are parametric, balanced, permutation, and antithetic. Importance resampling is specified by including importance weights; the type of importance resampling must still be specified but may only be ordinary or balanced in this case.
stype	string, indicates what the second argument of statistic represents. The default value is i for indices. Other possible values are f for frequencies and w for weights. It is not used when sim is set to parametric.
strata	vector of integer, specifies the strata for multi-sample problems. This may be specified for any simulation, but is ignored when sim is set to parametric. When strata is supplied for a nonparametric bootstrap, the simulations are done within the specified strata.
L	vector of influence values evaluated at the observations. This is used only when sim is set to antithetic. If not supplied, they are calculated through a call to empinf. This will use the infinitesimal jackknife provided that stype is set to w otherwise the usual jackknife is used.

m	the number of predictions which are to be made at each bootstrap replicate. This is most useful for (generalized) linear models. This can only be used when sim is ordinary. m will usually be a single integer but, if there are strata, it may be a vector with length equal to the number of strata, specifying how many of the errors for prediction should come from each strata. The actual predictions should be returned as the final part of the output of statistic, which should also take an argument giving the vector of indices of the errors to be used for the predictions.
weights	array of importance weights. If a vector then it should have as many elements as there are observations in the input data. When simulation from more than one set of weights is required, weights should be a matrix where each row of the matrix is one set of importance weights. If weights is a matrix then the number of bootstrap replicates R must be a vector of length nrow(weights). This parameter is ignored if sim is not set to ordinary or balanced.
ran.gen	function, used only when sim is set to parametric. It describes how random values are to be generated. It should be a function of two arguments. The first argument should be the observed data and the second argument consists of any other information needed (e.g. parameter estimates). The second argument may be a list, allowing any number of items to be passed to ran.gen. The returned value should be a simulated data set of the same form as the observed data which will be passed to statistic to get a bootstrap replicate. It is important that the returned value be of the same shape and type as the original dataset. If ran.gen is not specified, the default is a function which returns the original input data in which case all simulation should be included as part of statistic. Setting sim to parametric and using a suitable ran.gen allows the user to implement any types of nonparametric resampling which are not supported directly.
mle	secong argument to ran.gen, typically these will be maximum likelihood estimates of the parameters. For efficiency mle is often a list containing all of the objects needed by ran.gen which can be calculated using the original data set only.
simple	boolean, can only be set to TRUE if sim is set to ordinary, stype is set to I and n is set to 0. Otherwise it is ignored and generates a warning. By default a n by R index array is created which can be large. If simple is set to TRUE, this is avoided by sampling separately for each replication, which is slower but uses less memory.
...	other named arguments for statistic which are passed unchanged each time.

Details

This version is an early but fully working prototype. However, it is not compatible with other SPRINT functions, i.e. you cannot bootstrap other parallel functions from the SPRINT library. It is therefore recommended to use it only as a standalone function.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

boot SPRINT

---

pcombine *pcombine*

Description

Internal utility function.

---

pcor *Parallel Correlation*

Description

Parallel Pearson's correlation. It either takes a 2D array as input and correlates each row with every other row or takes two 2D arrays and correlates the columns of the first matrix with the columns of the second matrix. The output can either be the matrix of correlation coefficient or the distance matrix.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Usage

```
pcor(data_x, data_y = NULL, distance = FALSE, caching_ = "mmeachflush",
     filename_ = NULL)
```

Arguments

data_x	double precision 2D array of data
data_y	NULL or second double precision 2D array of data
distance	boolean, whether the distance or correlation coefficient matrix is returned
caching_	string, either "mmeachflush" or "mmnoflush" select the back-end caching scheme
filename_	string, name of the result file

Value

An ff_matrix object. The results of a correlation computation can be very large and so SPRINT returns a file-backed ff_matrix object instead of a standard R matrix object.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

cor SPRINT ff

pmaxT *Ajusted p-values for step down multiple testing*

Description

Function which implements a parallel version of the multtest "mt.maxT" function. It computes the adjusted p-values for step-down multiple testing procedures.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Usage

```
pmaxT(X, classlabel, test = "t", side = "abs", fixed.seed.sampling = "y",
      B = 10000, na = .naNUM, nonpara = "n")
```

Arguments

X	double precision 2D array of data
classlabel	class column labels of the input data array
test	one of the following statistical test: t. t.equalvar, Wilcoxon, F, Pair-T, Block-F
side	Type of rejection region, either abs, upper or lower
fixed.seed.sampling	whether the permutations are calculated on the fly or save to memory
B	number or permutations
na	missing value tag
nonpara	whether non-parametric test statistics or not

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

mt.maxT SPRINT

ppam *Parallel Partitioning Around Medoids*

Description

Parallel implementation of the Partitioning Around Medoids algorithm, based on the cluster "pam" serial function.

Usage

```
ppam(x, k, medoids = NULL, is_dist = inherits(x, "dist"),
     cluster.only = FALSE, do.swap = TRUE, trace.lev = 0)
```

Arguments

x	input data, either a 2D array or an ff object
k	positive integer, indicating for the number of clusters
medoids	vector, with the initial 'k' medoids or NULL to let the algorithm select the initial medoids
is_dist	boolean, whether the input data is a distance or dissimilarity matrix or a symmetric matrix
cluster.only	boolean, whether only the clustering is computed and returned
do.swap	boolean, whether the swap phase of the algorithm is required
trace.lev	positive integer for the level of details returned for diagnostics

Details

The interface and parameters to parallel function ppam() are similar to the serial function pam() but not identical. ppam() requires a distance matrix as input parameters. Although, ppam() does not include the option to calculate the distance matrix, this can easily be done using SPRINT pcor() function with the 'distance' parameter set to TRUE.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

pam ff pcor SPRINT

prandomForest *Parallel random forest generation*

Description

The machine learning function prandomForest() is an ensemble tree classifier that constructs a forest of classification trees from bootstrap samples of a dataset in parallel. The random forest algorithm can be used to classify both categorical and continuous variables. This function provides a parallel equivalent to the serial randomForest() function from the randomForest package. Note that the randomForest library must be loaded before calling the prandomForest function. library("randomForest")

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Usage

```
prandomForest(x, ...)
## Default S3 method:
prandomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
                     mtry = if (!is.null(y) && !is.factor(y))
                               max(floor(ncol(x)/3), 1)
                             else floor(sqrt(ncol(x))),
                     replace=TRUE, classwt=NULL, cutoff, strata,
                     sampsize = if (replace) nrow(x)
                               else ceiling(.632*nrow(x)),
                     nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
                     maxnodes=NULL, importance=FALSE, localImp=FALSE,
                     nPerm=1, proximity, oob.prox=proximity, norm.votes=TRUE,
                     do.trace=FALSE,
                     keep.forest = !is.null(y) && is.null(xtest),
                     corr.bias=FALSE, keep.inbag=FALSE, ...)
```

Arguments

x	array of data
...	optional parameters to be passed to the low level function randomForest.default.
y	vector, if a factor, classification is assumed, otherwise regression is assumed. If omitted, prandomForest() will run in unsupervised mode.
xtest	data array of predictors for the test set
ytest	response for the test set
ntree	integer, the number of trees to grow
mtry	integer, the number of variables randomly sampled as candidates at each split. The default value is sqrt(p) for classification and p/3 for regression, where p is the number of variables in the data matrix x.

replace	boolean, whether the sampling of cases is done with or without replacement. The default value is TRUE.
classwt	vector if priors of the classes. The default value is NULL.
cutoff	vector of k elements where k is the number of classes. The winning class for an observation is the one with the maximum ratio of proportion of votes to cutoff. The default value is 1/k.
strata	variable used for stratified sampling
sampsize	size of sample to draw. For classification, if sampsize is a vector of the length of the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	integer, the minimum size of the terminal nodes. The default value is 1 for classification and 5 for regression.
maxnodes	integer, maximum number of terminal nodes allowed for the trees. The default value is NULL.
importance	boolean, whether the importance of predictors is assessed. The default value is FALSE.
localImp	boolean, whether casewise importance measure is to be computed. The default value is FALSE.
nPerm	integer, the number of times the out-of-bag data are permuted per tree for assessing variable importance. The default value is one. Regression only.
proximity	boolean, whether the proximity measure among the rows is to be calculated.
oob.prox	boolean, whether the proximity is to be calculated for out-of-bag data. The default value is set to be the same as the value of the proximity parameter.
norm.votes	boolean, whether the final result of votes are expressed as fractions or whether the raw vote counts are returned. The default value is TRUE. Classification only.
do.trace	boolean, whether a verbose output is produced. The default value is FALSE. If set to an integer i then the output is printed for every i trees.
keep.forest	boolean, whether the forest is returned in the output object. The default value is FALSE.
corr.bias	boolean, whether to perform a bias correction. The default value is FALSE. Regression only.
keep.inbag	boolean, whether the matrix which keeps track of which samples are in-bag in which trees should be returned. The default value is FALSE.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

randomForest SPRINT

pRP *Parallel Rank Product*

Description

Parallel rank product function helps identifying differentially regulated genes in replicated microarray experiments.

Usage

```
pRP(data, cl, num.perm, logged = TRUE, na.rm = FALSE, gene.names = NULL,
    plot = FALSE, rand = NULL, sum = FALSE)
```

Arguments

`data`	array, input data
`cl`	vector, class labels of the samples
`num.perm`	integer, the number of permutations used in the calculation of the null density. The default value is 100.
`logged`	boolean, whether the data is logged or not. The default value is TRUE.
`na.rm`	boolean, whether missing values are to be replaced by the gene-wise mean of the non-missing values and used in computing rank. The default value is FALSE.
`gene.names`	the gene name to be assigned to the estimated percentage of false positive predictions. The default value is NULL.
`plot`	boolean, whether to plot the estimated percentage of false positive predictions against the rank of each gene. The default value is FALSE.
`rand`	number, the seed for the random number generator if specified. The default value is NULL.
`sum`	boolean, whether to perform a rank sum analysis. The default value is NULL.

Details

The SPRINT task parallel implementation of the rank product method is approximately twice as fast in serial as the existing RP() function from the RankProd package and it shows excellent scaling.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

RP SPRINT

pRPadvance *pRPadvance*

Description

Internal utility function.

pRS *pRS*

Description

Internal utility function.

pRSadvance *pRSadvance*

Description

Internal utility function.

pstringdistmatrix *Parallel compute hamming distance between strings*

Description

Calculates the hamming distance matrix between strings.

Usage

```
pstringdistmatrix(a, b, method = "h", filename = NULL, weight = NULL,
                  maxDist = 0, ncores = NULL)
```

Arguments

a	a: R object (target); will be converted by 'as.character'.
b	b: R object (source); will be converted by 'as.character'. Must be the same as argument a in this version of the software.
method	Method for distance calculation - only option 'h' for hamming distance is supported.
filename	Results will be stored here as binary data
weight	Not used in the hamming distance measure.
maxDist	Not used in the hamming distance measure.
ncores	Not used by SPRINT, please see the SPRINT user guide.

Details

Calculates the hamming distance between each pair of strings. Returns an ff result matrix.

N.B. Please see the SPRINT User Guide for how to run the code in parallel using the mpiexec command.

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

stringdistmatrix SPRINT

---

pterminate *SPRINT close*

Description

Function which indicates the end of the use of the SPRINT library. It terminates the use of MPI and shut down the SPRINT library. It must be used with every script that invokes the SPRINT package.

Usage

```
pterminate()
```

Arguments

None

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

SPRINT

ptest *SPRINT Installation Test*

Description

Function that test the correct installation of the SPRINT library. It simply prints a message identifying each processor in the compute cluster.

Usage

```
ptest()
```

Arguments

None

Author(s)

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> www.r-sprint.org

See Also

SPRINT

reset.rng *reset.rng*

Description

Internal utility function.

Index