

# Package ‘sprint’

April 27, 2012

**Title** Simple Parallel R INTerface

**Version** 1.0.2

**Date** 2012-04-19

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

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

**Depends** R (>= 2.9.2), rlecuyer

**Suggests** ff (>= 2.1-1), randomForest, RankProd, cluster, multtest, IRanges, RUnit

**Imports** boot

**SystemRequirements** MPI2, Unix

**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(). Call to the parallel R functions are included directly in standard R scripts.

**License** GPL (>= 3)

**URL** <http://www.sprint-r.org>

**Repository** CRAN

**Date/Publication** 2012-04-27 13:00:51

## R topics documented:

About SPRINT	2
init.rng	3
papply	3
pboot	4
pcombine	6
pcor	6
phamming.distance	7
pmaxT	7
ppam	8
prandomForest	9
pRP	11
pRPadvance	12
pRS	12
pRSadvance	12
pterminate	13
ptest	13
reset.rng	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

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 1.0.0: - `papply` - `pboot` - `pcor` - `pmaxT` - `ppam` - `prandomForest` - `pRP` - `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](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://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 used to perform the same operation over all the elements of data objects like matrices, data frames or lists. This function provides a parallel implementation of both the `apply()` and `lapply()` functions from the core of the R programming language. This parallel implementation `papply()` has been extended to accept an `ff` object as input which allows the processing of much larger data stored on disk.

**Usage**

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

**Arguments**

<code>data</code>	array, list or <code>ff</code> object
<code>fun</code>	function to be applied
<code>margin</code>	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.
<code>out_filename</code>	string, name of the result file when input is an <code>ff</code> object

**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.

**Author(s)**

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://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	second 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.

### Author(s)

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://www.r-sprint.org)

**See Also**

[boot SPRINT](#)

---

pcombine	<i>pcombine</i>
----------	-----------------

---

**Description**

Internal utility function.

---

pcor	<i>Parallel Correlation</i>
------	-----------------------------

---

**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.

**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

**Author(s)**

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://www.r-sprint.org)

**See Also**

[cor SPRINT](#)

---

phamming.distance      *Parallel hamming distance*

---

**Description**

Calculates the hamming distance matrix between a set of DNA sequences provided as a ShortReadQ or DNASTringSet objects.

**Usage**

```
phamming.distance(data, output_filename)
```

**Arguments**

data                    ShortReadQ or DNASTringSet objects  
output\_filename        results will be stored here as binary data

**Details**

This is an early but fully working prototype.

**Author(s)**

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> [www.r-sprint.org](http://www.r-sprint.org)

---

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.

**Usage**

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

**Arguments**

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

**Author(s)**

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://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**

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

**Author(s)**

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://www.r-sprint.org)

**See Also**

[pam](#) [ff](#) [pcor](#) [SPRINT](#)

---

prandomForest	<i>Parallel random forest generation</i>
---------------	--

---

**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.

**Usage**

```
prandomForest(x, ...)
```

**Arguments**

<code>x</code>	array of data
<code>...</code>	optional parameters to be passed to the low level function <code>randomForest.default</code> .

**Details**

`prandomForest` use relies on typical usage of the underlying `randomForest` function. The default paramaters are:

```
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, ...)
```

```
\item{y}{vector, if a factor, classification is assumed, otherwise
  regression is assumed. If omitted, prandomForest() will run
  in unsupervised mode.}
\item{xtest}{data array of predictors for the test set}
\item{ytest}{response for the test set}
\item{ntree}{integer, the number of trees to grow}
\item{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  $x$ .}
\item{replace}{boolean, whether the sampling of cases is done with or
  without replacement. The default value is TRUE.}
\item{classwt}{vector if priors of the classes. The default value is
  NULL.}
\item{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  $1/k$ .}
\item{strata}{variable used for stratified sampling}
\item{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.}
\item{nodesize}{integer, the minimum size of the terminal nodes. The
  default value is 1 for classification and 5 for
  regression.}
\item{maxnodes}{integer, maximum number of terminal nodes allowed for
  the trees. The default value is NULL.}
\item{importance}{boolean, whether the importance of predictors is
  assessed. The default value is FALSE.}
\item{localImp}{boolean, whether casewise importance measure is to be
  computed. The default value is FALSE.}
\item{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.}
\item{proximity}{boolean, whether the proximity measure among the rows
  is to be calculated.}
\item{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.}
\item{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.}
\item{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.}
\item{keep.forest}{boolean, whether the forest is returned in the
output object. The default value is FALSE.}
\item{corr.bias}{boolean, whether to perform a bias correction. The
default value is FALSE. Regression only.}
\item{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.}
\item{...}{optional parameters to be passed to the low level function
randomForest.default.}

```

**Author(s)**

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> [www.r-sprint.org](http://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.

<code>gene.names</code>	the gene name to be assigned to the estimated percentage of false positive predictions. The default value is NULL.
<code>plot</code>	boolean, whether to plot the estimated percentage of false positive predictions against the rank of each gene. The default value is FALSE.
<code>rand</code>	number, the seed for the random number generator if specified. The default value is NULL.
<code>sum</code>	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.

**Author(s)**

University of Edinburgh SPRINT Team <[sprint@ed.ac.uk](mailto:sprint@ed.ac.uk)> [www.r-sprint.org](http://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.

---

pterminate	<i>SPRINT close</i>
------------	---------------------

---

**Description**

Dunction 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](http://www.r-sprint.org)

**See Also**

[SPRINT](#)

---

pctest	<i>SPRINT Installation Test</i>
--------	---------------------------------

---

**Description**

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

**Usage**

```
pctest()
```

**Arguments**

None

**Author(s)**

University of Edinburgh SPRINT Team <sprint@ed.ac.uk> [www.r-sprint.org](http://www.r-sprint.org)

**See Also**

[SPRINT](#)

---

`reset.rng`*reset.rng*

---

**Description**

Internal utility function.

# Index

## \*Topic **interface**

About SPRINT, 2  
papply, 3  
pboot, 4  
pcor, 6  
phamming.distance, 7  
pmaxT, 7  
ppam, 8  
prandomForest, 9  
pRP, 11  
pterminate, 13  
ptest, 13

## \*Topic **utilities**

About SPRINT, 2  
papply, 3  
pboot, 4  
pcor, 6  
phamming.distance, 7  
pmaxT, 7  
ppam, 8  
prandomForest, 9  
pRP, 11  
pterminate, 13  
ptest, 13

About SPRINT, 2

apply, 4

boot, 6

cor, 6

ff, 4, 9

init.rng, 3

lapply, 4

mt.maxT, 8

pam, 9

papply, 3  
pboot, 4  
pcombine, 6  
pcor, 6, 9  
phamming.distance, 7  
pmaxT, 7  
ppam, 8  
prandomForest, 9  
pRP, 11  
pRPadvance, 12  
pRS, 12  
pRSadvance, 12  
pterminate, 13  
ptest, 13

randomForest, 11  
reset.rng, 14  
RP, 12

SPRINT, 4, 6, 8, 9, 11–13

SPRINT (About SPRINT), 2