GeneticsBase

April 19, 2009

ALZH	*Sample from National Institute of Mental Health (NIMH) Genetics Initiative Alzheimer's Disease*

Description

Sample from National Institute of Mental Health (NIMH) Genetics Initiative Alzheimer's Disease

Usage

```
data(ALZH)
```

Format

Object of class 'geneSet'

Details

"This data set is a subsample from the National Institute of Mental Health (NIMH) Genetics Initiative Alzheimer's Disease (AD) Sample. The ascertainment and assessment of AD families collected have been discussed in Blacker et al. (1997). None of the families in this data set have parental genotype information; practically all of them have both affected and unaffected offspring. In total there are 901 individuals contained in 301 nuclear families.

"Acknowledgements: We thank Genetics and Aging Research Unit, Rudolph E. Tanzi, PhD, Director, for providing the data.

"Genotypes

"The pedigree file, Alzh.ped, contains genotype information for two candidate genes, apoe and a2m. The apoe gene is multi-allelic, while the a2m gene is bi-allelic. The dataset also contains the affection status (2=affected, 1=unaffected, 0=missing).

"Phenotypes

"We only will be using the Alzheimer dataset to examine affection status, which is contained in the pedigree file, thus a phenotype file for these data is not necessary."

(quoted from Lange and Kraft 2005)

Source

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

References

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

Examples

```
library(GeneticsBase)
data(ALZH)
head(ALZH)
```

`Armitage` *Cochran-Armitage test for linear trends in proportions and frequencies*

Description

Cochran-Armitage test for linear trends in proportions and frequencies.

Usage

```
Armitage(geneSetObj, method="A")
Armitage.default(pedObj, method="A")
ArmitageTest(x, mem)
```

Arguments

`geneSetObj`	a geneSet object
`pedObj`	a pedigree object
`method`	genotype coding method. The default is additive coding (`A`). The other two available coding methods are recessive coding (`R`) and dominant coding (`D`), respectively.
`x`	a vector of bialleleic markers coded by additive, recessive, or dominant model. Denote B as common allele and A as minor allele.
additive model:	x=0 – BB; x=1 – AB; x=2 – AA
recessive model:	x=0 – BB; x=0 – AB; x=2 – AA
dominant model:	x=0 – BB; x=1 – AB; x=1 – AA
`mem`	disease membership. `1` – case; `0` – control

Value

The functions `Armitage` and `Armitage.default` return a matrix with `nMarkers` rows and 2 columns, where `nMarkers` is the number of markers. The two columns are test statistic (`stat`) and p-value (`pvalue`), respectively.

The function `ArmitageTest` returns a list of two elements:

`stat`	test statistic
`pvalue`	p-value of the test

Note

This implementation is based on the documentation at webpage: http://linkage.rockefeller.edu/pawe3d/help/Linear-trend-test-ncp.html.

Author(s)

Gregory Warnes <warnes@bst.rochester.edu> Ross Lazarus <ross.lazarus@channing.harvard.edu> Weiliang Qiu <stwxq@channing.harvard.edu>

References

Gordon D, Haynes C, Blumenfeld J, Finch SJ (2005) PAWE-3D: visualizing Power for Association With Error in case/control genetic studies of complex traits. Bioinformatics 21:3935-3937.

Gordon D, Finch SJ, Nothnagel M, Ott J (2002) Power and sample size calculations for case-control genetic association tests when errors are present: application to single nucleotide polymorphisms. Hum Hered 54:22-33.

Chapman, D.G. and Nam, J.M. (1968) Asymptotic power of chi square tests for linear trends in proportions. Biometrics. 24, 315-327.

Armitage, P. (1955) Tests for linear trends in proportions and frequencies. Biometrics. 11, 375-386.

Cochran, W.G. (1954) Some methods for strengthening the common chi-squared tests. Biometrics. 10, 417-451.

Examples

```
# not significant result
ArmitageTest(x=c(2,1,1,1,0,0,1,0,0,1), mem=c(1,1,1,1,1,0,0,0,0,0))

# significant result
ArmitageTest(x=c(2,2,1,1,0,0,0,0,0,0), mem=c(1,1,1,1,1,0,0,0,0,0))
```

CAMP *Genotype data from the Childhood Asthma Management Program (CAMP)*

Description

"This dataset comprises a collection of parent/child trios in the Childhood Asthma Management Program (CAMP) Ancillary Genetics Study. The CAMP study is a clinical trial of asthmatic children (mild to moderate asthma) who were randomized to three different asthma treatments (CAMP, 1999). The data set includes 699 complete parent/child trios. Some participants are siblings, and there are 2011 persons from 640 nuclear families in total. Both quantitative and qualitative traits are available.

"Genotypes:

"The pedigree file, Camp.ped, contains genotype information for eight candidate genes (m709, m654, m47, p46, p79, p252, p491, p523). The last seven markers (all but m709) lie in a haplotype block. All the markers are bi-allelic.

"Phenotypes:

"The phenotype file, campz.phe, contains 12 quantitative traits (zposfevp, zposfvcp, zlogpc20, zampfmea, zpmpfmea, zbdabs, zfevbd, zbdpred, zsxcmean, ztoteos, zlogige, zncorpos). All of the phenotypes have been mean centered and standardized. The fev (forced expiatory volume) and fvc (forced vital capacity) variables refer to the amount of air that can be expelled from the lungs. The pc20 variable refers to the amount of irritant required to cause a 20% drop in fev. The toteos variable refers to total eosinophil (a white blood cell) count, and Ige is a measurement of allergen reactivity."

(quoted from Lange and Kraft 2005)

Usage

```
data(CAMP)
```

Format

Object to class 'geneSet'. Covariate information described above.

Source

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

References

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

DeMeo, D. L., C. Lange, et al. (2002). "Univariate and multivariate family-based association analysis of the IL-13 ARG130GLN polymorphism in the Childhood Asthma Management Program." Genet Epidemiol 23(4): 335-48.

Examples

```
library(GeneticsBase)
data(CAMP)
head(CAMP)
```

HWE *Test the significances of Hardy-Weinberg Equilibrium (dis)equilibrium) statistics*

Description

Test the significances of Hardy-Weinberg Equilibrium (dis)equilibrium) statistics for each marker in a data set.

Usage

```
HWE(object,
    test = c("exact", "permutation", "chisq"),
    B = 10000,
    R = 1000,
    correct = TRUE,
    conf = c(0.95),
    na.rm = TRUE,
```

```
    founderOnly = TRUE,
    ...)
```

Arguments

`object`	a geneSet object.
`test`	specifys the test method. Available methods are "exact", "permutation", "chisq".
`B`	an integer specifying the number of replicates used in the Monte Carlo test. Defaults to 10000.
`R`	Number of bootstrap iterations to use when computing the confidence interval. Defaults to 1000.
`correct`	see `diseq.ci`.
`conf`	Confidence level to use when computing the confidence level for linkage disequilibrium measures. Defaults to 0.95, should be in (0,1).
`na.rm`	logical. Should missing values be removed?
`founderOnly`	Indicates if only founders are used to do the test.
`...`	othere arguments used by the function `chisq.test`.

Value

The function `HWE` returns a list:

`\dQuote{diseq}`	a character string.
`call`	the matched call.
`\dQuote{D}`	a matrix with m rows and $3 + p$ columns, where m is the number of markers in the geneSet, p is the number of elements of the argument `conf`. The first column is the estimated "D". The next p columns are estimated confidence limits for the confidence levels specified in the argument `conf`. The last two columns are the sample size and the p-value of the test that "D" is equal to zero.
`\dQuote{D'}`	a matrix with m rows and $3 + p$ columns, where m is the number of markers in the geneSet, p is the number of elements of the argument `conf`. The first column is the estimated "D"'. The next p columns are estimated confidence limits for the confidence levels specified in the argument `conf`. The last two columns are the sample size and the p-value of the test that "D"' is equal to zero.
`\dQuote{r}`	a matrix with m rows and $3 + p$ columns, where m is the number of markers in the geneSet, p is the number of elements of the argument `conf`. The first column is the estimated "r". The next p columns are estimated confidence limits for the confidence levels specified in the argument `conf`. The last two columns are the sample size and the p-value of the test that "r" is equal to zero.
`\dQuote{X$^2$}`	a matrix with m rows and $3 + p$ columns, where m is the number of markers in the geneSet, p is the number of elements of the argument `conf`. The first column is the test statistic "X^2" for HWE test. The next p columns are zeros. The last two columns are the sample size and the p-value of the test for Hardy-Weinberg equilibrium.

Author(s)

Gregory R. Warnes ⟨gregory.r.warnes@pfizer.com⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
library(GeneticsBase)
data(CAMP)

HWE(CAMP)
```

`HWE.chisq` *Hardy-Weinberg Equilibrium Significance test for a biallelic locus*

Description

Hardy-Weinberg Equilibrium Significance test for a biallelic locus .

Usage

```
HWE.chisq(object,
          marker,
          simulate.p.value = TRUE,
          B = 10000,
          founderOnly=TRUE,
          ...)
```

Arguments

`object`	a geneSet object.
`marker`	marker name for the biallelic locus.
`simulate.p.value`	a logical indicating whether to compute p-values by Monte Carlo simulation.
`B`	an integer specifying the number of replicates used in the Monte Carlo test. Defaults to 10000.
`founderOnly`	Indicates if only founders are used to do the test.
`...`	Othere arguments used by the function `chisq.test`.

Value

The values of the function `HWE.chisq` is the same as those of the function `chisq.test`. Part of the values are listed below:

`statistic`	the value the chi-squared test statistic.
`parameter`	the degrees of freedom of the approximate chi-squared distribution of the test statistic, 'NA' if the p-value is computed by Monte Carlo simulation.
`p.value`	the p-value for the test.

Author(s)

Gregory R. Warnes ⟨gregory.r.warnes@pfizer.com⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

See Also

`HWE.exact`

Examples

```
library(GeneticsBase)
data(CAMP)

HWE.chisq(CAMP, marker="m654")
```

`HWE.exact` *Exact test for Hardy-Weinberg Equilibrium for a biallelic locus*

Description

Exact test for Hardy-Weinberg Equilibrium for a biallelic locus.

Usage

```
HWE.exact(object,
          marker,
          founderOnly=TRUE)
```

Arguments

`object`	a geneSet object.
`marker`	marker name for the biallelic locus.
`founderOnly`	Indicates if only founders are used to do the test.

Value

The function `HWE` returns a list with class `htest` containing the following elements:

`statistic`	A 3-element vector records the genotype frequencies: `N11`, `N12`, `N22`.
`parameter`	A 2-element vector records the allele frequencies: `N1` and `N2`
`p.value`	p-value of the test.
`method`	Information indicates if `chisquare` method or `exact` method is used to do HWE test.
`data.name`	name of the data set
`observed`	a table lists allele levels, allele pairs, and allele codes

Note

This function only works for genotypes with exactly 2 alleles.

Author(s)

David Duffy davidD@qimr.edu.au with modifications by Gregory R. Warnes, and Nitin Jain

References

Emigh TH. (1980) "Comparison of tests for Hardy-Weinberg Equilibrium", Biometrics, 36, 627-642.

See Also

HWE.chisq

Examples

```
library(GeneticsBase)
data(CAMP)

HWE.exact(CAMP, marker="m654")
```

LD-class *Class "LD" ~~~*

Description

~~ A concise (1-5 lines) description of what the class is. ~~

Objects from the Class

Objects can be created by calls of the form `new("LD", ...)`. ~~ describe objects here ~~

Slots

`call`: Function call used to generate this object.

`D`: Linkage disequilibrium estimate

`D'`: Scaled linkage disequilibrium estimate

`r`: Correlation coefficient

`R^2`: squared correlation coefficient

`n`: Number of observations

`X^2`: Chi-square statistic for linkage equilibrium (i.e., D=D'=corr=0)

`P-value`: Chi-square p-value for marker independence

`LOD`: LOD scores

Methods

head `signature(x = "LD"): ...`

html `signature(x = "LD"): ...`

latex `signature(x = "LD"): ...`

left `signature(x = "LD"): ...`

plot `signature(x = "LD"): ...`

right `signature(x = "LD"): ...`

show `signature(object = "LD"): ...`

tail `signature(x = "LD"): ...`

Note

~~further notes~~

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
##---- Should be DIRECTLY executable !! ----
```

`LD` *Pairwise linkage disequilibrium between genetic markers.*

Description

Compute pairwise linkage disequilibrium between genetic markers

Usage

```
LD(object, pooling.threshold = 0.1, founderOnly = FALSE, quiet = TRUE )
LDband(object, width=31,pooling.threshold=0.1, founderOnly=FALSE, quiet=TRUE)
LDband.geneSet(object,width=31,pooling.threshold=0.1, founderOnly=FALSE, quiet=T
LDdist.geneSet(object,posVec, maxDist, pooling.threshold=0.1, founderOnly=FALSE,

summary.LD(object,
           which = c("D", "D'", "r", "R^2", "X^2", "P-value", "LOD", "n"),
           rowsep,
           digits = 3, ...)
```

Arguments

`object`	geneSet object
`width`	window width
`posVec`	marker position
`maxDist`	size of the window based on distance
`pooling.threshold`	Threshold for LD calculation
`which`	character string indicates which LD statistic should be print out
`rowsep`	separator for rows
`digits`	the desired number of digits after the decimal point
`founderOnly`	Indicates if only founders are used to do the test.
`quiet`	Indicates if intermediate results should be output
`...`	additional optional arguments

Details

Linkage disequilibrium (LD) is the non-random association of marker alleles and can arise from marker proximity or from selection bias.

`LD` estimates the extent of LD for all pairs of genotypes contained in a `object`. `LDband` computes the extent of LD of markers within a window containing `width` markers centered around each marker in `object`.

The current (temporary) code only computes LD for markers with exactly 2 variants. For other markers, `NA` is returned.

Three estimators of LD are computed:

D raw difference in frequency between the observed number of AB pairs and the expected number:

$$D = p_{AB} - p_A p_B$$

D' scaled D spanning the range [-1,1]

$$D' = \frac{D}{D_{max}}$$

where, if D > 0:

$$D_{max} = \min(p_A p_b, p_a p_B)$$

or if D < 0:

$$D_{max} = \max -p_A p_B, -p_a p_b$$

r correlation coefficient between the markers

$$r = \frac{-D}{\sqrt{(p_A * p_a * p_B * p_b)}}$$

where

- p_A is defined as the observed probability of allele 'A' for marker 1,
- $p_a = 1 - p_A$ is defined as the observed probability of allele 'a' for marker 1,
- p_B is defined as the observed probability of allele 'B' for marker 2, and
- $p_b = 1 - p_B$ is defined as the observed probability of allele 'b' for marker 2, and
- p_{AB} is defined as the probability of the marker allele pair 'AB'.

For genotype data, AB/ab cannot be distinguished from aB/Ab. Consequently, we estimate p_{AB} using maximum likelihood and use this value in the computations.

Value

`LD` returns an object of class `LD`, while `LDband` and `LDdist` return objects of classes `LDband` and `LDdist`, respectively. All classes contain these slots:

`call`	the matched call
`D`	Linkage disequilibrium estimate
`Dprime`	Scaled linkage disequilibrium estimate
`corr`	Correlation coefficient
`nobs`	Number of observations

chisq	Chi-square statistic for linkage equilibrium (i.e., D=D'=corr=0)
p.value	Chi-square p-value for marker independence
LOD	LOD scores
tab	Description of 'tab'
statlist	Description of 'statlist'
which	Description of 'which'
object	Description of 'object'

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩

See Also

geneSet-class, diseq

Examples

```
data(CAMP)

ld <- LD(CAMP)
print(ld)

ldb <- LDband(CAMP)
print(ldb)

ldd <- LDdist(CAMP, posVec=1:8, maxDist=3)
print(ldd)
```

LDband-class *Class "LDband" ~~~*

Description

~~ A concise (1-5 lines) description of what the class is. ~~

Objects from the Class

Objects can be created by calls of the form `new("LDband", ...)`. ~~ describe objects here ~~

Slots

call: Function call used to generate this object.

D: Linkage disequilibrium estimate

D': Scaled linkage disequilibrium estimate

r: Correlation coefficient

R^2: squared correlation coefficient

n: Number of observations

X^2: Chi-square statistic for linkage equilibrium (i.e., D=D'=corr=0)

P-value: Chi-square p-value for marker independence

LOD: LOD scores

Methods

head `signature(x = "LDband")`: ...

left `signature(x = "LDband")`: ...

right `signature(x = "LDband")`: ...

show `signature(object = "LDband")`: ...

tail `signature(x = "LDband")`: ...

Note

~~further notes~~

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
##---- Should be DIRECTLY executable !! ----
```

LDdist-class *Wrapper to efficiently store the result from performing LD calculatations over sliding windows containing markers within a distance*

Description

Wrapper to efficiently store the result from performing LD calculatations over sliding windows containing markers within a distance

Slots

call: Function call used to generate this object.

D: Linkage disequilibrium estimate

D': Scaled linkage disequilibrium estimate

r: Correlation coefficient

R^2: squared correlation coefficient

n: Number of observations

X^2: Chi-square statistic for linkage equilibrium (i.e., D=D'=corr=0)

P-value: Chi-square p-value for marker independence

LOD: LOD scores

Methods

head `signature(x = "LDdist")`: see `head`

left `signature(x = "LDdist")`: see `left`

right `signature(x = "LDdist")`: see `right`

show `signature(object = "LDdist")`: see `show`

tail `signature(x = "LDdist")`: see `tail`

Author(s)

Initial version by Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩, enhanced to use C routines by Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, and Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

`PGtables` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
PGtables(x, filename = "", sep="_", format = c("print", "html", "latex"), ...)
```

Arguments

`x`	~~Describe `x` here~~
`filename`	~~Describe `filename` here~~
`sep`	Character used in constructing file names from the prefix provide by `filename` and the table name generated internally.
`format`	~~Describe `format` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"PGtables"
```

PerlegenExample *Small example data set from Perlegen*

Description

Small example data set from Perlegen.

Usage

```
data(PerlegenExample)
```

Format

Object of class 'geneSet'.

Details

This data set is useful for testing, but the locus id's have been scrambled, and no covariate information is provided. As a consequence, the data is meaningless.

Examples

```
library(GeneticsBase)
data(PerlegenExample)
head(PerlegenExample)
```

PfizerExample *Small example data set from Pfizer*

Description

Small example data set from Pfizer.

Usage

```
data(PfizerExample)
```

Examples

```
library(GeneticsBase)
data(PfizerExample)
head(PfizerExample)
```

alleleCount *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
alleleCount(object, ...)
```

Arguments

object	~~Describe object here~~
...	~~Describe ... here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"alleleCount"
```

`alleleLevels` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
alleleLevels(object, ...)
```

Arguments

`object`	~~Describe `object` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"alleleLevels"
```

---

alleleSummary *Summary of allele information*

Description

Summary of allele information.

Usage

```
alleleSummary(object,
              by = NULL,
              confidence = 0.95,
              alpha = 1 - confidence,
              show = TRUE,
              verbose = FALSE,
              includeOverall = FALSE,
              omitRepeats = TRUE,
              ...)
```

Arguments

object	geneSet object
by	optional column name, by which the summary is desired. Default is NULL.
confidence	confidence intervals of Genotype frequencies within each marker (default is 95%)
alpha	Type -1 error rate = (1- confidence)
show	No longer used
verbose	Logical value (TRUE/FALSE), showing whether every 50th marker should be printed, default = FALSE

`includeOverall`
: logical value (TRUE/FALSE) indicating whether overall summary is also needed, default = FALSE

`omitRepeats`
: logical value (TRUE/FALSE) indicating whether Gene name and marker name should be printed repeatedly for each Genotype, default = TRUE

`...`
: Optional arguments

Author(s)

Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
library(GeneticsBase)
data(CAMP)

temp <- alleleSummary(CAMP)

print(temp) # display
txt(temp, filename="alleleSummary.txt")    # txt file
html(temp, filename="alleleSummary.html") # html file
latex(temp, filename="alleleSummary.tex") # latex file
```

---

alleles *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
alleles(object, ...)
```

Arguments

`object`
: ~~Describe `object` here~~

`...`
: ~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`
: Description of 'comp1'

`comp2`
: Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"alleles"
```

`as.geneSet` *Convert an existing object to a geneSet*

Description

Convert an existing object to a geneSet. Methods currenrly exist only for matrix and data.frame objects.

Usage

```
## S3 method for class 'matrix':
as.geneSet(x, ... )
## S3 method for class 'data.frame':
as.geneSet(x,
  gene.columns,
  format=c("single", "adjacent"),
  ploidy=2,
  alleles=NULL,
  sep="/",
  remove.spaces=TRUE,
  reorder=c("freq", "yes", "no", "default", "ascii"),
  allow.partial.missing=FALSE,
  markerNames,
  phase=list(F),
  ...)
```

Arguments

x	The matrix or data.frame object to be converted
gene.columns	Names or indexes of columns containing genotypes
format	One of "single", indiciating that each specified column contains a complete genotype (e.g. "A/C"), or "adjacent" indicating that sets of ploidy adjacent columns each contain as single allele (e.g. "A", "C")
ploidy	Number of allele copies per genotype. Defaults to 2
alleles	Not currently supported. In the future, this variable will allow specification of the allele strings for each genotype.
sep	A character value or a numeric index vector indicating how allels are spearated within genotypes, defaults to "/". If a character, it indicates that alleles within a genotype are separated by this character (e.g. for "A/C", sep="/"). If a numeric vector (of length ploidy-1), the value(s) indicate which positions separate allele names (e.g. for "AA", sep=1).
remove.spaces	Should whitespace be removed before processing.
reorder	One of "freq", "yes", "no", "default", or "ascii", indicating whether and how alleles should be reorderd within genotypes. If reorder="no", the observed order is preserved (important when phase is known). If reorder="freq", sort alleles within each individual by observed frequency. If reorder="yes", sort alleles in the order provided by the alleles argument. If reorder="ascii", reorder alleles in ASCII order (alphabetical, with all upper case before lower case). The default value is "freq".
allow.partial.missing	Logical value indicating whether a genotype is permitted to be partially missing. When allow.partial.missing=FALSE, if any individual allele is missing within a genotype, the entire genotype will be converted to a missing value. When allow.partial.missing=TRUE, the missingness of individual alleles will be preserved.
markerNames	Character vector of names to use for the genotype columns. This must have the same lenght as the number of genotype columns.
phase	List indicating whether phase is known for each genotype column. If the list has a single logical entry, this will apply to all genotype columns.
...	Optional arguments.

Value

An S4 object of class geneSet

Author(s)

Gregory R. Warnes ⟨greg@random-technologies-llc.com⟩

See Also

geneSet

Examples

```
## Create a test data set where there are several genotypes in columns
## of the form "A/T".
test1 <- data.frame(Tmt=sample(c("Control","Trt1","Trt2"),20, replace=TRUE),
                G1=sample(c("A/T","T/T","T/A",NA),20, replace=TRUE),
                N1=rnorm(20),
                I1=sample(1:100,20,replace=TRUE),
                G2=paste(sample(c("134","138","140","142","146"),20,
                                replace=TRUE),
                         sample(c("134","138","140","142","146"),20,
                                replace=TRUE),
                         sep=" / "),
                G3=sample(c("A /T","T /T","T /A"),20, replace=TRUE),
                comment=sample(c("Possible Bad Data/Lab Error",""),20,
                               rep=TRUE)
                )
test1

## now automatically convert genotype columns
geno1 <- as.geneSet(test1)
geno1
```

`binsearch` *Binary Search*

Description

Search within a specified range to locate an integer parameter which results in the the specified monotonic function obtaining a given value.

Usage

```
binsearch(fun, range, ..., target = 0, lower = ceiling(min(range)),
          upper = floor(max(range)), maxiter = 100, showiter = FALSE)
```

Arguments

`fun`	Monotonic function over which the search will be performed.
`range`	2-element vector giving the range for the search.
`...`	Additional parameters to the function `fun`.
`target`	Target value for `fun`. Defaults to 0.
`lower`	Lower limit of search range. Defaults to `min(range)`.
`upper`	Upper limit of search range. Defaults to `max(range)`.
`maxiter`	Maximum number of search iterations. Defaults to 100.
`showiter`	Boolean flag indicating whether the algorithm state should be printed at each iteration. Defaults to FALSE.

Details

This function implements an extension to the standard binary search algorithm for searching a sorted list. The algorithm has been extended to cope with cases where an exact match is not possible, to detect whether that the function may be monotonic increasing or decreasing and act appropriately, and to detect when the target value is outside the specified range.

The algorithm initializes two variable `lo` and `high` to the extremes values of `range`. It then generates a new value `center` halfway between `lo` and `hi`. If the value of `fun` at `center` exceeds `target`, it becomes the new value for `lo`, otherwise it becomes the new value for `hi`. This process is iterated until `lo` and `hi` are adjacent. If the function at one or the other equals the target, this value is returned, otherwise `lo`, `hi`, and the function value at both are returned.

Note that when the specified target value falls between integers, the *two closest values are returned. If the specified target falls outside of the specified `range`, the closest endpoint of the range will be returned, and an warning message will be generated. If the maximum number if iterations was reached, the endpoints of the current subset of the range under consideration will be returned.*

Value

A list containing:

`call`	How the function was called.
`numiter`	The number of iterations performed
`flag`	One of the strings, "Found", "Between Elements", "Maximum number of iterations reached", "Reached lower boundary", or "Reached upper boundary."
`where`	One or two values indicating where the search terminated.
`value`	Value of the function `fun` at the values of `where`.

Note

This function often returns two values for `where` and `value`. Be sure to check the `flag` parameter to see what these values mean.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩

See Also

`optim`, `optimize`, `uniroot`

Examples

```
### Toy examples

# search for x=10
binsearch( function(x) x-10, range=c(0,20) )

# search for x=10.1
binsearch( function(x) x-10.1, range=c(0,20) )

### Classical toy example
```

```
# binary search for the index of 'M' among the sorted letters
fun <- function(X) ifelse(LETTERS[X] > 'M', 1,
                          ifelse(LETTERS[X] < 'M', -1, 0 ) )

binsearch( fun, range=1:26 )
# returns $where=13
LETTERS[13]

### Substantive example, from genetics

# Determine the necessary sample size to detect all alleles with
# frequency 0.07 or greater with probability 0.95.
power.fun <- function(N) 1 - gregorius(N=N, freq=0.07)$missprob

binsearch( power.fun, range=c(0,100), target=0.95 )

# equivalent to
gregorius( freq=0.07, missprob=0.05)
```

`callCodes` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
callCodes(object, marker)
callCodes(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`marker`	~~Describe `marker` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"callCodes"
```

`ci.balance` *Experimental Function to Correct Confidence Intervals At or Near Boundaries of the Parameter Space by 'Sliding' the Interval on the Quantile Scale.*

Description

Experimental function to correct confidence intervals at or near boundaries of the parameter space by 'sliding' the interval on the quantile scale.

Usage

```
ci.balance(x, est, confidence=0.95, alpha=1-confidence, minval, maxval,
           na.rm=TRUE)
```

Arguments

`x`	Bootstrap parameter estimates.
`est`	Observed value of the parameter.
`confidence`	Confidence level for the interval. Defaults to 0.95.
`alpha`	Type I error rate (size) for the interval. Defaults to 1-`confidence`.
`minval`	A numeric value specifying the lower bound of the parameter space. Leave unspecified (the default) if there is no lower bound.
`maxval`	A numeric value specifying the upper bound of the parameter space. Leave unspecified (the default) if there is no upper bound.
`na.rm`	logical. Should missing values be removed?

Details

EXPERIMENTAL FUNCTION:

This function attempts to compute a proper `conf`*100% confidence interval for parameters at or near the boundary of the parameter space using bootstrapped parameter estimates by 'sliding' the confidence interval on the quantile scale.

This is accomplished by attempting to place a `conf` *100% interval symmetrically *on the quantile scale* about the observed value. If a symmetric interval would exceed the observed data at the upper (lower) end, a one-sided interval is computed with the upper (lower) boundary fixed at the the upper (lower) boundary of the parameter space.

Value

A list containing:

`ci` A 2-element vector containing the lower and upper confidence limits. The names of the elements of the vector give the actual quantile values used for the interval or one of the character strings "Upper Boundary" or "Lower Boundary".

`overflow.upper, overflow.lower`
The number of elements beyond those observed that would be needed to compute a symmetric (on the quantile scale) confidence interval.

`n.above, n.below`
The number of bootstrap values which are above (below) the observed value.

`lower.n, upper.n`
The index of the value used for the endpoint of the confidence interval or the character string "Upper Boundary" ("Lower Boundary").

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu ⟩

See Also

`boot`, `bootstrap`, Used by `diseq.ci`.

Examples

```
# These are nonsensical examples which simply exercise the
# computation. See the code to diseq.ci for a real example.
#
# FIXME: Add real example using boot or bootstrap.

set.seed(7981357)
x <- abs(rnorm(100,1))
ci.balance(x,1, minval=0)
ci.balance(x,1)

x <- rnorm(100,1)
x <- ifelse(x>1, 1, x)
ci.balance(x,1, maxval=1)
ci.balance(x,1)
```

convert *Efficienctly convert strings of characters into integer codes*

Description

Efficienctly convert strings of characters into integer codes.

Usage

```
convert(source, levels, byrow=FALSE, aslist=FALSE)
```

Arguments

source	Vector of character strings
levels	Vector of characters used to determine levels
byrow	Boolean. If FALSE (the default), return a matrix with one column per string. If TRUE, return a matrix with one row per string.
aslist	Boolean, return matrix (FALSE) or list of vectors (TRUE).

Details

This function efficiently converts character strings containing characters into vectors of integers. Its primary purpose is to allow translation of genotypes stored as character vectors, one character per genotype, to a factor-coded matrix. The equivalent code using `factor` is quite a bit slower, as shown by the last section of the example below.

The `levels` argument should be a vector of 1-character strings. This vector is used to determine the translation. The index of matching characters provides the returned integer values. Characters not present in `levels` will be converted to NA's.

Value

If `aslist=TRUE`, the return value is a a list of vectors. Each vector will contain the translation of the corresponding input string.

If `aslist=FALSE (the default)`, the return value will be a matrix. `byrow` controls whether each string is converted into a a column (`byrow=FALSE`, the default) or row (`byrow=TRUE`).

When `byrow=FALSE`, each element of the `source` vector is converted to a column, and the number of rows will be the number of characters in the longest element of the `source` vector. Any shorter vectors will be padded with NA's.

When `byrow=TRUE` the matrix is created with one row per element of the `source` vector, etc.

Note

Only of the first character of each element of `levels` is used. Any other characters will be ignored.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

See Also

factor, as.factor

Examples

```
###
# Toy Genetics Example
##
# 'c' = 'homozygote common allele'
# 'h' = 'heterozygone'
# 'r' = 'homozygote rare allele'
marker.data <- c( m1='cchchrcr', m2='chccccrr')
marker.data

convert(marker.data, c('c','h','r'))

###
# simple test example
###
source <- c(one='abcabcabc', two='abc','ggg',buckle='aaa',my='bbb',
            'shoe  '='bgb  ')
levels <- c('a','b','c','d')

convert(source,levels)
convert(source,levels,aslist=TRUE)
convert(source,levels,byrow=TRUE)

###
# compare efficiency with equivalent code using 'factor'
###
## Not run:
makestr <- function(n)
  paste(sample(letters, size=n, replace=T), sep='', collapse='')

timeit <- function( expr )
  {
    start <- Sys.time()
    expr
    end <- Sys.time()
    return( as.numeric(end-start ))
  }

# Step 1: create a large set of character strings
x <- unlist(lapply(1:100000, function(x) makestr(1000)))

# Step 2: Time convert  (~17 sec on Intel Xeon 3.0 GHz, 32 GB RAM)
newtime <- timeit( yn <- convert2(x, letters) )
newtime

# old method  (~4.7 min on Intex Xeon 3.0 GHz, 32 GB RAM)
oldmethod <- function(x)
  {
    yo <- factor(unlist(strsplit(x, split='')),levels=letters)
    attr(y1,'dim') <- c(nchar(x[1]), length(x))
    class(y1) <- 'matrix'
  }
```

```
oldtime <- timeit( oldmethod(x) )
oldtime

# time difference
oldtime - newtime
## End(Not run)
```

`decodeCallCodes` *Converts integer codes in a callCodes matrix to character string representations of geneSet.*

Description

Converts integer codes in a callCodes matrix to character string representations of geneSet.

Usage

```
decodeCallCodes(callCodes, transTables, markerInfo)
```

Arguments

`callCodes`	matrix of positive integers, such as the callCodes slot of a geneSet object.
`transTables`	list of code translation tables, such as the transTables slot of a geneSet object.
`markerInfo`	dataframe, with one row for each row in the callCodes matrix, such as the markerInfo slot of a geneSet object. Must include at least column "TransTable".

Value

a character matrix with the same dimension and dimnames as argument callCodes, giving character string representations of genotypes, as specified by a set of translation tables.

Author(s)

Scott D. Chasalow

See Also

geneSet

Examples

```
library(GeneticsBase)
tTabList <- makeTransTableList(c("Bb", "ATG"), c("Generic", "SNP"))
locus <- c("c104t", "c2249t")
ttab <- c("SNP", "Generic")
minfo <- makeMarkerInfo(locus, ttab)
misscodes <- list(Generic = c("NA", ".", "-"), SNP = "N")
r1 <- c(1, 2, 1, 1)
r2 <- c(1, 2, 3, 2)
data <- rbind(r1, r2) # callCodes
dimnames(data) <- list(locus, paste("Sample", seq(along = r1),
```

```
        sep = "."))
decodeCallCodes(data, tTabList, minfo)
```

`desMarkers` *Descriptive statistics for markers*

Description

Descriptive statistics for markers.

Usage

```
desMarkers(geneSetObj,
           founderOnly=TRUE,
           thrsh=0.05,
           HWE.method=c("simulate","exact"),
           simulate.p.value=FALSE,
           B=10000,
           markerThrsh=100,
           maxDist=5,
           LDmeasure="r2",
           plot=TRUE,
           ...)
```

Arguments

`geneSetObj`	a `geneSet` object.
`founderOnly`	indicates if using only founder info
`thrsh`	threshold for Hardy-Weinberg equilibrium test. If the pvalue of the HWE test for a marker is greater than `thrsh`, then the marker is a good marker.
`HWE.method`	method to do Hardy-Weinberg equilibrium test.
`simulate.p.value`	indicates if the pvalue of the HWE test is calculated by Monte Carlo simulation. `simulate.p.value=FALSE` means the pvalue is calculated from asymptotic chi-squared distribution of the test statistic. Otherwise, Monte Carlo simulation is used to calculate pvalue. For more details, please refers to the `R` function `chisq.test`.
`B`	the number of replicates used in Monte Carlo simulation to get the pvalue of HWE test. For more details, please refers to the `R` function `chisq.test`.
`markerThrsh`	if the number of markers is greater than this threshold, then 'LDdist' is called instead of 'LD'.
`maxDist`	the width of window used in 'LDdist' function.
`LDmeasure`	indicates if `r^2` or `D'` is to be plot.
`plot`	indicates if LD plot is output or not.
`...`	Other arguments that are used by HWE.chisq or HWE.exact.

Value

a data frame contains components:

`Name`	marker names
`Position`	marker positions
`ObsHET`	marker's observed heterozygosity (i.e., proportion of heterozygotes at markes). Missing alleles are excluded in the calculation.
`PredHET`	marker's predicted heterozygosity (i.e., `2*MAF*(1-MAF)`). Missing alleles are excluded in the calculation.
`HWpval`	pvalues for Hardy-Weinberg test
`pGeno`	percentage of non-missing genotypes for markes
`MAF`	minor allele frequencies. missing allele are excluded from calculation
`Rating`	`Rating[i]=1` means that the i-th marker passes HW test (do not reject H0 that HW equilibrium holds). `Rating[i]=0` means HW equilibrum does hold for the i-th marker.

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

Examples

```
#data(CAMP)
#res<-desMarkers(CAMP)
#print(res)
```

`description` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
description(object)
description(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as `help`, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"description"
```

`diseq` *Estimate or Compute Confidence Interval for the Single-Marker Disequilibrium*

Description

Estimate or compute confidence interval for single-marker disequilibrium.

Usage

```
diseq.ci(object, marker, R = 1000, conf = 0.95, correct = TRUE, na.rm =
TRUE, ...)
diseq.inner(object, marker, ...)
```

Arguments

object	geneSet object
marker	marker names
R	Number of bootstrap iterations to use when computing the confidence interval. Defaults to 1000.
conf	Confidence level to use when computing the confidence level for D-hat. Defaults to 0.95, should be in (0,1).
correct	See details.
na.rm	logical. Should missing values be removed?
...	optional additional parameters passed

Details

For a single-gene marker, diseq computes the Hardy-Weinberg (dis)equilibrium statistic D, D', r (the correlation coefficient), and r^2 for each pair of allele values, as well as an overall summary value for each measure across all alleles. print.diseq displays the contents of a diseq object. diseq.ci computes a bootstrap confidence interval for this estimate.

For consistency, I have applied the standard definitions for D, D', and r from the Linkage Disequilibrium case, replacing all marker probabilities with the appropriate allele probabilities.

Thus, for each allele pair,

D is defined as the half of the raw difference in frequency between the observed number of heterozygotes and the expected number:

$$D = \frac{1}{2}(p_{ij} + p_{ji}) - p_i p_j$$

D' rescales D to span the range [-1,1]

$$D' = \frac{D}{D_{max}}$$

where, if D > 0:

$$D_{max} = \min p_i p_j, p_j p_i = p_i p_j$$

or if D < 0:

$$D_{max} = \min p_i(1 - p_j), p_j(1 - p_i)$$

r is the correlation coefficient between two alleles, and can be computed by

$$r = \frac{-D}{\sqrt{(p_i * (1 - p_i)p(j)(1 - p_j))}}$$

where

- p_i defined as the observed probability of allele 'i',
- p_j defined as the observed probability of allele 'j', and
- p_{ij} defined as the observed probability of the allele pair 'ij'.

When there are more than two alleles, the summary values for these statistics are obtained by computing a weighted average of the absolute value of each allele pair, where the weight is determined by the expected frequency. For example:

$$D_{overall} = \sum_{i \neq j} |D_{ij}| * p_{ij}$$

Bootstrapping is used to generate confidence interval in order to avoid reliance on parametric assumptions, which will not hold for alleles with low frequencies (e.g. D' following a a Chi-square distribution).

See the function HWE from "genetics" package for testing Hardy-Weinberg Equilibrium, $D = 0$.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

`errorMetrics` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
errorMetrics(object)
errorMetrics(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"errorMetrics"
```

---

extractAlleles *functions for extracting allele levels, allele pairs, and allele codes*

Description

functions for extracting allele levels, allele pairs, and allele codes

Usage

```
extractAlleles(object, which = c(1, 2), codes = FALSE,
allow.partial.missing = FALSE, marker, obs)
```

Arguments

object	geneSet object
which	allele pair number - 1 or 2. Default if both
codes	If FALSE (default), then calleles are shown as characters, else numeric
allow.partial.missing	Whether partical matching should be allowed (default is FALSE)
marker	marker names
obs	Number of observations (samples) to be displayed. Default is all.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

fastGrid *Create a matrix giving all combinations of the elements of x*

Description

Create a matrix giving all combinations of the elements of x

Usage

```
fastGrid(x)
```

Arguments

x	a vector

Value

a matrix of the same mode as x, with dimension c(length(x)\textasciicircum2, 2). The rows give all points on a square lattice formed by pairing every element of x with every element of x. Here, order matters - that is, (1, 2) and (2, 1) both would be included - and points with an element taken twice - e.g. (1, 1) - also are included.

Author(s)

Scott Chasalow ⟨Scott.Chasalow@bms.com⟩

See Also

geneSet-class

founderGeneSet *Extract founder info from a geneSet object*

Description

Extract founder info from a geneSet object.

Usage

```
founderGeneSet(object)
```

Arguments

object	a geneSet object

Value

a geneSet object containing only founder info.

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

Examples

```
data(CAMP)
founders<-founderGeneSet(CAMP)
```

---

geneSet-class *Class "geneSet", a class for genetics data*

Description

A fundamental data structure for genetic data

Objects from the Class

Objects can be created by calls of the form `new("geneSet", ...)`. ~~ describe objects here ~~

Slots

callCodes: matrix of positive integers, giving genotype calls. Each row is a locus (marker); each column is an individual (sample). Each element is a row index into a matrix in the list of translation tables stored in the transTables slot. Must have row and column names.

errorMetrics: numeric matrix, parallel to the callCodes matrix. Each element gives an uncertainty measure for the corresponding element of the callCodes matrix. Must have row and column names.

transTables: list of code translation tables. The list must have names. Each component is a matrix, and must include a column named "levels".

missingCodes: list of allele missing-value codes, parallel to the transTables list. The list must have the same names as the list in the transTables slot. Each component is a character vector. Any allele symbol in component "abc" of the transTables list that appears in component "abc" of the missingCodes list is to be interpreted as a missing value by functions operating on the geneSet object. An empty list will be interpreted to mean that the data contains no missing values.

sampleInfo: Object of class `"data.frame"`

markerInfo: a dataframe, with one row for each row in the callCodes matrix. Must include columns "Name" and "TransTable".

studyInfo: Object of class `"list"` ~~

description: Object of class `"character"` ~~

notes: Object of class `"character"` ~~

ploidy: Object of class `"numeric"` ~~

phase: 1. Yes/No for all (logical scalar) 2. Yes/No for each Marker (logical vector) 3. phaseObject (TBD): observation by marker by phase probabilities + definitions of contigs + probability of contigs

Methods

HWE `signature(object = "geneSet")`: Hardy-Weinberg Equilibrium Significance Test

LD `signature(object = "geneSet")`: ...

LDband `signature(object = "geneSet")`: ...

LDdist `signature(object = "geneSet")`: ...

[`signature(x = "geneSet")`: ...

[[`signature(x = "geneSet")`: ...

alleleCount `signature(object = "geneSet")`: ...

alleleLevels `signature(object = "geneSet")`: ...

alleles `signature(object = "geneSet")`: ...

callCodes `signature(object = "geneSet")`: ...

callCodes<- `signature(object = "geneSet")`: ...

carrier `signature(object = "geneSet")`: ...

description `signature(object = "geneSet")`: ...

description<- `signature(object = "geneSet")`: ...

dominant `signature(object = "geneSet")`: ...

errorMetrics `signature(object = "geneSet")`: ...

errorMetrics<- `signature(object = "geneSet")`: ...

genotypeLevels `signature(object = "geneSet")`: ...

genotypes `signature(object = "geneSet")`: ...

head `signature(x = "geneSet")`: ...

heterozygote `signature(object = "geneSet")`: ...

homozygote `signature(object = "geneSet")`: ...

markerInfo `signature(object = "geneSet")`: ...

markerInfo<- `signature(object = "geneSet")`: ...

markerNames `signature(object = "geneSet")`: ...

missingCodes `signature(object = "geneSet")`: ...

missingCodes<- `signature(object = "geneSet")`: ...

nallele `signature(object = "geneSet")`: ...

nmarker `signature(object = "geneSet")`: ...

nobs `signature(x = "geneSet")`: ...

notes `signature(object = "geneSet")`: ...

notes<- `signature(object = "geneSet")`: ...

phase `signature(object = "geneSet")`: ...

phase<- `signature(object = "geneSet")`: ...

ploidy `signature(object = "geneSet")`: ...

ploidy<- `signature(object = "geneSet")`: ...

recessive `signature(object = "geneSet")`: ...

sampleInfo `signature(object = "geneSet")`: ...

sampleInfo<- `signature(object = "geneSet")`: ...

show `signature(object = "geneSet")`: ...

studyInfo `signature(object = "geneSet")`: ...

studyInfo<- `signature(object = "geneSet")`: ...

tail `signature(x = "geneSet")`: ...

transTables `signature(object = "geneSet")`: ...

transTables<- `signature(object = "geneSet")`: ...

Note

~~further notes~~

Author(s)

J.Cheng, modified by S. Chasalow, and Gregory R. Warnes <warnes@bst.rochester.edu>

References

~put references to the literature/web site here ~

Examples

```
##---- Should be DIRECTLY executable !! ----
```

`geneSet2Ped` *Translate a geneSet object to a ped matrix*

Description

Translate a geneSet object to a ped matrix.

Usage

```
geneSet2Ped(geneSetObj)
```

Arguments

`geneSetObj`	an object of geneSet class

Value

A list with five elements: `ped`, `columns`, `markerNames`, `Position`, and `fileName`. `ped` is a pedigree data frame whose first 6 columns are family (pedigree id), pid (patient id), father (father id), mother (mother id), sex, affected (affection status). The remaining columns are pairs of marker alleles. Each row corresponds to an individual; `columns` are the names of the first 5 (or 6) columns of ped file. It should be either equal to c("family","pid","father","mother","sex","affected") or equal to c("family","pid","father","mother","sex"); `founderOnly` indicates if using only founder info; `markerNames` is a vector of marker names; `Position` is a vector of marker positions; `fileName` is the pedigree file name.

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩, Gregory Warnes ⟨warnes@bst.rochester.edu⟩, Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
data(CAMP)
res<-geneSet2Ped(CAMP)
res$ped[1:5,]
res$columns
res$markerNames
res$Position
res$fileName
```

---

genotypeCoding *Get genotype coding*

Description

Get genotype coding.

Usage

```
genotypeCoding(geneSetObj, method = "A")
genotypeCoding.default(pedObj, method = "A")
```

Arguments

geneSetObj	a geneSet object
pedObj	a pedigreee object
method	genotype coding method. The default is additive coding (A). The other two available coding methods are recessive coding (R) and dominant coding (D), respectively.

Value

a matrix with nSubjects rows and nMarkers columns. Each column contains coded genotype.

Author(s)

Gregory Warnes <warnes@bst.rochester.edu> Ross Lazarus <ross.lazarus@channing.harvard.edu> Weiliang Qiu <stwxq@channing.harvard.edu>

Examples

```
data(CAMP)
res<-genotypeCoding(CAMP, method="A")
print(res[1:10,])
```

`genotypeLevels` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
genotypeLevels(object, ...)
```

Arguments

`object`	~~Describe `object` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as `help`, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"genotypeLevels"
```

`genotypeSummary` *print the summary of genotypes sorted by markers*

Description

print the summary of genotypes sorted by markers

Usage

```
genotypeSummary(object,
                by = NULL,
                confidence = 0.95,
                alpha = 1 - confidence,
                show = TRUE,
                HWE.method = c("simulate", "exact"),
                simulate.p.value = TRUE,
                B = 10000,
                verbose = FALSE,
                includeOverall = FALSE,
                omitRepeats = TRUE,
                founderOnly = FALSE,
                ...)
```

Arguments

`object`	geneSet object
`by`	optional column name, by which the summary is desired. Default is NULL.
`confidence`	confidence intervals of Genotype frequencies within each marker (default is 95%)
`alpha`	Type -1 error rate = (1- confidence)
`show`	No longer used
`HWE.method`	Method to be used for Hardy-Weinberg Equilibrium Significance Test, exact or simulate
`simulate.p.value`	No longer used
`B`	No longer used
`verbose`	No longer used
`includeOverall`	logical value (TRUE/FALSE) indicating whether overall summary is also needed, default = FALSE
`omitRepeats`	logical value (TRUE/FALSE) indicating whether Gene name and marker name should be printed repeatedly for each Genotype, default = TRUE
`founderOnly`	logical value (TRUE/FALSE) indicating whether founder information should be extracted from the geneSet object, default = FALSE
`...`	any further arguments to print

Details

We can print the genotypeSummary on screen, or save in .html format or .tex format

Author(s)

Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
library(GeneticsBase)
data(CAMP)

temp <- genotypeSummary(CAMP)

print(temp)
txt(temp, filename="genotypeSummary.txt")
html(temp, filename="genotypeSummary.html")
latex(temp, filename="genotypeSummary.tex")
```

---

genotypes *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
genotypes(object, ...)
```

Arguments

`object`	~~Describe `object` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"genotypes"
```

`gregorius` *Probability of Observing All Alleles with a Given Frequency in a Sample of a Specified Size.*

Description

Probability of observing all alleles with a given frequency in a sample of a specified size.

Usage

```
gregorius(freq, N, missprob, tol = 1e-10, maxN = 10000, maxiter=100, showiter =
```

Arguments

`freq`	(Minimum) Allele frequency (required)
`N`	Number of sampled genotypes
`missprob`	Desired maximum probability of failing to observe an allele.
`tol`	Omit computation for terms which contribute less than this value.
`maxN`	Largest value to consider when searching for N.
`maxiter`	Maximum number of iterations to use when searching for N.
`showiter`	Boolean flag indicating whether to show the iterations performed when searching for N.

Details

If `freq` and `N` are provided, but `missprob` is omitted, this function computes the probability of failing to observe all alleles with true underlying frequency `freq` when `N` diploid genotypes are sampled. This is accomplished using the sum provided in Corollary 2 of Gregorius (1980), omitting terms which contribute less than `tol` to the result.

When `freq` and `missprob` are provide, but `N` is omitted. A binary search on the range of [1,`maxN`] is performed to locate the smallest sample size, `N`, for which the probability of failing to observe all alleles with true underlying frequency `freq` is at most `missprob`. In this case, `maxiter` specifies the largest number of iterations to use in the binary search, and `showiter` controls whether the iterations of the search are displayed.

Value

A list containing the following values:

`call`	Function call used to generate this object.
`method`	One of the strings, "Compute missprob given N and freq", or "Determine minimal N given missprob and freq", indicating which type of computation was performed.
`retval$freq`	Specified allele frequency.
`retval$N`	Specified or computed sample size.
`retval$missprob`	Computed probability of failing to observe all of the alleles with frequency `freq`.

Note

This code produces sample sizes that are slightly larger than those given in table 1 of Gregorius (1980). This appears to be due to rounding of the computed `missprob`s by the authors of that paper.

Author(s)

Code submitted by David Duffy ⟨davidD@qumr.edu.au⟩, substantially enhanced by Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩.

References

Gregorius, H.R. 1980. The probability of losing an allele when diploid genotypes are sampled. Biometrics 36, 643-652.

Examples

```
# Compute the probability of missing an allele with frequency 0.15 when
# 20 genotypes are sampled:
gregorius(freq=0.15, N=20)

# Determine what sample size is required to observe all alleles with true
# frequency 0.15 with probability 0.95
gregorius(freq=0.15, missprob=1-0.95)
```

`haplo.em.w` *Wrapper for EM computation of haplotype probabilities, with Progressive Insertion*

Description

Wrapper for EM computation of haplotype probabilities, with Progressive Insertion.

Usage

```
haplo.em.w(geneSetObj,
           locus.label = NA,
           miss.val = c(0, NA),
           weight = NULL,
           control = haplo.em.control())
```

Arguments

`geneSetObj`	a geneSet object.
`locus.label`	vector of labels for loci.
`miss.val`	vector of values that represent missing alleles in geno.
`weight`	weights for observations.
`control`	list of control parameters. The default is constructed by the function `haplo.em.control`.

Details

Please refer to `haplo.em` for more details.

Value

list with components:

`converge`	indicator of convergence of the EM algorithm (1 = converge, 0 = failed).
`lnlike`	value of lnlike at last EM iteration (maximum lnlike if converged).
`lr`	likelihood ratio statistic to test the final lnlike against the lnlike that assumes complete linkage equilibrium among all loci (i.e., haplotype frequencies are products of allele frequencies).
`df.lr`	degrees of freedom for likelihood ratio statistic. The df for the unconstrained final model is the number of non-zero haplotype frequencies minus 1, and the df for the null model of complete linkage equilibrium is the sum, over all loci, of (number of alleles - 1). The df for the lr statistic is df[unconstrained] - df[null]. This can result in negative df, if many haplotypes are estimated to have zero frequency, or if a large amount of trimming occurs, when using large values of min.posterior in the list of control parameters.
`hap.prob`	vector of mle's of haplotype probabilities. The ith element of hap.prob corresponds to the ith row of haplotype.
`locus.label`	vector of labels for loci, of length K (see definition of input values).
`subj.id`	vector of id's for subjects used in the analysis, based on row number of input geno matrix. If subjects are removed, then their id will be missing from subj.id.

`rows.rem`	now defunct, but set equal to a vector of length 0, to be compatible with other functions that check for rows.rem.
`indx.subj`	vector for row index of subjects after expanding to all possible pairs of haplotypes for each person. If indx.subj=i, then i is the ith row of geno. If the ith subject has n possible pairs of haplotypes that correspond to their marker genotype, then i is repeated n times.
`nreps`	vector for the count of haplotype pairs that map to each subject's marker genotypes.
`max.pairs`	vector of maximum number of pairs of haplotypes per subject that are consistent with their marker data in the matrix geno. The length of max.pairs = nrow(geno). This vector is computed by geno.count.pairs.
`hap1code`	vector of codes for each subject's first haplotype. The values in hap1code are the row numbers of the unique haplotypes in the returned matrix haplotype.
`hap2code`	similar to hap1code, but for each subject's second haplotype.
`post`	vector of posterior probabilities of pairs of haplotypes for a person, given their marker phenotypes.
`haplotype`	matrix of unique haplotypes. Each row represents a unique haplotype, and the number of columns is the number of loci.
`control`	list of control parameters for algorithm. See haplo.em.control

Note

~~further notes~~

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

References

~put references to the literature/web site here ~

See Also

`haplo.scan.w`, `haplo.score.slide.w`, `haplo.score.w`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"haplo.em.w"
```

`haplo.scan.w` *Wrapper for searching for a trait-locus by sliding a fixed-width window over each marker locus and scanning all possible haplotype lengths within the window*

Description

Wrapper for searching for a trait-locus by sliding a fixed-width window over each marker locus and scanning all possible haplotype lengths within the window.

Usage

```
haplo.scan.w(geneSetObj,
             width = 4,
             miss.val = c(0, NA),
             em.control = haplo.em.control(),
             sim.control = score.sim.control())
```

Arguments

`geneSetObj`	A geneSet object
`width`	Width of sliding the window
`miss.val`	Vector of values that represent missing alleles in geno.
`em.control`	A list of control parameters to determine how to perform the EM algorithm for estimating haplotype frequencies when phase is unknown. The list is created by the function `haplo.em.control` - see this function for more details.
`sim.control`	List of control parameters to determine how simulations are performed for simulated p-values. The list is created by the function `score.sim.control` and the default values of this function can be changed as desired. See `score.sim.control` for details.

Details

Please refer to `haplo.scan` for more details.

Value

A list that has class haplo.scan, which contains the following items:

`call`	The call to haplo.scan.w
`scan.df`	A data frame containing the maximum test statistic for each window around each locus, and its simulated p-value.
`max.loc`	The loci (locus) which contain(s) the maximum observed test statistic over all haplotype lengths and all windows.
`globalp`	A p-value for the significance of the global maximum statistic.
`nsim`	Number of simulations performed

Note

~~further notes~~

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

References

~put references to the literature/web site here ~

See Also

haplo.em.w, haplo.score.slide.w, haplo.score.w

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"haplo.scan.w"
```

---

haplo.score.slide.w

Wrapper for haplo.score.slide in haplo.stats package, which is used to identify sub-haplotypes from a group of loci

---

Description

Wrapper for haplo.score.slide, which is used to identify sub-haplotypes from a group of loci. Run haplo.score on all contiguous subsets of size n.slide from the loci in a genotype matrix (geno). From each call to haplo.score, report the global score statistic p-value. Can also report global and maximum score statistics simulated p-values.

Usage

```
haplo.score.slide.w(geneSetObj,
                    trait.type = "gaussian",
                    n.slide = 2,
                    offset = NA,
                    x.adj = NA,
                    skip.haplo = 5/(2 * nrow(geno)),
                    locus.label = NA,
                    miss.val = c(0, NA),
                    simulate = FALSE,
                    sim.control = score.sim.control(),
                    em.control = haplo.em.control())
```

Arguments

geneSetObj	A geneSet object
trait.type	Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal".
n.slide	Number of loci in each contiguous subset. The first subset is the ordered loci numbered 1 to n.slide, the second subset is 2 through n.slide+1 and so on. If the total number of loci in geno is n.loci, then there are n.loci - n.slide + 1 total subsets.
offset	Vector of offset when trait.type = "poisson"
x.adj	Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should not be included, as it will be added in this function.
skip.haplo	Skip score statistics for haplotypes with frequencies < skip.haplo. The default is for an expected count of 5 out of the 2*N haplotype occurrences.
locus.label	Vector of labels for loci.
miss.val	Vector of values that represent missing alleles in geno.
simulate	Logical: if [F]alse, no empirical p-values are computed; if [T]rue, simulations are performed. Specific simulation parameters can be controlled in the sim.control parameter list.
sim.control	list of control parameters to determine how simulations are performed for simulated p-values. The list is created by the function score.sim.control and the default values of this function can be changed as desired. See score.sim.control for details.
em.control	A list of control parameters to determine how to perform the EM algorithm for estimating haplotype frequencies when phase is unknown. The list is created by the function haplo.em.control - see this function for more details

Details

Please refer to haplo.score.slide for more details.

Value

List with the following components:

df	Data frame with start locus, global p-value, simulated global p-value, and simulated maximum-score p-value.
n.loci	Number of loci given in the genotype matrix.
simulate	Same as parameter description above.
n.slide	Same as parameter description above.
locus.label	Same as parameter description above.
n.val.haplo	Vector containing the number of valid simulations used in the maximum-score statistic p-value simulation. The number of valid simulations can be less than the number of simulations requested (by sim.control) if simulated data sets produce unstable variables of the score statistics.
n.val.global	Vector containing the number of valid simulations used in the global score statistic p-value simulation.

Note

~~further notes~~

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

References

~put references to the literature/web site here ~

See Also

haplo.em.w, haplo.scan.w, haplo.score.w

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"haplo.score.slide.w"
```

---

haplo.score.w — *Wrapper for computing score statistics to evaluate the association of a trait with haplotypes, when linkage phase is unknown and diploid marker phenotypes are observed among unrelated subjects*

Description

wrapper for computing score statistics to evaluate the association of a trait with haplotypes, when linkage phase is unknown and diploid marker phenotypes are observed among unrelated subjects. For now, only autosomal loci are considered.

Usage

```
haplo.score.w(geneSetObj,
              trait.type = "gaussian",
              offset = NA,
              x.adj = NA,
              skip.haplo = 5/(2 * nrow(geno)),
              locus.label = NA,
              miss.val = c(0, NA),
              simulate = FALSE,
              sim.control = score.sim.control(),
              em.control = haplo.em.control())
```

Arguments

geneSetObj	A geneSet object
trait.type	Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal".
offset	Vector of offset when trait.type = "poisson"
x.adj	Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should not be included, as it will be added in this function.
skip.haplo	Skip score statistics for haplotypes with frequencies < skip.haplo. The default is for an expected count of 5 out of the 2*N haplotype occurrences.
locus.label	Vector of labels for loci.
miss.val	vector of values that represent missing alleles in geno.
simulate	Logical: if [F]alse, no empirical p-values are computed; if [T]rue, simulations are performed. Specific simulation parameters can be controlled in the sim.control parameter list.
sim.control	List of control parameters to determine how simulations are performed for simulated p-values. The list is created by the function score.sim.control and the default values of this function can be changed as desired. See score.sim.control for details.
em.control	A list of control parameters to determine how to perform the EM algorithm for estimating haplotype frequencies when phase is unknown. The list is created by the function haplo.em.control - see this function for more details

Details

Please refer to haplo.score for more details.

Value

List with the following components:

score.global	Global statistic to test association of trait with haplotypes that have frequencies >= skip.haplo.
df	Degrees of freedom for score.global.
score.global.p	P-value of score.global based on chi-square distribution, with degrees of freedom equal to df.
score.global.p.sim	P-value of score.global based on simulations (set equal to NA when simulate=F).
score.haplo	Vector of score statistics for individual haplotypes that have frequencies >= skip.haplo.
score.haplo.p	Vector of p-values for score.haplo, based on a chi-square distribution with 1 df.
score.haplo.p.sim	Vector of p-values for score.haplo, based on simulations (set equal to NA when simulate=F).
score.max.p.sim	Simulated p-value indicating for simulations the number of times a maximum score.haplo value exceeds the maximum score.haplo from the original data (equal to NA when simulate=F).

haplotype	Matrix of hapoltypes analyzed. The ith row of haplotype corresponds to the ith item of score.haplo, score.haplo.p, and score.haplo.p.sim.
hap.prob	Vector of haplotype probabilies, corresponding to the haplotypes in the matrix haplotype.
locus.label	Vector of labels for loci, of length K (same as input argument).
simulate	Same as function input parameter. If [T]rue, simulation results are included in the haplo.score object.
n.val.global	Vector containing the number of valid simulations used in the global score statistic simulation. The number of valid simulations can be less than the number of simulations requested (by sim.control) if simulated data sets produce unstable variances of the score statistics.
n.val.haplo	Vector containing the number of valid simulations used in the p-value simulations for maximum-score statistic and scores for the individual haplotypes.

Note

~~further notes~~

References

~put references to the literature/web site here ~

See Also

haplo.em.w, haplo.scan.w, haplo.score.slide.w

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"haplo.score.w"
```

---

hapmapchr22 *Chromosome 22 genotypes from International HapMap project*

Description

The sample data file, hapmapchr22, contains genotypes of hromosome 22 from the International HapMap project. This file contains genotypes from 30 CEPH trios (90 subjects) for all Phase 1 HapMap (about 1 SNP per 3kb on average) - see http://hapmap.org for the original data file, full details of samples, markers and methods. Note that the HapMap data files are deidentified and freely distributable without restriction.

Usage

```
data(hapmapchr22)
```

Format

The format is: chr "hapmapchr22"

Details

The file was created by converting chromosome 22 bulk data download data files from the HapMap file repository at http://hapmap.org/genotypes/2005-06_16c_phaseI/full/non-redundant/

Downloaded HapMap files were converted by transposing the layout of the data from one row per marker to the pedigree file convention of two columns per marker. There are many variants of the pedigree format but these files are compatible with the popular family based analysis software packages FBAT (http://www.biostat.harvard.edu/~fbat/fbat.htm) and PBAT (http://www.biostat.harvard.edu/~clange/default.htm)

The file has a header row containing only the marker names followed by one row per subject. Data rows always start with 6 fields - family_id, individual_id, father_id, mother_id, gender and affection status. Parents have zero for mother_id and father_id. The remaining columns in each row contain two allele codes for each marker. Alleles are coded as 0 for missing, $1 = A$, $2 = C$, $3 = G$ and $4 = T$. All fields in a row are delimited by one or more spaces. Note that affection status was arbitrarily set to 2 (affected) for children and 1 (unaffected) for adults - although in reality, HapMap CEPH subjects were not ascertained for any disease.

Source

http://hapmap.org/genotypes/2005-06_16c_phaseI/full/non-redundant/

References

http://www.biostat.harvard.edu/~fbat/fbat.htmhttp://www.biostat.harvard.edu/~clange/default.htm

Examples

```
data(hapmapchr22)
```

---

homozygote *Flag observations with specific allele patterns*

Description

homozygote Flag observatsions with identical alleles

heterozygote Flag observations with discordant alleles

carrier Flag observations containing a specified allele

dominant Flag observations containing one or more of the specified alleles.

recessive Flag observations containing only the specified alleles.

Usage

```
homozygote(object, ...)
homozygote.geneSet(object, allele.names, marker, ...)

heterozygote(object, ...)
heterozygote.geneSet(object, allele.names, marker, ...)

carrier(object, ...)
carrier.geneSet(object, allele.names, marker, ...)

dominant(object, ...)
dominant.geneSet(object, allele.names, marker, ...)

recessive(object, ...)
recessive.geneSet(object, allele.names, marker, ...)
```

Arguments

`object`	`geneSet` object
`allele.names`	(optional) allele names.
`marker`	(optional) marker names
`...`	(optional) additional arguments supplied

Value

matrix of logicals.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

See Also

`geneSet`, `extractAlleles`

Examples

```
data(CAMP)
lCAMP <- CAMP[,1:10] # 10 observations

# see the genotypes
genotypes(lCAMP)

# which ones are homozygotes?
homozygote(lCAMP)

# which ones are carriers  for allele "2"?
carrier(lCAMP, allele.names="2")

# which markers are heterozygotes for marker m709
heterozygote(lCAMP, marker="m709")
```

```
# if '1' is dominant, which ones will show the
# '1' phenotype for marker m47?
dominant(lCAMP, allele.names="1", marker="m47")

# if '2' is recessive, which ones will show the
# '2' phenotype for marker m523?
recessive(lCAMP, allele.names="2", marker="p523")
```

`html` *Generate summary table files for genotype objects*

Description

Generate summary table files for genotype objects

Usage

```
#### HTML file format
html
## S3 method for class 'LD':
html(x, filename = "", digits = 3, ...)
## S3 method for class 'GeneticsBaseSummary':
html(x, filename = "", ...)
## S3 method for class 'markerSummary':
html(x, filename = "", plot.format = "pdf",
                   sep=".", verbose=TRUE, ...)

#### plain text file format
txt
## Default S3 method:
txt(x, filename="", eol="\n", ...)
## S3 method for class 'markerSummary':
txt(x, filename = "", plot.format = "pdf",
                   sep=".", verbose=TRUE, ...)

#### LaTex file format
latex
## Default S3 method:
latex(x, filename="", ...)
## S3 method for class 'LD':
latex(x, filename = "", digits = 3, ...)
## S3 method for class 'GeneticsBaseSummary':
latex(x, filename = "", ...)
## S3 method for class 'markerSummary':
latex(x, filename = "", plot.format = "pdf",
                   sep=".", verbose=TRUE, ...)
```

Arguments

x	Object to be rendered to html/txt/latex
filename	Output filename, see below for details.
eol	End of line marker, defaults to "\n". MS-DOS/MS-Windows uses "\r\n"
digits	Number of digits to display
plot.format	Graphics format for LD plot. Only "pdf" is currently supported
sep	Separatior used to join the file prefix provided by filename and descriptive text when generating file names
verbose	Show names of created files.
...	Additional parameters to pass to component methods

Details

For alleleSummary, genotypeSummary, and LD objects, the filename argument is either the exact name of the file to be created, or "" which will print the output to the console.

For markerSummary objects, filename may be either "" or a prefix used to create file names. If filename="" all output is printed to the R console. Otherwise, filenames for each component are constructed by combining the prefix specified by filename, the separator specified by sep, a string descibing the file contents (one of "alleleSummary", "genotypeSummary", and "LD"), and the file extension ".html".

Value

Nothing of interest

Author(s)

Nitin Jain ⟨nitin_jain@pfizer.com⟩ and Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩

See Also

alleleSummary, genotypeSummary, markerSummary, LD

Examples

```
data(CAMP)

###
# Generate a plain text allele summary table
###
aS <- alleleSummary(CAMP)
# display inline
txt(aS, filename="")
# create CAMP_alleleSummary.txt
txt(aS, filename="CAMP.alleleSummary.html")

###
# Generate an HTML genotype summary table
###
gS <- genotypeSummary(CAMP)
# display inline
html(gS, filename="")
```

```
# create CAMP_genotypeSummary.html
html(gS, filename="CAMP.genotypeSummary.html")

###
# Generate a LaTeX Linkage Disequilibrium table
###
ld <- LD(CAMP)
# display inline
latex(ld, filename="")
# create CAMP_LDSummary.html
latex(ld, filename="CAMP.LD.html")

###
# Generate a complete set of summary tables
###
mS <- markerSummary(CAMP)
# Plain text format
txt(mS, filename="CAMP", sep="_")
# HTML format
html(mS, filename="CAMP", sep="_")
# LaTeX format
latex(mS, filename="CAMP", sep="_")
```

`left` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
left(x, ...)
right(x, ...)
```

Arguments

`x`	~~Describe `x` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"left"
```

---

makeMarkerInfo *makeMarkerInfo creates a dataframe of marker information*

Description

An auxilliary funtion to create a dataframe for the slot markerInfo of class geneSet

Usage

```
makeMarkerInfo(loci = character(0), transTable = loci)
```

Arguments

loci	a vector of makrer loci
transTable	corresponding translation table for the markers at each locus.must have same length with vector loci

Value

A dataframe with one row for each row in the callCodes slot of class geneSet. Include columns "Name" and "TransTable".

Author(s)

Scott Chasalow ⟨Scott.Chasalow@bms.com⟩, Junsheng Cheng ⟨cjsuicedu@yahoo.com⟩

`makeTransTable` *makeTransTable creates a single translation table of the markers*

Description

Create a single translation table of the markers. It is called by makeTransTableList to make a list of translation tables.

Usage

```
makeTransTable(alleleString = "Aa", sep = "/", ploidy = 2)
```

Arguments

`alleleString`	character vector such as c("12", "ACGT")
`sep`	separation symbol of the allels in alleleString
`ploidy`	Currently implemented only for ploidy=2

Value

a matrix of (# of alleles in alleleString)\textasciicircum2 by ploidy, and must include a column named "levels".

Author(s)

Scott Chasalow ⟨Scott.Chasalow@bms.com⟩, Junsheng Cheng ⟨cjsuicedu@yahoo.com⟩

`makeTransTableList` *makeTransTableList creates a list of translation tables*

Description

Calls makeTransTable to create each item of the list.

Usage

```
makeTransTableList(alleleStringVec, listNames = NULL)
```

Arguments

`alleleStringVec`	character vector such as c("12", "ACGT")
`listNames`	names of translation tables

Value

If it is a LIST, use:

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Author(s)

Scott Chasalow ⟨Scott.Chasalow@bms.com⟩, Junsheng Cheng ⟨cjsuicedu@yahoo.com⟩

Examples

```
## Not run:
## End(Not run) # End of \dontrun
```

`markerInfo` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
markerInfo(object, ...)
markerInfo(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"markerInfo"
```

`markerNames` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
markerNames(object, ...)
```

Arguments

`object`	~~Describe `object` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"markerNames"
```

`markerSummary` *Generate allele, genotype, or LD summary objects*

Description

Generate allele, genotype, or LD summary objects

Usage

```
markerSummary(object,
              covariate = NULL,
              confidence = 0.95,
              alpha = 1 - confidence,
              show = TRUE,
              ...)
## S3 method for class 'markerSummary':
print(x, ...)
```

Arguments

`object`	geneSet object
`covariate`	
`confidence`	desired confidence interval for genotype and allele frequencies in each marker
`alpha`	Type -1 error rate = (1- confidence)
`show`	No longer used
`x`	object of class 'markerSummary'
`...`	optional additional arguments

Details

We can print the alleleSummary and genotypeSummary on screen, or save in html, tex, or pdf format using appropriate methods.

Author(s)

Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
library(GeneticsBase)
data(CAMP)
temp <- markerSummary(CAMP)

print.markerSummary(temp)
html.markerSummary(temp, filename="test")
latex.markerSummary(temp, filename="test")
```

---

missingCodes *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
missingCodes(object, ...)
missingCodes(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"missingCodes"
```

`nallele` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
nallele(object, ...)
```

Arguments

`object`	~~Describe `object` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"nallele"
```

`nmarker` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
nmarker(object)
```

Arguments

`object`	~~Describe `object` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"nmarker"
```

---

notes *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
notes(object)
notes(object)<- value
```

Arguments

object	~~Describe object here~~
value	~~Describe value here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"notes"
```

`ped2geneSet` *translate a ped matrix to a geneSet object*

Description

Translate a ped matrix to a geneSet object.

Usage

```
ped2geneSet(pedObj, quiet=FALSE)
```

Arguments

`pedObj`	a list with five elements: `ped`, `columns`, `markerNames`, `Position`, and `fileName`. `ped` is a pedigree data frame whose first 6 columns are family (pedigree id), pid (patient id), father (father id), mother (mother id), sex, affected (affection status). The remaining columns are pairs of marker alleles. Each row corresponds to an individual; `columns` are the names of the first 5 (or 6) columns of ped file. It should be either equal to c("family","pid","father","mother","sex","affected") or equal to c("family","pid","father","mother","sex"); `founderOnly` indicates if using only founder info; `markerNames` is a vector of marker names; `Position` is a vector of marker positions; `fileName` is the pedidgree file name.
`quiet`	print intermediate results if `quiet=FALSE`.

Value

An object of geneSet class.

Author(s)

Weiliang Qiu ⟨stwxq@channing.harvard.edu⟩, Ross Lazarus ⟨ross.lazarus@channing.harvard.edu⟩

`phase` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
phase(object)
phase(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as `help`, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"phase"
```

`ploidy` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
ploidy(object)
ploidy(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as `help`, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"ploidy"
```

`plot.LD` *Textual and graphical display of linkage disequilibrium (LD) objects*

Description

Textual and graphical display of linkage disequilibrium (LD) objects

Usage

```
plot(x, y, ...)
plot.LD (x, y=NULL, cex, ...)
LDView(LdMat,
       SNPloc=NULL,
       is.triangle=TRUE,
       SNPnames=NULL,
       cexSNPnames=0.2+1/log10(nrow(LdMat)),
       margins=c(6,2,3,6),
       main=NULL,
       barCol=rainbow(256,start=0,end=1/6 ),
       widths=c(10,10),
       heights=c(1,10),
       ...)
```

Arguments

`x`	LD or LDband object
`y`	currently ignored
`cex`	Scaling factor for table text. If absent, text will be scaled to fit within the table cells.
`LdMat`	matrix of pair-wise LD measures
`SNPloc`	SNP locations
`is.triangle`	indicate if 'LdMat' is a lower triangle matrix or not. if 'LdMat' is an upper triangle, then the user has to transpose it before calling the function 'LDView'.
`SNPnames`	labels for SNPs
`cexSNPnames`	font size for SNPs labels
`margins`	margins for heatmap
`main`	title for the plot
`barCol`	specify the color scheme
`widths`	The plot is split into two parts – upper part and lower part 'widths' specifies the widths of the two parts
`heights`	The plot is split into two parts – upper part and lower part 'heights' specifies the heights of the two parts
`...`	Optional arguments (`plot.LD.data.frame` passes these to `LDtable` and `LDplot`) and other parameters for the function 'image'

Details

LDtable generates a graphical matrix of LD coefficients. It attempts to properly set the font size so that the estimated values fit into the boxes. It also colorized the boxes to whether the estimates are significantly different from no linkage.

LDplot generates a plot of the LD coeffients across markers. By default it will overlay plots for LD against all markers. LD against a specific subset of markers can be obtained using the `marker` argument.

Value

`SNPnames`	labels of the SNPs in `LDView`
`LdMat`	LD matrix
`SNPloc`	SNP positions

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩

See Also

`LD`, `geneSet`, `diseq`

Examples

```
# load the data
data(CAMP)

# compute pairwise LD
ld <- LD(CAMP)

print(ld)    # text display of LD coefficents and graphical display of LD estimates

LDView(t(ld@"R^2"), SNPloc=1:8, SNPnames=CAMP@markerInfo$Name)

## LDtable(ld) # graphical display of LD estimates
## LDtable(ld, which="D'", digits=2) # graphical display of D' only

## plot(CAMP) # two panel display
```

---

qtlex *Simulated pedigree with genotypes and one qtl covariate*

Description

Simulated dataset for a pedigree of 1000 trios with 51 SNPs, and 1 quantitative trait.

Usage

```
data(qtlex)
```

Format

'geneSet' object

Details

This data is the 'qtl' example from Lange and Kraft's "Short Course: Genetics Associateion Analysis."

Source

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

References

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Associateion Analysis."

DeMeo, D. L., C. Lange, et al. (2002). "Univariate and multivariate family-based association analysis of the IL-13 ARG130GLN polymorphism in the Childhood Asthma Management Program." Genet Epidemiol 23(4): 335-48.

Examples

```
library(GeneticsBase)
data(qtlex)
head(qtlex)
```

---

read.pfizer *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
read.pfizer.Listing(file, verbose = TRUE)
read.pfizer.Pivot(file, verbose = TRUE)
```

Arguments

file	~~Describe file here~~
verbose	~~Describe verbose here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.
```

---

read.phe *Read '.phe' phenotype file data*

Description

Read '.phe' phenotype file data

Usage

```
read.phe(filename, columns = c("family", "pid"), quiet = FALSE, ...)
read.fbat.phe(filename, columns = c("family", "pid"), quiet = FALSE, ...)
read.pbat.phe(filename, columns = c("family", "pid"), quiet = FALSE, ...)
```

Arguments

filename	Name of the file
columns	Column names for the first two columns of a pedigree file. Defaults to "family" and "pid", where "family" is a unique identifier for each family, and "pid" is a unique identifier for each individual.
quiet	Logical indicating whether progress display is suppressed, Defaults to TRUE.
...	Additional (optional) parameters provided to function matrix

Value

A data frame containing the file

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
# store where we are now
here <- getwd()

# move to the data directory
dir <- file.path(.path.package("GeneticsBase"),"data")
setwd(dir)

# load hapmap chromosome 1 data
xbat <- readGenes(gfile="xbat.ped", gformat="fbat.ped",
                  pfile="xbat.phe", pformat="fbat.phe")

# look at the data
xbat

# return to the original path
setwd(here)
```

`readGenes` *Import genetic data from standard file formats*

Description

Import genetic data from standard file formats.

Usage

```
readGenes(gfile, gformat=genotypeFileFormats,  goptions=list(),
          pfile, pformat=phenotypeFileFormats, poptions=list(),
          mfile, mformat=markerFileFormats,    moptions=list() )
```

Arguments

`gfile`	File containing genotype data
`gformat`	Function, function name, or a character file format specification from the list `genotypeFileFormats`.
`goptions`	Optional arguments for loading genotype data.
`pfile`	File containing phenotype data
`pformat`	Function, function name, or a character file format specification from the list `phenotypeFileFormats`.
`poptions`	Optional arguments for loading phenotype data.
`mfile`	File containing marker data
`mformat`	Function, function name, or a character file format specification from the list `markerFileFormats`.
`moptions`	Optional arguments for loading marker data.

Details

Load genotype and (optionally) phenotype and marker information from the specified files and generate a geneSet object containing the results.

A variety of file formats are available. See the variables `genotypeFileFormats`, `phenotypeFileFormats`, `phenotypeFileFormats` for formats.

Value

An object of class geneSet.

Note

Adding a new genotype (phenotype) file format requires creation of a function named `readGenes.newformat` (`read.newformat`) and adding the string "newformat" to the vector `genotypeFileFormats` (`phenotypeFileFormats`).

Please submit new format functions to the authors for inclusion in the package.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩

See Also

`read.table`, etc

Examples

```
# Perlegen data set (if we have access to it)
# (189K SNPS x 1008 patients)
path <- "~/clingen/Perlegen_Metabolic_Disorder/Data"
fname <- file.path(path,"PFZ_03/genotype.txt.gz")
if(file.exists(fname))
  {
    genef   <- gzfile(fname) # open the gzip compressed SNP file
    #phenof  <-
    markerf <-file.path(path,"PFZ_03/snp_info.txt")

    pgdata <- readGenes(gfile=genef, gformat="perlegen",
                        mfile=markerf, mformat="table",
                        moptions=list(sep="\t",header=TRUE))
    close(genef)
  }
```

readGenes.ped *Function to read pedigree file format*

Description

Function to read pedigree file format

Usage

```
readGenes.ped(filename, columns = c("family", "pid", "father", "mother", "sex"),
              phase = list(FALSE), quiet = FALSE, missingval = c(0), ...)
readGenes.hapmap.ped(filename, ...)
readGenes.fbat.ped(filename, ...)
readGenes.pbat.ped(filename, ...)
```

Arguments

filename	Name of the file in which data is stored
columns	Name of the columns in the pedigree file - by default points to "family", "pid", "father", "mother", "sex"
phase	1. Yes/No for all (logical scalar) 2. Yes/No for each Marker (logical vector) 3. phaseObject (TBD): observation by marker by phase probabilities + definitions of contigs + probability of contigs
quiet	Logical: whether the progress of reading the file should be displayed.
missingval	Missing values (if any) to obtain missingCodes
...	Additional arguments to function read.table

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

readGenes.perlegen *Read Perlegen data files.*

Description

Read Perlegen data files.

Usage

```
readGenes.perlegen(filename, ..., quiet = FALSE)
```

Arguments

filename	Name of the file in which data is stored
...	Additional arguments to scan
quiet	Whether the progress of loading data should be displayed.

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

`readGenes.pfizer` *Read genetics data files that use Pfizer's data format.*

Description

Read genetics data files that use Pfizer's data format.

Usage

```
readGenes.pfizer(filename, phase=list(F),
                 format=c("Pivot", "Listing"),...)
```

Arguments

`filename`	Name of the file in which data is stored
`phase`	see below
`format`	Format of the file to be read, one of "Pivot" or "Listing"
`...`	Additional arguments to function read.table

Details

The argument `phase` may be:

1. A single logical scalar value that applies for all markers
2. A logical vector with one element per marker
3. A phaseObject (TBD) providing observation by marker by phase probabilities + definitions of contigs + probability of contigs

Value

a geneSet object

Author(s)

Gregory R. Warnes ⟨warnes@bst.rochester.edu⟩ and Nitin Jain ⟨nitin.jain@pfizer.com⟩

Examples

```
# store where we are now
here <- getwd()

# move to the data directory
dir <- file.path(.path.package("GeneticsBase"),"data")
setwd(dir)

# load Pfizer Data
PfizerExample <- readGenes.pfizer("PfizerExample.txt", format="Listing")
```

```
# look at the data
PfizerExample

# return to the original path
setwd(here)
```

`sampleInfo` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
sampleInfo(object, ...)
sampleInfo(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as `help`, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"sampleInfo"
```

`studyInfo` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
studyInfo(object)
studyInfo(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"studyInfo"
```

`transTables` *~~function to do ... ~~*

Description

~~ A concise (1-5 lines) description of what the function does. ~~

Usage

```
transTables(object, ...)
transTables(object)<- value
```

Arguments

`object`	~~Describe `object` here~~
`value`	~~Describe `value` here~~
`...`	~~Describe `...` here~~

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

~~objects to See Also as help, ~~~

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
"transTables"
```

---

xbat *Simulated pedigree with genotypes and covariates*

Description

Simulated dataset for a pedigree of 1000 trios with 50 SNPs, with 8 quantitative traits, 2 binary traits, and 8 covariates."

Usage

```
data(xbat)
```

Format

'geneSet' object

Details

This data is the 'xbat' example from Lange and Kraft's "Short Course: Genetics Associateion Analysis." It is described there as:

"[This] simulated dataset comprises a pedigree file with genotype information for 1000 trios with 50 SNPs and a phenotype file that contains 8 quantitative traits, 2 binary traits, and 8 covariates.

"Genotypes

"The simulation generated complete genotype data for 1000 families with two parents and one offspring. The single nucleotide polymorphism (SNP) frequencies and haplotype blocks were estimated using real data. These estimates were fixed and used as parameters for the simulation of the parental genotypes. Offspring genotypes were generated by simulating random Mendelian transmission from their respective parents. In total, 50 SNPs were simulated, 28 of which lie in 1 of 5 variable length haplotype blocks (range: 4 to 10 SNPs per block). The blocks were simulated as a function of haplotype block frequency, assuming no recombination, resulting in varying degrees of linkage disequilibrium within each block. The remaining 22 SNPs that are not in a haplotype block were simulated randomly as a function of SNP frequency. The SNPs are indicated in the header line of the pedigree file, and named SNP1, SNP2, .., and SNP50. Note that the affectation status variable in the pedigree file is coded as missing (0) for all individuals. All phenotype data comes from the phenotype file (see below).

"Phenotypes

"Overall, 10 phenotypes (Y) were simulated additively as function of the genetic effect size a, marker score X, covariate effect size b, and covariate value Z as follows:

$Y_i = a_i X_i + b_i Z_i \ (i = 1, 2, .., 10)$

"Quantitative Traits

"Eight quantitative phenotypes were simulated from a random sample from a normal distribution: $Y \tilde{} N([aX + bZ], s2)$, where a is the additive effect for the phenotype and s2 is the variance. We measure the strength of the additive effect relative to the phenotypic variance by the heritability h2 [Falconer and Mackay, 1997], which is the proportion of phenotypic variation explained by genetic variation. We assume that the environment variance is 1. SNP23 was simulated as the "disease SNP" which is the 5th SNP in a 10 SNP haplotype block. The heritabilities were simulated from random uniform distribution ranging from -0.1 to 0.1. In addition, the simulation produced two correlated quantitative traits (QTL9 and QTL10; r2 = 0.40). The quantitative traits are indicated in the header line of the phenotype file and named QTL1, QTL2, .., and QTL10.

"Binary Traits

"Two binary traits were simulated simply by dichotomizing the first quantitative trait (QTL1). For the AFF1 trait, individuals were coded as affected (1) if their QTL1 value is above the sample mean and unaffected (0) if their QTL1 value was below the sample mean. For the AFF2 trait, individuals were coded as affected (1) if their QTL1 value is at least one standard deviation above the sample mean, and missing ("-") if their trait value did not reach that criteria.

"Covariates

"In addition to the additive genetic effect, each phenotype was simulated with one covariate effect. The quantitative covariates were sampled from normal distribution (mu = random, s2 = 10). The effect size for each covariate was sampled randomly from a uniform distribution (0, 1). The covariates are indicated in the header line of the phenotype file and named COV1, COV2, .., and COV10. Note that COV1 corresponds to QTL1, AFF1 and AFF2."

(quoted from Lange and Kraft 2005)

Source

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Association Analysis."

References

Lange, C. and Kraft, P. (2005). "Short Course: Genetics Association Analysis."

DeMeo, D. L., C. Lange, et al. (2002). "Univariate and multivariate family-based association analysis of the IL-13 ARG130GLN polymorphism in the Childhood Asthma Management Program." Genet Epidemiol 23(4): 335-48.

Examples

```
library(GeneticsBase)
data(xbat)
head(xbat)
```

Index