Package 'Haplin'

July 21, 2025

Title Analyzing Case-Parent Triad and/or Case-Control Data with SNP Haplotypes

Version 7.3.2

Date 2024-08-16

Type Package

Maintainer Hakon K. Gjessing <hakon.gjessing@uib.no>

Depends R (>= 3.5.0)

Imports tools, mgcv, MASS, ff, rlang, methods

Suggests knitr, Rmpi, ggplot2, testthat, rmarkdown

Description Performs genetic association analyses of case-parent triad (trio) data with multiple markers. It can also incorporate complete or incomplete control triads, for instance independent control children. Estimation is based on haplotypes, for instance SNP haplotypes, even though phase is not known from the genetic data. 'Haplin' estimates relative risk (RR + conf.int.) and p-value associated with each haplotype. It uses maximum likelihood estimation to make optimal use of data from triads with missing genotypic data, for instance if some SNPs has not been typed for some individuals. 'Haplin' also allows estimation of effects of maternal haplotypes and parent-of-origin effects, particularly appropriate in perinatal epidemiology. 'Haplin' allows special models, like X-inactivation, to be fitted on the X-chromosome. A GxE analysis allows testing interactions between environment and all estimated genetic effects. The models were originally described in ``Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396".

License GPL (>= 2)

URL https://haplin.bitbucket.io

VignetteBuilder knitr

RoxygenNote 7.3.1

BuildVignettes yes

NeedsCompilation yes

Author Hakon K. Gjessing [aut, cre],
Miriam Gjerdevik [ctb] (functions 'lineByLine', 'cbindFiles', 'rbindFiles', 'snpPower', 'snpSampleSize', 'hapSim', 'hapRun',

'hapPower', 'hapPowerAsymp', and 'hapRelEff'),
Julia Romanowska [ctb] (ORCID: <https://orcid.org/0000-0001-6733-1953>,
new data format, parallelisation, new documentation),
Oivind Skare [ctb] (TDT tests)

Repository CRAN

Date/Publication 2024-08-20 14:30:14 UTC

Contents

`cbindFiles` *Combine a sequence of files by columns*

Description

Takes a sequence of files and combines them column-wise (side-by-side), i.e. reads each file line by line, pastes corresponding lines, then writes to outfile. Combining files line by line is especially useful when working with large datasets, where the reading of entire files may be time consuming and require a large amount of memory.

Usage

```
cbindFiles(infiles, outfile, col.sep, ask = TRUE, verbose = TRUE)
```

Arguments

`infiles`	A character vector of names (and paths) of the files to combine.
`outfile`	A character string giving the name of the resulting file. The name of the file is relative to the current working directory, unless the file name contains a definite path.
`col.sep`	Specifies the column separator which will be inserted between files.
`ask`	Logical. Default is "TRUE". If set to "FALSE", an already existing outfile will be overwritten without asking.
`verbose`	Logical. Default is "TRUE", which means that the line number is displayed for each iteration, i.e. each combined line.

Details

The function `cbind` combines R objects by columns. However, reading large datafiles may require a large amount of memory and be extremely time consuming. `cbindFiles` combines the files column-wise, one line at a time, and then writes each line to `outfile`. This avoids reading the full files into memory.
If `infiles` contains only one file, this file will be copied directly to `outfile`, i.e. without any line-by-line modifications.

Value

There is no useful output; the objective of `cbindFiles` is to produce `outfile`.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

rbindFiles, lineByLine

Examples

```
## Not run:

# Combines the three infiles side-by-side
cbindFiles(infiles = c("myfile1.txt", "myfile2.txt",
"myfile3.txt"), outfile = "myfile_combined_by_columns.txt",
col.sep = " ", ask = TRUE, verbose = TRUE)


## End(Not run)
```

---

finishParallelRun *Closing the Rmpi cluster*

Description

This function closes all the slaves spawned in initParallelRun and finishes the mpi routines. This function MUST BE called after all the haplinSlide calls and right before exiting the script/R session!

Usage

```
finishParallelRun()
```

genDataGetPart *Extracting part of genetic data.*

Description

This function enables to extract (and save for later use) part of genetic data read in with genDataRead.

Usage

```
genDataGetPart(
  data.in = stop("No data given!", call. = FALSE),
  design = stop("Design type must be given!"),
  markers,
  indiv.ids,
  rows,
  cc,
  sex,
  file.out = "my_data_part",
  dir.out = ".",
  overwrite = NULL,
  ...
)
```

Arguments

data.in	The data object (in format as the output of genDataRead).
design	The design used in the study - choose from: • *triad* - (default), data includes genotypes of mother, father and child; • *cc* - classical case-control; • *cc.triad* - hybrid design: triads with cases and controls; .
	Any of the following can be given to narrow down the dataset:
markers	Vector with numbers or names indicating which markers to choose.
indiv.ids	Character vector giving IDs of individuals. **CAUTION:** in a standard PED file, individual IDs are not unique, so this will select all individuals with given IDs.
rows	Numeric vector giving the positions - this will select only these rows.
cc	One or more values to choose based on case-control status ('cc' column).
sex	One or more values to choose based on the 'sex' column.
file.out	The base for the output filename (default: "my_data_part").
dir.out	The path to the directory where the output files will be saved.
overwrite	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

... If any additional covariate data are available in `data.in`, the user can choose based on values of these (see the Examples section).

Details

The genetic data from GWAS studies can be quite large, and thus the analysis is time-consuming. If a user knows where they want to focus the analysis, they can use this function to extract part of the entire dataset and use only this part in subsequent Haplin analysis.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

Warning

No checks are performed when choosing a subset of the data - it is the user's obligation to check whether the data subset contains correct number of individuals (especially important when using the `triad` design study) and/or markers!

Examples

```
# The argument 'overwrite' is set to TRUE!
# Read the data:
examples.dir <- system.file( "extdata", package = "Haplin" )
example.file <- file.path( examples.dir, "HAPLIN.trialdata2.txt" )
my.gen.data.read <- genDataRead( file.in = example.file, file.out = "trial_data",
 dir.out = tempdir( check = TRUE ), format = "haplin", allele.sep = "", n.vars = 2,
 cov.header = c( "smoking", "sex" ), overwrite = TRUE )
my.gen.data.read
# Extract part with only men:
men.subset <- genDataGetPart( my.gen.data.read, design = "triad", sex = 1,
  dir.out = tempdir( check = TRUE ), file.out = "gen_data_men_only", overwrite = TRUE )
men.subset
# Extract the part with only smoking women:
women.smoke.subset <- genDataGetPart( my.gen.data.read, design = "triad",
  dir.out = tempdir( check = TRUE ), sex = 0, smoking = c( 1,2 ), overwrite = TRUE )
women.smoke.subset
```

genDataLoad *Loading the data previously read in and saved by "genDataRead"*

Description

This function loads the data from the saved .ffData and .RData files, and prepares the data to subsequent analysis.

Usage

```
genDataLoad(filename = stop("'filename' must be given!"), dir.in = ".")
```

Arguments

`filename`	The base of the filenames; i.e. if the data is saved in "my_data_gen.ffData", "my_data_gen.RData" and "my_data_cov.RData", then the 'filename' should be "my_data".
`dir.in`	The path to the directory where files were saved (defaults to the current directory).

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

genDataPreprocess *Pre-processing of the genetic data*

Description

This function prepares the data to be used in Haplin analysis

Usage

```
genDataPreprocess(
  data.in = stop("You have to give the object to preprocess!"),
  map.file,
  map.header = FALSE,
  design = "triad",
  file.out = "data_preprocessed",
  dir.out = ".",
  ncpu = 1,
  overwrite = NULL
)
```

Arguments

`data.in`	Input data, as loaded by genDataRead or genDataLoad.
`map.file`	Filename (with path if the file is not in current directory) of the .map file holding the SNP names, if available.
`map.header`	Logical: does the map.file contain a header in the first row? Default: FALSE.
`design`	The design used in the study - choose from: • *triad* - (default), data includes genotypes of mother, father and child; • *cc* - classical case-control; • *cc.triad* - hybrid design: triads with cases and controls .
`file.out`	The core name of the files that will contain the preprocessed data (character string); ready to load next time with genDataLoad function; default: "data_preprocessed".
`dir.out`	The directory that will contain the saved data; defaults to current working directory.
`ncpu`	The number of CPU cores to use - this speeds up the process for large datasets significantly. Default is 1 core, maximum is 1 less than the total number of cores available on a current machine (even if the number given by the user is more than that).
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters:
  - *variables* - tabulated information of the covariate data;
  - *variables.nas* - how many NA values per each column of covariate data;
  - *alleles* - all the possible alleles in each marker;
  - *alleles.nas* - how many NA values in each marker;
  - *nrows.with.missing* - how many rows contain any missing allele information;
  - *which.rows.with.missing* - vector of indices of rows with missing data (if any)

  .

Details

The .map file should contain at least two columns, where the second one contains SNP names. Any additional columns should be separated by a whitespace character, but will be ignored. The file should contain a header.

Examples

```
# The argument 'overwrite' is set to TRUE!
# First, read the data:
examples.dir <- system.file( "extdata", package = "Haplin" )
example.file <- file.path( examples.dir, "exmpl_data.ped" )
ped.data.read <- genDataRead( example.file, file.out = "exmpl_ped_data",
 dir.out = tempdir( check = TRUE ), format = "ped", overwrite = TRUE )
ped.data.read
# Take only part of the data (if needed)
ped.data.part <- genDataGetPart( ped.data.read, design = "triad", markers = 10:12,
 dir.out = tempdir( check = TRUE ), file.out = "exmpl_ped_data_part", overwrite = TRUE )
# Preprocess as "triad" data:
ped.data.preproc <- genDataPreprocess( ped.data.part, design = "triad",
 dir.out = tempdir( check = TRUE ), file.out = "exmpl_data_preproc", overwrite = TRUE )
ped.data.preproc
```

---

genDataRead *Reading the genetic data from a file*

Description

This function will read in data from PED or haplin formatted file.

Usage

```
genDataRead(
  file.in = stop("Filename must be given!", call. = FALSE),
  file.out = NULL,
  dir.out = ".",
  format = stop("Format parameter is required!"),
  header = FALSE,
  n.vars,
  cov.file.in,
  cov.header,
  map.file,
  map.header = FALSE,
  allele.sep = ";",
  na.strings = "NA",
  col.sep = "",
  overwrite = NULL
)
```

Arguments

file.in	The name of the main input file with genotype information.
file.out	The base for the output filename (by default, constructed from the input file name).

`dir.out` The path to the directory where the output files will be saved.

`format` Format of data (will influence how data is processed) - choose from:

- *haplin* - data already in one row per family,
- *ped* - data from .ped file, each row represents an individual.

.

`header` Whether the first line of the main input file contains column names; default: FALSE; NB: this is useful only for 'haplin'-formatted files!

`n.vars` The number of columns with covariate data (if any) in the main file; NB: if the main file is in PED format, it is assumed that the first 6 columns contain the standard PED-covariates (i.e., family ID, ID of the child, father and mother, sex and case-control status), so in this case setting 'n.vars' is useful only if the PED file contains more than 6 covariate columns.

`cov.file.in` Name of the file containing additional covariate data, if any. Caution: unless the 'cov.header' argument is used, it is assumed that the first line of this file contains the header (i.e., the column names of the additional data).

`cov.header` The character vector containing the names of covariate columns (in the file with additional covariate data if given by the 'cov.file.in' argument; or in the main file, if it's a "haplin"-formatted file).

`map.file` Filename (with path if the file is not in current directory) of the .map file holding the SNP names, if available (see Details).

`map.header` Logical: does the map.file contain a header in the first row? Default: FALSE.

`allele.sep` Character: separator between two alleles (default: ";").

`na.strings` Character or NA: how the missing data is coded (default: "NA").

`col.sep` Character: separator between the columns (i.e., markers; default: any whitespace character).

`overwrite` Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Details

The function reads in all the data in the file, creates ff objects to store the genetic information and data.frame to store covariate data (if any). These objects are saved in `.RData` and `.ffData` files, which can be later on easily uploaded to R (with genDataLoad) and re-used.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

Details

The .map file should contain at least two columns, where the second one contains SNP names. Any additional columns should be separated by a whitespace character, but will be ignored. The file should contain a header.

Usage note

When reading in a covariate file together with the genotype information, it is advised to include the header in the file, so that there is no doubt to the naming of the data columns.

Examples

```
# The argument 'overwrite' is set to TRUE!
examples.dir <- system.file( "extdata", package = "Haplin" )
# ped format:
example.file2 <- file.path( examples.dir, "exmpl_data.ped" )
ped.data.read <- genDataRead( example.file2, file.out = "exmpl_ped_data",
 dir.out = tempdir( check = TRUE ), format = "ped", overwrite = TRUE )
ped.data.read
# haplin format:
example.file1 <- file.path( examples.dir, "HAPLIN.trialdata2.txt" )
haplin.data.read <- genDataRead( file.in = example.file1,
 file.out = "exmpl_haplin_data", format = "haplin", allele.sep = "", n.vars = 2,
 cov.header = c( "smoking", "sex" ), overwrite = TRUE,
 dir.out = tempdir( check = TRUE ) )
haplin.data.read
```

---

getChildren *Getter for all rows with children data*

Description

Wrapper function for genDataGetPart that returns a subset of the data containing only children.

Usage

```
getChildren(
  data.in = stop("No data given!", call. = FALSE),
  file.out = "my_data_onlyChildren",
  dir.out = ".",
  overwrite = NULL
)
```

Arguments

`data.in`	The data object (in format as the output of genDataRead); note that the design of the data is assumed to be triad.
`file.out`	The base for the output filename (default: "my_data_onlyChildren").
`dir.out`	The path to the directory where the output files will be saved.
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

`getDyads` *Getter only for all dyads (child and one parent)*

Description

Wrapper function for genDataGetPart that returns a subset of the data containing only dyads (where the child and only one parent have genetic data), i.e., not triads.

Usage

```
getDyads(
  data.in = stop("No data given!", call. = FALSE),
  file.out = "my_data_onlyDyads",
  dir.out = ".",
  overwrite = NULL
)
```

Arguments

`data.in`	The data object (in format as the output of genDataRead); note that the design of the data is assumed to be "triad".
`file.out`	The base for the output filename (default: "my_data_onlyDyads").
`dir.out`	The path to the directory where the output files will be saved (default: ".", the current directory).
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

`getFathers` *Getter for all rows with fathers' data*

Description

Wrapper function for genDataGetPart that returns a subset of the data containing only fathers.

Usage

```
getFathers(
  data.in = stop("No data given!", call. = FALSE),
  file.out = "my_data_onlyFathers",
  dir.out = ".",
  overwrite = NULL
)
```

Arguments

`data.in`	The data object (in format as the output of genDataRead); note that the design of the data is assumed to be triad.
`file.out`	The base for the output filename (default: "my_data_onlyFathers").
`dir.out`	The path to the directory where the output files will be saved.
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

`getFullTriads` *Getter for all full triads*

Description

Wrapper function for genDataGetPart that returns a subset of the data containing only full triads (where all, the child, the mother and the father have genetic data).

Usage

```
getFullTriads(
  data.in = stop("No data given!", call. = FALSE),
  file.out = "my_data_onlyTriads",
  dir.out = ".",
  overwrite = NULL
)
```

Arguments

`data.in`	The data object (in format as the output of genDataRead); note that the design of the data is assumed to be triad.
`file.out`	The base for the output filename (default: "my_data_onlyTriads").
`dir.out`	The path to the directory where the output files will be saved.
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

`getMothers` *Getter for all rows with mothers' data*

Description

Wrapper function for genDataGetPart that returns a subset of the data containing only mothers.

Usage

```
getMothers(
  data.in = stop("No data given!", call. = FALSE),
  file.out = "my_data_onlyMothers",
  dir.out = ".",
  overwrite = NULL
)
```

Arguments

`data.in`	The data object (in format as the output of genDataRead); note that the design of the data is assumed to be triad.
`file.out`	The base for the output filename (default: "my_data_onlyMothers").
`dir.out`	The path to the directory where the output files will be saved.
`overwrite`	Whether to overwrite the output files: if NULL (default), will prompt the user to give answer; set to TRUE, will automatically overwrite any existing files; and set to FALSE, will stop if the output files exist.

Value

A list object with three elements:

- *cov.data* - a `data.frame` with covariate data (if available in the input file)
- *gen.data* - a list with chunks of the genetic data; the data is divided column-wise, using 10,000 columns per chunk; each element of this list is a ff matrix
- *aux* - a list with meta-data and important parameters.

This now contains only the selected subset of data.

gxe *Test for gene-environment interaction*

Description

Performes a gene-environment test to check if haplin estimates of relative risks change over strata of environment. It is typically applied to the output from haplinStrat

Usage

```
gxe(object.list)
```

Arguments

object.list	A list of haplin results, almost always the output from haplinStrat. The first element is the result of running haplin on all data; the remaining elements are the results for each stratum separately.

Details

haplinStrat runs haplin first on the entire input data file, then on each stratum separately. The results from haplinStrat are similar to just manually splitting the file into strata and running haplin on each, with one important difference, however: Since some strata may be small etc., haplin might conceivably choose different haplotypes in different strata, and also choose different reference haplotypes. When first running haplin on the entire file, haplinStrat saves the selected haplotypes and chosen reference category. Then, in the strata-specific runs haplinStrat forces haplin to choose the same haplotypes/reference category in all runs, so that results from different strata are comparable. When applying gxe to the output from haplinStrat, it will test whether there is a statistically significant change in parameter estimates from stratum to stratum, i.e. a gene-environment interaction since strata usually are defined by an environmental exposure. gxe uses Wald tests to test for interactions. It always tests whether there is change in haplotype frequencies from stratum to stratum. More importantly, it separately tests whether any genetic effects, such as fetal genetic effects, maternal effects, or parent-of-origin effects, change significantly over strata. gxe can also be run from within haplinSlide by using the strata argument in haplinSlide.

Value

A dataframe with one row for each test that is performed (haplo.freq is the first, the remaining depend on the model that has been estimated). The Wald chi-squared test value, degrees-of-freedom, and resulting p-value are reported.

NOTE:

In the future, the structure of the output from gxe will change. In particular, measures of ratios of relative risks will be reported in addition to the p-values

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin, haplinStrat, haplinSlide

Examples

```
## Not run:
# All standard haplin runs can be done with haplinStrat.
# Below is an illustration. See the haplin help page for more
# examples.
#
# Analyzing the effect of fetal genes, including triads with missing data,
# using a multiplicative response model. The first column of the data file
# in this example contains the stratification variable.
result <- haplinStrat("C:/work/data.dat", strata = 1, use.missing = T, response = "mult",
reference = "ref.cat", winlength = 1)
# Provide summary of separate results:
lapply(result, summary)
# Plot results separately:
par(ask = T)
lapply(result, plot)
#
# Convert results to table format and stack them over strata:
haptable(result)
# Test for interaction between haplotype risk estimates and the strata variable:
gxe(result)


## End(Not run)
```

haplin *Fitting log-linear models to case-parent triad and/or case-control data*

Description

haplin fits a log-linear model to case-parent triads, case-control data, or combined (hybrid) case-parent control-parent triads or dyads. It estimates marker or haplotype frequencies, and uses the EM algorithm to reconstruct haplotypes and, if requested, impute missing genotypes. haplin prints and plots estimates of relative risks associated with fetal and maternal haploypes, and in addition allows splitting fetal haplotype effects into maternally and paternally inherited effects. It allows special models, like x-inactivation, to be fitted on the X-chromosome. The result is an object of class haplin, which can be explored with summary, plot, and haptable.

Usage

```
haplin( data, markers = "ALL",
design = "triad", use.missing = FALSE,
xchrom = FALSE, maternal = FALSE, test.maternal = FALSE,
poo = FALSE, scoretest = "no", ccvar = NULL, strata = NULL,
sex = NULL, comb.sex = "double",
reference = "reciprocal", response = "free",
threshold = 0.01, max.haplos = NULL, haplo.file = NULL,
resampling = "no", max.EM.iter = 50, data.out = "no",
verbose = TRUE, printout = TRUE )
```

Arguments

data	An R-object which is the result of using genDataPreprocess. See the web page for a detailed description of how to use this function.
markers	Default is "ALL", which means haplin uses all available markers in the data set in the analysis. For the current version of haplin the number of markers used at a single run should probably not exceed 4 or 5 due to the computational burden. The markers argument can be used to select appropriate markers from the file without creating a new file for the selected markers. The relevant markers can be specified by giving a vector or numbers (e.g., markers = c(1, 3:10) will use the 10 first markers except marker 2) or characters (e.g., markers = c("m1", "m3", "rs35971")). When running haplin, it may be a good idea to start exploring a few markers at a time, using this argument.
design	The value "triad" is used for the standard case triad design, without independent controls. The value "cc.triad" means a combination of case triads and control triads. This requires the argument ccvar to point to the data column containing the case-control variable. The value "cc" means a simple case-control design, where the parents have not been genotyped (there are no data columns for parental genes). NOTE: design is also set in genDataPreprocess. Almost always, the two arguments should be equal. Occasionally, however, the user might want to override the original argument by switching from 'cc.triad' to 'triad' or vice versa.

`use.missing` A logical value used to determine whether triads with missing data should be included in the analysis. When set to TRUE, `haplin` uses the EM algorithm to obtain risk estimates, also taking into account triads with missing data. The standard errors and p-values are adjusted to correct for this. The default, however, is FALSE. When FALSE, all triads having any sort of missing data are excluded before the analysis is run. Note that `haplin` only looks at markers actually used in the analysis, so that if the markers argument (see below) is used to select a collection of markers for analysis, `haplin` only excludes triads with missing data on the included markers.

`xchrom` Logical, defaults to "FALSE". If set to "TRUE", `haplin` assumes the markers are on the x-chromosome. This option should be combined with specifying the `sex` argument. In addition, `comb.sex` can be useful. `xchrom = T` can be combined with `poo = T` and/or `maternal = T`.

`maternal` If TRUE, maternal effects are estimated as well as the standard fetal effects.

`test.maternal` Not yet implemented.

`poo` Parent-Of-Origin effects. If TRUE, `haplin` will split single-dose effects into two separate effect estimates, one for the maternally inherited haplotype, and one for the paternally inherited haplotype. Double dose will be estimated as before.

`scoretest` Special interest only. If "no", no score test is computed. If "yes", an overall score p-value is included in the output, and the individual score values are returned in the `haplin` object. If "only", `haplin` is only run under the null hypothesis, and a simple score object is returned instead of the full `haplin` object. Useful if only score testing is needed.

`ccvar` Numeric. Should give the column number for the column containing the case-control indicator in the data file. Needed for the "cc" and "cc.triad" designs. The column should contain two numeric values, of which the largest one is always used to denote cases.

`strata` Not yet implemented.

`sex` To be used with `xchrom = TRUE`. A numeric value specifying which of the data columns that contains the sex variable. The variable should be coded 1 for males and 2 for females.

`comb.sex` To be used with `xchrom = TRUE`. A character value that specifies how to handle gender differences on the X-chromosome. If set to "males" or "females", analyses are done either for just males or just females, respectively. If set to "single" or "double", males and females are used in a combined analysis. Specifically, when "single", the effect of a (single) allele in males is assumed to equal the effect of a single allele dose in females, and similarly, when "double", a single allele in males is assumed to have the same effect as a double allele dose in females. Default is "double", which corresponds to X-inactivation. See separate description for more details.

`reference` Decides how `haplin` chooses its reference category for the effect estimates. Default value is "reciprocal". With the reciprocal reference the effect of a single or double dose of each haplotype is measured relative to the remaining haplotypes. This means that a new reference category is used for each single haplotype. Other possible values are "population" (which is similar to reciprocal, but where

the reference category is always the total population), and "ref.cat", where a single haplotype is used as reference for all the rest. For ref.cat, the default is to choose the most frequent haplotype as the reference haplotype. The reference haplotype can be set explicitly by giving a numeric value for the reference argument. Note that the numeric value refers to the haplotype's position among the haplotypes selected for analysis by `haplin`. This means that one should run `haplin` once first to see what haplotypes are used before giving a numeric value to reference.

`response` The default value "free" means that both single- and double dose effects are estimated. Choosing "mult" instead specifies a multiplicative dose-response model.

`threshold` Sets the (approximate) lower limit for the haplotype frequencies of those haplotypes that should be retained in the analysis. Hapotypes that are less frequent are removed, and information about this is given in the output. Default is 0.01.

`max.haplos` Not yet implemented.

`haplo.file` Not yet implemented.

`resampling` Mostly for testing. Default is "no". When "no", the individual haplotypes reconstructed by the EM algorithm as assumed known when computing CIs and p-values. If set to "jackknife" a jackknife-based resampling procedure is used when computing confidence intervals and p-values for effect estimates. This takes more time, but corrects the CIs and p-values for the uncertainty contained in unphased data. Note: in all recent versions of `haplin`, the resampling is no longer needed since the confidence intervals and p-values are already corrected in the standard computation.

`max.EM.iter` The maximum number of iterations used by the EM algorithm. This value can be increased if necessary, which sometimes is the case with e.g. case-control data which a substantial amount of missing. However, for triad data with little missing information there is usually no need for many iterations.

`data.out` Character. Accepts values "no", "prelim", "null" or "full", with "no" as default. For values other than default, `haplin` returns the data file prepared for analysis rather than the usual `haplin` estimation results. The data file contains the haplotypes identified for each triad, and a vector of weights giving the probability distribution of different haplotype configurations within a triad. The probabilities are computed from preliminary haplotype frequency estimates, from the null model or from the full likelihood model. The "prelim" option will be much faster but somewhat less precise than the likelihood models.

`verbose` Default is T (=TRUE). During the EM algorithm, `haplin` prints the estimated parameters and deviance for each step. To avoid the output, set this argument to F (=FALSE).

`printout` Logical. If TRUE (default), `haplin` prints a full summary of the results after finishing the estimation. If FALSE, no such printout is given, but the `summary` function can later be applied to a saved result to get the same summary.

Details

Input data can be either a `haplin` format data file, or a PED data. These have to be loaded into R first, using genDataRead or genDataLoad functions, and then pre-processed with the genDataPreprocess function. If the PED data file is used, the arguments `filename`, `n.vars`, `sep`, `allele.sep`,

na.strings, ccvar, and sex need not be specified.
The output can be examined by print, summary, plot and haptable.

Value

An object of class haplin is returned. (The only exception is when data.out is set different from "no", where haplin will produce a data file with haplotypes identified.)

Warning

Typically, some of the included haplotypes will be relatively rare, such as a frequency of 1% - 5%. For those haplotypes there may be too little data to estimate the double doses properly, so the estimates may be unreliable. This is seen from the extremely wide confidence intervals. The rare double dose estimates should be disregarded, but the remaining single and double dose estimates are valid. To avoid the problem one can also reduce the model to a purely multiplicative model by setting response = "mult" combined with reference = "ref.cat".

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

summary.haplin, plot.haplin, pedToHaplin, haptable, haplinSlide, genDataLoad, genDataRead, genDataPreprocess

Examples

```
# setting up the directory with exemplary data
dir.in <- system.file( "extdata", package = "Haplin" )
file.in <- file.path( dir.in, "data.dat" )

# reading data in
data.in <- genDataRead( file.in, file.out = "poo_exmpl_data_read", format = "haplin",
  dir.out = tempdir( check = TRUE ), n.vars = 1, allele.sep = " ", col.sep = " ",
  overwrite = TRUE )
```

```
# preprocessing the data
data.preproc <- genDataPreprocess( data.in, design = "triad",
  file.out = "poo_exmpl_data_preproc", dir.out = tempdir( check = TRUE ), overwrite = TRUE )

# running haplin, calculating POO
res.POO <- haplin( data.preproc, markers = 2, poo = TRUE, response = "mult",
   reference = 2, use.missing = TRUE )
res.POO

## Not run:
# 1. Read the data:
my.haplin.data <- genDataRead( file.in = "HAPLIN.trialdata.txt", file.out =
  "trial_data1", dir.out = ".", format = "haplin", n.vars = 0 )

# 2. Run pre-processing:
haplin.data.prep <- genDataPreprocess( data.in = my.haplin.data, format =
  "haplin", design = "triad", file.out = "trial_data1_prep", dir.out = "." )

# 3. Analyze:
# Standard run:
haplin( haplin.data.prep )

# Specify path, estimate maternal effects:
haplin( haplin.data.prep, maternal = T )

# Specify path, use haplotype no. 2 as reference:
haplin( haplin.data.prep, reference = 2 )

# Remove more haplotypes from estimation by increasing the threshold
# to 5%:
haplin( haplin.data.prep, threshold = 0.05 )

# Estimate maternal effects, using the most frequent haplotype as reference.
# Use all data, including triads with missing data. Select
# markers 3, 4 and 8 from the supplied data.
haplin( haplin.data.prep, use.missing = T, maternal = T,
reference = "ref.cat", markers = c(3,4,8) )
# Note: in this version of haplin, the jackknife is
# no longer necessary since the standard errors are already corrected.

# Some examples showing how to save the haplin result and later
# recall plot and summary results:

# Same analysis as above, saving the result in the object "result.1":
result.1 <- haplin( haplin.data.prep, use.missing = T, maternal = T,
reference = "ref.cat", markers = c(3,4,8) )

# Replot the saved result (fetal effects):
plot( result.1 )

# Replot the saved result (maternal effects):
plot( result.1, plot.maternal = T )
```

```
# Print a very short summary of saved result:
result.1

# A full summary of saved result, with confidence intervals and
# p-values (the same as haplin prints when running):
summary( result.1 )

# Some examples when the data file contains two covariates,
# the second is the case-control variable:

# The following standard triad run is INCORRECT since it disregards
# case status:
haplin("data.dat", use.missing = T, n.vars = 2, design = "triad")

# Combined run on "hybrid" design, correctly using both case-parent
# triads and control-parent triads:
haplin( my.haplin.data, use.missing = T, n.vars = 2, ccvar = 2,
design = "cc.triad" )

# If parent columns are not in the file, a plain case-control
# run can be used:
haplin( my.haplin.data, use.missing = T, n.vars = 2, ccvar = 2,
design = "cc", response = "mult", reference = "ref.cat" )

# An example of how to produce a data file with all possible haplotypes
# identified for each triad, together with their probaility weights:
result.data <- haplin( my.haplin.data, use.missing = T,
markers = c(3,4,8), data.out = "prelim" )
# result.data will then contain the data file, with a vector of
# probabilities (freq) computed from the preliminary haplotype
# frequencies.

## End(Not run)
```

`haplinSlide` *Run haplin analysis in a series of sliding windows over a sequence of markers/SNPs*

Description

Produces a list, each element of which is an object of class `haplin`, which is the result of fitting the log-linear `haplin` models to the data one "window" at a time.

Usage

```
haplinSlide( data, markers = "ALL", winlength = 1,
strata = NULL, table.output = TRUE, cpus = 1, para.env = NULL, slaveOutfile = "",
printout = FALSE, verbose = FALSE, ...)
```

Arguments

data	R-object of class "haplin.ready", which is e.g., output from genDataPreprocess or genDataLoad, and contains covariate and genetic data.
markers	Default is "ALL", which means haplinSlide uses all available markers in the data set in the analysis. Alternatively, the relevant markers can be specified by giving a vector or numbers (e.g., markers = c(1, 3:10) will use the 10 first markers except marker 2) or characters (e.g., markers = c("m1", "m3", "rs35971")). haplinSlide will then run haplin on a series of windows selected from the supplied markers. The winlength argument decides the length of the windows. See details.
winlength	Length of the sliding, overlapping windows to be run along the markers. See details.
strata	A single numeric value specifying which data column contains the stratification variable.
table.output	If TRUE, the haptable function will be applied to each result after estimation, greatly reducing the size of the output. If FALSE, each element of the output list is a standard haplin object. To preserve memory, default is set to TRUE.
cpus	haplinSlide allows parallel processing of its analyses. The cpus argument should preferably be set to the number of available cpu's. If set lower, it will save some capacity for other processes to run. Setting it too high should not cause any serious problems.
para.env	The user can choose parallel environment to use — "parallel" (default) or "Rmpi" (for use on clusters); this option is used only when cpus argument is larger than 1.
slaveOutfile	Character. To be used when cpus > 1. If slaveOutfile = "" (default), output from all running cores will be printed in the standard R session window. Alternatively, the output can be saved to a file by specifying the file path and name.
printout	Default is FALSE. If TRUE, provides a full summary of each haplin result during the run of haplinSlide.
verbose	Same as for haplin, but defaults to FALSE to reduce output size.
...	Remaining arguments to be used by haplin in each run.

Details

haplinSlide runs haplin on a series of overlapping windows of the chosen markers. Except for the markers and winlength arguments, all arguments are used exactly as in haplin itself. For instance, if markers = c(1, 3, 4, 5, 7, 8) and winlength = 4, haplinSlide will run haplin on first the markers c(1, 3, 4, 5), then on c(3, 4, 5, 7), and finally on c(4, 5, 7, 8). The results are returned in a list. The elements are named "1-3-4-5" etc., and can be extracted with, say, summary(res[["1-3-4-5"]]) etc., where res is the saved result. Or the output can be examined by, for instance, using lapply(res, summary) and lapply(res, plot).
When running haplinSlide on a large number of markers, the output can become prohibitively large. In that case table.output should be set to TRUE, and haplinSlide will return a list of summary "haptables". This list can then be stacked into a single dataframe using toDataFrame. To avoid exessive memory use, the default is table.output = TRUE.

When multiple cores are available, set the cpus to the number of cores that should be used. This will run haplinSlide in parallel on the chosen number of cores. Note that feedback is provided by each of the cores separately, and some cores may start working on markers far out in the sequence.

Value

A list of objects of class haplin is returned.

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin, summary.haplin, plot.haplin, haptable, toDataFrame

Examples

```
## Not run:
# (Almost) all standard haplin runs can be done with haplinSlide.
# Below is an illustration. See the haplin help page for more
# examples.
#

# 1. Read the data:
my.haplin.data <- genDataRead( file.in = "HAPLIN.trialdata.txt", file.out =
  "trial_data1", dir.out = tempdir( check = TRUE ), format = "haplin", n.vars = 0 )

# 2. Run pre-processing:
haplin.data.prep <- genDataPreprocess( data.in = my.haplin.data,
  format = "haplin", design = "triad", file.out = "trial_data1_prep",
  dir.out = tempdir( check = TRUE ) )

# 3. Analyze:
# Analyzing the effect of fetal genes, including triads with missing data,
# using a multiplicative response model. When winlength = 1, separate
```

```
# markers are used. To make longer windows, winlength can be increased
# correspondingly:
result.1 <- haplinSlide( haplin.data.prep, use.missing = T, response = "mult",
reference = "ref.cat", winlength = 1, table.output = F)
# Provide summary of separate results:
lapply(result.1, summary)
# Plot results:
par(ask = T)
lapply(result.1, plot)


## End(Not run)
```

`haplinStrat` *Fit* `haplin` *to each subset/stratum of data, determined by the argument* `strata`

Description

Produces a list, each element of which is an object of class `haplin`, which is the result of fitting the log-linear `haplin` models to each `strata` stratum independently.

Usage

```
haplinStrat( data, strata = NULL, ...)
```

Arguments

`data`	R-object of class "haplin.ready", which is e.g., output from `genDataPreprocess` or `genDataLoad`, and contains covariate and genetic data.
`strata`	A single integer specifying the number of the column in the covariate data that contains the stratification variable
`...`	Remaining arguments to be used by `haplin` in each run.

Details

`haplinStrat` runs `haplin` first on the entire input data file, then on each stratum separately. Strata are defined by the `strata` variable, which can be coded as numerical or character. However, one should use only a moderate number of levels/strata, since `haplin` will be run independetly on each, and some strata may otherwise have an insufficient amount of data. Running `haplinStrat` is thus just a simplification of manually splitting the file into strata and running `haplin` on each; the end result would be the same. The main reason for running haplinStrat is to test for gene-environment interactions. This is achieved by running `postTest` on the result from `haplinStrat`. `haplinStrat` can also be run from within `haplinSlide` by using the `strata` argument in `haplinSlide`.

Value

A list of objects of class haplin is returned. The first element contains the result of running `haplin` on the entire data file; the remaining elements are the results from each of the strata. The names of the list correspond to the values of the `strata` variable.

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin, summary.haplin, plot.haplin, haptable, toDataFrame, haplinSlide, gxe

Examples

```
# setting up the directory with exemplary data
dir.in <- system.file( "extdata", package = "Haplin" )
file.in <- paste0( dir.in, "/data.dat" )

# reading data in
data.in <- genDataRead( file.in, file.out = "poo_exmpl_data_read", format = "haplin",
  dir.out = tempdir( check = TRUE ), n.vars = 1, allele.sep = " ", col.sep = " ",
  overwrite = TRUE )
# preprocessing the data
data.preproc <- genDataPreprocess( data.in, design = "triad",
 file.out = "poo_exmpl_data_preproc", dir.out = tempdir( check = TRUE ), overwrite = TRUE )

# running haplinStrat, checking for gene-environment interactions
res.GxE <- haplinStrat( data.preproc, markers = 2, strata = 1, poo = FALSE,
  response = "mult", reference = 2, use.missing = TRUE )
res.GxE

# running haplinStrat, checking for POO-environment interactions
res.POOxE <- haplinStrat( data.preproc, markers = c(1,2,3), strata = 1, poo = TRUE,
  response = "mult", reference = "ref.cat", use.missing = TRUE )
res.POOxE
```

```
## Not run:
# All standard haplin runs can be done with haplinStrat.
# Below is an illustration. See the haplin help page for more
# examples.
#
# Analyzing the effect of fetal genes, including triads with missing data,
# using a multiplicative response model. The first column of the data file
# in this example contains the stratification variable.
result <- haplinStrat("C:/work/data.dat", strata = 1, use.missing = T, response = "mult",
reference = "ref.cat", winlength = 1)
# Provide summary of separate results:
lapply(result, summary)
# Plot results separately:
par(ask = T)
lapply(result, plot)
#
# Convert results to table format and stack them over strata:
haptable(result)
# Test for interaction between haplotype risk estimates and the strata variable:
postTest(result)


## End(Not run)
```

---

hapPower *Power simulation for association analyses with Haplin*

Description

Simulates the statistical power of genetic analyses assessing fetal effects, maternal effects and/or parent-of-origin effects. Effects of X-chromosome genes and gene-environment interaction effects are also allowed.

Usage

```
hapPower(hapRun.result, alpha = 0.05)
```

Arguments

`hapRun.result`	The result of running `hapRun`
`alpha`	`alpha` is the Type I Error probablity. Equals 0.05 by default.

Details

The Haplin framework includes different modules for assessing genetic effects: `haplin`, `haplinStrat` and `haplinSlide`. `hapPower` simulates the power of these analyses, which enables power calculations of fetal effects, maternal effects and/or parent-of-origin effects. Various family designs, i.e.,

triads, case-control, the hybrid design, and all intermediate designs, are possible. It also allows power calculation of gene-environment interaction effects and effects on X-chromosome markers.

hapPower calculates statistical power using the result of hapRun, and the target effects must be specified in this function, see Examples below, and details in https://haplin.bitbucket.io/docu/Haplin_power.pdf.

Value

hapPower returns the simulated power.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, haplinSlide, hapSim, hapRun, snpPower, snpSampleSize , hapPowerAsymp

Examples

```
## Not run:
## Simulate power from 100 files using haplin.
## The files consist of fetal effects at two diallelic markers,
## corresponding to haplo.freq = rep(0.25, 4), RR = c(2,1,1,1) and RRstar = c(1,1,1,1).
## The power is simulated for the combination of 100 case triads
## and 100 control triads with no missing data at a 0.05 significance level,
## applying a multiplicative model.
hapRun.res <- hapRun(nall = c(2,2), n.strata = 1, cases = c(mfc=100), controls = c(mfc=100),
haplo.freq = rep(0.25,4), RR = c(2,1,1,1), RRstar = c(1,1,1,1),
hapfunc = "haplin", response = "mult", n.sim = 100, dire = "simfiles", ask = FALSE)
hapPower(hapRun.res)

## Simulate power from 100 files applying haplinStrat.
## The files consist of fetal and maternal effects at two diallelic markers.
## The data is simulated for 500 case triads and 200 control families in the first stratum,
## and 500 case triads and 500 control trids in the second.
## The fetal effects vary across strata,
## whereas the maternal effects are the same.
## One percent of the case triads are missing at random in the second stratum.
hapRun.res <- hapRun(nall = c(2,2), n.strata = 2, cases = c(mfc=500),
controls = list(c(mfc=200),c(mfc=500)), haplo.freq = rep(0.25,4), maternal = TRUE,
```

```
RR = list(c(1.5,1,1,1),c(1,1,1,1)), RRstar = c(1,1,1,1),
RR.mat = c(1.5,1,1,1), RRstar.mat = c(1,1,1,1), gen.missing.cases = list(NULL,0.01),
use.missing = TRUE, hapfunc = "haplinStrat", n.sim = 100, ask = FALSE)
hapPower(hapRun.res)

## Simulate power at the 0.1 significance level from 1000 files using haplin.
## The files consist of fetal effects at one diallelic locus,
## corresponding to haplo.freq = c(0.1,0.9), RR = c(2,1) and RRstar = c(1,1).
## The data consists of a combination of 100 case triads and 100 control triads.
hapRun.res <- hapRun(nall = c(2), cases = c(mfc=100), controls = c(mfc=100),
haplo.freq = c(0.1,0.9), RR = c(2,1), RRstar = c(1,1),
hapfunc = "haplin", response = "mult", n.sim = 1000, ask = FALSE)
hapPower(hapRun.res, alpha= 0.10)

## The latter example, applying response = "mult", should be comparable to
## the theoretic calculations of snpPower.
snpPower(cases = list(mfc=100), controls = list(mfc=100),
RR = 2, MAF = 0.1, alpha = 0.10)

## End(Not run)
```

`hapPowerAsymp` *Asymptotic power calculations for genetic association analyses with Haplin*

Description

Computes the asymptotic power for genetic analyses assessing fetal effects, maternal effects and/or parent-of-origin effects. Effects of X-chromosome genes and gene-environment interaction effects are also allowed.

Usage

```
hapPowerAsymp(nall = 2, n.strata = 1, cases, controls, haplo.freq,
RR, RRcm, RRcf, RRstar, RR.mat, RRstar.mat,
xchrom = F, sim.comb.sex = "double", BR.girls,
response = "mult", alpha = 0.05, ...)
```

Arguments

`nall`	A vector of the number of alleles at each locus. By default a diallellic SNP.
`n.strata`	The number of strata.
`cases`	A list of the number of case families. Each element is a vector of the number of families of the specified family design in the corresponding stratum. The possible family designs, i.e., the possible names of the elements, are `"mfc"` (full triad), `"mc"` (mother-child dyad), `"fc"` (father-child dyad) or `"c"` (a single case child).

controls	A list of the number of control families. Each element is a vector of the number of families of the specified family design in the corresponding stratum. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother) or "f" (a single control father).
haplo.freq	A list of which each element is a numeric vector of the haplotype frequencies in each stratum. The frequencies are normalized and sum to one. The Details section shows how to implement this argument in agreement with the possible haplotypes.
RR	A list of which each element is a numeric vector of the relative risks in each stratum. The Details section shows how to implement this argument in agreement with the possible haplotypes.
RRcm	A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the mother for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.
RRcf	A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the father for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.
RRstar	A list of numeric vectors. Estimates how much double-dose children would deviate from the risk expected in a multiplicative dose-response relationship.
RR.mat	The interpretation is similar to RR but for maternal genetic effects.
RRstar.mat	The interpretation is similar to RRstar but for maternal genetic effects.
xchrom	Logical. Equals FALSE by default, which indicates analyses of autosomal markers. If TRUE, hapPowerAsymp performs power analyses of X-linked markers.
sim.comb.sex	To be used with xchrom = TRUE. A character value that specifies how to handle gender differences on the X-chromosome. If "single", the effect of a (single) allele in males is equal to the effect of a single allele dose in females, and similarly, if "double", a single allele in males has the same effect as a double allele dose in females. Default is "double", which corresponds to X-inactivation.
BR.girls	To be used with xchrom = TRUE. Gives the ratio of baseline risk for females relative to the baseline risk for males.
response	The default value "mult" specifies a multiplicative dose-response model. response = "free" is not yet implemented.
alpha	alpha is the Type I Error probablity. Equals 0.05 by default.
...	Could include argument reference. By default, the most frequent allele or haplotype is chosen as reference. The reference haplotype can be set explicitly by giving a numeric value for the reference argument.

Details

The Haplin framework includes different modules for assessing genetic effects: haplin, haplinStrat and haplinSlide. hapPowerAsymp computes the asymptotic power for these analyses, which enables power calculations of fetal effects, maternal effects and/or parent-of-origin effects. Various

family designs, i.e., triads, case-control, the hybrid design, and all intermediate designs, are possible. It also allows power calculation of gene-environment interaction effects and effects on X-chromosome markers.

hapPower computes power through "brute force" simulations using hapRun. This is a robust way of checking software implementations, asymptotic approximations and attained significance level. However, both power and the corresponding sample size calculations can be performed much more efficiently using asymptotic approximations. The asymptotic power is calculated applying the non-centrality parameter of the Wald tests, which use the asymptotic normal distribution of the log-scale parameter. The function hapCovar (used by hapPowerAsymp) computes the variance-covariance matrix by applying the log-linear model combined with the EM algorithm.

Specifying haplotype risks:
The number of haplotypes used in the simulations is determined by the nall argument, since prod(nall) different haplotypes can be made from the specified number of markers, length(nall). The arguments haplo.freq, RR, RRcm, RRcf, RRstar, RR.mat, and RRstar.mat are all lists where each element represents a stratum. Within each stratum, the arguments are vectors of length equal to the number of haplotypes, specifying the relative risk etc. associated with each haplotype. The stratum specific arguments may be simplified if the number of strata is one, or if the arguments are equal across all strata. The haplotypes are determined by creating all possible haplotypes from the given markers, in a sequence where the first marker varies mostly quickly. For instance, if nall = c(3,2), the first marker has 3 alleles, the second has 2, and 6 haplotypes are possible. Taken in order, the haplotypes are 1-1, 2-1, 3-1, 1-2, 2-2, and 3-2. When specifying, say, RR = c(1,2,1,1,1,1) the haplotype 2-1 has a double risk compared to the rest. With, for instance, two strata, the specification RR = list(c(1,2,1,1,1,1), c(1,1,1,1,1,1)) would mean that the risk associated with 2-1 is elevated only in the first stratum, not the second. The simplest example would be with nall = c(2) and RR = c(1,2), which would simulate a single SNP where the second allele has a double risk.

Specifying genetic effects:
Standard fetal effects are specified by the arguments RR and RRstar, whereas parent-of-origin effects are addressed by the arguments RRcm, RRcf and RRstar. Maternal effects are included by the additional arguments RRmat and RRstar.mat.

Value

hapPowerAsymp returns the asymptotic power for the relevant genetic effects. The first element of the list depicts the power for each haplotype analyzed separately. If there are more than two possible haplotypes, the second element displays the overall power for all haplotypes combined.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, haplinSlide, haplinStrat, hapSim, hapRun, snpPower, hapPower

Examples

```
## Calculate the asymptotic power for a triad design
## when the minor allele increases the fetal risk by twofold.
## Assumes a multiplicative dose-response relationship.
hapPowerAsymp(nall = c(2), n.strata = 1, cases = list(c(mfc=120)),
haplo.freq = c(0.1,0.9), RR = c(2,1), RRstar = c(1,1))

## Calculate the asymptotic power for the hybrid design when
## the minor allele increases the fetal risk by twofold
## in the first stratum and no effect is seen in the second
## i.e., gene-environment interaction (GxE) effects.
hapPowerAsymp(nall = c(2), n.strata = 2, cases = list(c(mfc=100)),
controls = list(c(mfc=100)), haplo.freq = c(0.1,0.9),
RR = list(c(2,1), c(1,1)), RRstar = c(1,1))

## Calculate the asymptotic GxE power assessing maternal- and
## parent-of-origin effects at two diallelic loci.
hapPowerAsymp(nall = c(2,2), n.strata = 2, cases = c(mfc=500),
haplo.freq = c(0.1,0.2,0.3,0.4),
RRcm = list(c(3,1,1,1), c(1,1,1,1)), RRcf = c(1,1,1,1), RRstar = c(1,1,1,1),
RR.mat = list(c(1.5,1,1,1),c(1,1,1,1)), RRstar.mat = c(1,1,1,1))
```

---

hapRelEff *Relative efficiency comparing different study designs in genetic association analysis with Haplin*

Description

Computes the relative efficiency for different study designs in genetic association analysis.

Usage

```
hapRelEff(nall = 2, cases.comp, controls.comp,
cases.ref, controls.ref, haplo.freq,
RR, RRcm, RRcf, RRstar, RR.mat, RRstar.mat,
xchrom = F, sim.comb.sex = "double", BR.girls,
response = "mult", ...)
```

Arguments

nall	A vector of the number of alleles at each locus. By default a diallellic SNP.
cases.comp	A list of the number of case families in the comparison design. Its element is a vector of the number of families of the specified family design. The possible family designs, i.e., the possible names of the elements, are "mfc" (full triad), "mc" (mother-child dyad), "fc" (father-child dyad) or "c" (a single case child).
controls.comp	A list of the number of control families in the comparison design. Its element is a vector of the number of families of the specified family design. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother) or "f" (a single control father).
cases.ref	A list of the number of case families in the reference design. The options are the same as for cases.comp.
controls.ref	A list of the number of control families in the reference design. The options are the same as for controls.comp.
haplo.freq	A list of which its element is a numeric vector of the haplotype frequencies. The frequencies are normalized and sum to one. The Details section shows how to implement this argument in agreement with the possible haplotypes.
RR	A list of which its element is a numeric vector of the relative risks. The Details section shows how to implement this argument in agreement with the possible haplotypes.
RRcm	A numeric vector in list format containing the relative risks associated with the haplotypes transmitted from the mother. See Details for description of how to implement this argument in agreement with the possible haplotypes.
RRcf	A numeric vector in list format containing the relative risks associated with the haplotypes transmitted from the father. See Details for description of how to implement this argument in agreement with the possible haplotypes.
RRstar	A numeric vector in list format. Estimates how much double-dose children would deviate from the risk expected in a multiplicative dose-response relationship.
RR.mat	The interpretation is similar to RR but for maternal genetic effects.
RRstar.mat	The interpretation is similar to RRstar but for maternal genetic effects.
xchrom	Logical. Equals FALSE by default, which indicates analyses of autosomal markers. If TRUE, analyses are perfomed on X-linked markers.
sim.comb.sex	To be used with xchrom = TRUE. A character value that specifies how to handle gender differences on the X-chromosome. If "single", the effect of a (single) allele in males is equal to the effect of a single allele dose in females, and similarly, if "double", a single allele in males has the same effect as a double allele dose in females. Default is "double", which corresponds to X-inactivation.
BR.girls	To be used with xchrom = TRUE. Gives the ratio of baseline risk for females relative to the baseline risk for males.
response	The default value "mult" specifies a multiplicative dose-response model. response = "free" is not yet implemented.

... Could include argument reference. By default, the most frequent allele or haplotype is chosen as reference. The reference haplotype can be set explicitly by giving a numeric value for the reference argument.

Details

hapRelEff compares two study designs for genetic association analysis, using a term called relative efficiency. The relative efficiency is defined as the ratio of variances of estimators for the same parameter, computed from two different designs, or equivalently, the ratio of the sample sizes needed for each of the two designs to achieve the same level and power. The number of genotyped individuals within each design is accounted for. The relative efficiency estimated under the null hypothesis, i.e., when all relative risks are equal to one, is known as the Pitman efficiency. To compute the variance-covariance estimates for each design, hapRelEff calls the function hapCovar, which calculates the asymptotic variance-covariance matrix by applying the log-linear model combined with the EM algorithm.

The relative efficiency can be computed for fetal effects, maternal effects and parent-of-origin effect, as well as effects on X-chromosome markers. Various study designs, i.e., case-parent triads, the standard case-control design, the hybrid design, and all intermediate designs, can be compared.

Note that the exact number of case families and control families in the reference or comparison design is irrelevant, as this will be accounted for in the relative efficiency estimate. However, the ratio of control families to case families within the reference or comparison design must be specified correctly. See the Examples section.

Specifying haplotype risks:
The number of haplotypes used in the simulations is determined by the nall argument, since prod(nall) different haplotypes can be made from the specified number of markers, length(nall). The arguments haplo.freq, RR, RRcm, RRcf, RRstar, RR.mat, and RRstar.mat are all lists, containing vectors of length equal to the number of haplotypes, specifying the relative risk etc. associated with each haplotype. However, the function will work without using the list format. The haplotypes are determined by creating all possible haplotypes from the given markers, in a sequence where the first marker varies mostly quickly. For instance, if nall = c(3,2), the first marker has 3 alleles, the second has 2, and 6 haplotypes are possible. Taken in order, the haplotypes are 1-1, 2-1, 3-1, 1-2, 2-2, and 3-2. When specifying, say, RR = c(1,2,1,1,1,1) the haplotype 2-1 has a double risk compared to the rest. The simplest example would be with nall = c(2) and RR = c(1,2), which would simulate a single SNP where the second allele has a double risk.

Specifying genetic effects:
Standard fetal effects are specified by the arguments RR and RRstar, whereas parent-of-origin effects are addressed by the arguments RRcm, RRcf and RRstar. Maternal effects are included by the additional arguments RRmat and RRstar.mat.

Value

hapRelEff returns the relative efficiency estimate, comparing two study designs. The first element of the list depicts the relative efficiency for each haplotype analyzed separately. If there are more than two possible haplotypes, the second element displays the overall relative efficiency for all haplotypes combined.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, hapPowerAsymp, hapRun, snpPower, hapPower

Examples

```
## Child effects: Calculate the efficiency of the standard case-control design
## (with an equal number of case and control children)
## relative to the case-parent triad design
## under the null hypothesis when the minor allele frequency is 0.1.
hapRelEff(nall = c(2), cases.comp = c(c=1),
controls.comp = c(c=1), cases.ref = c(mfc=1),
haplo.freq = c(0.1,0.9), RR = c(1,1))

## Child effects: Calculate the efficiency of the standard case-control design,
## with twice as many cases as controls, relative to the case-parent triad design
## under the null hypothesis when the minor allele frequency is 0.2.
hapRelEff(nall = c(2), cases.comp = c(c=2),
controls.comp = c(c=1), cases.ref = c(mfc=1),
haplo.freq = c(0.2,0.8), RR = c(1,1))

## Child and maternal effects: Calculate the efficiency of the case-mother dyad design
## relative to the case-parent triad design
## under the null hypothesis when the minor allele frequency is 0.1.
hapRelEff(nall = c(2), cases.comp = c(mc=1), cases.ref = c(mfc=1),
haplo.freq = c(0.1,0.9), RR = c(1,1), RR.mat=c(1,1))

## PoO effects: Calculate the efficiency of the full hybrid design,
## with twice as many control families as case families,
## relative to the case-parent triad design
## under the null hypothesis when the minor allele frequency is 0.1.
hapRelEff(nall = c(2), cases.comp = c(mfc=1),
controls.comp = c(mfc=2), cases.ref = c(mfc=1),
haplo.freq = c(0.1,0.9), RRcm = c(1,1), RRcf = c(1,1))
```

hapRun *Simulates genetic data and runs Haplin for each simulation*

Description

Calculates Haplin results by first simulating genetic data, allowing a various number of family designs, and then running Haplin on the simulations. The simulated data may contain of fetal effects, maternal effects and/or parent-of-origin effects. The function allows for simulations and calculations on both autosomal and X-chromosome markers, assuming Hardy-Weinberg equilibrium. It enables simulation and calculation of gene-environment interaction effects, i.e, the input (relative risks, number of cases etc.) may vary across strata. hapRun calls haplin, haplinStrat or haplinSlide to run on the simulated data files.

Usage

```
hapRun(nall, n.strata= 1, cases, controls, haplo.freq,
RR, RRcm, RRcf, RRstar, RR.mat, RRstar.mat, hapfunc = "haplin",
gen.missing.cases = NULL, gen.missing.controls = NULL,
n.sim = 1000, xchrom = FALSE, sim.comb.sex = "double", BR.girls, dire,
ask = TRUE, cpus = 1, slaveOutfile = "", ...)
```

Arguments

nall	A vector of the number of alleles at each locus.
n.strata	The number of strata.
cases	A list of the number of case families. Each element is a vector of the number of families of the specified family design(s) in the corresponding stratum. The possible family designs, i.e., the possible names of the elements, are "mfc" (full triad), "mc" (mother-child dyad), "fc" (father-child dyad) or "c" (a single case child). See Details for a thorough description.
controls	A list of the number of control families. Each element is a vector of the number of families of the specified family design(s) in the corresponding stratum. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother) or "f" (a single control father). See Details for a thorough description.
haplo.freq	A list of which each element is a numeric vector of the haplotype frequencies in each stratum. The frequencies are normalized and sum to one. The Details section shows how to implement this argument in agreement with the possible haplotypes.
RR	A list of which each element is a numeric vector of the relative risks in each stratum. The Details section shows how to implement this argument in agreement with the possible haplotypes.

RRcm — A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the mother for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.

RRcf — A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the father for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.

RRstar — A list of numeric vectors. Estimates how much double-dose children would deviate from the risk expected in a multiplicative dose-response relationship.

RR.mat — The interpretation is similar to RR when simulating genetic data with maternal effects.

RRstar.mat — The interpretation is similar to RRstar when simulating genetic data with maternal effects.

hapfunc — Defines which haplin function to run, the options being "haplin", "haplinSlide" or "haplinStrat". "haplinSlide" is however only partially implemented.

gen.missing.cases — Generates missing values at random for the case families. Set to NULL by default, i.e., no missing values generated. See Details for description of how to implement this argument.

gen.missing.controls — Generates missing values at random for the control families. Set to NULL by default, i.e., no missing values generated. See Details for description of how to implement this argument.

n.sim — The number of simulations, i.e., the number of simulated data files.

xchrom — Logical. Equals FALSE by default, which indicates simulation of autosomal markers. If TRUE, hapSim simulates X-linked genes.

sim.comb.sex — To be used with xchrom = TRUE. A character value that specifies how to handle gender differences on the X-chromosome. If "single", the effect of a (single) allele in males is equal to the effect of a single allele dose in females, and similarly, if "double", a single allele in males has the same effect as a double allele dose in females. Default is "double", which corresponds to X-inactivation.

BR.girls — To be used with xchrom = TRUE. Gives the ratio of baseline risk for females to the baseline risk for males.

dire — Gives the directory of the simulated data files. Missing by default, which means that none of the files are saved to files.

ask — Logical. If TRUE, hapSim will ask before overwriting the files in an already existing directory.

cpus — Allows parallel processing of its analyses. The cpus argument should preferably be set to the number of available CPUs. If set lower, it will save some capacity for other processes to run. Setting it too high should not cause any serious problems.

slaveOutfile — Character. If slaveOutfile = "" (default), output from all running cores will be printed in the standard R session window. Alternatively, the output can be saved to a file by specifying the file path and name.

... — Arguments to be used by haplin, haplinSlide or haplinStrat.

Details

hapRun applies haplin, haplinSlide or haplinStrat on each data file simulated by hapSim. It provides simulations on various family designs, i.e., triads, case-control, the hybrid design, and all intermediate designs. The simulated files may accomodate fetal effects, maternal effects and/or parent-of-origin effects. hapRun allows simulation of both autosomal and X-chromosome markers, assuming Hardy-Weinberg equilibrium. It also enables simulation and calculation of gene-environment interaction effects.

Details on how to implement the arguments listed above are provided by hapSim and the Examples section below. The stratum specific arguments may be simplified if the number of strata is one, or if the arguments are equal across all strata.

haplin, haplinStrat and haplinSlide will run with default values unless otherwise specified by hapRun. For example, if hapfunc = "haplin", haplin will use response = "free" unless response = "mult" is explicitly given as an argument in hapRun. Moreover, triads with missing data are only included in the haplin analysis if the argument use.missing equals TRUE (default in hapRun). Please confer https://haplin.bitbucket.io/docu/Haplin_power.pdf for further details and examples.

For information on the arguments to be passed on to haplin, haplinStrat and haplinSlide, please consult their help pages.

Note that RR.mat and RRstar.mat and RRcm and RRcf are required for hapSim to simulate maternal and parent-of-origin effects, respectively. To calculate these effects, however, arguments maternal = TRUE and/or poo = TRUE must be specified.

gen.missing.cases and gen.missing.controls are flexible arguments. By default, both equal NULL, which means that no missing data are generated at random. If the arguments are single numbers, missing data are generated at random with this proportion for all cases and/or controls. If the arguments are vectors of length equal to the number of loci, missing data are generated with the corresponding proportion for each locus. The arguments can also be matrices with the number of rows equal to the number of loci and three columns. Each row corresponds to a locus, and the columns correspond to mothers, fathers and children, respectively.

Value

If hapfunc = "haplin", hapRun returns a dataframe consisting of results from running haplin on each simulated file. The first two columns are:

sim.no	The name of the directory from which the results are calculated, i.e., the simulation number
row.no	The row number within each simulation

haptable gives detailed information of the full dataframe.

If hapfunc = "haplinSlide", hapRun returns a list of which each element contains the results from a single run of haplinSlide. Consult suest for a thorough description of the output. Note, however, that hapfunc = "haplinSlide" is currently only implemented for diallelic markers, and the reference category is always chosen to be the first haplotype (see hapSim for a description of the haplotype grid).

If hapfunc = "haplinStrat", haplinStrat is used to estimate gene-effects in each stratum of the exposure covariate, and the results from all strata are compared using gxe. hapRun returns a list, where each element is the result of a single run of gxe.

Additionaly, if `dire` is not missing by default, the simulated files from which the Haplin results are calculated, are stored in the given directory.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, haplinSlide, hapSim, haptable, suest, hapPower, hapPowerAsymp

Examples

```
## Not run:
## Simulate Haplin results from 100 files using the multiplicative model in haplin.
## The files consist of fetal effects at two diallelic markers,
## corresponding to haplo.freq = rep(0.25, 4), RR = c(2,1,1,1)
## and RRstar = c(1,1,1,1). That is, the first allele has a doubled risk
## relative to the rest. The data consists of a combination of
## 100 case triads and 100 control triads with no missing data.
## No environmental factors are considered, i.e. the number of strata is one.
hapRun(nall = c(2,2), n.strata = 1, cases = c(mfc=100), controls = c(mfc=100),
haplo.freq = rep(0.25,4), RR = c(2,1,1,1), RRstar = c(1,1,1,1),
hapfunc = "haplin", response = "mult", n.sim = 100, dire = "simfiles", ask = FALSE)

## Simulate power from 100 files applying haplinStrat.
## The files consist of fetal and maternal effects at two diallelic markers.
## The data is simulated for 500 case triads and 200 control families in the first stratum,
## and 500 case triads and 500 control trids in the second.
## The fetal effects vary across strata,
## whereas the maternal effects are the same.
## One percent of the case triads are missing at random in the second stratum.
hapRun(nall = c(2,2), n.strata = 2, cases = c(mfc=500),
controls = list(c(mfc=200),c(mfc=500)), haplo.freq = rep(0.25,4), maternal = TRUE,
RR = list(c(1.5,1,1,1),c(1,1,1,1)), RRstar = c(1,1,1,1),
RR.mat = c(1.5,1,1,1), RRstar.mat = c(1,1,1,1),
gen.missing.cases = list(NULL,0.01), use.missing = TRUE, hapfunc = "haplinStrat",
n.sim = 100, ask = FALSE)

## Simulate Haplin results from 100 files using haplin.
## The files consist of fetal effects at one diallelic locus,
```

```
## corresponding to haplo.freq = rep(0.5,2), RR = c(1.5,1) and RRstar = c(1,1).
## We have a combination of 100 case triads and
## 100 control triads with no missing data.
## No environmental effects are considered.
hapRun(nall = c(2), n.strata = 1, cases = c(mfc=100), controls = c(mfc=100),
haplo.freq = rep(0.5,2), RR = c(1.5,1), RRstar = c(1,1),
hapfunc = "haplin", n.sim = 100, dire = "simfiles", ask = FALSE)

## End(Not run)
```

---

hapSim *Simulation of genetic data in Haplin format*

Description

Simulates genetic data in Haplin format, consisting of fetal effects, maternal effects and/or parent-of-origin effects. Allows for simulation of both autosomal and X-linked markers, assuming Hardy-Weinberg equilibrium. Enables stratified simulations for gene-environment interaction analyses, i.e the input (relative risks, number of cases etc) may vary across different strata.

Usage

```
hapSim(nall, n.strata = 1, cases, controls, haplo.freq,
RR, RRcm, RRcf, RRstar, RR.mat, RRstar.mat,
gen.missing.cases = NULL, gen.missing.controls = NULL,
n.sim = 1000, xchrom = F, sim.comb.sex = "double", BR.girls, dire = "simfiles",
ask = TRUE, verbose = TRUE, cpus = 1)
```

Arguments

nall	A vector of the number of alleles at each locus.
n.strata	The number of strata.
cases	A list of the number of case families. Each element is a vector of the number of families of the specified family design(s) in the corresponding stratum. The possible family designs, i.e., the possible names of the elements, are "mfc" (full triad), "mc" (mother-child dyad), "fc" (father-child dyad) or "c" (a single case child). See Details for a thorough description.
controls	A list of the number of control families. Each element is a vector of the number of families of the specified family design(s) in the corresponding stratum. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother) or "f" (a single control father). See Details for a thorough description.
haplo.freq	A list of which each element is a numeric vector of the haplotype frequencies in each stratum. The frequencies will be normalized so that they sum to one. The Details section shows how to implement this argument in agreement with the possible haplotypes.

RR — A list of which each element is a numeric vector of the relative risks in each stratum. The Details section shows how to implement this argument in agreement with the possible haplotypes.

RRcm — A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the mother for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.

RRcf — A list of numeric vectors. Each vector contains the relative risks associated with the haplotypes transmitted from the father for this stratum. See Details for description of how to implement this argument in agreement with the possible haplotypes.

RRstar — A list of numeric vectors. Estimates how much double-dose children would deviate from the risk expected in a multiplicative dose-response relationship.

RR.mat — The interpretation is similar to RR when simulating genetic data with maternal effects.

RRstar.mat — The interpretation is parallel to RRstar when simulating genetic data with maternal effects.

gen.missing.cases — Generates missing values at random for the case families. Set to NULL by default, i.e., no missing values generated. See Details for description of how to implement this argument.

gen.missing.controls — Generates missing values at random for the control families. Set to NULL by default, i.e., no missing values generated. See Details for description of how to implement this argument.

n.sim — The number of simulations, i.e., the number of simulated data files.

xchrom — Logical. Equals FALSE by default, which indicates simulation of autosomal markers. If TRUE, hapSim simulates X-linked genes.

sim.comb.sex — To be used with xchrom = TRUE. A character value which specifies how to handle gender differences on the X-chromosome. If "single", the effect of a (single) allele in males is equal to the effect of a single allele dose in females, and similarly, if "double", a single allele in males has the same effect as a double allele dose in females. Default is "double", which corresponds to X-inactivation.

BR.girls — To be used with xchrom = TRUE. Gives the ratio of baseline risk for females to the baseline risk for males.

dire — Gives the directory of the simulated data files.

ask — Logical. If TRUE, hapSim will ask before overwriting the files in an already existing directory.

verbose — Logical. Default is TRUE, which means that the file name is displayed for each iteration. Works only when cpus = 1.

cpus — Allows simulations to be performed in parallel. The cpus argument should preferably be set to the number of available cores. If set lower, it will save some capacity for other processes to run. Setting it too high should not cause any serious problems.

Details

hapSim simulates allele values for case and control families at multiple markers (typically in LD) simultaneously. The number of markers/SNPs involved will typically be in the range 1 to 6. Data are simulated to produce relative risks of disease as specified by the user input. Simulations can be performed separately over a number of strata so as to simulate gene-environment interactions.

Specifying haplotype risks:
The number of haplotypes used in the simulations is determined by the nall argument, since prod(nall) different haplotypes can be made from the specified number of markers, length(nall). The arguments haplo.freq, RR, RRcm, RRcf, RRstar, RR.mat, and RRstar.mat are all lists where each element represents a stratum. Within each stratum, the arguments are vectors of length equal to the number of haplotypes, specifying the relative risk etc. associated with each haplotype. The stratum specific arguments may be simplified if the number of strata is one, or if the arguments are equal across all strata.

The haplotypes are determined by creating all possible haplotypes from the given markers, in a sequence where the first marker varies mostly quickly. For instance, if nall = c(3,2), the first marker has 3 alleles, the second has 2, and 6 haplotypes are possible. Taken in order, the haplotypes are 1-1, 2-1, 3-1, 1-2, 2-2, and 3-2. When specifying, say, RR = c(1,2,1,1,1,1) the haplotype 2-1 has a double risk compared to the rest. With, for instance, two strata, the specification RR = list(c(1,2,1,1,1,1), c(1,1,1,1,1,1)) would mean that the risk associated with 2-1 is elevated only in the first stratum, not the second.

The simplest example would be with nall = c(2) and RR = c(1,2), which would simulate a single SNP where the second allele has a double risk.

Output file format:
The format of the simulated files is relatively flexible and allows multi-allelic markers and various designs. If both case and control families are present, the simulated files contain a leading column of the case/control status (1/0). If xchrom=TRUE, the neighboring column to the left of the genetic data contains the sex information (1 = male, 2 = female). Each line represents genotypes for a case or control triad.

There are six columns for each locus, two for the mother (M), two for the father (F) and two for the child (C). The columns are placed in the following sequence: M11 M12 F11 F12 C11 C12 M21 M22 F21 F22 C21 C22... etc, where the first number indicates marker, and the second number indicates the first or second allele at this locus. Columns are separated by a single white space, and missing data are coded as NA.

Intermediate designs, for instance mother-child dyads, are represented as full triads with columns of absent family members set to missing. In the case of a pure case-control design, however, each line represents a single individual, and there are no columns representing mothers and fathers.

There are no row or column names in the files.

Some examples are given below. See https://haplin.bitbucket.io/docu/haplin_data_format.html for a thorough description of the Haplin format. Note that this description separates the two alleles for an individual within a locus by a semi-colon, such as 1;2. This is, however, not necessary. Confer the document https://haplin.bitbucket.io/docu/Haplin_power.pdf for details and examples on how to perform the simulations.

gen.missing.cases and gen.missing.controls are flexible arguments. By default, both equal NULL, which means that no missing data are generated at random. If the arguments are single

numbers, missing data are generated at random with this proportion for all cases and/or controls. If the arguments are vectors of length equal to the number of loci, missing data are generated with the corresponding proportion for each locus. The arguments can also be matrices with the number of rows equal to the number of loci and three columns. Each row corresponds to a locus, and the columns correspond to mothers, fathers and children, respectively.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, hapRun, hapPower

Examples

```
## Not run:
## Simulate genetic data (100 files) at two diallelic markers, consisting of fetal effects
## corresponding to haplo.freq = rep(0.25, 4), RR = c(2,1,1,1) and RRstar = c(1,1,1,1),
## for the combination of 1000 case triads and 1000 control triads with no missing data.
## Only one stratum.
hapSim(nall = c(2,2), n.strata = 1, cases = c(mfc=1000),
controls = c(mfc=1000), haplo.freq = rep(0.25,4),
RR = c(2,1,1,1), RRstar = c(1,1,1,1), n.sim = 100, dire = "simfiles")

## Simulate genetic data (100 files) at two diallelic markers,
## consisting of fetal and maternal effects corresponding to
## haplo.freq = rep(0.25, 4), RR = c(2,1,1,1), RRstar = c(1,1,1,1),
## RR.mat = c(2,1,1,1) and RRstar.mat = c(1,1,1,1),
## for 1000 case triads and zero control families.
## One percent of the case triads are missing at random. One stratum only.
hapSim(nall = c(2,2), n.strata=1, cases = c(mfc=1000),
controls = c(mfc=0), haplo.freq = rep(0.25,4), RR = c(2,1,1,1),
RRstar = c(1,1,1,1), RR.mat = c(2,1,1,1), RRstar.mat = c(1,1,1,1),
gen.missing.cases = 0.01, n.sim = 100, dire = "simfiles")

## Simulate genetic data (100 files) at two diallelic markers. In the first stratum,
## we have a combination of 500 case triads and 500 control triads with
## haplo.freq = rep(0.25, 4), RR = c(2,1,1,1) and RRstar = c(1,1,1,1).
## In the second stratum, we have 300 case triads and 500 control triads with
## haplo.freq = rep(0.25, 4), RR = c(1,1,1,1) and RRstar = c(1,1,1,1).
```

```
## One percent of the control triads are missing at random in the first stratum.
hapSim(nall = c(2,2), n.strata= 2, cases = list(c(mfc=500),c(mfc=300)),
controls = c(mfc=500),haplo.freq = rep(0.25,4),
RR = list(c(2,1,1,1),c(1,1,1,1)), RRstar = c(1,1,1,1),
gen.missing.controls = list(0.01,NULL), n.sim = 100, dire = "simfiles")

## End(Not run)
```

---

haptable *Create haplin table*

Description

Create a comprehensive table of haplin output

Usage

```
haptable(object)
```

Arguments

object	A haplin object, i.e. the result of running haplin.

Details

haptable extracts the most important information from a haplin object to produce a summary table. The table can then be saved with, for instance, write.table, making the results easily accessible to other applications. You can also use output to produce and save the same results.

Value

—- A dataframe is returned, with the following columns: —-

marker	Name(s) of marker(s) investigated
alleles	A listing of the alleles found at each marker
counts	Frequency counts of alleles at each marker
HWE.pv	P-value from Hardy-Weinberg equilibrium test at each marker
Original	Number of triads before removal
After.rem.NA	Number of triads after removal of missing
After.rem.Mend.inc.	Number of triads after removal of Mendelian inconsistencies
After.rem.unused.haplos	Number of triads after removal of unused (rare) haplotypes
pv.overall	Overall likelihood ratio p-value (test of all genetic effects combined)
haplos	Haplotypes (or single-marker alleles) found during estimation
haplofreq	Estimated haplotype frequencies

haplofreq.lower	Lower 95% CI for estimated haplotype frequencies
haplofreq.upper	Upper 95% CI for estimated haplotype frequencies
reference	Reference method. If ref.cat is used, the reference category is labeled "ref"
RR.est.	Estimated single dose relative risk
RR.lower	Lower 95% CI for single dose relative risk
RR.upper	Upper 95% CI for single dose relative risk
RR.p.value	P-values for individual single dose effect
RRdd.est.	Estimated double dose relative risk
RRdd.lower	Lower 95% CI for double dose relative risk
RRdd.upper	Upper 95% CI for double dose relative risk
RRdd.p.value	P-values for individual double dose effect
NOTE1	When maternal = TRUE, there will be additional columns:
RRm.est.	Estimated single dose relative risk for maternal haplotype
RRm.lower	Lower 95% CI for single dose relative risk for maternal haplotype
RRm.upper	Upper 95% CI for single dose relative risk for maternal haplotype
RRm.p.value	P-values for individual single dose effect of maternal haplotype
RRmdd.est.	Estimated double dose relative risk for maternal haplotype
RRmdd.lower	Lower 95% CI for double dose relative risk for maternal haplotype
RRmdd.upper	Upper 95% CI for double dose relative risk for maternal haplotype
RRmdd.p.value	P-values for individual double dose effect of maternal haplotype
NOTE2	When poo = TRUE, the RR.est., RR.lower, and RR.upper columns will be replaced by the following columns:
RRcm.est.	Estimated single dose relative risk, when inherited from the mother
RRcm.lower	Lower 95% CI for single dose relative risk, when inherited from the mother
RRcm.upper	Upper 95% CI for single dose relative risk, when inherited from the mother
RRcm.p.value	P-values for individual single dose effects, when inherited from the mother
RRcf.est.	Estimated single dose relative risk, when inherited from the father
RRcf.lower	Lower 95% CI for single dose relative risk, when inherited from the father
RRcf.upper	Upper 95% CI for single dose relative risk, when inherited from the father
RRcf.p.value	P-values for individual single dose effects, when inherited from the father
RRcm_RRcf.est.	An estimate of parent-of-origin effect, i.e. the ratio RRcm/RRcf
RRcm_RRcf.lower	Lower 95% CI for ratio RRcm/RRcf
RRcm_RRcf.upper	Upper 95% CI for ratio RRcm/RRcf
RRcm_RRcf.p.value	P-value for parent-of-origin effect RRcm/RRcf at that marker

Note

Further information is found on the web page

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

haplin, output

Examples

```
## Not run:

# Produce a table containing the most important output from haplin:
res <- haplin("data.dat", use.missing = T, maternal = T)
haptable(res)

## End(Not run)
```

---

initParallelRun *Initialization of the Rmpi cluster*

Description

This function prepares a cluster using Rmpi package. The initialization is paired with closing of the cluster using the finishParallelRun function.

Usage

```
initParallelRun(cpus)
```

Arguments

cpus	Number of cores to use (optional). If given, only that number of CPUs will be used. By default (if not set), the Rmpi will check how many CPUs are available in the system and take the maximum number.

`lineByLine` *Line-by-line modification of files*

Description

Modifies a data file line by line, i.e. reads a file line by line, converts each line, then writes to the modified file. This method is especially useful when modifying large datasets, where the reading of entire files may be time consuming and require a large amount of memory.

Usage

```
lineByLine(infile, outfile, linefunc = identity, choose.lines = NULL,
choose.columns = NULL, col.sep = " ", ask = TRUE,
blank.lines.skip = TRUE, verbose = TRUE, ...)
```

Arguments

`infile`	A character string giving the name and path of the file to be modified.
`outfile`	A character string giving the name of the modified file. The name of the file is relative to the current working directory, unless the file name contains a definite path.
`linefunc`	`lineByLine` modifies each line using `linefunc`. Default is the identity function. The user may define his or her own line-modifying functions, see Details for a thorough description.
`choose.lines`	A numeric vector of lines to be selected or dropped from `infile`. Positive values refer to lines to be chosen, whereas negative values refer to lines to be skipped. The vector cannot include both positive and negative values at the same time. If "NULL" (default), all lines are selected.
`choose.columns`	A numeric vector of columns to be selected (positive values) or skipped (negative values) from `infile`. The vector cannot include both positive and negative values at the same time. By default, all columns are selected without reordering among the columns. Duplication and reordering among the selected columns will occur in the modified file corresponding to the order in which the columns are listed.
`col.sep`	Specifies the separator that splits the columns in `infile`. By default, `col.sep = " "` (space). To split at all types of spaces or blank characters, set `col.sep = "[[:space:]]"` or `col.sep = "[[:blank:]]"`.
`ask`	Logical. Default is "TRUE". If set to "FALSE", an already existing outfile will be overwritten without asking.
`blank.lines.skip`	Logical. If "TRUE" (default), `lineByLine` ignores blank lines in the input.
`verbose`	Logical. Default is "TRUE", which means that the line number is displayed for each iteration, in addition to output from `linefunc`. If `choose.columns` contains invalid column numbers, this will also be displayed.
`...`	Further arguments to be passed to `linefunc`.

Details

When reading large datafiles, functions such as `read.table` can use a large amount of memory and be extremely time consuming. Instead of reading the entire file at once, `lineByLine` reads one line at a time, modifies the line using `linefunc`, and then writes the line to `outfile`.
The user may specify his or her own line-converting function. This function must take the argument `x`, a character vector representing a single line of the file, split at spaces. However, additional arguments may be included. If `verbose` equals "TRUE", output should be displayed. The modified vector is returned.
The framework of the line-modifying function may look something like this:

```
lineModify <- function(x){
.xnew <- x

## Define any modifications, for instance recoding missing values in a dataset from NA to 0:
.xnew[is.na(.xnew)] <- 0

## Just to monitor progress, display, for instance, 10 first elements, without newline:
cat(paste(.xnew[1:min(10, length(.xnew))], collapse = " "))

## Return converted vector
return(.xnew)
}
```

See `Haplin:::lineConvert` for an additional example of a line-modifying function.

Value

`lineByLine` returns the number of lines read, although invisible. The main objective is the modified file.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

`convertPed`

Examples

```
## Not run:

## Extract the first ten columns from "myfile.txt",
## without reordering
lineByLine(infile = "myfile.txt", outfile = "myfile_modified.txt",
choose.columns = c(1:10))


## End(Not run)
```

`nfam` *Count the number of families in the data*

Description

This is a help function to count the number of families in an object read in with genDataRead (or loaded with genDataLoad). Note: it is assumed that the study design is either 'triad' or 'cc.triad'.

Usage

```
nfam(data.in)
```

Arguments

`data.in`	The data read in by genDataRead.

Value

How many families (integer).

`nindiv` *Count the number of individuals in the data*

Description

This is a help function to count the number of individuals in an object read in with genDataRead (or loaded with genDataLoad).

Usage

```
nindiv(data.in, design = "triad")
```

Arguments

`data.in`	The data read in by genDataRead.
`design`	The design used in the study - choose from: • *triad* (default) - data includes genotypes of mother, father and child; • *cc* - classical case-control; • *cc.triad* - hybrid design: triads with cases and controls

Value

How many individuals (integer).

`nsnps` *Count the number of markers in the data*

Description

This is a help function to count the number of markers in an object read in with genDataRead (or loaded with genDataLoad).

Usage

```
nsnps(data.in, design = "triad")
```

Arguments

`data.in`	The data read in by genDataRead.
`design`	The design used in the study - choose from: • *triad* (default) - data includes genotypes of mother, father and child; • *cc* - classical case-control; • *cc.triad* - hybrid design: triads with cases and controls

Value

How many markers (integer).

output *Save files with summary, table, and plot from a haplin object.*

Description

Create summary tables and figure from a haplin object. Save results as separate files in a specified directory.

Usage

```
output(object, dirname, ask = T)
```

Arguments

object	A haplin object, i.e. the result of running haplin.
dirname	Text string, for instance "c:/work/haplinresults". Name of directory where results should be saved. Default is to save results in the current working directory.
ask	Logical. If TRUE (default), you will be asked before overwriting any files with the same name. If FALSE, output will overwrite without warning.

Details

After having run haplin and saved the result (in the R workspace), the output function will extract summary results, a summary table, and a plot of the results and save them to the specified directory. The filenames will be haplin_summary.txt, haplin_table.txt and haplin_plot.jpg, respectively. output simply uses the available functions summary, haptable, and plot to produce the files, but is a quick way of saving all the relevant results.

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin

Examples

```
## Not run:

# Run haplin and save results in separate files
# in the c:\work\haplinresults directory:
res <- haplin("data.dat", use.missing = T, maternal = T)
output(res, dirname = "c:/work/haplinresults")

## End(Not run)
```

`plot.haplin` *Plot a haplin object*

Description

Plot a haplin object and (optionally) produce picture files

Usage

```
## S3 method for class 'haplin'
plot(x, reference, separate.plots = F, filename,
filetype = "png", use.dd, ...)
```

Arguments

`x`	A `haplin` object, i.e. the result of running `haplin`. This is the only required argument.
`reference`	Same as `reference` argument in `haplin`. Note that when plotting, you can only choose `"reciprocal"`, `"population"` or `"ref.cat"`. You cannot use a numeric value to change the reference category, to do that `haplin` must be run over again. (See the `reference` argument of `haplin`.)
`separate.plots`	Logical. If you estimate effects of both fetal and maternal genes you can decide whether or not to plot them in the same plot. The default is the same plot (TRUE), the alternative (FALSE) means in separate plots. If you choose separate plots you may have to set the graphics window to "recording" to make sure you can scroll back to the first plot.
`filename`	If you want a file containing the plot to be produced, give a character string for the filename.
`filetype`	The default filetype is `"png"`, alternatively you can choose `"jpeg"`.
`use.dd`	Numeric vector indicating which double dose estimates should be plotted. For instance, if set to c(1,3) only the first and third haplotypes will be drawn with double dose estimates. This is useful if some haplotypes are rare and you want to exclude the uncertain estimates from the plot.
`...`	Further arguments to be passed on to the plot function

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin

Examples

```
## Not run:

# Produce separate plots for child and mother, dump plots to files:
res <- haplin("data.dat", use.missing = T, maternal = T)
plot(res, separate.plots = T, filename = "Haplinres.png")


## End(Not run)
```

---

plot.haplinSlide *Plotter function for haplinSlide.*

Description

This will plot any haplinSlide object in one figure.

Usage

```
## S3 method for class 'haplinSlide'
plot(
  x,
  filename,
  title,
  windows,
```

```
  plot.signif.only = FALSE,
  signif.thresh = 0.05,
  y.lim,
  x.title,
  y.title,
  file.width,
  file.height,
  ...
)
```

Arguments

x	The haplinSlide object (NB: only the output produced by running haplinSlide with the table.output argument set to TRUE!)
filename	If the plot should be saved to the disk, give the name of the output file including the file extension.
title	If the user wishes to override the default title of the plot, give it here.
windows	Numerical vector. If given, this will only plot the chosen windows.
plot.signif.only	Logical: whether to filter out the "non-significant" markers from the plot. Default: FALSE, i.e., plot everything.
signif.thresh	The threshold defining the significant p-values: if plot.signif.only == TRUE, then only the markers with relative risk p-values lower than the threshold will be kept for plotting. Default: 0.05.
y.lim	Vector with two numbers setting the Y limits of the plotted graph.
x.title	Title for the X axis (default: "marker").
y.title	Title for the Y axis (default: "RR (log scale)").
file.width	Width (in inches) for the output plot, if a filename was given.
file.height	Height (in inches) for the output plot, if a filename was given.
...	other arguments (ignored).

Details

The haplinSlide object is a list of haplin results - by default in haptable form. This is used to plot the relative risk estimates for all the markers in one plot, to enable easy comparison. NB: those estimates that have infinite confidence interval will not be plotted.

Value

ggplot object.

`plot.haplinStrat` *Plotter function for haplinStrat results.*

Description

This will automatically plot all haplinStrat results on one figure.

Usage

```
## S3 method for class 'haplinStrat'
plot(
  x,
  filename,
  title,
  windows,
  plot.signif.only = FALSE,
  signif.thresh = 0.05,
  y.lim,
  x.title,
  y.title,
  file.width,
  file.height,
  ...
)
```

Arguments

`x`	The `haplinSlide` object (NB: only the output produced by running `haplinSlide` with the `table.output` argument set to TRUE!)
`filename`	If the plot should be saved to the disk, give the name of the output file including the file extension.
`title`	If the user wishes to override the default title of the plot, give it here.
`windows`	Numerical vector. If given, this will only plot the chosen windows.
`plot.signif.only`	Logical: whether to filter out the "non-significant" markers from the plot. Default: FALSE, i.e., plot everything.
`signif.thresh`	The threshold defining the significant p-values: if `plot.signif.only` == `TRUE`, then only the markers with relative risk p-values lower than the threshold will be kept for plotting. Default: 0.05.
`y.lim`	Vector with two numbers setting the Y limits of the plotted graph.
`x.title`	Title for the X axis (default: "marker").
`y.title`	Title for the Y axis (default: "RR (log scale)").
`file.width`	Width (in inches) for the output plot, if a filename was given.
`file.height`	Height (in inches) for the output plot, if a filename was given.
`...`	other arguments (ignored).

Details

This function uses the same style as plot.haplinSlide and plots all of the haplinStrat results on one figure, for easy comparison. NB: those estimates that have infinite confidence interval will not be plotted.

Value

ggplot object.

---

plot.haptable *Plot a haptable object*

Description

Plot a haptable object, which is the result of running haptable on a haplin result, and (optionally) produce picture files.

Usage

```
## S3 method for class 'haptable'
plot(x, separate.plots = F, filename,
filetype = "png", use.dd, verbose = T, ...)
```

Arguments

x	A haptable object, i.e. the result of running haptable on a result from haplin. This is the only required argument.
separate.plots	Logical. If you estimate effects of both fetal and maternal genes you can decide whether or not to plot them in the same plot. The default is the same plot (TRUE), the alternative (FALSE) means in separate plots. If you choose separate plots you may have to set the graphics window to "recording" to make sure you can scroll back to the first plot.
filename	If you want a file containing the plot to be produced, give a character string for the filename.
filetype	The default filetype is "png", alternatively you can choose "jpeg".
use.dd	Numeric vector indicating which double dose estimates should be plotted. For instance, if set to c(1,3) only the first and third haplotypes will be drawn with double dose estimates. This is useful if some haplotypes are rare and you want to exclude the uncertain estimates from the plot.
verbose	Turns on or off some minor comments when plotting
...	Further arguments to be passed on to the plot function

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin

Examples

```
## Not run:

# Directly plotting the haplin result. Produce separate plots for child and mother,
# dump plots to files:
res <- haplin("data.dat", use.missing = T, maternal = T)
plot(res, separate.plots = T, filename = "Haplinres.png")

# Create haptable from the haplin result and plot the results in the table:
res <- haplin("data.dat", use.missing = T, maternal = T)
tab <- haptable(res)
plot(tab)

# Create haptables for the 10 first markers of a data file using haplinSlide.
# Create plots for each result:
res <- haplinSlide("data.dat", markers = 1:10, use.missing = T, maternal = T, table.output = T)
lapply(res, plot)

## End(Not run)
```

---

plotPValues *Plotting p-values for relative risks*

Description

This function plots p-values for the relative risks calculated by haplinSlide.

Usage

```
plotPValues(
  object,
  windows,
  which.p.val = "overall",
  plot.signif.only = FALSE,
  signif.thresh = 0.05,
  title,
  filename
)
```

Arguments

`object`	The `haplinSlide` results: list of `haptable` objects.
`windows`	Numerical vector; if given, the plot will be restricted to only those.
`which.p.val`	Character string specifying which p-values to choose for plotting: "overall" (default), "child", "child.double", "maternal", "maternal.double", "paternal". The last three options can be chosen only if `haplinSlide` was run with `maternal = TRUE` or `poo = TRUE`.
`plot.signif.only`	Logical: whether to filter out the "non-significant" markers from the plot. Default: FALSE, i.e., plot everything.
`signif.thresh`	The threshold defining the significant p-values: if `plot.signif.only == TRUE`, then only the markers with relative risk p-values lower than the threshold will be kept for plotting. Default: 0.05.
`title`	Optional character string for the title of the figure.
`filename`	If the plot should be saved to the disk, give the name of the output file including the file extension.

Details

The output of `haplinSlide` can be very lengthy and not suitable for an overall plot of all the relative risks (RR) on one figure. Therefore, it's advised to first plot only the p-values for each window (user can choose which p-values to plot - see parameter `which.p.val`), and only then plot the RRs for specific windows, for which the p-values are significant.

Value

Invisibly returns the table with only the plotted p-values.

pQQ *QQ-plot with confidence intervals for a vector of p-values*

Description

Produces a QQ-plot of p-values. The x-axis is -log10 of the expected p-values (under a null hypothesis of no effects), the y-axis is -log10 values of the actual p-values. A (pointwise) confidence interval can be drawn, and names of snps/genes corresponding to the most significant ones can be added.

Usage

```
pQQ(pvals, nlabs = 6, conf = 0.95, lim, mark = 0.05, ...)
```

Arguments

pvals	A vector of p-values.
nlabs	The number of (most significant) p-values to be labeled using `names(pvals)`.
conf	The confidence level of the pointwise confidence band. The default is 0.95. The confidence intervals are computed under the assumption of the p-values being drawn independently from a uniform [0,1] distribution. To leave out the confidence interval, set this to FALSE.
lim	A vector of length 2 giving the plot limits (on a log10 scal, for instance c(0,4)). Plot limits are computed automatically. However, if other plot limits are desireable, they can be set using this argument.
mark	By default, the 0.05 significance level is marked by lines. Can be changed to a different value, or set to FALSE.
...	Other arguments passed on to the plotting function.

Details

The `pvals` argument should be a vector of p-values to be plotted. If the vector has names corresponding to marker (snp) names, the plot will label some of the most significant points with the marker names.

Value

No value is returned.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

`print.haplin` *Print a haplin object*

Description

Print basic information about a haplin object

Usage

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

Arguments

`x`	A `haplin` object, i.e. the result of running `haplin`.
`...`	Other arguments, passed on to `print`.

Note

Further information is found on the web page

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

`haplin`

`print.summary.haplin` *Print the summary of a haplin object*

Description

Print the result of applying `summary` to a haplin object

Usage

```
## S3 method for class 'summary.haplin'
print(x, digits, ...)
```

Arguments

`x`	A `haplin` object, i.e. the result of running `haplin`.
`digits`	The number of digits to be used in the printout. Defaults to 3.
`...`	Other arguments (ignored).

Note

Further information is found on the web page

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

`haplin`

Examples

```
## Not run:

# Standard summary:
res <- haplin("data.dat", use.missing = T, maternal = T)
summary(res)

# Increase number of digits in printout
print(summary(res), digits = 8)


## End(Not run)
```

`rbindFiles` *Combine a sequence of files by rows*

Description

Takes a sequence of files and combines them by rows, without reading the full files into memory. This is especially useful when dealing with large datasets, where the reading of entire files may be time consuming and require a large amount of memory.

Usage

```
rbindFiles(infiles, outfile, col.sep, header = FALSE, ask = TRUE,
verbose = FALSE, add.file.number = FALSE, blank.lines.skip = FALSE)
```

Arguments

`infiles`	A character vector of names (and paths) of the files to combine.
`outfile`	A character string giving the name of the modified file. The name of the file is relative to the current working directory, unless the file name contains a definite path.
`col.sep`	Specifies the separator used to split the columns in the files. To split at all types of spaces or blank characters, set `col.sep = "[[:space:]]"` or `col.sep = "[[:blank:]]"`.
`header`	A logical variable which indicates if the first line in each file contains the names of the variables. If "TRUE", `outfile` will display this header in its first row, assuming the headers for each file are identical. Equals FALSE by default, i.e. no headers assumed.
`ask`	Logical. Default is "TRUE". If set to "FALSE", an already existing outfile will be overwritten without asking.
`verbose`	Logical. Default is "TRUE", which means that the line number is displayed for each iteration, i.e. each combined line.

`add.file.number`
: A logical variable which equals "FALSE" by deafult. If "TRUE", an extra first column will be added to the outfile, consisting of the file numbers for each line.

`blank.lines.skip`
: Logical. If "TRUE" (default), `lineByLine` ignores blank lines in the input.

Details

The function `rbind` combines R objects by rows. However, reading large data files may require a large amount of memory and be extremely time consuming. `rbindFiles` avoids reading the full files into memory. It reads the files line by line, possibly modifies each line, then writes to outfile. If however, `header`, `verbose`, `add.file.number` and `blank.lines.skip` are all set to "FALSE" (their default values), the files are appended directly, thus evading line-by-line modifications. In the case where `infiles` contains only one file and no output or modifications are requested (`verbose`, `add.file.number` and `blank.lines.skip` equal "FALSE"), an identical copy of this file is made.

Value

There is no useful output; the objective of `rbindFiles` is to produce `outfile`.

Note

Combining the files by reading each file line by line is less time efficient than appending the files directly. For this reason, if `header = FALSE`, changing the values of the logical variables `verbose`, `add.file.number` and `blank.lines.skip` from "FALSE" to "TRUE" should not be done unless absolutely necessary.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

`cbindFiles`, `lineByLine`

Examples

```
## Not run:

# Combines the three infiles, by rows
rbindFiles(file.names = c("myfile1.txt", "myfile2.txt", "myfile3.txt"),
```

```
outfile = "myfile_combined_by_rows.txt", col.sep = " ", header = TRUE, verbose = TRUE)

## End(Not run)
```

---

showGen *Display chosen genotypes*

Description

This is a help function to extract genotypes from an object read in with genDataRead (or loaded with genDataLoad).

Usage

```
showGen(data.in, design = "triad", n = 5, from, to, sex, markers = 1:5)
```

Arguments

data.in	The data read in by genDataRead.
design	The design used in the study - choose from: • *triad* - data includes genotypes of mother, father and child; • *cc* - classical case-control; • *cc.triad* - hybrid design: triads with cases and controls; .
n	Number of rows to display or "all" (default: 5).
from	From which row to display (optional, default: from the first).
to	To which row to display (optional).
sex	If the sex column is part of the phenotypic information, the user can choose based on one of the categories used in this column (optional); NB: this does not combine with the 'to' and 'from' arguments.
markers	A vector specifying which markers to display or "all" (default: first 5); NB: the user can specify the markers by numbers or by their names.

Value

A table with genotypes extracted from 'data.in'.

showPheno *Display phenotype part of data*

Description

This is a help function to extract phenotypic (and covariate) data from an object read in with genDataRead (or loaded with genDataLoad).

Usage

```
showPheno(data.in, n = 5, from, to, sex)
```

Arguments

data.in	The data read in by genDataRead).
n	Number of rows to display or "all" (default: 5).
from	From which row to display (optional, default: from the first).
to	To which row to display (optional).
sex	If the sex column is part of the phenotypic information, the user can choose based on one of the categories used in this column (optional); NB: this does not combine with the 'to' and 'from' arguments.

Value

A table with phenotypic and covariate data (if any) extracted from 'data.in'.

showSNPnames *Display marker names*

Description

This is a help function to extract marker names from an object generated by genDataRead, genDataLoad, genDataGetPart or genDataPreprocess.

Usage

```
showSNPnames(data.in, n = 5, from, to)
```

Arguments

data.in	The data as outputted by genDataRead, genDataLoad, genDataGetPart or genDataPreprocess.
n	Number of names to display or "all" (default: 5).
from	From which marker to display (optional, default: from the first).
to	To which marker to display (optional).

Value

A vector with marker names, as read in from map.file or generated dummy names.

---

snpPos *Find the column numbers of SNP identifiers/SNP numbers in a ped file*

Description

Gives the column numbers of SNP identifiers or SNP numbers in a standard ped file, calculated from the SNP's positions in the corresponding map file. The column numbers are sorted in the same order as `snp.select`. These positions may be useful when extracting a selection of SNPs from a ped file.

Usage

```
snpPos(snp.select, map.file, blank.lines.skip = TRUE)
```

Arguments

`snp.select`	A character vector of the SNP identifiers (RS codes) or a numeric vector of the SNP numbers.
`map.file`	A character string giving the name and path of the standard map file to be used. See Details for a description of the standard map format.
`blank.lines.skip`	Logical. If "TRUE" (default), `snpPos` ignores blank lines in `map.file`.

Details

To extract certain SNPs from a standard ped file, one has to know their positions in the ped file. This can be obtained from the corresponding map file.

The map file should look something like this:

```
Chromosome SNP-identifier Base-pair-position
1               RS9629043             554636
1              RS12565286             711153
1              RS12138618             740098
```

Alternatively, the map file could contain four columns. The column values should then be: Chromosome, SNP-identifier, Genetic-distance, Base-pair-position.
A header must be added to the map file if this does not already exist.

The format of the corresponding ped file should be something like this:

```
1104  1104-1  1104-2  1104-3  1  2  4  1  3  2
1104  1104-2       0       0  1  1  4  1  2  2
1104  1104-3       0       0  2  1  0  0  0  0
1105  1105-1  1105-2  1105-3  2  2  1  1  2  2
1105  1105-2       0       0  1  1  1  1  2  2
1105  1105-3       0       0  2  1  1  1  3  2
```

The column values are: Family id, Individual id, Father's id, Mother's id, Sex (1 = male, 2 = female, alternatively: 1 = male, 0 = female), and Case-control status (1 = controls, 2 = cases, alternatively: 0 = controls, 1 = cases).
Column 7 and onwards contain the genotype data, with alleles in separate columns. A "0" is used to denote missing data.

Value

A vector of the column numbers of the SNP identifiers/SNP numbers in the ped file, sorted in the same order as given in `snp.select`.

Note

The function does not check if the map file is formatted correctly or if the map and ped file have the same number of SNPs. The corresponding positions of the SNPs in the ped file may not be correct if the ped file has a different format from the given example.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Web Site: https://haplin.bitbucket.io

See Also

convertPed, lineByLine

Examples

```
## Not run:

# Find the column numbers of the SNP identifiers "RS9629043" and "RS12565286" in
# a standard ped file
snpPos(snp.select = c("RS9629043", "RS12565286"), map.file = "mygwas.map")
```

```
## End(Not run)
```

---

snpPower *Power calculations for a single SNP*

Description

Calculates power for a single SNP. Allows for power computations of several combinations simultaneously.

Usage

```
snpPower(cases, controls, RR, MAF, alpha = 0.05)
```

Arguments

cases	A list of the number of case families. Each element contains the number of families of a specified family design. The possible family designs, i.e. the possible names of the elements, are "mfc" (full triad), "mc" (mother-child dyad), "fc" (father-child dyad) or "c" (a single case child).
controls	A list of the number of control families. Each element contains the number of families of a specified family design. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother) or "f" (a single control father).
RR	A numeric vector of the relative risks (the effect sizes of interest).
MAF	A numeric vector of the minor allele frequencies.
alpha	A numeric vector of the Type I Errors. Equals 0.05 by default.

Details

snpPower computes power for a single SNP by counting the number of "real" case alleles, "real" control alleles and pseudo-control alleles. The pseudo-control alleles are the non-transmitted alleles, possibly from both case families and control families. It assumes a multiplicative dose-response model. snpPower uses the asymptotic normal approximation for the natural logarithm of the odds ratio for calculating power (the relative risks and odds ratios are used interchangeably due to the "rare disease assumption").

snpPower allows for power calculations for mixtures of the possible case family designs and control family designs. The argument cases could, for example, consist of a combination of 1000 full case triads (family design "mfc") and 500 single case children (family design "c"). It is also feasible to compute power for several combinations of the input variables simultaneously. See Examples for further details.

Value

snpPower returns a data frame containing the combinations of input variables and the corresponding power calculations.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Skare O, Jugessur A, Lie RT, Wilcox AJ, Murray JC, Lunde A, Nguyen TT, Gjessing HK. Application of a novel hybrid study design to explore gene-environment interactions in orofacial clefts. Annals of Human Genetics (2012) 76, pp. 221-236.

Web Site: https://haplin.bitbucket.io

See Also

haplin, snpSampleSize, hapRun, hapPower, hapPowerAsymp

Examples

```
## Compute power for a single SNP,
## for the combination of 1000 case triads, RR = 1.2, MAF = 0.1 and alpha = 0.05
snpPower(cases = list(mfc = 1000), controls = list(mfc = 0), RR = 1.2, MAF = 0.1)

## Compute power for a single SNP,
## for the combination of 1000 case triads and 500 single case children (altogether),
## 5000 control triads, RR = 1.1, MAF = 0.1 and alpha = 0.05
snpPower(cases = list(mfc = 1000, c = 500), controls = list(mfc=5000),
RR = 1.1, MAF = 0.1, alpha = 0.05)

## Compute power for a single SNP,
## for the combination of 500 case triads, 10000 control triads,
## relative risk of 1.2 and minor allele frequency of 0.1,
## and also for the combination of 1000 case triads, 10000 control triads,
## relative risk of 1.1 and minor allele frequency of 0.1
snpPower(cases = list(mfc = c(500, 1000)), controls = list(mfc = 10000),
RR = c(1.2, 1.1), MAF = 0.1)
```

snpSampleSize *Sample size calculations for a single SNP*

Description

Sample size calculations for a single SNP. Allows for sample size calculations of several combinations simultaneously.

Usage

```
snpSampleSize(fam.cases, fam.controls, fraction = 0.5,
RR, MAF, alpha = 0.05, power = 0.80)
```

Arguments

fam.cases	A character vector of the case family design. The possible family designs are "mfc" (full triad), "mc" (mother-child dyad), "fc" (father-child dyad) and "c" (a single case child).
fam.controls	A character vector of the control family design. The possible family designs are "mfc" (full triad), "mc" (mother-child-dyad), "fc" (father-child dyad), "mf" (mother-father dyad), "c" (a single control child), "m" (a single control mother), "f" (a single control father) or "no_controls" (no control families).
fraction	A numeric vector of the proportion of case families. Equals 0.5 by default, i.e. there are as many case families as control families. If fam.controls equals "no_controls", fraction is automatically set to 1.
RR	A numeric vector of the relative risks (the effect sizes of interest).
MAF	A numeric vector of the minor allele frequencies.
alpha	A numeric vector of the Type I Errors. Equals 0.05 by default.
power	A numeric vector of the desired probability of identifying a difference in the relative risks. Default is 0.80.

Details

snpSampleSize computes the number of case and control families required for a single SNP to attain the desired power. It assumes a multiplicative dose-response model. snpSampleSize calculates the fraction of case alleles corresponding to the given family designs and then uses the asymptotic normal approximation for the natural logarithm of the odds ratio for calculating the sample sizes (the relative risks and odds ratios are used interchangeably due to the "rare disease assumption").

snpSampleSize allows for sample size calculations of several combinations of the input variables at once. The Examples section provides further details.

Value

snpSampleSize returns a data frame containing the combinations of input variables and the corresponding sample size calculations.

Author(s)

Miriam Gjerdevik,
with Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health

<hakon.gjessing@uib.no>

References

Skare O, Jugessur A, Lie RT, Wilcox AJ, Murray JC, Lunde A, Nguyen TT, Gjessing HK. Application of a novel hybrid study design to explore gene-environment interactions in orofacial clefts. Annals of Human Genetics (2012) 76, pp. 221-236.

Web Site: https://haplin.bitbucket.io

See Also

haplin, snpPower

Examples

```
## Compute sample sizes for a single SNP,
## when the specified case family design is a full triad, there are no control families,
## RR = 1.1, MAF = 0.1, alpha = 0.05 and power = 0.9
snpSampleSize(fam.cases = "mfc", fam.controls = "no_controls",
RR = 1.1, MAF = 0.1, alpha = 0.05, power = 0.9)

## Compute sample sizes for a single SNP,
## for the combination of case triads, control triads, fraction = 0.5, RR = 1.2,
## MAF = 0.1, alpha = 0.05 and power = 0.8, and also for the combination of case triads,
## control children, fraction = 0.5, RR = 1.2, MAF = 0.1, alpha = 0.05 and power = 0.9
snpSampleSize(fam.cases = "mfc", fam.controls = c("mfc", "c"),
RR = 1.2, MAF = 0.1, alpha = 0.05, power = c(0.8, 0.9))
```

---

suest *Compute a joint p-value for a list of haplin fits (usually from a sliding window approach), correcting for multiple testing.*

Description

The first argument to suest should be a list of haplin estimation results (from the same data file), usually the output from haplinSlide. suest produces as a result a joint overall p-value based on aggregating the individual p-values and then correcting for multiple testing. The correction is achieved by using the principle of "seemingly unrelated" estimation, taking into account the correlation between the individual estimation results.

Usage

```
suest(reslist)
```

Arguments

reslist	A list whose elements are different haplin runs on the same data file, typically the output of haplinSlide.

Details

haplinSlide runs haplin on a series of overlapping windows of markers from the same data file, typically within the same gene. Since each run produces a separate overall p-value, suest computes a joint overall p-value for the gene (or region) that has been scanned. It corrects the overall p-value for multiple testing, also taking into account the fact that the sequence of estimates produced by haplinSlide will be dependent, both because they are computed on the same data set and also since the windows are overlapping (if the window length is larger than 1). If the suest estimation fails (which doesn't happen very often), a standard Bonferroni correction is used instead. Important: haplinSlide must be run with the option table.output = FALSE to provide suest with enough information.

Value

A list is returned, the most important elements of which are:

pval.obs	The overall score p-values from each haplin run
pval.obs.corr	The joint p-value, corrected for multiple testing
bonferroni	A logical, usually FALSE, which means the suest estimation went well. If TRUE, it means that the suest estimation failed for some reason, and a standard Bonferroni correction was used instead.

Note

Further information is found on the web page.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin, haplinSlide

Examples

```
## Not run:
# (Almost) all standard haplin runs can be done with haplinSlide.
# Below is an illustration. See the haplin help page for more
# examples.
#
# Analyzing the effect of fetal genes, including triads with missing data,
# using a multiplicative response model. When winlength = 1, separate
# markers are used. To make longer windows, winlength can be increased
# correspondingly:
result.1 <- haplinSlide("C:/work/data.dat", use.missing = T, response = "mult",
reference = "ref.cat", winlength = 1, table.output = F)
# Provide summary of separate results:
lapply(result.1, summary)
# Plot results:
par(ask = T)
lapply(result.1, plot)
# Compute an overall p-value for the scan, corrected for multiple testing
# and dependencies between windows:
suest(result.1)


## End(Not run)
```

---

summary.haplin *Summary of a haplin object*

Description

Provides detailed information about estimation results from a haplin object.

Usage

```
## S3 method for class 'haplin'
summary(object, reference, ...)
```

Arguments

object	A haplin object, i.e. the result of running haplin.
reference	Same as reference argument in haplin. Note that when producing the summary, you can only choose "reciprocal", "population" or "ref.cat". You cannot use a numeric value to change the reference category, to do that haplin must be run over again. (See the reference argument of haplin.)
...	Other arguments (ignored).

Note

Further information is found on the web page

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

See Also

haplin

Examples

```
## Not run:

# Produce separate plots for child and mother, dump plots to files:
res <- haplin("data.dat", use.missing = T, maternal = T)
summary(res)


## End(Not run)
```

---

toDataFrame *Stack dataframes from haplinSlide into a single dataframe*

Description

When haplinSlide is run with the option `table.output = T`, the result is a list of `haptables`, i.e. tables with summary haplin results for each window `haplinSlide` is run on. `toDataFrame` stacks the separate dataframes into one large dataframe containing all results.

Usage

```
toDataFrame(x, reduce = F)
```

Arguments

x	The output from haplinSlide run with option table.output = TRUE.
reduce	Reduce output to one line per marker

Details

When haplinSlide is run with winlength = 1 on SNP markers, each table in the output has only two rows, and can be condensed to a single row. By setting the argument reduce to TRUE, toDataFrame reduces each table to one line and returns a dataframe with one line for each SNP. In more general situations, with multi-allelic markers or, more commonly, winlength set to 2 or more, each output table will typically have more than two rows and cannot be reduced, so reduce should be set to FALSE.

Value

The output is a dataframe. First column contains the marker names. Second column are row numbers, counted within each output table. The remaining columns are identical to the individual output columns, which are described in more detail in the help file for haptable.

Author(s)

Hakon K. Gjessing
Professor of Biostatistics
Division of Epidemiology
Norwegian Institute of Public Health
<hakon.gjessing@uib.no>

References

Gjessing HK and Lie RT. Case-parent triads: Estimating single- and double-dose effects of fetal and maternal disease gene haplotypes. Annals of Human Genetics (2006) 70, pp. 382-396.

Web Site: https://haplin.bitbucket.io

Index