Package: pbatR (via r-universe)

Version 2.2-17

Date 2024-01-22

Title Pedigree/Family-Based Genetic Association Tests Analysis and Power

Author Thomas Hoffmann <tjhoffm@gmail.com>, with contributions from Christoph Lange <clange@hsph.harvard.edu>

Maintainer Thomas Hoffmann <tjhoffm@gmail.com>

Depends R (>= 2.0.0)

Imports survival, rootSolve

Suggests kinship2, tcltk

Description This R package provides power calculations via internal simulation methods. The package also provides a frontend to the now abandoned PBAT program (developed by Christoph Lange), and reads in the corresponding output and displays results and figures when appropriate. The license of this R package itself is GPL. However, to have the program interact with the PBAT program for some functionality of the R package, users must additionally obtain the PBAT program from Christoph Lange, and accept his license. Both the data analysis and power calculations have command line and graphical interfaces using tcltk.

License GPL

URL https://academic.oup.com/bioinformatics/article-abstract/22/24/3103/208723

LazyLoad true

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-01-29 10:20:03 UTC

Contents

affectionPhe *Affection Phenotype Object*

Description

Creates a phenotype object from a pedigree object (see 'ped' and 'phe'), preserving affection status. Mostly used internally, or as a substitute offset fix for the GUI.

Usage

```
affectionPhe(ped, trait="affected", offset=0.0)
```

Arguments

ped	An object of class ped.
trait	Name for the trait in the generated phenotype file.
offset	The offset to be used.

Details

Returns an object of class phe.

See Also

[read.phe](), [read.ped]()

---

c2bat	*C2BAT*

Description

C2BAT (c) Matt McQueen, Jessica Su, Christoph Lange.

Usage

```
c2bat(
  snps,
  case.aa, case.Aa, case.AA,
  control.aa, control.Aa, control.AA,
  replicates=0,
  statistic="armitage",
  method="exact")
```

Arguments

snps	Names of the snps (vector). Can be strings.
case.aa	Number of cases with aa genotype (a vector corresponding to the snps).
case.Aa	Number of cases with Aa genotype.
case.AA	Number of cases with AA genotype.
control.aa	Number of controls with aa genotype.
control.Aa	Number of controls with Aa genotype.
control.AA	Number of controls with AA genotype.
replicates	This value should be set to zero for the first analysis. Then, once the top snps have been selected, this should be rerun with replicates>1000 for those selected snps.
statistic	"armitage" = the Armitage trend test. "logrank" = logrank test
method	"exact" = exact likelihood function "approximate" = approximate likelihood function

Details

The output is formatted as follows: SNP, case.aa, case.Aa, case.AA, control.aa, control.Aa, control.AA, Monte-Carlo p-value, p-value, non-centrality parameter (for power calculations), modelc2-based OR, allelic OR.

Note that the non-centrality parameter and the ORs are independent of the pvalues.

Examples

```
## Not run:
c2bat( snps=c("snp1","snp2"),
       case.aa=c(10,11), case.Aa=c(30,31), case.AA=c(14,44),
       control.aa=c(33,32), control.Aa=c(100,99), control.AA=c(0,0) )

## End(Not run)
```

---

clean   *Cleaning pbat(), pbat.last(), pbat.m() output.*

Description

Removes many of the temporary output files from communicating with PBAT. Make sure you save how PBAT was run for future reference. Most useful for simulations.

Usage

```
pbat.clean(pbatObj, plots=TRUE, all.output=FALSE)
```

Arguments

pbatObj	'pbat' object
plots	TRUE/FALSE - delete plot output
all.output	DEPRECATED: No longer does anything, deleting files from other (potentially running) PBAT runs can be dangerous.

See Also

pbat

---

cluster   *Cluster result functions*

Description

Options when 'clusterRefresh' has been set to 0 (so that the current process is batched). See pbat.set for details.

is.finished() returns if PBAT execution has terminated.

pbat.load() loads in PBAT output.

pbat.concatenate() concatenates output files (without reading them into an R object).

Usage

```
is.finished(pbatObj=NULL, clean=TRUE)
pbat.load(pbatObj=NULL)
pbat.concatenate(pbatObj=NULL,
   filename="myResults.txt", clean=FALSE)
```

Arguments

pbatObj   'pbat' object (result of pbat.m, pbat.obj, pbat.files). If NULL / left empty, then
              the results are taken from what was run in the GUI interface.

clean   *pbat.load*: If TRUE it will remove intermediate files that setup the job submis-
              sion interface (but no other results, and not the command-files sent to pbat); see
              also pbat.clean. If FALSE, nothing else is done. NOTE, when TRUE (which is
              the default), if you run this more than once, you will get TRUE only once, and
              then FALSE the rest of the times.

              pbat.concatenate: If clean is TRUE, then output that went into the concatenated
              file will be deleted. Default is FALSE, as this has been a primary area of contin-
              ual changes in PBAT, and so you may not want to do this until you have checked
              the output.

filename   output filename

See Also

pbat.set

pbat

---

cped   *Phenotype Object*

Description

Creates, tests, reads, or writes an object of class cped.

Usage

```
as.cped(x,
  pid="pid", id="id", idfath="idfath",
  idmoth="idmoth", sex="sex", affection="AffectionStatus",
  clearSym=FALSE)

is.cped(obj)

read.cped(filename, lowercase=TRUE, sym=TRUE, max=100, ...)
fread.cped(filename, ...)

write.cped(file, cped)
```

```
## S3 method for class 'cped'
sort(x,decreasing=FALSE,...)

plotCPed( cped, sink=NULL )
```

Arguments

x	An object of class cped or data.frame as described below.
	If x is of class cped, no other options are used.
	When x is of class data.frame, the columns have entries that match the string parameters idped,...,AffectionStatus; copy number variation measurements markers consist of *subsequent* columns formated as follows: e.g. two copy number variants 'cnv1' and 'cnv2' with three intensities would need a total of six columns (three per cnv) past the first six named 'cnv1.1', 'cnv1.2', 'cnv1.3', 'cnv2.1', 'cnv2.2', 'cnv2.3'. See the examples below.
	A slightly different format can be used when reading data from disk, that is more or less consistent with how a pedigree file is loaded from disk. See the details below.
pid	String corresponding to column name for pedigree id.
id	String corresponding to column name for subject id.
idfath	String corresponding to column name for father id.
idmoth	String corresponding to column name for mother id.
sex	String corresponding to column name for sex.
affection	String corresponding to column name for affection status.
filename	Filename to open; does not need .phe extension. See the details below for the file format.
lowercase	When TRUE (and sym is FALSE), enforces all headers to lowercase for convenience.
...	Options for [read.table](), used only when sym is FALSE. Do *not* put in header=TRUE, as this will cause an error, as the header is automatically loaded.
	With the proper file formatting, this should not be used.
file	string representing filename, or a connection for file output
cped	an object of class cped
obj	an object
sym	When TRUE, only the header of the file is read in; only PBAT will load in the file. When FALSE, the entire file will be read in, and can be modified before using with PBAT.
max	When sym is TRUE, the amount of headers to read in before going pure symbolic (so that the SNP usage consistency will not be assessed by pbatR, only by PBAT).
clearSym	When TRUE, if a symbolic file is found, it will be read in; otherwise, it will stay symbolic.

| decreasing | Whether to sort in decreasing/increasing order. |
| sink | For 'plot.cped', this is the name of a pdf file to output all of the plots to (there will be one plot per page). |

Details

When reading in a file on disk using read.cped, a '.cped' file should have the following format. The file should be formatted as follows. The first six columns are *unlabeled* (1) the pedigree id, (2) the individual id, (3) the father id, (4) the mother id, (5) sex [0=missing, 1=male, 2=female], and (6) AffectionStatus [0=missing, 1=unaffected, 2=affected]. The subsequent columns correspond to the intensities. So, suppose we have cnv1 and cnv2. The first line of the file would contain 'cnv1 cnv2'. Then the subsequent lines would correspond to each individual, the first six columns being as described, and then NUMINTENSITY columns per cnv for a total of 6+2*NUMINTENSITY data columns. NUMINTENSITY is just however many intensities there are per cnv, you will need to specify this number at analysis time. NOTE: MISSING DATA in a cped file should be coded as '-1234.0', rather than the usual '.' or '-' (technically the '.' and '-' should still work with fread.cped, and when sym=FALSE).

The best way to see all of this in action is to run the code in the examples below, and look at the cped file produced from it.

'plotCPed' plots the data similar to the 'plotPed' routine (in fact it transforms the data to use it).

See Also

read.ped, read.cped write.cped, plotCPed

Examples

```
####################
## First Example ##

## A highly artificial example with not enough subjects to be run;
##  however, it demonstrates how to put data in it.
## We have two cnvs here, cnv1 and cnv2.
## The data is just completely random.
set.seed(13)
x <- data.frame( pid       = c(1,1,1,1,1),
                 id        = c(1,2,3,4,5),
                 idfath    = c(4,4,4,0,0),
                 idmoth    = c(5,5,5,0,0),
                 sex       = c(1,2,1,1,2),
                 AffectionStatus = c(1,0,0,1,0),
                 cnv1.1    = runif(5),
                 cnv1.2    = runif(5),
                 cnv1.3    = runif(5),
                 cnv2.1    = runif(5),
                 cnv2.2    = runif(5),
                 cnv2.3    = runif(5) )
x
myCPed <- as.cped( x ) # Mark it with the class 'cped'
myCPed
```

```
## Not run:
####################
## Second Example ##

## Again, a completely random dataset.
## Here we go through an analysis of it.
## However, see pbat.m for many more details on all of the options.
## Create a completely random dataset with one cnv.
set.seed(13)
NUMTRIOS <- 500
## The data is completely random, it does not really make any sense.
cped <- as.cped(data.frame(
  pid            = kronecker(1:NUMTRIOS, rep(1,3)),
  id             = rep(1:3, NUMTRIOS),
  idfath         = rep(c(0,0,1), NUMTRIOS),
  idmoth         = rep(c(0,0,2), NUMTRIOS),
  sex            = rep(c(2,1,1), NUMTRIOS),
  AffectionStatus = rep(c(0,0,2), NUMTRIOS),
  cnv1.1         = runif(3*NUMTRIOS),
  cnv1.2         = runif(3*NUMTRIOS),
  cnv1.3         = runif(3*NUMTRIOS)))

## Print out part of the dataset
print(head(cped))

## Command line run
pbat.work() ## Makes the intermediate files go in ./pbatRwork directory

## - Analyzing the first intensity
res1 <- pbat.m(AffectionStatus ~ NONE, ped=cped, phe=NULL, fbat="gee",
               cnv.intensity=1, cnv.intensity.num=3, offset="none")
pbat.clean(res1, all.output=TRUE) ## Removes all intermediate files
## - Analyzing the second intensity
res2 <- pbat.m(AffectionStatus ~ NONE, ped=cped, phe=NULL, fbat="gee",
               cnv.intensity=2, cnv.intensity.num=3, offset="none")
pbat.clean(res2, all.output=TRUE)
## - Analyzing the third intensity
res3 <- pbat.m(AffectionStatus ~ NONE, ped=cped, phe=NULL, fbat="gee",
               cnv.intensity=3, cnv.intensity.num=3, offset="none")
pbat.clean(res3, all.output=TRUE)

pbat.unwork() ## Close up work (head to original working directory)

## Print all of the results
print(res1$results)
print(res2$results)
print(res3$results)

## Or put all the results together and write to file
res1$results <- rbind(res1$results, res2$results, res3$results)
write.pbat(res1, "cpedResults.csv")
```

```
## Otherwise, we could write the data to disk,
##  and run with the GUI interface
## Write the data to disk:
write.cped("cped.cped", cped)

## End(Not run)
```

---

generic   *'pbat' Object Generic Routines*

Description

Summary routines for the results of 'pbat', 'pbat.last', 'pbat.m', 'pbat.obj', and 'pbat.files'.

Only a logrank analysis can be plotted.

Usage

```
## S3 method for class 'pbat'
print(x,...)

## S3 method for class 'pbat'
summary(object,...)

## S3 method for class 'pbat'
plot(x,...)
```

Arguments

x	'pbat' object (result of pbat.m(...) or pbat()).
object	'pbat' object (result of pbat.m(...) or pbat()).
...	Ignored. Needed for S3 generic method consistency.

References

Jiang, H., Harrington, D., Raby, B. A., Bertram, L., Blacker, D., Weiss, S. T. & Lange, C. (2005) Family-based association test for time-to-onset data with time-dependent differences between the hazard functions. Genetic Epidemiology.

See Also

pbat, pbat.last

---

obfuscate　　　　　　　　　　　　　*Obfuscating Pedigree and Phenotype Files*

Description

Randomly permutes the data in a pedigree or phenotype file and changes column headers so as to mangle the data. Used for debugging requests, in the hopes that the mangled data will produce the same bad output, but will not be identifiable.

Usage

```
obfuscate(obj)
```

Arguments

obj　　　　　　　object of class 'ped' or 'phe' to be used (the same object type is returned, only mangled; you must write this out to file).

See Also

pbat pbat.help

Examples

```
## Not run:
ped <- read.ped( "myped" );  ## reads in myped.ped
oped <- obfuscate( "myped" );
write.ped( "obfuscate.ped", ped );

phe <- read.phe( "myphe" );  ## reads in myphe.phe
ophe <- obfuscate( "myphe" );
write.phe( "obfuscate.phe", ped );

## End(Not run)
```

---

pbat　　　　　　　　　　　　　*PBAT Graphical and Command Line Interface*

Description

The following routines are for the graphical and command line pbat interface. The command line interfaces are listed in an order of suggested usage. Most users of the command line will only want to use pbat.m.

pbat runs a GUI (Graphical User Interface) for pbat.

pbat.last returns an object of class pbat of the last command file run from running pbat(). Note this is also returned from pbat. However, this command is provided because rerunning a command in pbat can be a very time-consuming process).

pbat.last.rawResults prints out the raw text file of the output (particularly useful if the output of pbat cannot be parsed properly, in the unexpected event the output could not be parsed correctly). This should work even with the new option of not loading the output in.

pbat.m runs pbat according to an expression, from phe class (phenotype information), ped class (pedigree information), and various options.

pbat.obj runs pbat with a ped class object (pedigree information), a 'phe' class object (phenotype information), and various other options.

pbat.files runs pbat according to a set of filenames and commands.

pbat.create.commandfile creates a command file for Christoph Lange's pbat software with respect to two files on disk (.phe, .ped).

Some options are only available for the respective pbat-gee (G), pbat-pc (P), pbat-logrank (L). If a parameter is 'R'equired for a specific version, it will be denoted, for example, by (G-R).

Usage

```
pbat()

pbat.last()

pbat.last.rawResults()

pbat.m(
  formula, phe, ped, fbat="",
  max.pheno=1, min.pheno=1,
  null="no linkage, no association", alpha=0.05,
  trans.pheno="none", trans.pred="none", trans.inter="none",
  scan.pred="all", scan.inter="all",
  scan.genetic="additive",
  offset="gee",
  screening="conditional power", distribution="default",
  logfile="",
  max.gee=1,
  max.ped=14, min.info=0,
  incl.ambhaplos=TRUE, infer.mis.snp=FALSE,
  sub.haplos=FALSE, length.haplos=2, adj.snps=TRUE,
  overall.haplo=FALSE, cutoff.haplo=FALSE,
  output="normal",
  max.mating.types=10000,
  commandfile="",
  future.expansion=NULL,
  LOAD.OUTPUT=TRUE,
  monte=0,
  mminsnps=NULL, mmaxsnps=NULL,
  mminphenos=NULL, mmaxphenos=NULL,
```

```
    env.cor.adjust=FALSE,
    gwa=FALSE,
    snppedfile=FALSE,
    extended.pedigree.snp.fix=FALSE,
    new.ped.algo=FALSE,
    cnv.intensity=2, cnv.intensity.num=3 )

pbat.obj(
    phe, ped, file.prefix, phenos="",
    offset="gee", LOAD.OUTPUT=TRUE, ...)

pbat.files(
    pedfile, phefile, fbat="gee",
    commandfile="",
    logrank.outfile="",
    preds="", preds.order="",
    max.pheno=1,
    LOAD.OUTPUT=TRUE,
    ...)

pbat.create.commandfile(
    pedfile, phefile="",
    snps="",
    phenos="", time="", # (set only one)
    preds="", preds.order="",
    inters="",
    groups.var="", groups="",
    fbat="gee",
    censor="",
    max.pheno=1, min.pheno=1,
    null="no linkage, no association", alpha=0.05,
    trans.pheno="none", trans.pred="none", trans.inter="none",
    scan.pred="all", scan.inter="all",
    scan.genetic="additive",
    offset="gee",
    screening="conditional power", distribution="default",
    logfile="",
    max.gee=1,
    max.ped=7, min.info=0,
    haplos=NULL, incl.ambhaplos=TRUE, infer.mis.snp=FALSE,
    sub.haplos=FALSE, length.haplos=2, adj.snps=TRUE,
    overall.haplo=FALSE, cutoff.haplo=FALSE,
    output="normal",
    max.mating.types=10000,
    commandfile="",
    future.expansion=NULL,
    LOGFILE.OVERRIDE=TRUE,
    monte=0,
```

```
            mminsnps=NULL, mmaxsnps=NULL,
            mminphenos=NULL, mmaxphenos=NULL,
            env.cor.adjust=FALSE,
            gwa=FALSE,
            snppedfile=FALSE,
            extended.pedigree.snp.fix=FALSE,
            new.ped.algo=FALSE,
            cnv.intensity=2, cnv.intensity.num=3 )
```

Arguments

formula	Symbolic expression describing what should be processed. See 'examples' for more information.
phe	'phe' object as described in `write.phe`. If you do not have a phe file set this to NULL (i.e when you are only using AffectionStatus from the pedigree).
ped	'ped' object as described in `write.ped`.
file.prefix	Prefix of the output datafile (phe & ped must match)
pedfile	Name of the pedigree file (.ped/.pped/.cped) in PBAT-format (extension '.ped' is optional).
phefile	Name of the phenotype file (.phe) in PBAT-format. The default assumes the same prefix as that in 'pedfile'. Leave empty or set to the empty string "" if you do not have a phenotype file (i.e. you are only using AffecitonStatus). In the case of no phenotype file, one must be created; it will be in empty_phe.phe, and requires loading in the pedigree file into R.
...	Options in higher level functions to be passed to 'pbat.create.commandfile'.
fbat	Selects the fbat statistic used the data analysis.
	"gee" = The FBAT-GEE statistic simplifies to the standard univariate FBAT-statistic. If several phenotypes are selected, all phenotypes are tested simultaneously, using FBAT-GEE. The FBAT-GEE statistic can handle any type of multivariate data.
	"pc" = FBAT extension for longitudinal phenotypes and repeated measurements.
	"logrank" = FBAT-extensions of the classical LOGRANK and WILCOXON tests for time-on-onset data. Kaplan-Meier plots for the analyzed data set will be generated and plotted.
max.pheno	(G,P) The maximum number of phenotypes that will be analyzed in the FBAT-statistic.
min.pheno	(G,P) The minimum number of phenotypes that will be analyzed in the FBAT-statistic.
null	Specification of the null-hypothesis.
	"no linkage, no association" = Null-hypothesis of no linkage and no association.
	"linkage, no association" = Null-hypothesis of linkage, but no association.
alpha	Specification of the significance level.

trans.pheno	Transformation of the selected phenotypes.
	"none" = no transformation
	"ranks" = transformation to ranks
	"normal score" = transformation to normal score
	The default choice is "none", although it recommended to use transformation to normal scores for quantitative phenotypes.
trans.pred	Transformation of the selected predictor variables/covariates:
	"none" = no transformation
	"ranks" = transformation to ranks
	"normal score" = transformation to normal score
	The default choice is "none", although it recommended to use transformation to normal scores for quantitative covariates.
trans.inter	Transformation of the selected interaction variables
	"none" = no transformation
	"ranks" = transformation to ranks
	"normal score" = transformation to normal score
	The default choice is "none", although it recommended to use transformation to normal scores for quantitative interaction variables.
scan.pred	(G,P) Computation of all covariate sub-models:
	"all" = The selected FBAT statistic is computed with adjustment for all selected covariates/predictors.
	"subsets" = The selected FBAT statistic is computed for all possible subsets of the selected covariates/predictor variables. The command is particularly useful to examine the dependence of significant results on the selection of a covariate model.
scan.inter	(G,P) Computation of all interaction sub-models:
	"all" = The selected FBAT statistic is computed including all selected interaction variables.
	"subsets" = The selected FBAT statistic is computed for all posible subsets of the interaction variables.
scan.genetic	Specification of the mode of inheritance:
	"additive" = Additive model
	"dominant" = Dominant model
	"recessive" = Recessive model
	"heterozygous advantage" = Heterozygous advantage model
	"all" = The FBAT-statistics are computed for all 4 genetic models
offset	Specification of the covariate/predictor variables adjustment:
	"none" = No adjustments for covariates/predictor variables. You need to select this for dichotomous traits.
	"max power" = Offset (=FBAT adjustment for covariates and interaction variables) that maximizes the power of the FBAT-statistic (computationally slow, efficiency dependent on the correct choice of the mode of inheritance)

"gee + marker score" = Offset (=FBAT adjustment for covariates and interaction variables) based on standard phenotypic residuals obtained by GEE-estimation including the expected marker score (E(X|H0)), all covariates and interaction variables.

"gee" = Offset (=FBAT adjustment for covariates and interaction variables) based on standard phenotypic residuals obtained by GEE-estimation including all covariates and interaction variables. (default - most of the time, with the exception of selecting from the gui interface (not the command line) Affection-Status)

(numeric value) = This only works for AffectionStatus; set a numeric value (i.e. '0.13' without the '' marks) to this.

screening	Specification of the screening methods to handle the multiple comparison problem for multiple SNPs/haplotypes and a set of phenotypes.
	"conditional power" = Screening based on conditional power (parametric approach)
	"wald" = Screening based on Wald-tests (non-parametric approach)
distribution	Screening specification of the empirical phenotypic distribution
	"default"
	"jiang" = Approach by Jiang et al (2006)
	"murphy" = Approach by Murphy et al (2006)
	"naive" = Naive allele freq estimator
	"observed" = Observed allele frequencies
logfile	Specification of the log-file. By default, PBAT selects an unique file-name for the log-file, i.e. "pbatlog...".
max.gee	(G) Specification of the maximal number of iterations in the GEE-estimation procedure.
max.ped	Specification of the maximal number of proband in one extended pedigrees.
min.info	Specification of the minimum number of informative families required for the computation of the FBAT-statistics.
incl.ambhaplos	This command defines the handling of ambiguous haplotypes in the haplotypes analysis. Choices:
	TRUE = Ambiguous haplotypes (phase can not be inferred) are included in the analysis and are weighted according to their estimated frequencies in the probands.
	FALSE = Ambiguous haplotypes are excluded from the analysis.
infer.mis.snp	Handling of missing genotype information in the haplotypes analysis.
	FALSE = Individuals with missing genotype information are excluded from the analysis. This is the analysis also implemented in the HBAT option of the FBAT-program.
	TRUE = Individuals with missing genotype information are included in the analysis. The algorithm of Horvath et al (2004) is applied to all individuals, even if they have missing genotype information. This results in more ambiguous haplotypes.
sub.haplos	FALSE = The haplotypes defined by the all SNPs given in the haplotype-block definition are analyzed.

	TRUE = All haplotypes are analyzed that are defined by any subset of SNPs in the haplotypes block definition.
length.haplos	Defines the haplotype length when subhaplos=TRUE.
adj.snps	Takes effect when subhaplos=TRUE.
	FALSE = All sub-haplotypes are analyzed
	TRUE = Only the sub-haplotypes are analyzed for which the first constituting SNPs are adjacent.
overall.haplo	Specification of an overall haplotypes test. When this command is included in the batch-file, only one level of the "groups" variable can be specified.
	FALSE = no overall test
	TRUE = an overall test is computed testing all haplotypes defined by the same set of SNPs simultaneously. This option can not be applied when sub.haplos=TRUE.
cutoff.haplo	The minimum haplotypes frequency so that a haplotypes is included in the overall test.
output	"normal" = Normal PBAT output.
	"short" = Shorter output. This is mostly for use in conjunction with 'gwa', where there is a lot of output.
	"detailed" = Detailed output for each family is created.
max.mating.types	
	Maximal number of mating types in the haplotype analysis.
commandfile	Name of the temporary command file that will be created to send to the pbat. It is suggested to leave this blank, and an appropriate name will be chosen with a time stamp.
future.expansion	
	(Only included for future expansion of pbat.) A vector of strings for extra lines to write to the batchfile for pbat.
logrank.outfile	
	(L) Name of the file to store the R source code to generate the plots for logrank analysis.
snps	Vector of strings for the SNPs to process. Default processes all of the SNPs.
phenos	(G,P) Vector of strings for the phenotypes/traits for the analysis. If none are specified, then all are analyzed. (Note: this *must be left empty for logrank analysis*, instead specify the time to onset with the time variable.
time	(L-R) Time to onset variable. 'phenos' cannot be specified when this is used, but it *must be set for logrank*.
preds	Vector of strings for the covariates for the test statistic.
preds.order	Vector of integers indicating the order of 'preds' - the order for the vector of covariates for the test statistic.
inters	Vector of strings for the interaction variables.
groups.var	String for the grouping variable.
groups	Vector of strings corresponding to the groups of the grouping variable (groupsVar).
censor	(L-R) String of the censoring variables. In the corresponding data, this variable has to be binary.

haplos List of string vectors representing the haplotype blocks for the haplotype analysis. For example, `list(block1=c("m1","m2"), block2=c("m3","m4"))` defines 2 haplotype-blocks where the first block is defined by SNPs m1 and m2, and the second by SNPs m3 and m4.

LOGFILE.OVERRIDE
When using the 'sym' option in read.ped and read.phe, when this is set to TRUE (default), the PBAT logfile is put in the current working directory; if FALSE, then it is put in the same directory as the datafile.

LOAD.OUTPUT When TRUE, loads the output into R (generally recommended). When FALSE, it leaves it in the output left from PBAT (in case output is too large to load into memory).

monte When this is nonzero, monte-carlo based methods are used to compute the p-values instead, according to the number of iterations supplied. 1000 iterations is suggested.

mminsnps Multi-marker multi-phenotype tests: the minimum number of snps to be tested.

mmaxsnps Multi-marker multi-phenotype tests: the maximum number of snps to be tested.

mminphenos Multi-marker multi-phenotype tests: the minimum number of phenotypes to be tested.

mmaxphenos Multi-marker multi-phenotype tests: the maximum number of phenotypes to be tested.

env.cor.adjust Whether to adjust for environmental correlation.

gwa Whether to use (g)enome (w)ide (a)cceleration mode. This is faster for genome-wide association tests, and has slightly less output.

snppedfile Whether the pedigree file contains just snps. When this is true, it employs a more optimal storage technique and uses much less memory. It is especially advantageous for genome-wide studies.

extended.pedigree.snp.fix
Set to TRUE when you are using a dataset with large extended pedigrees. This will not work with any mode but 'single' mode currently [see pbat.set(...)]. This is also sometimes necessary for multi-allelic markers (i.e. not binary markers).

new.ped.algo Set to TRUE (default is FALSE) to use the new, 10-100 times faster and more memory efficient algorithm. Somewhat experimental with extended pedigrees, so use with caution.

cnv.intensity The CNV intensity number that should be analyzed.

cnv.intensity.num
The number of CNV intensities per CNV in the .cped file.

Details

IF YOU ARE HAVING PROBLEMS: Try setting extended.pedigree.snp.fix, the slowest but most robust method.

INTERPRITING THE OUTPUT:

1) I make every attempt to try to properly header the output, but sometimes this is not possible. You will often see a warning message to this regards, which is generally safe to ignore.

2) 'a'=additive, 'd'=dominant, 'r'=recessive, 'h'=heterozygous advantage

FURTHER USEFUL COMMENTS:

These commands require 'pbatdata.txt' to be in the working directory; if not found, the program will attempt to (1) copy the file from the directory where pbat is, (2) copy it from anywhere in the path, or (3) error and exit.

Linux warning: the file 'pbatdata.txt' appears not to have shipped with the current (as of writing this) linux version; to fix this just download the windows version as well and copy the file from there to the same directory as pbat.

It is recommended to set 'LOAD.OUTPUT' to 'FALSE' when dealing with very large numbers of SNPs.

These commands will also generate a lot of output files in the current working directory when interfacing with pbat. These files will be time-stamped so concurrent analysis in the same directory can be run. *Race condition*: if two logrank analysis finish at *exactly the same time*, then the plots for one might be lost and/or get linked to the wrong analysis. This should be a rather rare occurence, and is an unpreventable result of pbat always sending this output to only one filename. Workaround to race condition: create another directory and use that as your current working directory instead.

Note that multi-marker / multi-phenotype mode is not supported in parallel at this time, so if you are having problems try running the command pbat.setmode("single"), or setting it to single from the graphical interface before running these tests.

WARNING: Note the 'extended.pedigree.snp.fix' option, which is important for getting more accurate results in very extended pedigrees. It uses a slower but more accurate pedigree reconstruction method.

Value

'pbat', 'pbat.last', 'pbat.m', 'pbat.obj', and 'pbat.files' return an object of class pbat. Methods supported by this include plot(...), summary(...), and print(...). Follow the first three links in the 'see also' section of this file for more details.

References

This was taken with only slight modification to accomodate the interface from Christoph Lange's description of the commands for the pbat program, (which was available with the software at the time of this writing).

P2BAT webpage.

FBAT webpage (lists a lot of references in relation to both of these programs).

More pbat references:

Hoffmann, T. and Lange, C. (2006) P2BAT: a massive parallel implementation of PBAT for genome-wide association studies in R. Bioinformatics. Dec 15;22(24):3103-5.

Jiang, H., et al. (2006) Family-based association test for time-to-onset data with time-dependent differences between the hazard functions. Genet. Epidemiol, 30, 124-132.

Laird, N.M. and Lange, C. (2006) Family-based designs in the age of large-scale gene-association studies. Nat. Rev. Genet, 7.

Lange, C., et al. (2003) Using the noninformative families in family-based association tests: a powerful new testing strategy. Am. J. Hum. Genet, 73, 801-811.

Lange, C., et al. (2004a) A family-based association test for repeatedly measured quantitative traits adjusting for unknown environmental and/or polygenic effects. Stat. Appl. Genet. Mol. Biol, 3.

Lange, C., et al. (2004b) Family-based association tests for survival and times-to-onset analysis. Stat. Med, 23, 179-189.

Van Steen, K., et al. (2005) Genomic screening and replication using the same data set in family-based association testing. Nat. Genet, 37, 683-691.

See Also

summary.pbat, plot.pbat, print.pbat,

as.ped, as.pedlist, read.ped

as.phe, read.phe,

top

Examples

```
###########################
## pbat.m(...) examples ##
###########################

## Not run:

## Note, when you run the example (or anything else) you will generally
##  get a warning message that the column headers were guessed.
## This means they were guessed, and while I've tried to catch most
##  cases, the warning stands for ones I might have missed.

## These cannot be run verbatim, and are just meant to be examples.

###############################
## Further formula examples ##
###############################


# load in the data
# Here we assume that:
#  data.phe contains 'preds1', 'preds2', 'preds3', 'time',
#                     'censor', 'phenos1', ... 'phenos4'
#  data.ped contains 'snp1', 'snp2', 'snp3',
#                     'block1snp1','block1snp2',
#                     'block2snp1','block2snp2'
data.phe <- read.phe( "data" )
data.ped <- read.ped( "data" )

# This model does just the affection status (always given as
#  AffectionStatus) as the phenotype, no predictor covariates, and all
#  the snps for a snps analysis.
# Since affection status is dichotomous, we additionally set
#  distribution='categorical'
#  offset='none'
```

```
# NONE is a special keyword to indicate none, and can be only used in
#  this case (note that it is _case_ _sensative_);
#  otherwise one specifies values from the phenotype object, after and
#  including AffectionStatus.
res <- pbat.m( AffectionStatus ~ NONE, phe, ped, fbat="gee",
                distribution='categorical', offset='none', ... )
summary( res )
res  # equivalent to print(res)

# basic model with one phenotype, does all snps (if none specified)
pbat.m( phenos1 ~ preds1, phe, ped, fbat="gee" )

# same model, but with more phenotypes; here we test them all at once
pbat.m( phenos1 + phenos2 + phenos3 ~ preds1, phe, ped, fbat="gee" )

# same model as just before, but now supposing that these phenotypes are
#  instead from a longitudinal study
pbat.m( phenos1 + phenos2 + phenos3 ~ preds1, phe, ped, fbat="pc" )

# like our second model, but the mi() tells it should be a marker
#  interaction
pbat.m( phenos1 ~ mi(preds1), phe, ped, fbat="gee" )

# logrank analysis - fbat need not be set
# uses more than one predictor variable
res <- pbat.m( time & censor ~ preds1 + preds2 + preds3, phe, ped )
plot( res )

# single snp analysis (because each snp is seperated by a vertical bar
#  '|'), and stratified by group (presence of censor auto-indicates
#  log-rank analysis).  Note that the group is at the end of the
#  expression, and _must_ be at the end of the expression
res <- pbat.m( time & censor ~ preds1^3 + preds2 | snp1 | snp2 |
        snp3 / group, temp )
plot( res )

# haplotype analysis, stratified by group
res <- pbat.m( time & censor ~ preds1^2 + preds2^3 | block1snp1
                + block1snp2 | block2snp1 + block2snp2 / group, temp )

# set any of the various options
res <- pbat.m( phenos ~ preds, phe, ped, fbat="pc",
                null="linkage, no association", alpha=0.1 )

## New multimarker test (as described above)
# mmaxphenos and mmaxsnps are set to the minimum if not specified
res <- pbat.m( phenos1 + phenos2 + phenos3 ~ preds | m1 | m2 | m3 | m4,
                phe, ped, fbat="pc", mminphenos=2, mminsnps=2 )

## And the top markers by conditional power
top( res )

## End(Not run)
```

```
#############################
## pbat.obj(...) examples ##
#############################

## Not run:
# These will not function; they only serve as examples.

# ... just indicates there are various options to be put here!
res <- pbat.obj("pedfile", snps=c("snp1,snp2"), preds="pred1", ... )
summary(res)
res

# plot is only available for "logrank"
res <- pbat.obj(..., fbat="logrank")
plot( res )

## End(Not run)


###############################
## pbat.files(...) examples ##
###############################

## Not run:
# These will not function, but only serve as examples.

# Note in the following example, both "pedfile.ped" and "pedfile.phe"
#  must exist.  If the names differed, then you must specify the
#  option 'phe="phefile.phe"', for example.
res <- pbat.files( "pedfile", phenos=c("phenos1","phenos2"),
                   screening="conditional power" )
summary(res)
res

## End(Not run)
```

---

pbat.help   *Help and Reporting Information*

Description

Provides some helpful suggestions and information if you are having some troubles getting the package set up (often tends to be version related). Information useful for reporting.

Usage

```
pbat.help(bug=FALSE, full=TRUE, ped=NULL, phe=NULL, lib.loc=NULL)
pbat.firsttime()
```

Arguments

bug	Whether you think a bug was found on a run. Will attempt to generate some useful information for the developers.
full	When bug=TRUE, runs a full bug report (slower, reruns analysis, but recommended.
ped	Pedigree object that can be obfuscated (see 'obfuscate').
phe	Phenotype object that can be obfuscated (see 'obfuscate').
lib.loc	a character vector describing the location of R library trees to search through, or 'NULL', i.e what/if you specified for this when the library was loaded.

See Also

[pbat](#) [obfuscate](#)

---

    pbat.set   *Set Pbat Parameters*

Description

All of these values are retained, even after closing and restarting R.

pbat.set sets the name of the pbat executable.

pbat.get gets the current stored name of the pbat executable.

pbat.setmode sets the mode (single, multiple, or cluster), with pbat.setmode.defaults returning it to default values.

pbat.getmode returns the current mode specifications.

pbat.setVersionCheck sets whether to check the version on startup.

pbat.setwine sets pbat up for use with wine. Intended for mac OS (darwine), or linux 32-bit version (temporarily with wine). Download the windows version of pbat, and use pbat.set to point it to pbat*.exe. See details below. pbat.getwine returns the value, leave empty for windows

Usage

```
pbat.set(executableStr="", CLEAR=FALSE)
pbat.get()

pbat.setmode.defaults(save=TRUE)
pbat.setmode(mode=NULL, jobs=NULL, clusterCommand=NULL,
             clusterRefresh=NULL )
pbat.getmode()

pbat.setVersionCheck(check=TRUE)

pbat.setwine(wineStr="", CLEAR=FALSE)
pbat.getwine()
```

Arguments

executableStr	String of the pbat executable name, e.g. "c:/pbat/pbat25.exe". If the string is omitted, then a file-choose dialogue will appear to selet the file (suggested).
CLEAR	If set to TRUE, the executable name is cleared, regardless of the string passed in the other option.
save	Used internally.
mode	'single', 'multiple', or 'cluster'. See details.
jobs	The number of jobs to partition into for 'multiple' or 'cluster'.
clusterCommand	The command used to submit a job to your cluster. The default is 'bsub'.
clusterRefresh	How often (in seconds) to check if jobs have finished.
check	Whether to perform the version check on startup (recommended).
wineStr	Executable binary for wine. For linux 32-bit (64-bit is natively compiled), "wine" should suffice, assuming it is installed. For Mac, assuming you have installed Darwine (as detailed in the package), you should set it to be "/Applications/Darwine/Wine.bundle/Contents/bin/wine", NOT the wineHelper.

Details

The default (upon installation) assumes that the executable is 'pbat' and is in the path; pbat.set allows you to set the version you wish to use, and easily toggle between other versions of the software (in case you are testing the software it is easy to toggle to an older version).

————————————————————————————

The 'single' mode is meant for only one processor on a users personal computer (see also cluster for some special features of that mode). However, with putting several cores on one chip, it may be more advantageous to use 'multiple' or 'cluster'.

WARNING: if the number of jobs is 1, it will always be in this 'single' mode.

————————————————————————————

The 'multiple' mode is meant for a single multiprocessor system, or a cluster that acts like one. You can then set 'jobs' to be however many processors that you have, or more depending on some of the newer technologies. If you have a cluster that acts like a multi-processor system, then this can be more efficient than the 'cluster' mode *if and only if you are waiting for the output*. The 'multiple' mode will work in all operating systems.

————————————————————————————

In 'cluster' mode, you must specify the command and any additional flags (to keep it generic for all platforms) to submit a job that is represented in a file (i.e 'bsub -q normal sh' on some clusters, 'batch -f' or 'at now -f' in a very basic unix environment - see 'man batch' from the command line or search for it online).

When the 'refresh' option is set to '0' (zero), then the output is batched, and R will not wait for the output to be finished. If you do this, be sure to save your workspace when exiting; then you can reload your output back in. See is.finished, pbat.load, and pbat.concatenate for more information on this.

When 'refresh' is set to an integer greater than zero, it indicates the seconds that should elapse before the current R session checks to see if the other processes are done. The goal was to keep this as generic as possible.

Cluster mode may work in windows, but I am unfamiliar with any batching command system available in windows. Would there actually be any interest in this? Please e-mail me if this might be useful.

_____

System administrators could copy the '.pbat.Rmeta' file to the users home directory after using `pbat.set` to set it themselves.

Value

`pbat.get` returns a string of the currently stored name of the pbat executable.

See Also

[pbat.m](#)

[is.finished](#), [pbat.load](#), [pbat.concatenate](#)

_____

pbat.status   *PBAT Status of Run*
_____

Description

Provides status information, i.e. if there were any errors.

Usage

```
pbat.status(n=1,workFirst=FALSE)
```

Arguments

n   Returns the last 'n' lines from the status file. If n=0, all lines are returned.

workFirst   Whether to check for './pbatRwork/pbatstatus.txt' or './pbatstatus.txt' first. Generally only semi-important internally, but set this to true if you just typed `pbat.unwork()`.

See Also

[pbat](#)

---

pbat.work   *pbatRwork Temp Directory*

Description

Creates a temporary 'pbatRwork' directory, and moves to it. This can be created in the same directory as a pedigree or phenotype file by passing those as parameters instead.

Usage

```
pbat.work( pedOrPhe=NULL )
pbat.unwork( cur=NULL )
```

Arguments

pedOrPhe   If a 'ped' or 'phe' symbolic object (read in without 'sym=TRUE' which is the
                default), then uses that directory + '/pbatRwork' as the current working directory
                (so all output will go there). Otherwise it just creates a 'pbatRwork' directory in
                the current working directory.

cur   Pass in the output of pbat.work(...), see example below.

Examples

```
## Not run:
ped <- read.ped("mydata")
phe <- read.phe("myphe")

cur <- pbat.work( ped )

res <- pbat.m(trait ~ NONE, ped=ped, phe=phe, fbat="gee")
...
## whatever you want to do with the results

pbat.unwork( cur ) ## sends you back to where you were

## End(Not run)
```

---

ped   *Pedigree Object*

Description

Creates, tests, reads, or writes objects of type ped or pedlist to be used with the pbat commands.

The ped class inherits the data.frame structure, and is almost identical to the data.frame object described to create it, only with some special reserved names.

The pedlist class inherits the list structure, and is almost identical to the list object described to create it, only with some special reserved names.

The 'pped' functions provide support for a more compressed form of input, that can be read in faster, and so may be faster when running in clustered mode.

COMPATIBILITY ISSUE 2024-01-27: The function ped.markerNames was renamed and is now pedMarkerNames. Failed new S3 check (which enforced that, as named, needed to be an S3 method).

Usage

```
as.ped(x,
  pid="pid", id="id", idfath="idfath",
  idmoth="idmoth", sex="sex", affection="AffectionStatus",
  clearSym=FALSE)

as.pedlist(x,
  pid="pid", id="id", idfath="idfath",
  idmoth="idmoth", sex="sex", affection="AffectionStatus",
  clearSym=FALSE)

is.ped(obj, pure.ped=FALSE)

is.pedlist(obj)

read.ped(filename, format="ped", lowercase=TRUE, sym=TRUE, max=100, ...)
fread.ped(filename, ...)

write.ped(file, ped)

is.pped(obj)

read.pped(filename, max=100)

as.pped(ped, ppedname="")

## S3 method for class 'ped'
sort(x,decreasing=FALSE,...)

plotPed(ped, sink=NULL, geno="percent", phe=NULL, pheCols=NULL)

pedMarkerNames(ped)
```

Arguments

x	An object of class ped, pedlist, data.frame, or list as described below.
	If x is of class ped or pedlist, no other options are used.
	When x is of class data.frame, the columns have entries that match the string parameters idped,...,AffectionStatus; genetic markers consist of *subsequent* columns formated as follows: e.g. marker 'p5' would need two columns known as 'p5.a' 'p5.b' (or really just where the characters a and b are different).
	When x is of class list, the entries still include idped,...,censor as discussed previously, and genetic markers consist of a list object of two vectors for each unphased haplotype.
pid	String corresponding to column name for pedigree id.
id	String corresponding to column name for subject id.
idfath	String corresponding to column name for father id.
idmoth	String corresponding to column name for mother id.
sex	String corresponding to column name for sex.
affection	String corresponding to column name for affection status.
filename	Filename to open; does not need .phe extension.
format	Toggles the return structure, set to "ped" or "pedlist".
lowercase	When TRUE (and sym is FALSE), enforces all headers to lowercase for convenience.
...	Options for [read.table](), used only when sym is FALSE. Do *not* put in header=TRUE, as this will cause an error, as the header is automatically loaded.
	With the proper file formatting, this should not be used.
file	string representing filename, or a connection for file output
ped	an object of class ped or pedlist (see as.ped or as.pedlist)
obj	an object
sym	When TRUE, only the header of the file is read in; only PBAT will load in the file. When FALSE, the entire file will be read in, and can be modified before using with PBAT.
max	When sym is TRUE, the amount of headers to read in before going pure symbolic (so that the SNP usage consistency will not be assessed by pbatR, only by PBAT).
clearSym	When TRUE, if a symbolic file is found, it will be read in; otherwise, it will stay symbolic.
pure.ped	When FALSE, tests if an object is a 'ped' or 'pped'. When TRUE, tests only if the object is a 'ped'.
ppedname	Name of the 'pped' file. If a symbolic ped, it defaults to that name except with a pped extension; otherwise, it defaults to 'pped.pped'.
decreasing	Whether to sort in decreasing/increasing order.
sink	For 'plot.ped', this is the name of a pdf file to output all of the plots to (there will be one plot per page).

geno	For 'plot.ped'; "none" includes no genotype summary information, "percent" gives an overall summary of how much each individual is genotyped, i.e. non-missing (the default), and "each" gives all of the genotypes of each individual.
phe	For 'plot.ped', a phenotype object for extra information.
pheCols	For 'plot.ped', names in the phenotype object of columns that should be displayed in the pedigree.

Details

When reading in a file on disk using read.ped, a '.ped' file should have the following format (taken from the PBAT web-page). The first line of the PBAT pedigree file contains the names of the markers. Each subsequent line stands for one individual/subject, starting with the pedigree id, followed by the individual/subject id, the id of the father, the id of the mother, the individual's sex and affection status. After this information, for each marker, both marker alleles are listed. The order of the markers has to correspond to the order of the marker names in the first line of the file. Missing values here must be encoded with a '0', unlike the phenotype file. Examples of this type of file can be found on the PBAT webpage.

The usage of as.ped and as.pedlist should also follow the same missingness convention.

'plot.ped' attempts to make use of the 'kinship' package to draw the pedigrees. In my personal experience, this package cannot handle all pedigrees. My preferred alternative would be to use Madeline, which makes beautiful pictures (amongst other things): http://eyegene.ophthy.med.umich.edu/

ped.markerNames returns the names of the markers, without the '.a' and '.b' extension for markers (and not duplicated for markers).

See Also

read.ped, write.ped, as.pedlist

Examples

```
# A highly artificial example with not enough subjects to be run;
#  however, it demonstrates how to put data in it.
x <- data.frame( pid        = c(1,1,1,1,1),
                 id         = c(1,2,3,4,5),
                 idfath     = c(4,4,4,0,0),
                 idmoth     = c(5,5,5,0,0),
                 sex        = c(1,2,1,1,2),
                 AffectionStatus = c(1,0,0,1,0),
                 m1.a       = c(1,1,1,1,1),
                 m1.b       = c(1,2,1,1,2),
                 m2.a       = c(4,4,4,4,4),
                 m2.b       = c(3,3,3,4,3) )
x
myPed <- as.ped( x )        # Mark it with the class 'ped'
myPedlist <- as.pedlist( x )  # Instead mark it with 'pedlist'
myPed
myPedlist

# an alternate example of creating
```

```
names( x )[1:6] <- c( "mypedid", "subid", "fathid",
                      "mothid", "gender", "affection" );
x
myPed <- as.ped( x, pid="mypedid", id="subid", idfath="fathid",
                 idmoth="mothid", sex="gender", affection="affection" )
myPed  # Note it's the same as before!

myPed <- as.ped( myPedlist )      # Easy conversion back
myPedlist <- as.pedlist( myPed )  #  and forth between formats.
```

---

phe   *Phenotype Object*

Description

Creates, tests, reads, or writes an object of class phe.

Usage

```
as.phe(df, pid="pid", id="id")

is.phe(obj)

read.phe(
  filename, na.strings=c("-",".","NA"),
  lowercase=TRUE, sym=TRUE, ...)
fread.phe(filename, ...)

write.phe(file, phe)

## S3 method for class 'phe'
sort(x,decreasing=FALSE,...)
```

Arguments

df	Dataframe with the data
pid	String for the column header for 'pid' - pedigree ID.
id	String for the column header for 'id' - subject ID.
obj	any object
filename	Filename to open; does not need .phe extension.
na.strings	Strings that represent NA; defaults should be fine here.
lowercase	When TRUE (default), enforces all headers to lowercase for convenience.
sym	When TRUE, only the header of the file is read in; only PBAT will load in the file (* - see exception). When FALSE, the entire file will be read in, and can be modified before using with PBAT.

...	Options for `read.table`. Do *not* put in `header=TRUE`, as this will cause an error, as the header is automatically loaded. With the proper file formatting, this should not be used.
file	string representing filename, or a connection for file output
phe	An object of class 'phe' (see `as.phe`).
x	An object of class 'phe' (see `as.phe`).
decreasing	Whether to sort in decreasing/increasing order.

Details

When reading in a file on disk using `read.ped`, a '.phe' file should have the following format (taken from the PBAT web-page). The first line contains the names of the covariates and phenotypes and the subsequent lines contain the pedigree id, the id of the subject, followed by the values of the covariates and phenotypes for that subject. Here missing values must be indicate with a '.' or '-', unlike the pedigree file. Examples of this type of file can be found on the PBAT webpage.

Once the dataset is read in, missing values are converted into the usual R format of NA (see NA in the help files).

When using `as.phe`, missing values should be in the native R format. The `write.ped` function will convert back into the missing format necessary for PBAT.

(*) Exception. If symbolic is true, the dataset will be temporarily read in under one special circumstance. This is when a stratification variable is used in pbat.m, pbat.obj, or pbat.files. This is because the grouping values must be read in. Alternatively, you can specify these values when calling those functions (see the groups.* options) in pbat.

Note

'read.phe' mild warning: This function *might change the names of headers*, so they may not correspond to what pbat is looking for if you call any of the pbat*files() commands (*even* if lowercase=FALSE, as some symbols might be modified by R naming conventions). In general this should not happen.

See Also

`read.ped`, `write.ped`, `as.ped`, `as.pedlist`

Examples

```
# A highly artificial example just to get you used to the syntax
#  of using 'as.phe'.
x <- data.frame( pid    = c(1,1,2,2,2),
                 id     = c(1,2,3,4,5),
                 age    = c(14,45,33,22,21),
                 weight = c(150,100,180,185,110) )
x
myPhe <- as.phe( x );
myPhe

# And just another e.g. capitalizing on prior code
```

```
names(x)[1:2] <- c("thepid","theid")
x
myPhe <- as.phe( x, pid="thepid", id="theid" )
myPhe  # same as before!
```

---

power *Power Exploration*

Description

Power has been completely rewritten from scratch, and is all done via monte carlo simulation internally now. These routines do not require pbat, and should run on any machine.

Usage

```
pbat.power(mode="continuous")

pbat.powerCmd(
  numOffspring=1, numParents=2, numFamilies=500,
  additionalOffspringPhenos=TRUE,
  ascertainment="affected",
  modelGen="additive", modelTest=modelGen,
  afreqMarker=NA,
  penAA=0.8, penAB=0.5, penBB=0.3,
  heritability=0.0, contsAscertainmentLower=0.0,
  contsAscertainmentUpper=1.0,
  pDiseaseAlleleGivenMarkerAllele=1.0, afreqDSL=0.1,
  alpha=0.01,
  offset="default",
  numSim=1000,
  ITERATION_KILLER=200 )
```

Arguments

mode	"continuous" or "dichotomous"
numOffspring	Family - number of offspring
numParents	Family - number of parents (0,1,2)
numFamilies	Family - number of families
additionalOffspringPhenos	
	Only used when you have missing parents; additional offspring phenotypes. 1 for yes, 0 for no.
ascertainment	'unaffected', 'affected', or 'na' for anyone
modelGen	The model used when generating the simulated data - one of 'additive', 'dominant', 'recessive'.
modelTest	The model used to test the simulated data.

afreqMarker   allele frequency at the marker

penAA   penetrance of AA genotype

penAB   penetrance of AB genotype

penBB   penetrance of BB genotype

heritability   heritibility - when this is zero, a binary trait according to the previously defined parameters is used; when it is nonzero, a continuous trait is used.

contsAscertainmentLower

      Lower bound for affected ascertainment with a continuous trait, this is a vector for each member after the proband, defaulting to '0'. It represents the quantiles, so 0.05 would indicate that the lower 5 percent of the phenotypes should be removed.

contsAscertainmentUpper

      Upper bound, defaults to '1'.

pDiseaseAlleleGivenMarkerAllele

      Pr(Disease allele A|marker allele A)

afreqDSL   allele frequency at DSL, defaults to marker frequency.

alpha   significance level

offset   "default" uses the population prevalence for dichotomous traits and the population mean for continuous traits. If a number is specified, then that number is used as the offset.

numSim   Number of monte-carlo simulations. I'd use a smaller number while starting out with it, and then turn it up to a much higher number of iterations later on.

ITERATION_KILLER

      Controls how many times to try to draw data when simulating, before giving up. A value of 0 indicates to never stop. This is useful if you are playing around and are considering a situation that is too difficult for this program to be able to simulate.

Details

pbat.powerCmd(...) does not really do any range checking, primarily because I don't expect most will use it directly, and instead will use the friendly GUI interface for power exploration.

Be careful with the number of simulations! When you are first exploring, you can keep this low, but you should turn this all the way up before doing your final computation.

Note that some values of 'pDiseaseAlleleGivenMarkerAllele' in combination with 'afreqMarker' are not possible. These will return negative values (these are error codes for the GUI, which will provide more helpful messages).

Lastly, you might want to look into something like set.seed(1) e.g., if you want the results to be reproducable (set it to any number, but make note of this number, see set.seed for more details).

References

Hoffmann, T. and Lange, C. (2006) P2BAT: a massive parallel implementation of PBAT for genome-wide association studies in R. Bioinformatics. Dec 15;22(24):3103-5.

Horvath, Steve, Xu, Xin, and Laird, Nan M. The family based association test method: computing means and variances for general statistics. Tech Report.

See Also

[pbat](#), [pbat.last](#)

---

top	*top*

Description

The top n markers by conditional power.

Usage

```
top(pbatObj, n=10, sortBy=NULL)
```

Arguments

pbatObj	An object of class 'pbat', i.e. the results of 'pbat.m(...)'.
n	How many markers to print. 0 means all will be returned.
sortBy	Name of conditional power column to sort by, for instance if you wanted to look at the GxE, you might specify "FBATGxE" instead.

Details

Prints out the top markers according to conditional power.

---

write.pbat	*Saving 'pbat' Object*

Description

Writes a 'pbat' object (the result of calling pbat.m) to disk.

Usage

```
write.pbat(x, filename, resultsOnly=FALSE)
write.pbat.csv(x, filename, resultsOnly=FALSE)
```

Arguments

x	Object of class pbat.
filename	Filename to save to.
resultsOnly	When TRUE, this only prints the results to file, and not how the function was called. Typically not used.

Details

With `write.pbat`, a text file is written out with an '&' symbol delimiting the output section. Both the calling and results section are preserved. If the file has the '.csv' extension, then `write.pbat.csv` is called.

With `write.pbat.csv`, the text file is of the standard 'csv' format to delimit the text file. This should be readable with any spreadsheet program. Note that to use the R output (for the logrank plots), you should open the file in a spreadsheet program and just copy and paste it into R.

Index