

Package ‘SHARE’

January 2, 2012

Type Package

Title SNP-Haplotype Adaptive REgression (SHARE)

Version 1.1.0

Date 2010-11-23

Author James Y. Dai

Maintainer Ting-Yuan Liu <ting@fhcrc.org>

Description An adaptive algorithm to select the most informative set of SNPs for genetic association

License GPL (>= 2)

LazyLoad no

Depends haplo.stats, MASS, methods

Collate S4-class.R genoSet-class.R haploSet-class.R haplo-class.R
share-class.R finalsubset.R cshare.R zzz.SHARE.R

Repository CRAN

Date/Publication 2010-11-24 07:49:55

R topics documented:

SHARE-package	2
cshare	2
genoSet	4
haplo	5
haplo-class	6
haploSet-class	7
keremRand	8
nameSeq-methods	11
nameSNP-methods	11
nSeq-methods	12
nSNP-methods	12
share-class	12
shareTest	14

Index**16**

SHARE-package	<i>An adaptive algorithm to select the most informative set of SNPs for genetic association</i>
---------------	---

Description

This is the R package to perform the adaptive algorithm, developed by James Dai, et al., to select the most informative SNP set for genetic association.

Details

Association studies have been widely used to identify genetic liability variants for complex diseases. While scanning the chromosomal region one SNP at a time may not fully explore linkage disequilibrium (LD), haplotype analyses tend to require a fairly large number of parameters, thus potentially losing power. Clustering algorithms, such as the cladistic approach, have been proposed to reduce the dimensionality, yet they have important limitations. We propose the SHARE algorithm that seeks the most informative set of SNPs for genetic association in a targeted region by growing/shrinking haplotypes with one more/less SNP in a stepwise fashion, and comparing prediction errors of different models via cross-validation. The model can also accommodate for non-genetic covariates.

Author(s)

James Y. Dai and Ting-Yuan Liu

References

Dai, J. Y., LeBlanc, M., Smith, N. L., Psaty, B. M. and Kooperberg, C. (2009). SHARE: an adaptive algorithm to select the most informative set of SNPs for genetic association. *Biostatistics*, In Press.

cshare	<i>Stepwise search for the most informative haplotypes</i>
--------	--

Description

The cshare function seeks the most informative set of SNPs for genetic association in a targeted region by growing/shrinking haplotypes with one more/less SNP in a stepwise fashion, and comparing prediction errors of different models via cross-validation or BIC.

Usage

```
cshare(haploObj, status, ncover = 0, covar = NULL, nfold = 10, maxsnps, tol = 1e-08,
       verbose = FALSE, ModSelMethod = c("Cross-Val", "BIC"),
       Minherit = c("additive", "dominant", "recessive"))
```

Arguments

haploObj	A object of calss "haplo". haploObj could be generated by the function haplo.
status	A character string indicating the column name of the phenotype in haploObj@pheno to be used as the clinical status in the analysis.
ncovar	An integer indicating the number of non-genetic covariates.
covar	A matrix of non-genetic covariates where the first column is the subject ID numbers that match the names in haploObj@nPosHapPair. Missing values are not allowed.
ifold	An integer that determines how many folds in the cross-validation, if ModSelMethod="Cross-Val"
maxsnps	An integer that determines the maximal number of SNPs to be chosen. The default is 6.
tol	The convergence parameters in haplotype logistic regression
verbose	TRUE/FALSE to decide whether to create log file for debug
ModSelMethod	Model selection method. Possible methods are "Cross-Val" for cross-validation, and "BIC" for Bayesian information criterion
Minherit	Mode of inheritance. Possible mode are "additive", "dominant", and "recessive"

Details

This function takes input from a phased genotype dataset and case-control status, performs stepwise search for the most informative haplotypes based on deviance criterion, either by cross-validation or by BIC. The function output the prediction deviances of a ladder of models up to the size of "maxsnps" values and best haplotype model selected.

Value

cshare returns a object of class "share"

Author(s)

James Y. Dai

References

Dai, J. Y., LeBlanc, M., Smith, N. L., Psaty, B. M. and Kooperberg, C. (2009). SHARE: an adaptive algorithm to select the most informative set of SNPs for genetic association. Biostatistics, In Press.

See Also

[haplo](#), [share-class](#)

Examples

```
## See vignette for more details
## Not run:
unphasedKerem <- list()
unphasedKerem[["Cross-Val"]] <- cshare(unphasedHaplo, status="CF",
                                     nfold=20, maxsnps=5,
                                     ModSelMethod="Cross-Val",
                                     Minherit="additive")
unphasedKerem[["Cross-Val"]]

unphasedKerem[["BIC"]] <- cshare(unphasedHaplo, , status="CF",
                                 maxsnps=5, ModSelMethod="BIC", Minherit="additive", verbose=1)
unphasedKerem[["BIC"]]

## End(Not run)
```

genoSet

Genotype Sequence Set

Description

A class for storing genotype sequences and phenotype information.

Objects from the Class

Objects can be created by calls of the form `new("genoSet", ...)`.

Slots

genoSeq: Object of class "data.frame" to store allelic data with sequence names as row names and SNP names as column names.

phenoData: Object of class "data.frame" to store phenotype information for each genotype sequence. The names of the phenotype are used as the column names, and the row names should match the sequence names in the genoSeq slot.

Methods

genoSeq signature(.Object = "genoSet"): extract the "data.frame" object in the genoSeq slot.

haplo signature(.Object = "genoSet"): to estimate the haplotype sequences by EM algorithm in the haplo.stats package.

phenoData signature(.Object = "genoSet"): return the "data.frame" object in the phenoData slot.

Author(s)

Ting-Yuan Liu

Examples

```

showClass("genoSet")
## See vignette for more details
## Not run:
unphasedGeno <- new("genoSet",
                    genoSeq = data.frame(keremRandAllele),
                    phenoData = data.frame(CF=keremRandStatus)
                    )

unphasedGeno

## End(Not run)

```

 haplo

Haplotype Estimation

Description

Estimate haplotype sequences and frequencies from genotype sequences by EM algorithm

Usage

```
haplo(.Object, ...)
```

Arguments

<code>.Object</code>	A object of calss "genoSet".
<code>...</code>	Other arguments used in haplo.em from haplo.stats package.

Details

By using the haplo.em function in haplo.stats package, haplo convert unphased genotype sequences into phased haplotype sequences by EM algorithm. Haplotype frequencies are also estimated by haplp.em function.

Value

haplo returns an object of class "haplo" which results in the following slots:

haploSeq	unique haplotypes. This is from the "haplotype" component in the result of haplo.em funciton.
haploFreq	MLE's of haplotype probabilities. This is from the "hap.prob" component in the result of haplo.em funciton.
hap1	the code of first haplotype of each subject. This is from the "hap1code" component in the result of haplo.em funciton.
hap2	the code of first haplotype of each subject. This is from the "hap2code" component in the result of haplo.em funciton.

poolHapPair	Subject IDs. This is from the "subj.id" component in the result of haplo.em funciton.
nPosHapPair	vector for the count of haplotype pairs that map to each subject's marker genotypes. This is from the "nreps" component in the result of haplo.em funciton.
post	vector of posterior probabilities of pairs of haplotypes for a person, given their marker phenotypes. This is from the "post" component in the result of haplo.em funciton.

Author(s)

James Y. Dai

References

haplo.em

See Also

[genoSet](#), [haplo-class](#)

Examples

```
## See vignette for more details
## Not run:
unphasedHaplo <- haplo(unphasedGeno)
unphasedHaplo

## End(Not run)
```

haplo-class

Haplotype Estimation Set

Description

A class for storing haplotype estimation information.

Objects from the Class

Objects can be created by calls of the form `new("haplo", ...)`.

Slots

hap1: the code of first haplotype of each subject. This is from the "hap1code" component in the result of haplo.em function.

hap2: the code of first haplotype of each subject. This is from the "hap2code" component in the result of haplo.em function.

poolHapPair Subject IDs. This is from the "subj.id" component in the result of haplo.em function.

nPosHapPair: vector for the count of haplotype pairs that map to each subject's marker genotypes. This is from the "nreps" component in the result of haplo.em function.

post: vector of posterior probabilities of pairs of haplotypes for a person, given their marker phenotypes. This is from the "post" component in the result of haplo.em function.

haploSeq: Object of class "data.frame" to store the unique haplotype sequences. This is from the "haplotype" component in the result of haplo.em function.

haploFreq: Object of class "vector" to store the MLE's of haplotype probabilities. This is from the "hap.prob" component in the result of haplo.em function.

pheno: Object of class "data.frame" to store the phenotype information

Extends

Class "[haploSet](#)", directly.

Author(s)

Ting-Yuan Liu

See Also

[haplo](#), [haploSet](#)

Examples

```
showClass("haplo")
```

haploSet-class

Haplotype Set

Description

A class to store the haplotype sequences and their frequencies.

Objects from the Class

Objects can be created by calls of the form `new("haploSet", ...)`.

Slots

haploSeq: Object of class "data.frame" to store the haplotype sequences. Names of haplotype sequences are used as the row names of the "data.frame", and names of SNPs are used as the column names.

haploFreq: Object of class "vector" to store the frequencies of the haplotype sequences in the haploSeq slot. Names of each elements must match the names of haplotype sequences.

Methods

haploFreq signature(.Object = "haploSet"): the "vector" object in the haploFreq slot.

haploSeq signature(.Object = "haploSet"): the "data.frame" object in the haploSeq slot.

Author(s)

Ting-Yuan Liu

See Also

[haplo-class](#)

Examples

```
showClass("haploSet")
```

keremRand

Pseudo-subjects from Kerem's Cystic Fibrosis Data

Description

This datasets contains the psudo-subjects created from cystic fibrosis data in Kerem et al. (1989).

Usage

```
data(keremRand)
```

Format

Here is the list of the 23 alleles:

```
locus_01 Probe: metD; Enzyme: Ban I
locus_02 Probe: metD; Enzyme: Taq I
locus_03 Probe: methH; Enzyme: Taq I
locus_04 Probe: E6; Enzyme: Taq I
locus_05 Probe: E7; Enzyme: Taq I
locus_06 Probe: pH131; Enzyme: Hinf I
```

locus_07 Probe: W3D1.4; Enzyme: Hind III
locus_08 Probe: H2.3A (XV2C); Enzyme: Taq I
locus_09 Probe: EG1.4; Enzyme: Hinc II
locus_10 Probe: EG1.4; Enzyme: Bgl II
locus_11 Probe: JG2E1 (KM19); Enzyme: Pst I
locus_12 Probe: E2.6 (E.9); Enzyme: Msp I
locus_13 Probe: H2.8A; Enzyme: Nco I
locus_14 Probe: E4.1 (Mp6d.9); Enzyme: Msp I
locus_15 Probe: J44; Enzyme: Xba I
locus_16 Probe: 10-1X.6; Enzyme: Acc I
locus_17 Probe: 10-1X.6; Enzyme: Hae III
locus_18 Probe: T6/20; Enzyme: Msp I
locus_19 Probe: H1.3; Enzyme: Nco I
locus_20 Probe: CEL.0; Enzyme: Nde I
locus_21 Probe: J32; Enzyme: Sac I
locus_22 Probe: J3.11; Enzyme: Msp I
locus_23 Probe: J29; Enzyme: Pu II

Details

SHARE algorithm requires subject-level information, i.e., it needs to know the haplotype/genotype sequences of every subjects in both case and control groups. However, the original data in Kerem et al. (1989) only provide the sequence-level information, meaning that we only know what group (case/control) each haplotype sequence belongs to. We need to simulate subject-level information to demonstrate SHARE algorithm. Two haplotypes with the same clinical status (having cystic fibrosis or not) are then randomly paired to form a pseudo-subject with the that status.

Three objects will be attached after loading the dataset keremRand:

The data.frame object keremRandSeq contains 186 sequences with 23 SNPs. The row names show the subject id and the sequence id within this subject. The SNPs are coded as 1 referring to the large allele of the RFLP, and 2 referring to the smaller allele.

The vector object keremRandStatus provides the CF/control status of each subject. 1 indicates subjects in case group (i.e., CF), and 0 indicates control group. There are 47 subjects in CF group and 46 in control group.

The data.frame object keremRandAllele contains allelic data for 23 SNPs, coded as 0, 1, 2 as the number of minor alleles.

How these three objects were created is shown in the example section.

Source

This dataset was originally released in Kerem et al. (1989), and was converted to R objects in Browning (2006). Browning's dataset could be found in the HapVLMC package (<http://www.stat.auckland.ac.nz/~browning/HapVLMC/index.htm>).

References

S. R. Browning. Multilocus association mapping using variable-length markov chains. *American Journal of Human Genetics*, 78(6):903-913, Jun 2006.

B. Kerem, J. M. Rommens, J. A. Buchanan, D. Markiewicz, T. K. Cox, A. Chakravarti, M. Buchwald, and L. C. Tsui. Identification of the cystic fibrosis gene: genetic analysis. *Science (New York, N.Y.)*, 245(4922):1073-1080, Sep 8 1989.

Examples

```
## Not run:
## Here are how the psudo-subjects are simulated
#### loading HapVLMC package and the dataset
library(HapVLMC)
data(Kerem)
set.seed(20090313)
randOrder <- runif(nrow(kerem.snps.data))
keremRandSeq <- rbind(## randomly order the TRUE part
                    kerem.snps.data[kerem.status, ][order(randOrder[kerem.status]), ],
                    ## randomly order the FALSE part
                    kerem.snps.data[!kerem.status, ][order(randOrder[!kerem.status]), ]
                    )

nLoci <- ncol(keremRandSeq)
lociNum <- unlist(sapply(1:nLoci,
                      function(x){
paste(paste(
rep("0", ceiling(log10(nLoci)) - nchar(as.character(x))), collapse=""),
x, sep="", collapse="")
      })
      )
colnames(keremRandSeq) <- paste("locus_", lociNum, sep="")

nSubj <- nrow(keremRandSeq)/2
subjNum <- unlist(sapply(1:nSubj,
                      function(x){
paste(paste(
rep("0", ceiling(log10(nSubj)) - nchar(as.character(x))), collapse=""),
x, sep="", collapse="")
      })
      )
subjLabel <- paste("subj_", subjNum, sep="")
seqLabel <- paste("seq", 1:2, sep="_")
rownames(keremRandSeq) <- paste(rep(subjLabel, each=2), seqLabel, sep="_")

keremRandStatus <- c(rep(1, sum(kerem.status)/2), rep(0, sum(!kerem.status)/2))

keremRandAllele <- NULL
for(i in seq(1, nrow(keremRandSeq), by=2)){
  keremRandAllele <- rbind(keremRandAllele,
                          apply(keremRandSeq[c(i, i+1), ], 2,
                              function(x){
```

```

        ## counting how many small alleles
        sum(x==2)
      }
    )
  }
  rownames(keremRandAllele) <- unique(gsub("^(subj_.*)_seq_(.*)$", "\\1", rownames(keremRandSeq)))

## End(Not run)

## load keremRand
data(keremRand)

## check which objects are attached
ls()

## dimation of psedu-subject data
dim(keremRandSeq)

## number of CF (TRUE) and control (FALSE) subjects
table(keremRandStatus)

```

nameSeq-methods	<i>Name of Sequences</i>
-----------------	--------------------------

Description

Methods to return the names of the sequences in the object.

Methods

.Object = "genoSet" Names of the genotype sequences in the genoSet object

.Object = "haploSet" Names of the haplotype sequences in the haploSet object

nameSNP-methods	<i>Name of SNPs</i>
-----------------	---------------------

Description

Methods to return the names of the SNPs in the object.

Methods

.Object = "genoSet" Names of the SNPs in the genotype sequences in the genoSet object

.Object = "haploSet" Names of the SNPs in the haplotype sequences in the haploSet object

nSeq-methods	<i>Number of Sequences</i>
--------------	----------------------------

Description

Methods to count how many sequences in the object.

Methods

.Object = "genoSet" Number of genotype sequences in the genoSet object

.Object = "haploSet" Number of haplotype sequences in the haploSet object

nSNP-methods	<i>Number of SNPs</i>
--------------	-----------------------

Description

Methods to count how many SNPs in the object.

Methods

.Object = "genoSet" Number of SNPs in the genoSet object

.Object = "haploSet" Number of SNPs in the haploSet object

share-class	<i>SHARE output</i>
-------------	---------------------

Description

A class to store the result of SHARE algorithm

Objects from the Class

Objects can be created by calls of the form `new("share", ...)`.

Slots

uhap: Object of class "vector"; the concatenated haplotype sequences
weight: Object of class "vector"; the weight of each haplotype
nFold: Object of class "numeric"; the fold size specified while calling the cshare function to perform the cross-validation approach
maxSNP: Object of class "numeric"; the maximum SNP size specified while calling the cshare function used
deviance: Object of class "vector"; the prediction deviance from cross-validation approach, or the BIC from BIC approach
bestsize: Object of class "numeric"; the size of most informative SNPs
finalHap: Object of class "data.frame"; the result of estimated haplotype sequences with the selected SNPs
finalHapFreq: Object of class "vector"; the frequencies of the estimated haplotype sequences with the selected SNPs
finalHapTest: Object of class "data.frame"; the hypothesis test results of the estimated haplotype sequences with the selected SNPs
globalP: Object of class "numeric"; the global p-values
modelmethod: Object of class "character"; the method of model selection specified while calling the cshare function
hap1 the code of first haplotype of each subject.
hap2 the code of first haplotype of each subject.
poolHapPair Subject IDs.
nPosHapPair vector for the count of haplotype pairs that map to each subject's marker genotypes.
post vector of posterior probabilities of pairs of haplotypes for a person, given their marker phenotypes.
haploSeq: Object of class "data.frame"; the original haplotype sequences with all SNPs
haploFreq: Object of class "vector"; the frequencies of the original haplotype sequences with all SNPs
pheno: Object of class "data.frame" to store the phenotype information
nngcov: Object of class "numeric"; the number of non-genetic covariates
ngcov: Object of class "vector"; the vector of non-genetic covariates
inherit: Character string to show which mode of inheritance for haplotype effects is specified.

Extends

Class "[haplo](#)", directly. Class "[haploSet](#)", by class "haplo", distance 2.

Methods

dplot signature(shareObj = "share"): to create the deviance plot to show the estimation of SNP size

Author(s)

James Y. Dai & Ting-Yuan Liu

See Also[cshare](#), [haplo](#), [haploSet](#)**Examples**

```
showClass("share")

## See vignette for more details
## Not run:
dplot(unphasedKerem[["Cross-Val"]])
dplot(unphasedKerem[["BIC"]])

## End(Not run)
```

shareTest

*Permutation Test for the Results from SHARE Algorithm***Description**

Permutation tests to compute the experimentwise p-values that account for model searching.

Usage

```
shareTest(outObj, haploObj, status, ncover = 0, covar = NULL, tol = 1e-08, verbose = FALSE,
          nperm = 1000, seed=38329832)
```

Arguments

outObj	the share object outputed from cshare function
haploObj	The haplo object cshare applied to
status	A character string indicating the column name of the phenotype in haploObj@pheno to be used as the clinical status in the analysis.
ncover	An integer indicating the number of non-genetic covariates.
covar	A matrix of non-genetic covariates where the first column is the subject ID numbers that match the names in haploObj@nPosHapPair. Missing values are not allowed.
tol	The convergence parameter for the haplotype logistic regression.
verbose	TRUE/FALSE to decide whether to create log file for debug
nperm	maximal number of permutation tests
seed	Seed for randomization in permutation tests

Details

If non-genetic covariates were used in creating `outObj`, they must also be entered in the `shareTest` function. If the best model size is zero, there appears to be no genetic association in the region of interest. There is no need to perform a permutation test. For final models with at least 1 SNPs, we permute case-control labels 1000 times regardless of the genotypic data, carry out model searching for each permuted dataset, and compute the nominal p-value using a Wald test. Finally the experiment-wise p-value is computed by comparing the observed p-value to its null distribution.

Value

The experiment-wise p-value from the permutation test will be returned.

Author(s)

James Y. Dai

References

- J. Y. Dai, M. LeBlanc, N. L. Smith, B. M. Psaty, and C. Kooperberg. SHARE: an adaptive algorithm to select the most informative set of SNPs for genetic association. *Biostatistics*, 2009. In press.
- J. Besag and P. Clifford. Sequential monte carlo p-values. *Biometrika*, 78(2):301, June 1, 1991.

See Also

[cshare](#)

Examples

```
## Not run:
## See vignette for more details
permuPValue <- shareTest(outObj=kerem[["Cross-Val"]],
  haploObj=keremHaplo,
  status = "CF",
  nperm=1000
)

## End(Not run)
```

Index

- *Topic **classes**
 - genoSet, 4
 - haplo-class, 6
 - haploSet-class, 7
 - share-class, 12
- *Topic **datasets**
 - keremRand, 8
- *Topic **htest**
 - shareTest, 14
- *Topic **methods**
 - nameSeq-methods, 11
 - nameSNP-methods, 11
 - nSeq-methods, 12
 - nSNP-methods, 12
- *Topic **package**
 - SHARE-package, 2

- cshare, 2, 14, 15

- dplot (share-class), 12
- dplot, share-method (share-class), 12

- genoSeq (genoSet), 4
- genoSeq, genoSet-method (genoSet), 4
- genoSet, 4, 6
- genoSet-class (genoSet), 4

- haplo, 3, 5, 7, 13, 14
- haplo, genoSet-method (genoSet), 4
- haplo-class, 6, 8
- haplo-class, 6
- haploFreq (haploSet-class), 7
- haploFreq, haploSet-method (haploSet-class), 7
- haploSeq (haploSet-class), 7
- haploSeq, haploSet-method (haploSet-class), 7
- haploSet, 7, 13, 14
- haploSet (haploSet-class), 7
- haploSet-class, 7

- keremRand, 8
- keremRandAllele (keremRand), 8
- keremRandSeq (keremRand), 8
- keremRandStatus (keremRand), 8

- nameSeq (nameSeq-methods), 11
- nameSeq, genoSet-method (nameSeq-methods), 11
- nameSeq, haploSet-method (nameSeq-methods), 11
- nameSeq-methods, 11
- nameSNP (nameSNP-methods), 11
- nameSNP, genoSet-method (nameSNP-methods), 11
- nameSNP, haploSet-method (nameSNP-methods), 11
- nameSNP-methods, 11
- nSeq (nSeq-methods), 12
- nSeq, genoSet-method (nSeq-methods), 12
- nSeq, haploSet-method (nSeq-methods), 12
- nSeq-methods, 12
- nSNP (nSNP-methods), 12
- nSNP, genoSet-method (nSNP-methods), 12
- nSNP, haploSet-method (nSNP-methods), 12
- nSNP-methods, 12

- phenoData (genoSet), 4
- phenoData, genoSet-method (genoSet), 4

- SHARE (SHARE-package), 2
- share (share-class), 12
- share-class, 3
- share-class, 12
- SHARE-package, 2
- shareTest, 14