

# Package ‘fbati’

April 17, 2009

**Version** 0.6.2

**Date** 2009-03-10

**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), fgui, rootSolve, tcltk

**Imports** tcltk, fgui

**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** <http://www.people.fas.harvard.edu/~tjhoffm/fbati.html>

**LazyLoad** true

**Repository** CRAN

**Date/Publication** 2009-03-11 15:58:51

## R topics documented:

fbatc . . . . .	2
fbatcStrategyStep . . . . .	5
fbati . . . . .	6
fbatj . . . . .	10
launchpad . . . . .	12
nuclify . . . . .	12
strataReduce . . . . .	13
<b>Index</b>	<b>16</b>

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**

<code>ped</code>	Object from <code>(f) read.ped</code> or <code>as.ped</code> (non-symbolic). See <code>write.ped</code> in the <code>pbatR</code> R package for more details on the file format.
<code>phe</code>	Object from <code>(f) read.phe</code> or <code>as.phe</code> (non-symbolic). See <code>write.phe</code> in the <code>pbatR</code> R package for more details on the file format.
<code>data</code>	a <code>data.frame</code> 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.
<code>trait</code>	Trait to be analyzed. Defaults to <code>AffectionStatus</code> .
<code>traitType</code>	“auto”, “binary”, or “continuous”: if set to “auto”, then “binary” will be chosen if there is only two levels of outcome, otherwise “continuous”.
<code>markerAnalyze</code>	Names of markers to analyze (without <code>.a</code> , e.g.).
<code>markerCondition</code>	Names of markers to condition on. If none are specified, then each marker will be conditioned on in turn.
<code>offset</code>	If set to <code>NULL</code> (i.e. left unset, the default) then for a continuous trait this is estimated by the trait mean.
<code>tempPrefix</code>	Temporary prefix to use for output files. These are safe to delete later.
<code>MAXITER</code>	Maximum iterations before giving up on convergence for the nuisance parameters.
<code>TOL</code>	Relative tolerance for convergence for the nuisance parameters.
<code>verbose</code>	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=0.05, sortByCorrelation=TRUE, tempPrefix="" )
fbatcStrategyStepDown( ped, phe, markers=ped.markerNames(ped), markersChosen=ped.markerNames(ped), trait="trait", traitType="auto", alphaMMarker=0.05, alphaStep=0.05, sortByCorrelation=TRUE, tempPrefix="" )
fbatcStrategyStep( ped, phe, markers=ped.markerNames(ped), trait="trait", traitType="auto", alphaMMarker=0.05, alphaStep=0.05, sortByCorrelation=TRUE, tempPrefix="" )
fbatcStrategyStepLatex( res, digits=4, ffile="", preamble=FALSE, build=preamble, po=0 )
## S3 method for class 'fbatcSStep':
print(x, ...)
```

## Arguments

<code>ped</code>	Object from <code>(f) read.ped</code> or <code>as.ped</code> . See <code>write.ped</code> in the <code>pbatR</code> R package for more details on the file format.
<code>phe</code>	Object from <code>(f) read.phe</code> or <code>as.phe</code> . See <code>write.phe</code> in the <code>pbatR</code> R package for more details on the file format.
<code>markers</code>	Names of the markers to analyze.
<code>trait</code>	Name of the trait to analyze. Can be dichotomous or continuous.
<code>traitType</code>	"auto", "dichotomous", or "continuous". If "auto" (the default), then "dichotomous" will be set if there are only two levels of the phenotype.
<code>alphaMMarker</code>	Alpha value for the multimarker test.
<code>alphaStep</code>	Alpha value used in the stepwise procedure.
<code>sortByCorrelation</code>	Whether to sort the markers by putting those in highest correlation closest to each other.
<code>tempPrefix</code>	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.
<code>sim</code>	Developer use only.
<code>res</code>	Result of 'fbatcStrategyStep' routine.
<code>digits</code>	Number of significant digits to display.
<code>ffile</code>	If set to a filename, then the output is redirected to that file instead of the standard output.

<code>preamble</code>	Whether to produce a latex file that can be compiled, or only the code for the chart.
<code>build</code>	Whether to run <code>pdflatex</code> on the file (requires <code>preamble=TRUE</code> ), <code>pdflatex</code> must be in your path (generally true in linux, but not in Windows).
<code>pdf</code>	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.
<code>markersChosen</code>	In the step-down approach, the markers to start with for the model-based approach.
<code>markersChosenR</code>	In the step-down approach, the markers to start with for the model-free approach.
<code>debug</code>	Developer use only.
<code>x</code>	Result of <code>fbatcStrategyStep</code> , <code>fbatcStrategyStepUp</code> , <code>fbatcStrategyStepDown</code> .
<code>...</code>	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`.

---

`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 sib trio, 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

## -- environment 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
      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).

## 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) <code>read.ped</code> or <code>as.ped</code> . See <code>write.ped</code> in the <code>pbatR</code> R package for more details on the file format.
phe	Object from (f) <code>read.phe</code> or <code>as.phe</code> . See <code>write.phe</code> in the <code>pbatR</code> R package for more details on the file format.
data	a <code>data.frame</code> 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 <code>NULL</code> for all markers. Otherwise, it can be the names of the marker (if you load in with <code>read.ped</code> , 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 <code>TRUE</code> 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) <code>read.ped(...)</code> routine).
verbose	Developer use only (extended output).

### Details

Returns a `data.frame` object with the results. 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](#)

---

 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 )
nuclify( ped, phe )
```

### Arguments

<code>ped</code>	Object from <code>(f) read.ped</code> or <code>as.ped</code> .
<code>phe</code>	Object from <code>(f) read.phe</code> or <code>as.phe</code> .
<code>data</code>	<code>data.frame</code> 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.

## 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  $[100 * (\text{previous pid}) + (\text{nuclear family index})]$ . 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.
```

## 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

## \*Topic **interface**

- fbatc, [2](#)
- fbatcStrategyStep, [5](#)
- fbati, [6](#)
- nuclify, [12](#)
- strataReduce, [13](#)

- fbatc, [2](#)
- fbatcStrategyStep, [5](#)
- fbatcStrategyStepDown
  - (*fbatcStrategyStep*), [5](#)
- fbatcStrategyStepLatex
  - (*fbatcStrategyStep*), [5](#)
- fbatcStrategyStepUp
  - (*fbatcStrategyStep*), [5](#)
- fbati, [6](#), [12](#)
- fbatj, [8](#), [10](#), [11](#), [12](#)

- launchpad, [12](#)

- mergePhePed(*nuclify*), [12](#)

- nuclify, [12](#)
- nuclifyMerged(*nuclify*), [12](#)

- print.fbatcSStep
  - (*fbatcStrategyStep*), [5](#)

- strataReduce, [13](#)