# Package 'EXPERT'

December 6, 2010

**Title** A package for Extremely small P-value Evaluation for Resampling-based Tests

**Version** 0.0.1

**Date** 2010-12-06

**Author** Kai Yu and Faming Liang

**Maintainer** Kai Yu <yuka@mail.nih.gov>

**Description** A package for efficient p-value evaluation for resampling-based tests

**Depends** R (>= 2.10.1)

**License** GPL-2

## R topics documented:



---

EXPERT *A package for Extremely small P-value Evaluation for Resampling-based Tests*

---

### Description

An R package for efficient p-value estimation for resampling-based tests.

### Details

The main function in this package is SAMC.adapt which uses the SAMC (Stochastic Approximation Monte Carlo) method to estimate the p-value for an observed test statistic. In order to use this function, the user must provide two functions: one function for calculating the test statistic, and another function for updating the input data.

### Author(s)

Kai Yu <yuka@mail.nih.gov> and Faming Liang

### References

Yu K, Liang F, Ciampa J, Chatterjee N. Efficient p-value evaluation for resampling-based tests (Submitted manuscript).

---

## MAX.test.statistic *Calculate the maximum value of multiple Cochran-Armitage trend test statistics*

---

### Description

A function to calculate the MAX test statistic, i.e., the maximum value of multiple Cochran-Armitage trend test statistics performed on a set genetic markers (e.g. SNPs).

### Usage

```
MAX.test.statistic(data.input)
```

### Arguments

| | |
|---|---|
| `data.input` | A list of two components, x and y, which are matrices containing genotypes on a set of genetic markers from cases and controls, where the rows are for subjects and columns for the markers. Each element takes the value 0, 1, or 2, representing the number of copies of a certain allele in the genotype. |

### Value

The value of the test statistic

### See Also

`proposal.permute.matrix`

### Examples

```
MAX.test.statistic
```

proposal.permute.matrix *Example function for the fun.proposal argument of the SAMC.adapt() function*

## Description

This function updates the inpute data by randomly switching any chosen pair of rows from two matrices.

## Usage

```
proposal.permute.matrix(data.input, prop.change)
```

## Arguments

| | |
|---|---|
| data.input | A list of two components, x and y, which are matrices containing genotypes on a set of genetic markers from cases and controls, where the rows are for subjects and columns for the markers. Each element takes the value 0, 1, or 2, representing the number of copies of a certain allele in the genotype. |
| prop.change | The proportion of the rows in the smaller matrix to be switched with chosen rows from the other matrix. |

## Details

This function can be used together with MAX.test.statistic. See the reference on exactly how the permutations are conducted.

## Value

The updated version of data.input.

## References

Yu K, Liang F, Ciampa J, Chatterjee N. Efficient p-value evaluation for resampling-based tests (Submitted manuscript).

## See Also

MAX.test.statistic, SAMC.adapt

## Examples

```
proposal.permute.matrix
```

`proposal.permute.vector` *Example function for the fun.proposal argument of the SAMC.adapt() function*

## Description

This function updates the inpute data by randomly switching any chosen pair of elements from two vectors.

## Usage

```
proposal.permute.vector(data.input, prop.change)
```

## Arguments

| | |
|---|---|
| `data.input` | The example input data object |
| `prop.change` | The proportion of data to be updated |

## Details

This function can be used together with `t.test.statistic`. See the reference on exactly how the permutations are conducted.

## Value

The updated version of `data.input`.

## References

Yu K, Liang F, Ciampa J, Chatterjee N. Efficient p-value evaluation for resampling-based tests (Submitted manuscript).

## See Also

`t.test.statistic`, `SAMC.adapt`

## Examples

```
proposal.permute.vector
```

SAMC.adapt *SAMC (Stochastic Approximation Monte Carlo) method for small p-value estimation*

## Description

To estimate the p-value for an observed test statistic based on the SAMC (Stochastic Approximation Monte Carlo) method.

## Usage

```
SAMC.adapt(data.input, t.obs, t.start=0, n.iter.1=200000, n.iter.2=1000000,
      n.region.1=101, n.region.2=301, prop.change=0.05, gain.factor.t0=1000,
      fun.test.statistic=NULL, fun.proposal=NULL, ...)
```

## Arguments

| | |
|---|---|
| `data.input` | The input data, which can be any data object, e.g. data frame, list, etc., as long as it is compatible with the two user provided functions `fun.test.statistic` and `fun.proposal`. |
| `t.obs` | The threshold at which the tail probability is to be calculated. For example, this value can be set to the observed test statistic. The estimated p-value for this value will be computed. |
| `t.start` | The minimum value for the considered test statistic. The default is 0. |
| `n.iter.1` | The number of iterations used for the initial run with subregions defined uniformly in the interval [`t.start`, `t.obs`], with final region [`t.obs`, $\infty$). The default is 200000. |
| `n.iter.2` | The number of iterations used for the final run with subregions defined according to the results from the initial run. If `n.iter.2` = 0, then only the initial run is performed. The default is 1000000. |
| `n.region.1` | The number of subregions used by SAMC in the initial run. The default is 101. |
| `n.region.2` | The number of subregions used by SAMC in the final run. The default is 301. |
| `prop.change` | The proportion of data to be updated by `fun.proposal` in each iteration. The default is 0.05. |
| `gain.factor.t0` | This is used for defining the gain factor sequence. The default is 1000. |
| `fun.test.statistic` | A function written by the user to calculate and return the test statistic based on `data.input`. It has the following format: `fun.test.statistic(data.input, ...)`. See the example function `t.test.statistic`. |
| `fun.proposal` | A function written by the user to update the input data and return the updated data. It should be in the following format: `fun.proposal(data.input, prop.change, ...)`. See the example function `proposal.permute.vector`. |
| `...` | Other arguments for `fun.test.statistic` and/or `fun.proposal` |

## Value

A list with two components `hist.mat` and `p.value`. The object `hist.mat` will be a data frame of dimension `n.region.2` by 4 (or `n.region.1` by 4 if `n.iter.2` = 0), which contains information used for checking convergence. The other component of each sublist is `p.value`, which is the estimated p-value. The `samplefreq` column of `hist.mat` provides the number of times the SAMC iterations generate a sample falling into each individual sub-region. A barplot of `samplefreq` (barplot(hist.mat[, "samplefreq"]) should show a flat pattern.

## Author(s)

Kai Yu

## References

Yu K, Liang F, Ciampa J, Chatterjee N. Efficient p-value evaluation for resampling-based tests (Submitted manuscript).

## See Also

`t.test.statistic`, `proposal.permute.vector`, `SAMC.fun`

## Examples

```
# Example 1: An example of two-sample t test.
# Suppose we do not know the theoretical distribution of t test statistic,
# but instead want to rely on a permutation procedure to evaluate the significance
# level for a given observed statistic value. In this permutation procedure, we
# randomly shuffle the group IDs among the two samples and generate new versions of
# the data. We can apply the SAMC algorithm to estimate the p-value (tail probability)
# at a much reduced number of iterations than the standard permutation procedure.
# First, we need to define the two functions for running SAMC, fun.test.statistic,
# and fun.proposal. In the SPERT package, we provide such two functions. They are
# t.test.statistic, which calculates the two sided t-test statistic, and
# proposal.permute.vector, which update the data through permutation.

 set.seed(1)
x<-rnorm(200, mean=0, sd=1)
y<-rnorm(200, mean=0.5, sd=1)
data.input<-list(x=x, y=y)
t.obs<-t.test.statistic(data.input)

# True p-value
p.true <- 2.0*(1-pt(t.obs, df=398))
p.true

# Run SAMC to estimate the p-value
res<-SAMC.adapt(data.input, t.obs, t.start=0, n.iter.1=5000,
            n.iter.2=10000, n.region.1=101, n.region.2=201,
          prop.change=0.05, gain.factor.t0=1000,
          fun.test.statistic=t.test.statistic, fun.proposal=proposal.permute.vector)

# The estimated p-value
res$p.value

# To check for convergence:
barplot(res$hist.mat[, "samplefreq"])
```

```
# Note that the barplot is not flat, so the algorithm has not converged yet.
# To get a more precise estimate run the code below.
# WARNING: the following call to SAMC.adapt could take around 20 minutes to run.
ptm <- proc.time()
#res<-SAMC.adapt(data.input, t.obs, t.start=0, n.iter.1=200000,
#            n.iter.2=1000000, n.region.1=101, n.region.2=301,
#         prop.change=0.05, gain.factor.t0=1000,
#         fun.test.statistic=t.test.statistic, fun.proposal=proposal.permute.vector)
proc.time()-ptm
```

---

`SAMC.fun` *SAMC algorithm*

---

## Description

This function is to run the SAMC algorithm with user defined subregion definition.

## Usage

```
SAMC.fun(data.input, n.iter, partition.vec, prop.change, theta.start,
      gain.factor.t0, fun.test.statistic, fun.proposal, ...)
```

## Arguments

| | |
|---|---|
| `data.input` | The input data, which can be any data object, e.g. data frame, list, etc., as long as it is compatible with the two user provided functions `fun.test.statistic` and `fun.proposal`. |
| `n.iter` | The number of iterations |
| `partition.vec` | A vector of length m of cut points at the domain of the test statistic. The first subregion is defined as [`partition.vec[1]`, `partition.vec[2]`], the second subregion is [`partition.vec[2]`, `partition.vec[3]`], ..., with final subregion being [`partition.vec[m]`, $\infty$). Usually `partition.vec[1]` is the minimum value of the test statistic (e.g. 0), and `partition.vec[m]` can be set as the observed test statistic whose p-value is to be estimated. |
| `prop.change` | The percentage of data to be updated by `fun.proposal` in each iteration. |
| `theta.start` | theta.start |
| `gain.factor.t0` | This is used for defining the gain factor sequence. |
| `fun.test.statistic` | A function written by the user to calculate and return the test statistic based on `data.input`. It has the following format: `fun.test.statistic(data.input, ...)`. See the example function `t.test.statistic`. |
| `fun.proposal` | A function written by the user to update the input data and return the updated data. It should be in the following format: `fun.proposal(data.input, prop.change, ...)`. See the example function `proposal.permute.vector`. |
| `...` | Other arguments for `fun.test.statistic` and/or `fun.proposal` |

### Details

This function estimates the tail probability

$$P(\lambda(x) \geq partition.vec[m])$$

### Value

A list with two components `hist.mat` and `p.value` (see `SAMC.adapt`).

### Author(s)

Kai Yu

### References

Yu K, Liang F, Ciampa J, Chatterjee N. Efficient p-value evaluation for resampling-based tests (Submitted manuscript).

### See Also

`SAMC.adapt`, `t.test.statistic`, `proposal.permute.vector`

---

## `t.test.statistic` *Calculate a 2 sample t-test statistic*

---

### Description

A function to calculate a 2 sample t-test statistic which can be used as an example function for the fun.test.statistic argument of the SAMC.adapt function.

### Usage

```
t.test.statistic(data.input)
```

### Arguments

| | |
|---|---|
| `data.input` | A list of two components, x and y, which are vectors containing measurements from one of the two samples. |

### Value

The value of the test statistic

### See Also

`proposal.permute.vector`

### Examples

```
t.test.statistic
```

# Index