Package 'fbati'

April 23, 2018

Version 1.0-3

Date 2018-04-19

Title Gene by Environment Interaction and Conditional Gene Tests for Nuclear Families

Author Thomas Hoffmann <tjhoffm@gmail.com>

Maintainer Thomas Hoffmann <tjhoffm@gmail.com>

Depends pbatR(>= 2.0.0)

Imports tcltk, fgui, rootSolve

Description Does family-based gene by environment interaction tests, joint gene, gene-environment interaction test, and a test of a set of genes conditional on another set of genes.

License GPL

URL https://sites.google.com/site/thomashoffmannproject/software/fbati

LazyLoad true

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-23 15:42:07 UTC

R topics documented:

fbatc *fbatc*

Description

Family based test for a group of markers conditional on another group of markers (typically conditional on a single marker). To start the graphical interface, provide no options, i.e. type `fbatc()` and press return.

Usage

```
fbatc( ped=NULL, phe=NULL, data=mergePhePed( ped, phe),
       trait="AffectionStatus", traitType="auto",
       markerAnalyze=NULL, markerCondition=NULL,
       offset=NULL,
       tempPrefix="temp",
       MAXITER=1000, TOL=sqrt(.Machine$double.eps),
       verbose=FALSE )
```

Arguments

ped	Object from (f)read.ped or as.ped (non-symbolic). See write.ped in the pbatR R package for more details on the file format.
phe	Object from (f)read.phe or as.phe (non-symbolic). See write.phe in the pbatR R package for more details on the file format.
data	a data.frame object containing required data, or formed from merging a pedigree and phenotype object together. The first columns of it must be as in a 'ped' object, while the next can be in any order representing marker or phenotype information.
trait	Trait to be analyzed. Defaults to AffectionStatus.
traitType	"auto","binary", or "continuous": if set to "auto", then "binary" will be chosen if there is only two levels of outcome, otherwise "continuous".
markerAnalyze	Names of markers to analyze (without .a, e.g.).
markerCondition	Names of markers to condition on. If none are specified, then each marker will be conditioned on in turn.
offset	If set to NULL (i.e. left unset, the default) then for a continuous trait this is estimated by the trait mean.
tempPrefix	Temporary prefix to use for output files. These are safe to delete later.
MAXITER	Maximum iterations before giving up on convergence for the nuisance parameters.
TOL	Relative tolerance for convergence for the nuisance parameters.
verbose	For debug.

Details

Implements the test as described in Hoffmann et. al. (please see References).

The results returned are a data.frame object. The column 'pvalue' and 'rank' are the pvalue and rank of the empirical covariance matrix of the model-based test (dichotomous or normal). The column 'pvalueR' and 'rankR' are the pvalue and rank of the robust test. The model-based test has considerable more power over the robust test, but must assume a disease model. Please see Hoffmann et. al. for more details.

This requires that FBAT be installed. If it is not, then the routine will attempt to automatically install it when given permission to do so by the user.

References

Hoffmann, Thomas J. and Laird, Nan M. Parsing the Effects of Individual SNPs in Candidate Genes in Families. Submitted.

Examples

```
## Not run:
  set.seed(13)

  ## We simulate NO LD HERE, and a completely random trait!
  ## Data here is only to show how to use the function

  ####################
  ## IGNORE START: ##
  ####################

  ##  You can safely ignore how the data is generated,
  ##  and just see how to use it afterward.
  NUM_FAMILIES <- 500
  AFREQ <- c(0.2,0.2)
  ped <- as.ped( data.frame( pid = kronecker(1:NUM_FAMILIES,c(1,1,1)),
                            id = kronecker( rep(1,NUM_FAMILIES), c(1,2,3) ),
                            idfath = kronecker( rep(1,NUM_FAMILIES), c(0,0,1) ),
                            idmoth = kronecker( rep(1,NUM_FAMILIES), c(0,0,2) ),
                            sex = rep(0,NUM_FAMILIES*3),
                           AffectionStatus = kronecker( rep(1,NUM_FAMILIES), c(0,0,2) ),
                            m0.a = rep(0,NUM_FAMILIES*3),      ## missing for now
                            m0.b = rep(0,NUM_FAMILIES*3),
                            m1.a = rep(0,NUM_FAMILIES*3),
                            m1.b = rep(0,NUM_FAMILIES*3) ) )
  CUR_FAMILY <- 1
  while( CUR_FAMILY<=NUM_FAMILIES ) {
    ## Indexing: start=father, (start+1)=mother, (start+2)=child
    start <- CUR_FAMILY*3-2

    ## Draw the parental genotypes from the population
    ped$m0.a[start:(start+1)] <- rbinom( 1, 1, AFREQ[1] ) + 1
    ped$m0.b[start:(start+1)] <- rbinom( 1, 1, AFREQ[1] ) + 1
    ped$m1.a[start:(start+1)] <- rbinom( 1, 1, AFREQ[2] ) + 1
    ped$m1.b[start:(start+1)] <- rbinom( 1, 1, AFREQ[2] ) + 1
```

```
  ## Draw the children's genotype from the parents
  ma <- rbinom( 1, 1, 0.5 )
  mb <- rbinom( 1, 1, 0.5 )
  if( rbinom( 1, 1, 0.5 ) == 0 ) {
    ped$m0.a[start+2] <- ped$m0.a[start]
    ped$m1.a[start+2] <- ped$m1.a[start]
  }else{
    ped$m0.a[start+2] <- ped$m0.b[start]
    ped$m1.a[start+2] <- ped$m1.b[start]
  }
  if( rbinom( 1, 1, 0.5 ) == 0 ) {
    ped$m0.b[start+2] <- ped$m0.a[start+1]
    ped$m1.b[start+2] <- ped$m1.a[start+1]
  }else{
    ped$m0.b[start+2] <- ped$m0.b[start+1]
    ped$m1.b[start+2] <- ped$m1.b[start+1]
  }

  CUR_FAMILY <- CUR_FAMILY + 1
}

## Create a completely random phenotype as well
phe <- as.phe( data.frame( pid=ped$pid, id=ped$id, qtl=rnorm(nrow(ped)) ) )

################
## IGNORE END ##
################

## Look at the first part of the pedigree
print( head( ped ) )
## Look at the first part of the phenotype
print( head( phe ) )

## Binary trait
## -- fbatc default trait is AffectionStatus
## -- fbatc default trait type is 'auto'
## - Test marker m1 conditional on m0
print(  fbatc( ped=ped, markerAnalyze="m1", markerCondition="m0" )  )
## - Do the test the other way around, m0 conditional on m1
print(  fbatc( ped=ped, markerAnalyze="m0", markerCondition="m1" )  )
## - Otherwise, we could have done this in one step;
##    if markerCondition is not specified, each member
##     of markerAnalyze is used.
print(  fbatc( ped=ped, markerAnalyze=c("m0","m1") )  )

## QTL
print(  fbatc( ped=ped, phe=phe, trait="qtl", markerAnalyze="m1", markerCondition="m0" )  )
print(  fbatc( ped=ped, phe=phe, trait="qtl", markerAnalyze="m0", markerCondition="m1" )  )

## Additionally, we could write out the data that we
##  generated to disk so that we can then use it.
write.ped( "simulated", ped ) ## to simulated.ped
```

```
  write.phe( "simulated", phe ) ## to simulated.phe

## End(Not run)
```

`fbatcStrategyStep` *FBAT-C Stepwise Strategy*

Description

Apply the FBAT-C test in a stepwise fashion using `fbatcStrategyStep` (which does forward selection with `fbatcStrategyStepUp`, followed by backwards selection with `fbatcStrategyStepDown`) and get the results ready for publication with `fbatcStrategyStepLatex`.

Usage

```
fbatcStrategyStepUp(ped, phe, markers=ped.markerNames(ped), trait="trait",
  traitType="auto", alphaMMarker=0.05, alphaStep=alphaMMarker, sortByCorrelation=TRUE,
  tempPrefix="temp_", sim=FALSE, debug=FALSE )
fbatcStrategyStepDown(ped, phe,  markers=ped.markerNames(ped),
  markersChosen=ped.markerNames(ped), markersChosenR=markersChosen, trait="trait",
  traitType="auto", alphaMMarker=0.05, alphaStep=alphaMMarker, sortByCorrelation=TRUE,
  tempPrefix="temp_", sim=FALSE, debug=FALSE )
fbatcStrategyStep(ped, phe, markers=ped.markerNames(ped), trait="trait",
  traitType="auto", alphaMMarker=0.05, alphaStep=alphaMMarker, sortByCorrelation=TRUE,
  tempPrefix="temp_", sim=FALSE, debug=FALSE )
fbatcStrategyStepLatex(res, digits=4, ffile="", preamble=FALSE, build=preamble, pdf="")
## S3 method for class 'fbatcSStep'
print(x,...)
```

Arguments

`ped`	Object from `(f)read.ped` or `as.ped`. See `write.ped` in the pbatR R package for more details on the file format.
`phe`	Object from `(f)read.phe` or `as.phe`. See `write.phe` in the pbatR R package for more details on the file format.
`markers`	Names of the markers to analyze.
`trait`	Name of the trait to analyze. Can be dichotomous or continuous.
`traitType`	"auto","dichotomous", or "continuous". If "auto" (the default), then "dichotomous" will be set if there are only two levels of the phenotype.
`alphaMMarker`	Alpha value for the multimarker test.
`alphaStep`	Alpha value used in the stepwise procedure.
`sortByCorrelation`	Whether to sort the markers by putting those in highest correlation closest to each other.

tempPrefix	The prefix to use for some intermittent files. Changing this is only necessary when you want to run this routine in parallel when each process shares the same disk.
sim	Developer use only.
res	Result of 'fbatcStrategyStep' routine.
digits	Number of significant digits to display.
ffile	If set to a filename, then the output is redirected to that file instead of the standard output.
preamble	Whether to produce a latex file that can be compiled, or only the code for the chart.
build	Whether to run pdflatex on the file (requires preamble=TRUE), pdflatex must be in your path (generally true in linux, but not in Windows).
pdf	Name of the pdf viewer executable, if you also want to open the compiled file immediately. Note that in this case, you may not be able to return to the R session until you close this window.
markersChosen	In the step-down approach, the markers to start with for the model-based approach.
markersChosenR	In the step-down approach, the markers to start with for the model-free approach.
debug	Developer use only.
x	Result of fbatcStrategyStep, fbatcStrategyStepUp, fbatcStrategyStepDown.
...	Extra arguments.

Details

fbatcStrategy returns a list with the following components.

mmarkerPvalue: p-value of the multi-marker test on those markers (Rakovski et. al 2008).

correlation: correlation matrix of the markers

univariate: univariate results

step: (model-based test) list of components pvalue (ith pvalue of the conditional test of markersAnalyze[i] on all markersCondition), numInf (number of informative families in the ith test), markersAnalyze, and markersCondition

markersChosen: (model-based test) results from applying step-up strategy

stepR, markersChosenR: (model-free test) results similar to step and markersChosen.

fbatge *fbatge*

Description

Family based test for gene-environment interaction utilizing arbitrary family structures and multiple affected offspring. This method is recommended over the `fbati` routine in most scenarios.

If no arguments are passed, then a friendly graphical interface is presented to the user.

fbatge [GxE test], fbatj (see fbatj help) [G,GxE test], fbatme (see fbatme help) [G test] generally have more options than fbatgeAll. fbatgeAll runs all three tests, and gives results of all of them, and so uses only the options that are common to all three functions.

Usage

```
fbatge( ped=NULL, phe=NULL,
        env=NULL, cov=NULL,
        trait="AffectionStatus", geno=NULL,
        strategy="hybrid", model="additive" )
fbatgeAll( ped=NULL, phe=NULL, env=NULL, trait="AffectionStatus" )
```

Arguments

`ped`	Object from `(f)read.ped` or `as.ped` (non-symbolic). See `write.ped` in the pbatR R package for more details on the file format.
`phe`	Object from `(f)read.phe` or `as.phe` (non-symbolic). See `write.phe` in the pbatR R package for more details on the file format.
`env`	Environmental Exposure. Should be a string, corresponding to the name in the 'phe' object.
`cov`	Any covariates to adjust for (does not apply to RR method). Should be a vector of strings, corresponding to names in the 'phe' object.
`trait`	Dichotomous trait name. Should be either "AffectionStatus", corresponding to the affection status in the pedigree object, or a string in the phenotype object.
`geno`	Names of the genetic markers, from the 'ped' object. If NULL (default), then all genetic markers are tested.
`strategy`	One of 'hybrid' (recommended, most efficient, requires rare disease), 'RR' (relative risk model, generally for a rare disease), or 'CLR' (conditional logistic regression).
`model`	Either 'additive' for the additive genetic model, or 'codominant' for the codominant genetic model (indicator variables for the genotypes).

Details

Implements the test as described in Hoffmann et. al. (please see References).

NOTE: The allele frequency is simply based on the allele frequency in all genotyped individuals, and is not the best choice.

References

Hoffmann, Thomas J., and Laird, Nan M. Combining Multiple Disease Models for a More Powerful Gene-Environment Interaction Test in Nuclear Families.

Examples

```
example( fbati ) ## See fbati, creates a dataset for us in 'phe' and 'ped'
print( fbatge( ped=ped, phe=phe, env="env" ) )
## The results are very close to the FBAT-I function, which
##  we would expect for trios.
```

`fbati` *fbati*

Description

Family based test for gene-environment interaction for bi-allelic snps, command/line or GUI (provide no options to start the graphical interface, i.e. just type `fbati()` and press return).

Usage

```
fbati( ped=NULL, phe=NULL,
       data=mergePhePed(ped,phe),
       marker=NULL, ## pairs...
       env,
       method="fbati",
       model="additive",
       iter=10000,
       seed=7,
       maxSib=3,
       fixNames=TRUE,
       debug=FALSE )
```

Arguments

`ped`	Object from `(f)read.ped` or `as.ped`. See `write.ped` in the pbatR R package for more details on the file format.
`phe`	Object from `(f)read.phe` or `as.phe`. See `write.phe` in the pbatR R package for more details on the file format.
`data`	a data.frame object containing required data, or formed from merging a pedigree and phenotype object together. The first columns of it must be as in a 'ped' object, while the next can be in any order representing marker or phenotype information.

marker	Default is NULL for all markers. Otherwise, it can be the names of the marker (if you load in with read.ped, this should be without the '.a'/'.b' added to differentiate the two markers). If you are using more specialized loading routines, this represents the numbers of the columns where the markers are at. For example, 7:10 would mean that columns 7 and 8 represent one locus, and columns 9 and 10 represent another locus.
env	Character string representing the name of the environmental variable to use (a column header name of the 'data' parameter).
method	Currently only 'fbati' is supported.
model	one of "additive", "dominant", or "recessive".
iter	The number of Monte-Carlo iterations to perform.
seed	The random seed, so consistent answers are maintained. See NOTE 1 for more details. NA/NULL disables this, but is not recommended.
maxSib	When nonzero, employs the following rules to minimize the number of strata, to improve the number of informative transmissions. When there are parents, a random affected child is chosen. When parents are missing, a random affected child with environmental exposure is chosen, and random genotyped siblings are chosen to maxSib total offspring (so 2 indicates a sibpair, 3 a sibtrio, etc.), and parents are treated as missing (even if there is one). See the 'strataReduce' routine for more details and examples.
fixNames	Just leave this to TRUE if creating from ped/phe objects (pops off the '.a' and '.b' added on to the names of the two markers that are added on when read in via the (f)read.ped(...) routine).
debug	Developer use only (extended output).

Details

Returns a data.frame object with the results. The columns entitled GX...X indicate the number of informative families in each strata for the given marker. If these columns do not show up, it indicates there was only one type of strata.

The parents need not be in the dataset if they have completely missing genotypes (they will be inserted), but the snps must currently be bi-allelic (or you will get error messages).

`fread.ped` and `fread.phe` are suggested to enforce loading the whole dataset.

NOTE 1: The fbati test was developed for families with at least one affected, so if there is more than one affected individual per family, only a random affected one will be used, and a random unaffected to reduce strata, unless `strataFix` is disabled. This is done on a per marker basis, thus the seed is set before every marker to obtain reproducible results.

NOTE 2: The data is converted into nuclear families. This is done by a call to 'nuclifyMerged' to the passed in dataset to enforce this consistency.

References

Hoffmann, Thomas J., Lange, Christoph, Vansteelandt, Stijn, and Laird, Nan M. Gene-Environment Interaction Test for Dichotomous Traits in Trios and Sibships. Submitted.

S. L. Lake and N. M. Laird. Tests of gene-environment interaction for case-parent triads with general environmental exposures. Ann Hum Genet, 68(Pt 1):55-64, Jan 2004.

See Also

fbatj,

Examples

```
## Data is simulated according to the formula in the
##  paper (you can see it from the code).

## Set the seed so you get the same results
set.seed(13)

## Constants (you can vary these)
NUM_FAMILIES <- 500
AFREQ <- 0.1  ## Allele frequency
BG <- -0.25   ## main effect of gene
BE <- 0       ## main effect of environment
BGE <- 0.75   ## main gene-environment effect
ENV <- 0.2    ## environmental exposure frequency

## (but don't modify this one)
MAX_PROB <- exp( BG*2 + BE*1 + BGE*2*1 )

####################################
## Create a random dataset (trios) ##
####################################

## -- genotypes are set to missing for now,
##    everyone will be affected
ped <- as.ped( data.frame( pid = kronecker(1:NUM_FAMILIES,c(1,1,1)),
                           id = kronecker( rep(1,NUM_FAMILIES), c(1,2,3) ),
                           idfath = kronecker( rep(1,NUM_FAMILIES), c(0,0,1) ),
                           idmoth = kronecker( rep(1,NUM_FAMILIES), c(0,0,2) ),
                           sex = rep(0,NUM_FAMILIES*3),
                           AffectionStatus = kronecker( rep(1,NUM_FAMILIES), c(0,0,2) ),
                           m0.a = rep(0,NUM_FAMILIES*3),      ## missing for now
                           m0.b = rep(0,NUM_FAMILIES*3) ) )   ## missing for now
## -- envioronment not generated yet
phe <- as.phe( data.frame( pid = ped$pid,
                           id = ped$id,
                           env = rep(NA,NUM_FAMILIES*3) ) )   ## missing for now

## 50/50 chance of each parents alleles
mendelTransmission <- function( a, b ) {
  r <- rbinom( length(a), 1, 0.75 )
  return( a*r + b*(1-r) )
}

## Not the most efficient code, but it gets it done;
##  takes < 5 sec on pentium M 1.8Ghz
CUR_FAMILY <- 1
while( CUR_FAMILY<=NUM_FAMILIES ) {
  ## Indexing: start=father, (start+1)=mother, (start+2)=child
```

```
  start <- CUR_FAMILY*3-2

  ## Draw the parental genotypes from the population
  ped$m0.a[start:(start+1)] <- rbinom( 1, 1, AFREQ ) + 1
  ped$m0.b[start:(start+1)] <- rbinom( 1, 1, AFREQ ) + 1

  ## Draw the children's genotype from the parents
  ped$m0.a[start+2] <- mendelTransmission( ped$m0.a[start], ped$m0.b[start] )
  ped$m0.b[start+2] <- mendelTransmission( ped$m0.a[start+1], ped$m0.b[start+1] )

  ## Generate the environment
  phe$env[start+2] <- rbinom( 1, 1, ENV )

  ## and the affection status
  Xg <- as.integer(ped$m0.a[start+2]==2) + as.integer(ped$m0.b[start+2]==2)
  if( rbinom( 1, 1, exp( BG*Xg + BE*phe$env[start+2] + BGE*Xg*phe$env[start+2] ) / MAX_PROB ) == 1 )
    CUR_FAMILY <- CUR_FAMILY + 1
  ## otherwise it wasn't a valid drawn individual
}


###############
## Analysis ##
###############

## Print the first 4 families
print( head( ped, n=12 ) )
print( head( phe, n=12 ) )

## NOTE: We could have just put all of this info into a single dataframe otherwise,
##  that would look like just the results of this
data <- mergePhePed( ped, phe )
print( head( data, n=12 ) )

## And run the analysis on all the markers
fbati( ped=ped, phe=phe, env="env" )

## Or do it via the merged data.frame object
##  7 and 8 correspond to the marker columns
fbati( data=data, env="env", marker=c(7,8) )

## You may also want to up the number of Monte-Carlo
##  iterations from the default

## And we could also run a joint test instead
##  (see fbatj)
fbatj( ped=ped, phe=phe, env="env" )
fbatj( data=data, env="env", marker=c(7,8) )

## Not run:
## This won't be run, but we could do this with the gui.
## It requires the data to be written to disk, so we do so:
write.ped( ped, "simulated" )
```

```
write.phe( phe, "simulated" )
## Then start the GUI -- specify the options as before,
##  but for the first two, navigate to the 'simulated.ped' and 'simulated.phe' files.
fbati()

## End(Not run)
```

---

fbatj *fbatj*

Description

Family based joint test for both the main effect of the gene and gene-environment interaction, command/line or GUI (provide no options to start gui, i.e. just type `fbatj()` and press return). Null hypothesis is no linkage and no association.

Usage

```
fbatj( ped=NULL, phe=NULL,
       data=mergePhePed(ped,phe),
       marker = NULL,
       trait = "AffectionStatus",
       env = "env",
       model = "additive",
       mode = "univariate",
       fixNames = TRUE,
       verbose = FALSE )
```

Arguments

`ped`	Object from `(f)read.ped` or `as.ped`. See `write.ped` in the pbatR R package for more details on the file format.
`phe`	Object from `(f)read.phe` or `as.phe`. See `write.phe` in the pbatR R package for more details on the file format.
`data`	a data.frame object containing required data, or formed from merging a pedigree and phenotype object together. The first columns of it must be as in a 'ped' object, while the next can be in any order representing marker or phenotype information.
`marker`	Default is NULL for all markers. Otherwise, it can be the names of the marker (if you load in with read.ped, this should be without the '.a'/'.b' added to differentiate the two markers). If you are using more specialized loading routines, this represents the numbers of the columns where the markers are at. For example, 7:10 would mean that columns 7 and 8 represent one locus, and columns 9 and 10 represent another locus.
`trait`	Character string representing the name of the trait variable to use (a column header name of the 'data' parameter).

env	Character string representing the name of the environmental variable to use (a column header name of the 'data' parameter).
model	one of "additive", "dominant", or "recessive".
mode	"univariate" is the only one supported.
fixNames	Just leave this to TRUE if creating from ped/phe objects (pops off the '.a' and '.b' added on to the names of the two markers that are added on when read in via the (f)read.ped(...) routine).
verbose	Developer use only (extended output).

Details

Returns a data.frame object with the results. Uses variance based on Mendelian transmissions (no longer uses the empirical variance).

For an example, see the fbati function.

References

K Lunetta, S V Faraone, J Biederman, and N M Laird. Family-based tests of association and linkage that use unaffected sibs, covariates, and interactions. Am J Hum Genet, 2000, 66, 605-614.

See Also

fbatj

---

fbatme *FBAT Main effects Test*

Description

Family based test for the main genetic effect, using the variance based on Mendelian transmissions. The null hypothesis is that there is no linkage and no association.

Usage

```
fbatme( ped=NULL, phe=NULL,
        data=mergePhePed(ped,phe),
        marker=NULL,
        trait="AffectionStatus",
        model="additive",
        fixNames=TRUE,
        verbose = FALSE )
```

Arguments

ped	Object from (f)read.ped or as.ped. See write.ped in the pbatR R package for more details on the file format.
phe	Object from (f)read.phe or as.phe. See write.phe in the pbatR R package for more details on the file format.
data	a data.frame object containing required data, or formed from merging a pedigree and phenotype object together. The first columns of it must be as in a 'ped' object, while the next can be in any order representing marker or phenotype information.
marker	Default is NULL for all markers. Otherwise, it can be the names of the marker (if you load in with read.ped, this should be without the '.a'/'.b' added to differentiate the two markers). If you are using more specialized loading routines, this represents the numbers of the columns where the markers are at. For example, 7:10 would mean that columns 7 and 8 represent one locus, and columns 9 and 10 represent another locus.
trait	Character string representing the name of the trait variable to use (a column header name of the 'data' parameter).
model	one of "additive", "dominant", or "recessive".
fixNames	Just leave this to TRUE if creating from ped/phe objects (pops off the '.a' and '.b' added on to the names of the two markers that are added on when read in via the (f)read.ped(...) routine).
verbose	Developer use only (extended output).

Details

Returns a data.frame object with the results. Uses the variance based on Mendelian transmissions.

NOTE: The allele frequency is simply based on the allele frequency in all genotyped individuals, and is not the best choice.

---

launchpad *Launchpad*

Description

Provides a GUI launchpad for routines in the fbati (i.e. this package) and pbatR (a dependency of this package) R packages.

Usage

```
launchpad()
```

See Also

fbati fbatj

nuclify *Nuclify and Merge*

Description

mergePhePed merges a phenotype and pedigree object into a single data.frame object.

nuclifyMerged chops a merged object into nuclear families of a dataset, generally a necessary preprocessing option for tests.

nuclify chops instead a 'ped' and 'phe' object separately.

Usage

```
mergePhePed(ped, phe)
nuclifyMerged(data, OUT_MULT=2)
nuclify(ped, phe)
```

Arguments

ped	Object from (f)read.ped or as.ped.
phe	Object from (f)read.phe or as.phe.
data	data.frame containing required data, or formed from merging a pedigree and phenotype object together. The first columns of it must be as in a 'ped' object, while the next can be in any order representing marker or phenotype information.
OUT_MULT	Hint for size of output, doesn't matter if wrong.

Details

mergePhePed and nuclifyMerged both return data.frame objects. nuclify returns a list that contains the 'phe' object and the 'ped' object with those respective names (see pbatR documentation, both objects extend data.frame objects, and can be used for the most part as if data.frame objects). When the data is nuclified, the parents of the nuclified families parents are lost.

NOTE: nuclifyMerged will modify the pedigree id (pid) to be [100*(previous pid) + (nuclear family index)]. This should make it easy to observe the results of this call to your dataset.

Examples

```
## Create some pedigree structure
##
##   100 --- 101
##        |
##       201---202
##           |
##     -------------
##     |   |   |   |
##    301 302 303 304
```

```
ped <- as.ped( data.frame( pid    = rep(1,8),
                           id     = c(100,101, 201,202, 301,302,303,304),
                           idfath = c(0,0,     100,0,   201,201,201,201),
                           idmoth = c(0,0,     101,0,   202,202,202,202),
                           sex    = c(1,2,     1,2,     2,2,2,2),
                           AffectionStatus = rep(0,8),
                           m0.a = rep(2,8),
                           m0.b = rep(2,8) ) )

## Which should chop up into
##   100 --- 101    201---202
##        |             |
##       201      -------------
##                |   |   |   |
##               301 302 303 304
nuclifyMerged( ped )

## NOTE: We could have merged the ped with a phe object,
##        via the 'mergePhePed' routine before running.
```

`strataReduce` *Strata Reduction*

Description

Reduces the number of strata in nuclear pedigrees for testing (for use with FBAT-I). For nuclear families with both parents, a random affected child is drawn. For nuclear families with at least one parent missing, a random affected and another random sib is used (parents ignored).

Usage

```
strataReduce( data, envCol, m0, m1=m0+1, maxSib=3 )
```

Arguments

`data`	data.frame of a merged pedigree/phenotype (see mergePhePed(...)).
`envCol`	Integer representing environment column.
`m0`	Integer representing column of first marker.
`m1`	Integer representing column of second marker.
`maxSib`	Maximum number of sibs to use per family to reduce the number of strata.

Examples

```
## Function creates a family with the specified markers and statuses
## NOTE: affection is false/true, whereas it is coded 1/2 in the ped file
createFam <- function( pa=c(0,0), pb=c(0,0),
                       ca, cb,
                       caffected=rep(TRUE,length(ca)),
```

```
                       env=1:length(ca) ) {
  ## pid, id, idfath, idmoth, sex, affection, m0a, m0b
  numC <- length(ca)
  return( data.frame( pid=rep(1,2+numC),
                      id=1:(2+numC),
                      idfath=c(0,0,rep(1,numC)),
                      idmoth=c(0,0,rep(2,numC)),
                      sex=c(2,1,rep(0,numC)),
                      affection=c(0,0,as.integer(caffected)+1),
                      m0.a=c(pa,ca), m0.b=c(pb,cb),
                      env=c(NA,NA,env) ) )
}
## Function tests/exemplifies the strataReduce(...) routine
srFam <- function( ... ) {
  data <- createFam( ... )
  data2 <- strataReduce( data=data, envCol=9, m0=7, maxSib=2 )
  cat( "Original data:\n" )
  print( data )
  cat( "Reduced stratification data:\n" )
  print( data2 )
}

## Basic sib test
srFam( ca=c(1,1,2), cb=c(1,2,2) )

## Basic trio test
srFam( ca=c(1,1,2), cb=c(1,2,2), pa=c(1,1), pb=c(2,2) )

## a fairly comprehensive test here
## The affected should always be one of the first three,
##  the unaffected could be one the first eight
for( i in 1:10 )
  srFam( ca=c(1:8,0,0), cb=c(1:8,0,0),
         pa=c(1,1),
         caffected=c(rep(TRUE,6),rep(FALSE,4)),
         env=c(1:3,rep(NA,7)) )

## Now just to make sure, a full pedigree, rather than just one family
data <- createFam( ca=1:2, cb=1:2 )
for( i in 2:10 )
  data <- rbind( data, createFam( ca=1:2, cb=1:2 ) )
  cat( "Original data (full pedigree):\n" )
  print( data )
  cat( "Reduced stratification data (full pedigree), maxSib=3\n" )
  print( strataReduce( data=data, envCol=9, m0=7 ) )
```

Index