

---

add.phdata	<i>Adds phenotypic variables to gwaa.data object</i>
------------	--

---

## Description

Adds phenotypic variables to @phdata slot of an `gwaa.data-class` object

## Usage

```
add.phdata(data, phdata)
```

## Arguments

data	an object of <code>gwaa.data-class</code>
phdata	data frame

## Details

The "phdata" data frame is simply merged to the @phdata slot of the "data", and is sorted according to the right order.

The "phdata" frame should contain single variable named "id", preferably of "character" class. It may contain "sex" variable, but that will be re-named to avoid duplication with the default sex variable presented in @phdata.

## Value

Updated object of `gwaa.data-class`

## Author(s)

Yurii Auelhenko

## See Also

`merge.gwaa.data`, `merge.snp.data`

## Examples

```
data(srdta)
# take a small subset for this example
srdta <- srdta[1:10,1:5]
srdta
# generate id names
ids <- paste("p",c(2,1,7,3,5,9,11,22,27),sep="")
# generate some random trait values
newtra <- rnorm(9)
# make data frame
```

```

      x <- data.frame(id=ids,newtra=newtra)
      x
# now add this new trait to the data
      srdta1 <- add.phdata(srdta,x)
      srdta1

```

---

add.plot

*function to plot additional GWAA results*


---

## Description

Add plot of results of GWA analysis

## Usage

```
add.plot(x, ..., df = 1, col=c("lightgreen","lightblue"), sort=TRUE, delta = 1)
```

## Arguments

x	object of type scan.gwaa-class, as returned by <code>scan.glm</code> , <code>qtscore</code> , <code>ccfast</code> , <code>emp.ccfast</code> , <code>emp.qtscore</code> , or <code>scan.haplo</code> ; or of type scan.gwaa.2D-class, as returned by <code>scan.haplo.2D</code> or <code>scan.glm.2D</code> .
...	additional arguments to be passed to plot
df	P-value at which df to add (1, 2 or "Pc1df")
col	which colors to use to depict consecutive chromosomes
sort	whether results should be plotted after sorting by chromosome and position
delta	gap width between chromosomes

## Value

No value returned.

## Author(s)

Yurii Aulchenko

## See Also

`plot`, `snpsubset`, `scan.glm`, `qttscore`, `ccfast`, `emp.qtscore`, `emp.ccfast`, `scan.haplo`, `scan.haplo.2D`, `scan.glm.2D`

## Examples

```

data(srdta)
a <- ccfast("bt",srdta,snps=c(1:100))
plot(a)
a1 <- qttscore(bt,srdta,snps=c(1:100))
add.plot(a1,col="red",type="l")

```

---

```
as.character.gwaa.data
```

*Attempts to convert genotypic part of gwaa.data to character*

---

## Description

A function to convert @gtdata slot of an object of `gwaa.data-class` to "character"

## Usage

```
as.character.gwaa.data(x, ...)
```

## Arguments

x	An object of <code>gwaa.data-class</code>
...	...

## Value

A matrix containing genotypes in character format

## Author(s)

Yurii Aulchenko

## See Also

`as.character.snp.data`, `as.double.gwaa.data`, `as.double.snp.data`, `as.hsgeno`,  
`as.genotype.gwaa.data`, `as.genotype.snp.data`

## Examples

```
data(srdta)
as.character(srdta[1:5,1:10])
```

```
as.character.snp.coding
```

*Attempts to convert internal snp.coding-class to character*

---

## Description

A function to convert an object of `snp.coding-class` to "character"

## Usage

```
as.character.snp.coding(x, ...)
```

## Arguments

<code>x</code>	An object of <code>snp.coding-class</code>
<code>...</code>	...

## Value

A vector containing actual (nucleotide) coding, for corresponding SNPs, in character format

## Author(s)

Yurii Aulchenko

## See Also

`as.character.snp.strand`, `as.character.snp.data`, `as.double.snp.data`, `as.hsgeno`,  
`as.genotype.snp.data`

## Examples

```
data(srdta)
as.character(srdta@gtdata@coding[1:5])
```

---

```
as.character.snp.data
```

*Attempts to convert snp.data to character*

---

### Description

A function to convert an object of `snp.data-class` to "character"

### Usage

```
as.character.snp.data(x, ...)
```

### Arguments

x	An object of <code>snp.data-class</code>
...	...

### Value

A matrix containing genotypes in character format

### Author(s)

Yurii Aulchenko

### See Also

`as.double.snp.data`, `as.hsgeno`, `as.genotype.snp.data`

### Examples

```
data(srdta)
as.character(srdta@gtdata[1:5,1:10])
```

---

```
as.character.snp.strand
```

*Attempts to convert internal strand-class to character*

---

### Description

A function to convert an object of `snp.strand-class` to "character"

**Usage**

```
as.character.snp.strand(x, ...)
```

**Arguments**

x	An object of <code>snp.strand-class</code>
...	...

**Value**

A vector containing strand ("+", "-" or "u"), for corresponding SNPs, in character format

**Author(s)**

Yurii Aulchenko

**See Also**

`as.character.snp.coding`, `as.character.snp.data`, `as.double.snp.data`, `as.hsgeno`,  
`as.genotype.snp.data`

**Examples**

```
data(srdta)
as.character(srdta@gtdata@strand[1:5])
```

---

```
as.data.frame.gwaa.data
```

*Attempts to convert snp.data to "hsgeno"*

---

**Description**

A function taking @phdata part (data.frame) of the object of `gwaa.data-class`

**Usage**

```
as.data.frame.gwaa.data(x, ...)
```

**Arguments**

x	An object of <code>data.frame-class</code>
...	...

**Details**

Use is mainly internal

**Value**

A data-frame containing phenotypic data

**Author(s)**

Yurii Aulchenko

**See Also**

[as.character.snp.data](#), [as.double.snp.data](#), [as.genotype.snp.data](#)

**Examples**

```
data(srdta)
as.data.frame(srdta[1:5,])
```

---

```
as.double.gwaa.data
```

*Attempts to convert gwaa.data to double*

---

**Description**

A function to convert an object of [gwaa.data-class](#) to "double"

**Usage**

```
as.double.gwaa.data(x, ...)
```

**Arguments**

x	An object of <a href="#">gwaa.data-class</a>
...	...

**Value**

A matrix containing genotypes in double (numeric) format

**Author(s)**

Yurii Aulchenko

**See Also**

[as.character.gwaa.data](#), [as.character.snp.data](#), [as.double.gwaa.data](#), [as.double.snp.data](#),  
[as.hsgeno](#), [as.genotypes.gwaa.data](#), [as.genotype.snp.data](#)

## Examples

```
data(srdta)
as.double(srdta[1:5,1:10])
```

---

`as.double.snp.data` *Attempts to convert snp.data to double*

---

## Description

A function to convert an object of `snp.data-class` to "double"

## Usage

```
as.double.snp.data(x, ...)
```

## Arguments

<code>x</code>	An object of <code>snp.data-class</code>
<code>...</code>	...

## Value

A matrix containing genotypes in double (numeric) format

## Author(s)

Yurii Aulchenko

## See Also

`as.character.snp.data`, `as.hsgeno`, `as.genotype.snp.data`

## Examples

```
data(srdta)
as.double(srdta@gtdata[1:5,1:10])
```



---

```
as.genotype.gwaa.data
```

*Attempts to convert gwaa.data to "genotype"*

---

## Description

A function to convert @gtdata slot of an object of `gwaa.data-class` to "genotype" data frame

## Usage

```
as.genotype.gwaa.data(x, ...)
```

## Arguments

`x`                      An object of `gwaa.data-class`

`...`                    ...

## Value

A data-frame containing genotypes consumable by "genetics" library

## Author(s)

Yurii Aulchenko

## See Also

`as.character.gwaa.data`, `as.character.snp.data`, `as.double.gwaa.data`, `as.double.snp.data`,  
`as.hsgeno`, `as.genotype.gwaa.data`, `as.genotype.snp.data`

## Examples

```
data(srdta)
as.genotype(srdta[1:5,1:10])
```

---

```
as.genotype.snp.data
```

*Attempts to convert snp.data to "genotype"*

---

### Description

A function to convert an object of `snp.data-class` to "genotype" data frame

### Usage

```
as.genotype.snp.data(x, ...)
```

### Arguments

<code>x</code>	An object of <code>snp.data-class</code>
<code>...</code>	...

### Value

A data-frame containing genotypes consumable by "genetics" library

### Author(s)

Yurii Aulchenko

### See Also

`as.character.snp.data`, `as.double.snp.data`, `as.hsgeno`

### Examples

```
data(srdta)
as.genotype(srdta@gtdata[1:5,1:10])
```

---

```
as.genotype
```

*Attempts to convert object to "genotype"*

---

### Description

A function to convert an object to "genotype" data frame

### Usage

```
as.genotype(x, ...)
```

**Arguments**

`x`                      An object of `snp.data-class`  
`...`                    ...

**Value**

A data-frame containing "genotype" data class, consumable by "genetics" library

**Author(s)**

Yurii Aulchenko

**See Also**

`as.character.gwaa.data`, `as.character.snp.data`, `as.double.gwaa.data`, `as.double.snp.data`,  
`as.hsgeno`, `as.genotype.gwaa.data`, `as.genotype.snp.data`

**Examples**

```
data(srdta)
as.genotype(srdta@gtdata[1:5,1:10])
```

---

```
as.hsgeno.gwaa.data
```

*Attempts to convert gwaa.data to "hsgeno"*

---

**Description**

A function to convert @gtdata slot of an object of `gwaa.data-class` to "hsgeno" data frame

**Usage**

```
as.hsgeno.gwaa.data(x, ...)
```

**Arguments**

`x`                      An object of `gwaa.data-class`  
`...`                    ...

**Value**

A data-frame containing alleles, consumable by "haplo.stats" library

**Author(s)**

Yurii Aulchenko

**See Also**

[as.character.gwaa.data](#), [as.character.snp.data](#), [as.double.gwaa.data](#), [as.double.snp.data](#),  
[as.hsgeno](#), [as.genotype.gwaa.data](#), [as.genotype.snp.data](#)

**Examples**

```
data(srdta)
as.hsgeno(srdta[1:5,1:10])
```

---

`as.hsgeno.snp.data` *Attempts to convert snp.data to "hsgeno"*

---

**Description**

A function to convert an object of [snp.data-class](#) to "hsgeno" data frame

**Usage**

```
as.hsgeno.snp.data(x, ...)
```

**Arguments**

x	An object of <a href="#">snp.data-class</a>
...	...

**Value**

A data-frame containing alleles, consumable by "haplo.stats" library

**Author(s)**

Yurii Aulchenko

**See Also**

[as.character.snp.data](#), [as.double.snp.data](#), [as.genotype.snp.data](#)

**Examples**

```
data(srdta)
as.hsgeno(srdta@gtdata[1:5,1:10])
```

---

as.hsgeno	<i>Attempts to convert object to "hsgeno"</i>
-----------	---

---

**Description**

A function to convert an object to "hsgeno" data frame, to be used by "haplo.stats" library

**Usage**

```
as.hsgeno(x, ...)
```

**Arguments**

x	An object of <code>snp.data-class</code> or <code>gwaa.data-class</code>
...	...

**Value**

A data-frame containing alleles, consumable by "haplo.stats" library

**Author(s)**

Yurii Aulchenko

**See Also**

`as.character.snp.data`, `as.double.snp.data`, `as.genotype.snp.data`

**Examples**

```
data(srdta)
as.hsgeno(srdta[1:5,1:3])
as.hsgeno(srdta@gtdata[1:5,1:3])
```

---

autosomal	<i>Function telling all autosomal SNPs</i>
-----------	--

---

**Description**

Function telling all autosomal SNPs

**Usage**

```
autosomal(data)
```

**Arguments**

data                      object of gwaa.data-class or snp.data-class

**Details**

For every SNP, looks up the chromosome, and, when it is an utosome (not X, Y, XY, or mt), reports the name back

**Value**

Vector of SNP names

**Author(s)**

Yurii Aulchenko

**Examples**

```
data(ge03d2)
autosomal(ge03d2)[1:10]
```

---

catable	<i>function to generate summary table for quantitative data</i>
---------	---

---

**Description**

This function makes a table with number of observations which fall between user-defined categories

**Usage**

```
catable(data, categories = c(quantile(data,c(0.01,0.1,0.5,0.9,0.99),na.rm=TRUE)), c
```

**Arguments**

data	A vector of numerics
categories	vector containing desired cut-off levels
cumulative	whether cumulative distribution should be shown
na.rm	how to treat NAs
digits	number of digits to be saved in rounding

**Value**

table with number and proportion of observations falling between categories

**Author(s)**

Yurii Aulchenko

**See Also**[summary.snp.data](#), [perid.summary](#)**Examples**

```
data(srdta)
callr <- summary(srdta@gtdata)[,"CallRate"]
catable(callr,c(0.93,0.95,0.99))
catable(callr)
catable(callr,cum=TRUE)
```

ccfast

*fast case-control analysis***Description**

Fast case-control analysis by computing chi-square test from 2x2 (allelic) or 2x3 (genotypic) tables

**Usage**

```
ccfast(y, data, snpsubset, idsubset, times=1, quiet=FALSE, bcast=10, clambda=TRUE, propPs)
```

**Arguments**

y	character name of the vector of case-control status. Cases are denoted as 1 and controls as 0.
data	An object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data are used for analysis.
times	If more then one, the number of replicas to be used in derivation of empirical genome-wide significance. See <a href="#">emp.qtscore</a> , which calls qtscore with times>1 for details
quiet	do not print warning messages
bcast	If the argument times > 1, progress is reported once in bcast replicas
clambda	If inflation facot Lambda is estimated as lower then one, this parameter controls if the original P1df (clambda=TRUE) to be reported in Pc1df, or the original 1df statistics is to be multiplied onto this "deflation" factor (clambda=FALSE). If a numeric value is provided, it is used as a correction factor.
propPs	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <a href="#">estlambda</a>

**Value**

Object of class `scan.gwaa-class`

**Author(s)**

Yurii Aulchenko

**See Also**

`emp.ccfast`, `plot.scan.gwaa`, `scan.gwaa-class`

**Examples**

```
data(srdta)
a <- ccfast("bt", data=srdta, snps=c(1:10), ids=c(1:100))
a
a <- ccfast("bt", data=srdta)
plot(a)
```

---

`check.marker-class` *Class "check.marker"*

---

**Description**

This class contains results of genotypic quality control. This is an list object, usually generated by `check.marker`.

**Names**

`snpok` Markers which passed all criteria

`idok` People which passed all criteria

`nohwe` Markers which did not pass HWE check

`Pex.nohwe` Exact HWE P-values for markers which did not pass HWE check

`nocall` Markers with call rate < specified callrate

`nofreq` Markers with MAF < specified maf

`Xmrkfail` X-linked markers with too many heterozygous male genotypes

`redundant` Redundant markers

`details.redundancy` List with details on redundant markers (reference-marker <-> redundant-markers)

`idnocall` People with too low SNP call rate across al SNPs



hetfail People having too high heterozygosity

ibsfail People having too high IBS with other people

Xidfail Men with too many heterozygous X-linked markers

call List with details on call: call, name (of marker), map, chromosome

### Methods

**summary** signature(object = "check.marker"): gives a cross table summarising how many markers did not pass because of this or that criteria

**plot** signature(object = "check.marker"): Plots summary of genotypic data QC

### Author(s)

Yurii Aulchenko

### See Also

[check.marker](#), [summary.check.marker](#), [redundant](#), [plot.check.marker](#)

### Examples

```
data(srdta)
mc <- check.marker(data=srdta@gtdata[,1:100], redundant="all", maf=0.01, minconcordance=0.9, fd=0.01)
class(mc)
names(mc)
names(mc$call)
mc$nohwe
mc$Pex.nohwe
summary(mc)
plot(mc)
```

---

check.marker

*function to do genotypic quality control*

---

### Description

This function helps selecting the marker which should enter into GWA analysis based on call rate, minor allele frequency, value of the chi-square test for Hardy-Weinberg equilibrium, and redundancy, defined as concordance between the distributions of the genotypes (including missing values).

**Usage**

```
check.marker(data, snpsubset, idsubset, callrate = 0.95,
             perid.call=0.95, extr.call = 0.1, extr.perid.call = 0.1, het.fdr =
             ibs.threshold = 0.95, ibs.mrk = 2000, ibs.exclude="lower", maf, p.l
             odds = 1000, hweidssubset, redundant = "no",
             minconcordance = 2.0, qoption = "bh95", imphetasmissing = TRUE, XXY
```

**Arguments**

<code>data</code>	gwaa.data or snp.data object
<code>snpsubset</code>	a subset of SNPs to check (names, indexes, logical), default is all from data
<code>idsubset</code>	a subset of people to check (names, indexes, logical), default is all from data
<code>callrate</code>	cut-off SNP call rate
<code>perid.call</code>	cut-off individual call rate (maximum percent of missing genotypes in a person)
<code>extr.call</code>	SNPs with this low call rate are dropped prior to main analysis
<code>extr.perid.call</code>	people with this low call rate are dropped prior to main analysis
<code>het.fdr</code>	FDR rate for unacceptably high individual heterozygosity
<code>ibs.threshold</code>	threshold value for acceptable IBS
<code>ibs.mrk</code>	How many random markers should be used to estimate IBS. When <code>ibs.mrk &lt; 1</code> , IBS checks are turned off. When "all" all markers are used.
<code>ibs.exclude</code>	"both" or "lower" – whether both samples with <code>IBS&gt;ibs.threshold</code> should be excluded, or the one with lower call rate.
<code>maf</code>	cut-off Minor Allele Frequency. If not specified, the default value is 5 chromosomes (5/data@nsnpa)
<code>p.level</code>	cut-off p-value in check for Hardy-Weinberg Equilibrium. If negative, FDR is applied
<code>fdrate</code>	cut-off FDR level in check for Hardy-Weinberg Equilibrium
<code>odds</code>	cut-off odds to decide whether marker/person should be excluded based on sex/X-linked marker data inconsistency
<code>hweidssubset</code>	a subset of people to check (names, indexes, logical) to use for HWE check
<code>redundant</code>	if "bychrom", redundancy is checked within chromosomes; "all" – all pairs of markers; "no" – no redundancy checks
<code>minconcordance</code>	a parameter passed to "redundant" function. If "minconcordance" is > 1.0 only pairs of markers which are exactly the same, including NA pattern, are considered as redundant; if $0 < \text{"minconcordance"} < 1$ , then pairs of markers with concordance > "minconcordance" are considered redundant. see <a href="#">redundant</a> for details. Note that if "minconcordance" < 1 the program will take much longer time to run
<code>qoption</code>	if "bh95", BH95 FDR used; if "storey", qvalue package is used

imphetasmissing	If "impossible heterozygotes" (e.g. heterozygous mtDNA, and male Y- and X-chromosome markers) should be treated as missing genotypes in the QC procedure
XXY.call	What proportion of Y-chromosome markers should be called to consider that Y-chromosome is present (in presence of XX)

## Details

In this procedure, sex errors are identified initially and then possible residual errors are removed iteratively. At the first step, of the iterative procedure, per-marker (minor allele frequency, call rate, exact P-value for Hardy-Weinberg equilibrium) and between-marker statistics are generated and controlled for, mostly using the internal call to the function `summary.snp.data`.

At the second step of the iterative procedure, per-person statistics, such call rate within a person, heterozygosity and between-person statistics (identity by state across a random sample of markers) are generated, using `perid.summary` and `ibs` functions. Heterozygosity and IBS are estimated using only autosomal data. If IBS is over `ibs.threshold` for a pair, one person from the pair is added to the `ibsfail` list and excluded from the `idok` list. At the second step, only the markers passing the first step are used.

The procedure is applied recursively till no further markers and people are eliminated.

## Value

Object of class `check.marker-class`

## Author(s)

Yurii Aulchenko

## See Also

`check.trait`, `ibs`, `summary.snp.data`, `perid.summary`, `plot.check.marker`, `summary.check.marker`, `redundant`, `HWE.show`, `check.marker-class`

## Examples

```
# usual way
data(ge03d2c)
# many errors
mc0 <- check.marker(ge03d2c)
# take only people and markers passing QC
fixed0 <- ge03d2c[mc0$idok,mc0$snpok]
# major errors fixed, still few males are heterozygous for X-chromosome markers
mc1 <- check.marker(fixed0)
# fix minor X-chromosome problems
fixed1 <- Xfix(fixed0)
# no errors
mc2 <- check.marker(fixed1)
summary(mc2)
# ready to use fixed1 for analysis
```

```
# let us look into redundancy
data(srdta)
mc <- check.marker(data=srdta,ids=c(1:300),call=.92,perid.call=.92)
names(mc)
mc$nohwe
mc <- check.marker(data=srdta@gtdata[,1:100],call=0.95,perid.call=0.9,maf=0.02,minconcordance=0.9)
summary(mc)
HWE.show(data=srdta,snps=mc$nohwe)
plot(mc)
```

---

check.trait	<i>function to do primitive trait quality control</i>
-------------	---

---

## Description

This function check for outliers (using FDR framework) and plots the raw data.

## Usage

```
check.trait(trait, data, fdrate = 0.05, graph = TRUE, binshow = FALSE,
            qoption = "bh95")
```

## Arguments

trait	name (or list of names) of trait(s) to be checked
data	gwaa.data object or data frame containing the trait
fdrate	false discovery rate to apply for QC
graph	if graphical output should be produced
binshow	if binary traits should be plotted
qoption	how to compute q-values (not implemented, currently using only BH95)

## Details

The P-value that a particular measurement is an outlier is computed as following. Consider trait vector  $Y$  with particular  $i^{th}$  measurement denoted as  $y_i$ . Let  $Y(-i)$  is vector, which is the same as  $Y$ , except that  $i^{th}$  measurement is dropped. Then Chi-square for measurement  $i$  is computed as

$$Chi_i = (mean(Y(-i)) - y_i)^2 / var(Y(-i))$$

P-value is computed using 1 d.f., and the vector of P-values enters FDR computation procedure (BH95 by default).

**Value**

No value returned, output is made to the screen and graphical device.

**Author(s)**

Yurii Aulchenko

**See Also**

[check.marker](#)

**Examples**

```
data(srdta)
check.trait("qt3", data=srdta)
n <- names(srdta@phdata)
check.trait(n, data=srdta)
```

---

`convert.snp.affymetrix`

*function to convert genotypic data from Affymetrix to internal format*

---

**Description**

Converts genotypic data from Affymetrix format to internal genotypic data formatted file

**Usage**

```
convert.snp.affymetrix(dir, map, outfile, skipaffym)
```

**Arguments**

<code>dir</code>	Directory which affymetrix files storages.
<code>map</code>	File name with map (annotation) information.
<code>outfile</code>	Output data file.
<code>skipaffym</code>	Number of lines to skip in the Affymetrix file.

**Details**

Affymetrix file has following format:

some information...

some information...

some information...

SNPID Call Confidence others column ...

```

AFFX-7317060 AB 0.01709367 ...
AFFX-7317061 BB 0.01683776 ...
AFFX-7317067 AB 0.01704767 ...
AFFX-7317077 AB 0.01817814 ...
AFFX-7317078 AA 0.0006741961 ...
AFFX-7317079 AA 0.004776776 ...
AFFX-7317063 AB 0.006349149 ...
AFFX-7317064 AB 0.04771883 ...
AFFX-7317067 AA 0.04387166 ...

```

The first several lines do not contain genotype information and have to be skipped. Skipped numbers of lines can be setted. by setting skipaffym input parameter. For above example it has to be skipaffym=3.

Every row corresponds to a SNP. The first column is snp name, the second - genotype. The second column can contain letters (AA, AB, BB) or figures (1, 2, 3). Another values consider as unmeasured.

All affymetrix files must have same SNP amount and same SNP order.

The first two lines in the map file will be skipped.

If SNP does not exist in map (annotation) file this SNP will be skipped.

Output will be written into file pointed in outfile.

### Value

Does not return any value, but writes file with GenABEL raw data

### Note

The function does not check if "outfile" already exists, thus it is always over-written

### Author(s)

Maksim Struchalin

### See Also

[load.gwaa.data](#), [convert.snp.text](#), [convert.snp.mach](#), [convert.snp.tped](#) [convert.snp.illumina](#)

### Examples

```

## Not run:
      convert.snp.affymetrix(dir="where_is_our_aff_files", map="map_file", outfile="output")
## End(Not run)

```

---

```
convert.snp.illumina
```

*function to convert genotypic data from Illumina/Affymetrix to internal format*

---

## Description

Converts genotypic data from Illumina/Affymetrix-like format to internal genotypic data formatted file

## Usage

```
convert.snp.illumina(infile, outfile, strand = "+", bcast = 10000000)
```

## Arguments

<code>infile</code>	Pre-madekep linkage genotypic data file name
<code>outfile</code>	Output data file
<code>strand</code>	Specification of strand, one of "u" (unknown), "+", "-" or "file". In the latter case, extra column specifying the strand (again, one of "u", "+", or "-") should be included on the infile.
<code>bcast</code>	Reports progress after reading bcast portion of SNP genotypes

## Details

Input file is the one which could be typically obtained from Illumina BeadStudio software. For example:

```
Name Chr Pos id1 id2 id3
rs1001 2 12897 AC AA AA
rs2401 3 12357 AG GG AG
rs123 3 5327 TC TT CC
```

Here, every row corresponds to a SNP, and each column, starting with the 4th, corresponds to a person.

When strand information is available (option strand="file"), the file should look like

Accepted allele codes: 1/2, A/B, A/T, A/G, A/C, T/G, T/C, G/C, A/-, T/-, G/-, C/-. Here, "-" stands of a deletion. Missing data can be coded as "-" or "00". Make sure that the coding for missing is "00" if you use one of the codings A/-, T/-, G/-, C/-!

```
Name Chr Pos Strand id1 id2 id3
rs1001 2 12897 + AC AA AA
rs2401 3 12357 - AG GG AG
rs123 3 5327 + TC TT CC
```

Accepted strand coding: +, -, u (unknown)

The procedure always codes genotypes that "0", "1" and "2" correspond to AA, AB, and BB, where B is the less frequent allele. Thus GWA analysis procedures will return effect of the minor allele.

### Value

Does not return any value, but writes file with GenABEL raw data

### Note

The function does not check if "outfile" already exists, thus it is always over-written

### Author(s)

Yurii Aulchenko

### See Also

[load.gwaa.data](#), [convert.snp.text](#), [convert.snp.mach](#), [convert.snp.tped](#)

### Examples

```
#
# convert.snp.illumina(infile="pedin.18",out="genos.raw",strand="+")
#
```

---

<code>convert.snp.mach</code>	<i>function to convert genotypic data from MACH format to internal data format</i>
-------------------------------	--

---

### Description

Converts genotypic data from MACH format to internal genotypic data formatted file

### Usage

```
convert.snp.mach(pedfile, mapfile, infofile, outfile, quality = 0.9, column.quality)
```

### Arguments

<code>pedfile</code>	File with genotypic data from MACH (geno or mlgeno)
<code>mapfile</code>	Name of the map file
<code>infofile</code>	Name MACH info-file
<code>outfile</code>	Output data file
<code>quality</code>	Drop the SNPs with quality (as specified in some column of info-file) lower than this threshold.



column.quality	What column of the info-file provides "quality". Default = 7 or r2; possible values include 6 (average posterior probability).
strand	Specification of strand, one of "u" (unknown), "+", "-" or "file". In the latter case, map-file should contain an extended map (the one including strand and coding). See options to <code>convert.snp.ped</code> for details.
...	Other arguments passed to <code>convert.snp.ped</code>

### Details

This is a simple script converting the MACH data with `convert.snp.ped`, re-loading data, and filtering the `snp.data` object based on quality as specified in MACH info-file

### Value

Does not return any value, but writes file with GenABEL raw data

### Note

The function does not check if "outfile" already exists, thus it is always over-written

### Author(s)

Yurii Aulchenko

### See Also

`load.gwaa.data`, `convert.snp.illumina`, `convert.snp.text`, `convert.snp.ped`,  
`convert.snp.tped`

### Examples

```
#
# convert.snp.mach(ped="pedin.18", map="map.18", out="genos.raw")
#
```

---

<code>convert.snp.ped</code>	<i>function to convert genotypic data in pedigree format (+map) to internal data format</i>
------------------------------	---

---

### Description

Converts genotypic data in a variety of pedigree formats (+map) to internal genotypic data formatted file

**Usage**

```
convert.snp.ped(pedfile, mapfile, outfile, format = "premakeped", traits = 1,
               strand = "u", bcast = 10000000, wslash=F)
```

**Arguments**

pedfile	Pre-madekep linkage genotypic data file name
mapfile	Name of the map file
outfile	Output data file
format	Input data format, either "premakeped" (default, also works with Merlin files), or "mach"
traits	How many traits are specified in the pedigree file (usually 1 – affection – or 2 – affection and liability). Has no effect when format = "mach".
strand	Specification of strand, one of "u" (unknown), "+", "-" or "file". In the latter case, map-file should contain an extended map (the one including strand and coding)
bcast	Reports progress after reading bcast portion of SNPs
wslash	Whether alleles are separated with slash (is true for Mach/Merlin format), otherwise it is assumed that alleles are separated with space. When wslash=T it is assumed that genotypes are coded with single characters, separated with slash (no spaces), e.g. "A/G", and not "A/ G" or "A / G".

**Details**

Pedfile must be standard pre-madekep/Merlin linkage file, or a Mach file. In pre-madekep linkage file, columns are

```
ped id fa mo sex trait snp1.allele1 snp1.allele2 snp2.allele1 snp2.allele2 ...
```

For example

```
1 1 0 0 1 2 A A G T ...
```

```
1 2 0 0 1 0 A G T T ...
```

```
1 3 0 0 2 1 A A T T ...
```

...

Would imply that persons 1, 2 and 3 are "founders" (which would be typical for a case-control study), 1 and 2 are males and 3 is female. Person 1 is homozygous for allele 1 at locus 1 and heterozygous at locus 2. Person 2 is heterozygous at both loci. Person 3 is homozygous for allele 2 at locus 1 and allele 1 at locus 2.

Only the second and the marker columns are used, thus make sure the IDs are unique!

Accepted allele codes: 1/2, A/B, A/T, A/G, A/C, T/G, T/C, G/C, A/-, T/-, G/-, C/-. Here, "-" stands of a deletion.

The map file is standard Merlin map. For example:

```
chrom name position
```

```
18 rs679153 2859916
```

```
18 rs9965482 2860891
```

Says that locus 1 is named rs679153 and located at chromosome 18 position 2859916. Locus 2 (rs9965482) is located at chromosome 18, position 2860891.

In extended map format, there should be 4th column specifying the strand

chrom name position strand

```
18 rs679153 2859916 -
```

```
18 rs9965482 2860891 +
```

Accepted strand coding: +, -, u (unknown)

Please note that the header line (e.g. "chrom name position") SHOULD be present in your file!

### Value

Does not return any value, but writes file with GenABEL raw data

### Note

The function does not check if "outfile" already exists, thus it is always over-written

### Author(s)

Yurii Aulchenko

### See Also

[load.gwaa.data](#), [convert.snp.illumina](#), [convert.snp.mach](#), [convert.snp.text](#),  
[convert.snp.tped](#)

### Examples

```
#
# convert.snp.ped(ped="pedin.18", map="map.18", out="genos.raw")
#
```

---

<code>convert.snp.text</code>	<i>function to convert integer genotypic data file to raw internal data formatted file</i>
-------------------------------	--

---

### Description

Converts integer genotypic data file to raw internal data formatted file

### Usage

```
convert.snp.text(infile, outfile, bcast = 10000)
```

**Arguments**

<code>infile</code>	Input data file name
<code>outfile</code>	Output data file
<code>bcast</code>	Reports progress after reading bcast portion of SNPs

**Details**

Input genotypic data file contains all kind of genetic information. The first line of this file contains IDs of all study subjects. The second line gives names of all SNPs in the study. The third line list the chromosomes the SNPs belong to. Sequential numbers are used for autosomes and "X" (capital!) is used for the sex-chromosome. The forth line lists genomic position of the SNPs, in order which is the same as order in the line 2. The genomic position can be chromosome-specific (each chromosome starts with "0") or, better, a true genomic position (chromosome 1 starts with 0 and chromosome 2 continues at the point chromosome 1 ends).

Starting with the line five, genetic data are presented. The 5th line contains the data for SNP, which is listed first on the second line. The first column of this line specifies the genotype for the person, who is listed first on the line 1; the second column gives the genotype for the second person, so on. The genotypes are coded as 0 (missing), 1 (for AA), 2 (for AB) and 3 (for BB). Here is a small example:

```
289982 325286 357273 872422 1005389
SNP-1886933 SNP-2264565 SNP-2305014
1 1 1
825852 2137143 2585920
3 3 3 3 2
3 2 3 3 3
2 2 1 1 1
```

In this example, we can see that SNP-2305014 (number 3 in the second line) is located on chromosome 1 at the position 2585920. If we would like to know what is genotype of person with ID 325286 (second in the first line), we need to take second column and the third line of the genotypic data. This cell contains 1, thus, person 325286 has genotype "AA" at SNP-2305014.

In the event that you do not want to use a map for some reason (such as prior ordering of the polymorphisms in the genotype file), make a dummy map-line, which contains order information.

The above described genotypic data file is (more or less) human-readable; actually, to achieve the aim of effective data storage GWAA package uses internal format. In this format, four genotypes are stored in single byte; "raw" data format of R is used.

**Value**

Does not return any value

**Note**

The function does not check if "outfile" already exists, thus it is always over-written

**Author(s)**

Yurii Aulchenko

**See Also**

`load.gwaa.data`, `convert.snp.illumina`, `convert.snp.ped`, `convert.snp.mach`,  
`convert.snp.tped`

**Examples**

```
#  
# convert.snp.text("genos.dat", "genos.raw")  
#
```

---

<code>convert.snp.tped</code>	<i>function to convert genotypic data in transposed-ped format (.tped and .tfam) to internal genotypic data formatted file</i>
-------------------------------	--

---

**Description**

Converts genotypic data in transposed-ped format (.tped and .tfam) to internal genotypic data formatted file

**Usage**

```
convert.snp.tped(tpedfile, tfamfile, outfile, strand = "+", bcast = 10000)
```

**Arguments**

<code>tpedfile</code>	Name of transposed-ped format (.tped) file to read
<code>tfamfile</code>	Name of individual data (.tfam) file to read
<code>outfile</code>	Name for output data file
<code>strand</code>	Specification of strand, one of "u" (unknown), "+", "-" or "file". In the latter case, extra column specifying the strand (again, one of "u", "+", or "-") should be included on the infile.
<code>bcast</code>	Reports progress every time this number of SNPs have been read

**Details**

The transposed-ped file format may be preferred when extremely large numbers of markers have been genotyped. This file format is supported by plink! See <http://pngu.mgh.harvard.edu/purcell/plink/> for details.

The conversion is performed by C++ code that is both fast and memory efficient.

The genotype data are stored in the main transposed-ped format file, usually with a .tped file extension. If there are NSNP markers genotyped in NIND individuals, this file has NSNP rows and  $4+NIND*2$  columns. There is one row per marker, and no header. The first four columns are:

Chromosome

Marker name (e.g. rs number)

Genetic position (in Morgans)

Physical position (in bp)

These are followed by two columns per individual, which contain the genotype, coded as two characters. The '0' character is used for missing data. For example, a file containing data for six individuals genotyped at two SNPs would look like:

```
1 rs1234 0 5000650 A A 0 0 C C A C C C C C
```

```
1 rs5678 0 5000830 G T G T G G T T G T T T
```

In this example, the second individual is missing data for SNP rs1234, etc. The alleles can be coded by any two distinct characters, e.g. 'C' and 'G', or '1' and '2'. The '0' character is reserved for missing data, and each individual genotype must be either complete, or completely missing. In the current implementation, only the physical positions of the SNPs are read, and the genetic positions are ignored.

The indices for the columns are stored in a separate file, usually with a .tfam file extension. Traditionally, this file has six columns, and no header. In the current implementation, only the second column is used. This column must contain the individual id. Other columns are ignored.

### Value

Does not return any value

### Note

The function does not check if "outfile" already exists, thus it is always over-written

### Author(s)

Toby Johnson <toby.johnson@unil.ch>

### See Also

[convert.snp.ped](#), [convert.snp.illumina](#), [convert.snp.text](#), [convert.snp.mach](#),  
[load.gwaa.data](#)

### Examples

```
#
# convert.snp.tped("c21.tped",map="c21.tfam",out="c21.raw")
#
```

---

crnames	<i>Return column and row names</i>
---------	------------------------------------

---

**Description**

Given a dimnames, returns column and row names for index cells

**Usage**

```
crnames(dnames, idx)
```

**Arguments**

dnames	object <code>dimnames</code>
idx	index (or logical condition on the original object)

**Examples**

```
data(ge03d2ex)
a <- as.numeric(ge03d2ex[1:20,1:3])
crnames(dimnames(a), a==1)
```

---

descriptives.marker
---------------------

*Function to generate descriptive summary tables for genotypic data*

---

**Description**

Function to generate descriptive summary tables for genotypic data

**Usage**

```
descriptives.marker(data, snpsubset, idsubset, file, mafc, hwec, snpc, idcc, digits)
```

**Arguments**

data	an object of <code>snp.data-class</code> or <code>gwaa.data-class</code>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data</code> are used for analysis.
file	A string specifying the name of a file to write the tables to (default is missing).
mafc	vector containing desired cut-off levels for minor allele frequency

hwec	vector containing desired cut-off levels for exact HWE P-values
snp	vector containing desired cut-off levels for SNP call rate
idcc	vector containing desired cut-off levels for individual SNP call rate
digits	number of digits to be printed

**Value**

A list containing descriptive tables and statistics

**Author(s)**

Yurii Aulchenko

**Examples**

```
data(srdta)
descriptives.marker(srdta)
```

---

`descriptives.scan` *Function to describe "top" hits in GWA scan*

---

**Description**

Describes "top" hits in GWA scan

**Usage**

```
descriptives.scan(data, file, top=10, sortby="P1df", digits = 10, sep = "\t")
```

**Arguments**

data	an object of <code>snp.data-class</code> or <code>gwaa.data-class</code>
file	A string specifying the name of a file to write the tables to (default is no file output).
top	How many "top" hits to describe
sortby	How to pick up "top" hits ("P1df", "P2df", "Pgw1df", "Pgw2df")
digits	number of digits to be printed
sep	field separator (takes effect only if file argument provided)

**Value**

A descriptive table

**Author(s)**

Yurii Aulchenko



## Examples

```
data(srdta)
x <- qtscore(qt2,srdta)
descriptives.scan(x)
```

---

`descriptives.trait` *Function to generate descriptive summary tables for phenotypic data*

---

## Description

Function to generate descriptive summary tables for phenotypic data

## Usage

```
descriptives.trait(data, subset, file, by.var=NULL, digits = 3)
```

## Arguments

<code>data</code>	an object of <code>snp.data-class</code> or <code>gwaa.data-class</code>
<code>subset</code>	Subset of people to run analysis on. If missing, all people from data are used for analysis.
<code>file</code>	A string specifying the name of a file to write the tables to (default is no file output).
<code>by.var</code>	a binary trait; statistics will be produced separately for the groups and compared
<code>digits</code>	number of digits to be printed

## Value

A table with descriptive statistics. Ptt: t-test; Pkw: kruskal.test; Pex: Fisher exact test (for factors with <5 levels)

## Author(s)

Yurii Aulchenko

## Examples

```
data(srdta)
descriptives.trait(srdta)
descriptives.trait(srdta, by.var=srdta@phdata$sex)
```

---

`dprfast`*Estimates  $D'$  between multiple markers*

---

**Description**

Given a set of SNPs, computes a matrix of  $D'$

**Usage**

```
dprfast(data, snpsubset, idsubset)
```

**Arguments**

<code>data</code>	object of <a href="#">snp.data-class</a>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data</code> are used for analysis.

**Details**

The function is based on slightly modified code of Hao et al.

**Value**

A (Nsnps X Nsnps) matrix giving  $D'$  values between a pairs of SNPs above the diagonal and number of SNP genotype measured for both SNPs below the diagonal

**Author(s)**

Yurii Aulchenko

**References**

Hao K, Di X, Cawley S. (2006) LdCompare: rapid computation of single- and multiple-marker  $D'$  and genetic coverage. *Bioinformatics*, 23: 252-254.

**See Also**

[rhofast](#)

## Examples

```
data(ge03d2)
# D's using D'fast
a <- dprfast(ge03d2,snps=c(1:10))
# D's using package genetics
b <- LD(as.genotype(ge03d2[,1:10]))$"D'"
# see that the D's are not exactly the same
cor(a[upper.tri(a)],b[upper.tri(b)])
plot(a[upper.tri(a)],b[upper.tri(b)])
```

---

egscore.old

*Fast score test for association, corrected with PC*


---

## Description

Fast score test for association between a trait and genetic polymorphism, adjusted for possible stratification by principal components.

## Usage

```
egscore.old(formula,data,snpsubset,idsubset,kinship.matrix,naxes=3,strata,times=1,quiet=FALSE,bcast=FALSE)
```

## Arguments

formula	Formula describing fixed effects to be used in analysis, e.g. $y = a + b$ means that outcome (y) depends on two covariates, a and b. If no covariates used in analysis, skip the right-hand side of the equation.
data	An object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
kinship.matrix	kinship matrix, as returned by <a href="#">ibs</a> , (use weight="freq"!)
naxes	Number of axes of variation to be used in adjustment (should be much smaller than number of subjects)
strata	Stratification variable. If provided, scores are computed within strata and then added up.
times	If more than one, the number of replicas to be used in derivation of empirical genome-wide significance.
quiet	do not print warning messages
bcast	If the argument times > 1, progress is reported once in bcast replicas

<code>clambda</code>	If inflation facot Lambda is estimated as lower then one, this parameter controls if the original P1df ( <code>clambda=TRUE</code> ) to be reported in Pc1df, or the original 1df statistics is to be multiplied onto this "deflation" factor ( <code>clambda=FALSE</code> ). If a numeric value is provided, it is used as a correction factor.
<code>propPs</code>	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <a href="#">estlambda</a>

## Details

The idea of this test is to use genomic kinship matrix to first, derive axes of genetic variation (principal components), and, second, adjust both trait and genotypes onto these axes.

The traits is first analysed using LM and with covariates as specified with formula and also with axes of variation as predictors. Corrected genotypes are defined as residuals from regression of genotypes onto axes (which are orthogonal). Correlaton between corrected genotypes and phenotype is computed, and test statistics is defined as square of this correlation times  $(N - K - 1)$ , where N is number of genotyped subjects and K is the number of axes.

This test is defined only for 1 d.f.

## Value

Object of class [scan.gwaa-class](#)

## Author(s)

Yurii Aulchenko

## References

Price A. L. et al, Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet 38: 904-909.

## See Also

[qtscore](#), [mmscore](#), [ibs](#), [scan.gwaa-class](#)

## Examples

```
data(ge03d2ex)
#egscore.old with stratification
gkin <- ibs(ge03d2ex,w="freq")
a <- egscore.old(dm2~sex+age,data=ge03d2ex,kin=gkin)
plot(a,df="Pc1df")
```

egscore

*Fast score test for association, corrected with PC***Description**

Fast score test for association between a trait and genetic polymorphism, adjusted for possible stratification by principal components.

**Usage**

```
egscore(formula, data, snpsubset, idsubset, kinship.matrix, naxes=3, strata, times=1, quiet)
```

**Arguments**

<code>formula</code>	Formula describing fixed effects to be used in analysis, e.g. <code>y ~ a + b</code> means that outcome (y) depends on two covariates, a and b. If no covariates used in analysis, skip the right-hand side of the equation.
<code>data</code>	An object of <code>gwaa.data-class</code>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data/cc</code> are used for analysis.
<code>kinship.matrix</code>	kinship matrix, as returned by <code>ibs</code> , (use <code>weight="freq"</code> !)
<code>naxes</code>	Number of axes of variation to be used in adjustment (should be much smaller than number of subjects)
<code>strata</code>	Stratification variable. If provided, scores are computed within strata and then added up.
<code>times</code>	If more than one, the number of replicas to be used in derivation of empirical genome-wide significance.
<code>quiet</code>	do not print warning messages
<code>bcast</code>	If the argument <code>times &gt; 1</code> , progress is reported once in bcast replicas
<code>clambda</code>	If inflation factor Lambda is estimated as lower than one, this parameter controls if the original P1df ( <code>clambda=TRUE</code> ) to be reported in Pc1df, or the original 1df statistics is to be multiplied onto this "deflation" factor ( <code>clambda=FALSE</code> ). If a numeric value is provided, it is used as a correction factor.
<code>propPs</code>	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <code>estlambda</code>

## Details

The idea of this test is to use genomic kinship matrix to first, derive axes of genetic variation (principal components), and, second, adjust both trait and genotypes onto these axes.

The traits is first analysed using LM and with covariates as specified with formula and also with axes of variation as predictors. Corrected genotypes are defined as residuals from regression of genotypes onto axes (which are orthogonal). Correlation between corrected genotypes and phenotype is computed, and test statistics is defined as square of this correlation times  $(N - K - 1)$ , where  $N$  is number of genotyped subjects and  $K$  is the number of axes.

This test is defined only for 1 d.f.

## Value

Object of class `scan.gwaa-class`

## Note

Original method of Price et al. uses covariance for both diagonal and off-diagonal elements of the matrix on which PC are computed. Kinship matrix returned by GenABEL contains  $0.5 + F$  ( $\neq$  covariance) on the diagonal. As a temporary solution we now replace diagonal elements with 0.5

## Author(s)

Yurii Aulchenko

## References

Price A. L. et al, Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet 38: 904-909.

## See Also

`qtscore`, `mmscore`, `ibs`, `scan.gwaa-class`

## Examples

```
data(ge03d2ex)
#egscore with stratification
gkin <- ibs(ge03d2ex,w="freq")
a <- egscore(dm2~sex+age,data=ge03d2ex,kin=gkin)
plot(a,df="Pc1df")
```

emp.ccfast

*Genome-wide significance for a case-control GWA scan***Description**

Genome-wide significance for a case-control GWA scan. Analysis function is [ccfast](#).

**Usage**

```
emp.ccfast(y, data, snpsubset, idsubset, times = 100, quiet=FALSE,
          bcast = 10)
```

**Arguments**

All arguments are the same as in and passed intact to the [ccfast](#). See help for this function.

y	character name of the vector of case-control status. Cases are denoted as 1 and controls as 0.
data	An object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data are used for analysis.
times	If more then one, the number of replicas to be used in derivation of empirical genome-wide significance. See <a href="#">emp.qtscore</a> , which calls qtscore with times>1 for details
quiet	do not print warning messages
bcast	If the argument times > 1, progress is reported once in bcast replicas

**Details**

In the analysis of empirical significance, first time the function [ccfast](#) is called and result object is saved. Later, the function [ccfast](#) is called `times` times with `replace=FALSE` in order to generate the distribution under the null. Each call, minimal P-value is extracted and compared with original P-values. For a particular SNP, empirical P-value is obtained as a proportion of times minimal Ps from resampled data was less then the original P.

The list elements `effB`, `effAB` and `effBB` are the ones obtained from the analysis of the original (not permuted) data set

**Value**

Object of class [scan.gwaa-class](#)

**Author(s)**

Yurii Aulchenko

**See Also**[ccfast](#), [emp.qtscore](#), [scan.gwaa-class](#)**Examples**

```

data(srdta)
a<-ccfast("bt",data=srdta,snps=c(500:800))
plot(a)
# this does not make sense, as the whole experiment must be analysed, not a small region!
b<-emp.ccfast("bt",data=srdta,snps=c(500:800),bcast=10)
plot(b)
# compare qvalues and empirical P
qv<-qvaluebh95(a$P1df)$qval
qv
b$P1df
plot(qv,b$P1df,xlim=c(0,1),ylim=c(0,1))
abline(a=0,b=1)

```

emp.qtscore

*Genome-wide significance for a GWA scan***Description**

Genome-wide significance for a GWA scan. Analysis function is [qtscore](#).

**Usage**

```

emp.qtscore(formula , data, snpsubset, idsubset, strata, trait.type="gaussian",
             quiet=FALSE, bcast = 10)

```

**Arguments**

All arguments are the same as in and passed intact to the [qtscore](#). See help for this function.

formula	Formula describing fixed effects to be used in analysis, e.g. $y = a + b$ means that outcome (y) depends on two covariates, a and b. If no covariates used in analysis, skip the right-hand side of the equation.
data	An object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.



strata	Stratification variable. If provided, scores are computed within strata and then added up.
trait.type	"gaussian" or "binomial". If not specified, the procedure guesses the type
times	If more then one, the number of replicas to be used in derivation of empirical genome-wide significance. See <a href="#">emp.qtscore</a> , which calls qtscore with times>1 for details
quiet	do not print warning messages
bcast	If the argument times > 1, progress is reported once in bcast replicas

## Details

In the analysis of empirical significance, first time the function [qtscore](#) is called and result object is saved. Later, the function [qtscore](#) is called `times` times with `replace=FALSE` in order to generate distribution under the null. Each call, minimal P-value is extracted and compared with original P-values. For a particular SNP, empirical P-value is obtained as a proportion of times minimal Ps from resampled data was less then original P.

The list elements `effB`, `effAB` and `effBB` are the ones obtained from the analysis of the original (not permuted) data set

The function does not yet implement correct analysis for X-linked data.

## Value

Object of class [scan.gwaa-class](#)

## Author(s)

Yurii Aulchenko

## See Also

[qtscore](#), [emp.ccfast](#), [scan.gwaa-class](#)

## Examples

```
data(srdta)
a<-qtscore(qt3~age+sex,data=srdta,snps=c(1:200))
plot(a)
# this does not make sense, as the whole experiment must be analysed, not a small region!
b<-emp.qtscore(qt3~age+sex,data=srdta,snps=c(1:200))
plot(b)
```

estlambda

*Estimate the inflation factor for a distribution of P-values*

---

**Description**

Estimate the inflation factor for a distribution of P-values or 1df chi-square test. The major use of this procedure is the Genomic Control, but can also be used to visualise the distribution of P-values coming from other tests.

**Usage**

```
estlambda(data, plot = TRUE, proportion = 1.0, ... )
```

**Arguments**

data	A vector of reals. If all are $\leq 1$ , it is assumed that this is a vector of P-values, else it is treated as a vector of chi-squares with 1 d.f.
plot	Whether the plot should be presented
proportion	The proportion of lowest P (Chi2) to be used when estimating the inflation factor Lambda
...	arguments passed to <a href="#">plot</a> function

**Value**

A list with elements	
estimate	Estimate of Lambda
se	Standard error of the estimate

**Author(s)**

Yurii Aulchenko

**See Also**

[ccfast](#), [qtscore](#)

**Examples**

```
data(srdta)
pex <- summary(srdta@gtdata)[,"Pexact"]
estlambda(pex)
a <- ccfast("bt", srdta)
a$lambda
```

---

export.merlin	<i>function to export GenABEL data in merlin format</i>
---------------	---

---

## Description

Exports GenABEL data to Merlin and other pedigree formats

## Usage

```
export.merlin(data, pedfile = "merlin.ped", datafile = "merlin.dat", mapfile = "merlin.map",  
format = "merlin", fixstrand = "no", extendedmap = TRUE, traits = 1)
```

## Arguments

data	gwaa.data object
pedfile	Output pedigree data file name
datafile	Output data (information) file name
mapfile	Output map file name
format	Output format: reserved for future use, currently only "merlin"
fixstrand	"no" – the strand information and coding comes from the data; "+" – change all coding to "+" strand, "-" – change all coding to "-" strand
extendedmap	if TRUE extended map (+ strand, + coding) is saved with the name "mapfile.ext", where "mapfile" is the parameter supplied by user
traits	How many fake traits to insert before first column of marker data

## Details

The use is straightforward, with only the "fixstrand" option requiring some explanation. Consider a SNP on "-" strand with alleles G and A. If this SNP is accessed on "+" strand, the corresponding alleles would be C and T. While for example Affymetrix reports SNPs on bot "+" and "-" strands, HapMap reports coding on "+" strand only. To make data compatible, and/or to run imputations, one will need to convert all SNP codes to "+" strand. This can be achieved by running export.merlin() with fixstrand="+" parameter.

## Value

No value returned

## Author(s)

Yurii Aulchenko

## See Also

To load the data to GenABEL again, use `convert.snp.ped`, `load.gwaa.data`.

## Examples

```
#
# load(srdta)
# export.merlin(srdta[1:50,1:3])
#
```

---

formetascore	<i>function to run GWA analysis oriented for future meta-analysis</i>
--------------	---

---

## Description

Function to run GWA analysis – using all functions available in GenABEL – and produce output oriented for future meta-analysis

## Usage

```
formetascore(formula, data, stat = qtscore, transform = ztransform,
              build = "unknown", verbosity = 1, ...)
```

## Arguments

formula	standard formula
data	object of <code>gwaa.data-class</code>
stat	which GWA analysis function to apply. Could be <code>mlreg</code> , <code>qtscore</code> , <code>mmscore</code> , <code>grammar</code> , <code>egscore</code> , etc.
transform	Which trait transform to apply, could be <code>ztransform</code> or <code>rntransform</code> . It can also be equal to "no" if no transformation is required.
build	if you need that in output, specify genomic build here (e.g. "35")
verbosity	how much output is produced? Possible values are 0, 1, and 2
...	further arguments, passed to the "stat" GWA analysis function

## Details

This function should be applied to analysis of quantitative traits, if meta-analysis is aimed afterwards.

A transformation is applied to the formula-defined residual, and the resulting trait is analysed with specified function. Results are arranged as data-frame.

## Value

Data frame, containing GWA summary. The fields include: (1) SNP name (2) chromosome (3) position (4) number of people with available data (5) effect of the allele (6) standard error of the effect (7) P-value for the test (8) corrected P-value (we will use Genomic Control) (9) coding, with reference allele coming first (10) strand (11) frequency of the reference allele (12) Exact P-value for HWE test, etc. (depends on "verbosity" parameter).

**Author(s)**

Yurii Aulchenko

**See Also**[ztransform](#), [qtscore](#)**Examples**

```
data(ge03d2c)
x <- formetascore(bmi ~ sex+age, ge03d2c)
x[1:10,]
x <- formetascore(bmi ~ sex+age, ge03d2c, trans=rntransform)
x[1:10,]
x <- formetascore(bmi ~ sex+age, ge03d2c, trans=rntransform, verbosity=2)
x[1:10,]
```

---

GASurv*Makes survival data object for reg.gwaa*

---

**Description**Helper to [mlreg](#): makes survival data object**Usage**

```
GASurv(fuptime, status)
```

**Arguments**

fuptime	Follow-up time
status	status (1=event, 0=censored)

**Value**

Matrix with column 1 = follow-up time, and 2 = status

**Author(s)**

Yurii Aulchenko

**See Also**[mlreg](#)

---

ge03d2

GWA-type data on few small region

---

### Description

ge03d2 A small data set (approximately 1,000 people and 8,000 SNPs) containing data on 3 autosomes and X chromosome. Is a good set for demonstration of the QC procedures (different genotyping errors are introduced) and GWA analysis. Run `demo(ge03d2)` to see a demo. This data set was developed for the "Advances in population- based studies" (Ge03) course of the Nihes.

ge03d2c A small data set (approximately 200 people and 8,000 SNPs) containing data on 3 autosomes and X chromosome. This data set is complementary to [ge03d2](#).

ge03d2ex A small data set (approximately 150 people and 4,000 SNPs) containing data on 3 autosomes and X chromosome. Is a good set for demonstration of the QC procedures (different genotyping errors are introduced) and GWA analysis. This data set was developed for the "Advances in population- based studies" (Ge03) course of the Nihes. See vignette "GenABEL-tutorial.pdf" for details.

The data sets with extension ".clean" are sets after QC.

### Usage

```
data(ge03d2)
```

### Examples

```
#main example: use this to see full functionality
# demo(ge03d2)

# load and work with ge03d2
data(ge03d2)
a <- qtscore(dm2, ge03d2)
plot(a)
```

---

GenABEL-package

*GenABEL: an R package for Genome Wide Association Analysis*


---

### Description

Genome-wide association (GWA) analysis is a tool of choice for identification of genes for complex traits. Effective storage, handling and analysis of GWA data represent a challenge to modern computational genetics. GWA studies generate large amount of data: hundreds of thousands of single nucleotide polymorphisms (SNPs) are genotyped in hundreds or thousands of patients and controls. Data on each SNP undergoes several types of analysis: characterization of frequency

distribution, testing of Hardy-Weinberg equilibrium, analysis of association between single SNPs and haplotypes and different traits, and so on. Because SNP genotypes in dense marker sets are correlated, significance testing in GWA analysis is preferably performed using computationally intensive permutation test procedures, further increasing the computational burden.

To make GWA analysis possible on standard desktop computers we developed GenABEL library which addresses the following objectives:

(1) Minimisation of the amount of rapid access memory (RAM) used and the time required for data transactions. For this, we developed an effective data storage and manipulation model.

(2) Maximisation of the throughput of GWA analysis. For this, we designed optimal fast procedures for specific genetic tests.

Imbedding GenABEL into R environment allows for easy data characterisation, exploration and presentation of the results and gives access to a wide range of standard and special statistical analysis functions available in base R and specific R packages, such as "haplo.stats", "genetics", etc.

## Details

Package:	GenABEL
Type:	Package
Version:	1.4-1
Date:	2008-11-26
License:	GPL (>= 2)

To see (more or less complete) functionality of GenABEL, try running `demo(ge03d2)`.

Other demo of interest could be run with `demo(srda)`. Depending on your user privileges in Windows, it may well not run. In this case, try `demo(srdaWIN)`.

The most important functions and classes are:

For converting data from other formats, see `convert.snp.illumina` (Illumina/Affymetrix-like format). This is our preferred converting function, very extensively tested. Other conversion functions include: `convert.snp.text` (conversion from human-readable GenABEL format), `convert.snp.ped` (Linkage, Merlin, Mach, and similar files), `convert.snp.mach` (Mach-format), `convert.snp.tped` (from PLINK TPED format), `convert.snp.affymetrix` (BRML-style files).

To load the data, see `load.gwa.data`.

For data management and manipulations see `merge.gwa.data`, `merge.snp.data`, `gwa.data-class`, `snp.data-class`, `snp.names`, `snp.subset`.

For merging extra data to the phenotypic part of `gwa.data-class` object, see `add.phdata`.

For traits manipulations see `ztransform` (transformation to standard Normal), `rntransform` (rank-transformation to normality), `npsubtreated` (non-parametric routine to "impute" trait's values in these medicated).

For quality control, see `check.trait`, `check.marker`, `HWE.show`, `summary.snp.data`, `perid.summary`, `ibs`, `hom`.

For fast analysis function, see `scan.gwa-class`, `ccfast`, `qtscore`, `mmscore`, `egscore`, `ibs`, `r2fast`, `dprfast`, `rhofast`

For specific tools facilitating analysis of the data with stratification (population stratification or (possibly unknown) pedigree structure), see `qtscore` (implements basic Genomic Control), `ibs` (computations of IBS / genomic IBD), `egscore` (stratification adjustment following Price et al.), `polygenic` (heritability analysis), `mmscore` (score test of Chen and Abecasis), `grammar` (grammar test of Aulchenko et al.).

For functions facilitating construction of tables for your manuscript, see `descriptives.marker`, `descriptives.trait`, `descriptives.scan`.

For meta-analysis and related, see help on `formetascore`.

For link to WEB databases, see `show.ncbi`.

For interfaces to other packages and standard R functions, also for 2D scans, see `scan.glm`, `scan.glm.2D`, `scan.haplo`, `scan.haplo.2D`, `scan.gwaa-class`, `scan.gwaa.2D-class`.

For graphical facilities, see `plot.scan.gwaa`, `plot.check.marker`.

## Author(s)

Yurii Aulchenko

Maintainer: Yurii Aulchenko <i.aoulchenko@erasmusmc.nl>

## References

If you use GenABEL package in your analysis, please cite the following work:

Aulchenko Y.S., Ripke S., Isaacs A., van Duijn C.M. GenABEL: an R package for genome-wide association analysis. *Bioinformatics*. 2007 23(10):1294-6.

If you used `polygenic`, please cite

Thompson EA, Shaw RG (1990) Pedigree analysis for quantitative traits: variance components without matrix inversion. *Biometrics* 46, 399-413.

If you used environmental residuals from `polygenic` for `qtscore`, used GRAMMAR and/or GRAMMAS analysis, please cite

Aulchenko YS, de Koning DJ, Haley C. Genomewide rapid association using mixed model and regression: a fast and simple method for genome-wide pedigree-based quantitative trait loci association analysis. *Genetics*. 2007 177(1):577-85.

Amin N, van Duijn CM, Aulchenko YS. A genomic background based method for association analysis in related individuals. *PLoS ONE*. 2007 Dec 5;2(12):e1274.

If you used `mmscore`, please cite

Chen WM, Abecasis GR. Family-based association tests for genome-wide association scans. *Am J Hum Genet*. 2007 Nov;81(5):913-26.

For exact HWE (used in `summary.snp.data`), please cite:

Wigginton G.E., Cutler D.J., Abecasis G.R. A note on exact tests of Hardy-Weinberg equilibrium. *Am J Hum Genet*. 2005 76: 887-893.

For haplo.stats (`scan.haplo`, `scan.haplo.2D`), please cite:

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Score tests for association between traits and haplotypes when linkage phase is ambiguous. *Am J Hum Genet*. 2002 70:425-434.

For fast LD computations (function `dprfast`, `r2fast`), please cite:



Hao K, Di X, Cawley S. LdCompare: rapid computation of single- and multiple-marker  $r^2$  and genetic coverage. *Bioinformatics*. 2006 23:252-254.

If you used [npsubtreated](#), please cite

Levy D, DeStefano AL, Larson MG, O'Donnell CJ, Lifton RP, Gavras H, Cupples LA, Myers RH. Evidence for a gene influencing blood pressure on chromosome 17. Genome scan linkage results for longitudinal blood pressure phenotypes in subjects from the framingham heart study. *Hypertension*. 2000 Oct;36(4):477-83.

### See Also

Packages `genetics`, `haplo.stats`, `qvalue`,

### Examples

```
## Not run:
demo(ge03d2)
demo(srdta)
demo(srwtawin)

## End(Not run)
```

---

grammar

*Approximate score test for association in related people*

---

### Description

Fast approximate score test for association between a trait and genetic polymorphism, in samples of related individuals. When used with argument "times=1", it is equivalent to running [qtscore](#) on "environmental residuals" from [polygenic](#). However, it does not produce correct results with permutations, because the raw trait values, which are not exchangeable, are permuted. Use [qtscore](#) on "environmental residuals" when you want to have empirical GW significance with GRAMMAR method.

### Usage

```
grammar(h2object, data, snpsubset, idsubset, strata, times=1, quiet=FALSE, bcast=10, clambda=1)
```

### Arguments

<code>h2object</code>	An object returned by <a href="#">polygenic</a> polygenic mixed model analysis routine. The sub-objects used are measuredIDs, residualY, h2an\$estimates (last element, total variance, only), and InvSigma. One can supply grammar with a fake h2object, containing these list elements.
<code>data</code>	An object of <a href="#">gwaa.data-class</a>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.

<code>idssubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data/cc</code> are used for analysis.
<code>strata</code>	Stratification variable. If provided, scores are computed within strata and then added up.
<code>times</code>	If more then one, the number of replicas to be used in derivation of empirical genome-wide significance. NOTE: do not use times > 1 unless you are really sure you understand what you are doing!
<code>quiet</code>	do not print warning messages
<code>bcast</code>	If the argument times > 1, progress is reported once in bcast replicas
<code>clambda</code>	If inflation facor Lambda is estimated as lower then one, this parameter controls if the original P1df ( <code>clambda=TRUE</code> ) to be reported in Pc1df, or the original ldf statistics is to be multiplied onto this "deflation" factor ( <code>clambda=FALSE</code> ). With GRAMMAR, Lambda is expected ot be less than 1. If a numeric value is provided, it is used as a correction factor.
<code>propPs</code>	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <code>estlambda</code>

## Details

Approximate score test is performed using the formula

$$\sigma^4 \frac{((G - E[G])V^{-1}residualY)^2}{(G - E[G])(G - E[G])}$$

where  $\sigma^4$  is the square of the residual variance,  $G$  is the vector of genotypes (coded 0, 1, 2) and  $E[G]$  is a vector of (strata-specific) mean genotypic values;  $V^{-1}$  is the `InvSigma` and `residualY` are residuals from the trait analysis with `polygenic` procedure.

Compared to score test implemented in `mmscore`, grammar test is faster and computation time grows only linearly with the number of subjects (with `mmscore` this relation is quadratic). While raw P1df from grammar are not quite correct, the GC p-values correspond very closely to these from the `mmscore`.

## Value

Object of class `scan.gwaa-class`; only 1 d.f. test is implemented currently.

## Author(s)

Yurii Aulchenko

## References

Aulchenko YS, de Koning DJ, Haley C. Genomewide rapid association using mixed model and regression: a fast and simple method for genome-wide pedigree-based quantitative trait loci association analysis. *Genetics*. 2007 177(1):577-85.

Amin N, van Duijn CM, Aulchenko YS. A genomic background based method for association analysis in related individuals. *PLoS ONE*. 2007 Dec 5;2(12):e1274.

**See Also**

[grammar](#), [qtsscore](#), [plot.scan.gwaa](#), [scan.gwaa-class](#)

**Examples**

```
# ge03d2.clean is rather bad data set to demonstrate grammar,
# because this is a population-based study
data(ge03d2.clean)
#take half for speed
ge03d2.clean <- ge03d2.clean[1:450,]
gkin <- ibs(ge03d2.clean,w="freq")
h2ht <- polygenic(height ~ sex + age,kin=gkin,ge03d2.clean)
h2ht$est
# can use "grammar", but ...
a <- grammar(h2ht,data=ge03d2.clean)
# ... use rather qtsscore (note clam=FALSE), which is a better alternative for the same thing
a <- qtsscore(h2ht$pgres,data=ge03d2.clean,clam=FALSE)
# compare to GC:
b <- qtsscore(height ~ sex + age,data=ge03d2.clean)
plot(b,df="Pc1df")
add.plot(a,df="Pc1df")
# relatively large difference is due to high heritability
# note that locus at chromosome 2 should indeed be there...
```

---

gwaa.data-class	<i>Class "gwaa.data"</i>
-----------------	--------------------------

---

**Description**

This class contains objects holding all GWAA data – phenotypes, genotypes and other relevant information

**Slots**

**phdata:** dataframe with phenotypic data used in GWAA

**gtdata:** object of class [snp.data-class](#) used to store genotypic data, map, etc.

**Methods**

[ signature(x = "gwaa.data", i = "ANY", j = "ANY", drop = "ANY"): subset operations. x[i,j] will select people listed in i and SNPs listed in j.

**show** signature(object = "gwaa.data"): shows both parts of the object. Take care that the objects are usually very large!

**summary** signature(object = "gwaa.data"): Calls standard summary to describe phenotypic part and calls [summary.snp.data](#) to [snp.data-class](#)

**Author(s)**

Yurii Aulchenko

**See Also**[snp.data-class](#), [load.gwaa.data](#), [snp.mx-class](#)**Examples**

```
data(srdta)
srdta@phdata[1:10,]
srdta@gtdata[1:10,1:12]
srdta[1:10,1:12]
as.numeric(srdta@gtdata[1:12,1:10])
# very long output:
summary(srdta)
```

---

hom.old

*function to compute average homozygosity within a person*


---

**Description**

This function computes average homozygosity (inbreeding) for a set of people, across multiple markers. Can be used for Quality Control (e.g. contamination checks)

**Usage**

```
hom.old(data, snpsubset, idsubset, weight="no")
```

**Arguments**

data	Object of <a href="#">gwaa.data-class</a> or <a href="#">snp.data-class</a>
snpsubset	Subset of SNPs to be used
idsubset	People for whom average homozygosity is to be computed
weight	When "no", homozygosity is computed as a proportion of homozygous genotypes. When "freq", an estimate of inbreeding coefficient is computed (see details).

**Details**

With "freq" option, for person  $i$  inbreeding is estimated with

$$f_i = \frac{O_i - E_i}{(L_i - E_i)}$$

where  $O_i$  is observed homozygosity,  $L_i$  is the number of SNPs measured in individual  $i$  and

$$E_i = \sum_{j=1}^{L_i} (1 - 2p_j(1 - p_j)) \frac{T_{Aj}}{T_{Aj} - 1}$$

where  $T_{Aj}$  is the number of measured genotypes at locus  $j$ .

Only polymorphic loci with number of measured genotypes >1 are used with this option.

This measure is the same as used by PLINK (see reference).

You should use as many people and markers as possible when estimating inbreeding from marker data.

### Value

With option `weight="no"`: A matrix with rows corresponding to the ID names and columns showing the number of genotypes measured (`NoMeasured`) and homozygosity (`Hom`).

With option `weight="freq"`: the same as above + expected homozygosity (`E(Hom)`) and the estimate of inbreeding, `F`.

### Author(s)

Yurii Aulchenko

### References

Purcell S. et al, (2007) PLINK: a toolset for whole genome association and population-based linkage analyses. *Am. J. Hum. Genet.*

### See Also

[ibs](#), [gwaa.data-class](#), [snp.data-class](#)

### Examples

```
data(ge03d2)
h <- hom(ge03d2[,c(1:100)])
homsem <- h[, "Hom"] * (1-h[, "Hom"]) / h[, "NoMeasured"]
plot(h[, "Hom"], homsem)
# wrong analysis: one should use all people (for right frequency) and markers (for right F)
h <- hom(ge03d2[,c(1:10)], weight="freq")
h
```

hom

*function to compute average homozygosity within a person***Description**

This function computes average homozygosity (inbreeding) for a set of people, across multiple markers. Can be used for Quality Control (e.g. contamination checks)

**Usage**

```
hom(data, snpsubset, idsubset, weight="no", snpfreq, n.snpfreq = 1000)
```

**Arguments**

data	Object of <a href="#">gwaa.data-class</a> or <a href="#">snp.data-class</a>
snpsubset	Subset of SNPs to be used
idsubset	People for whom average homozygosity is to be computed
weight	When "no", homozygosity is computed as a proportion of homozygous genotypes. When "freq", an estimate of inbreeding coefficient is computed (see details).
snpfreq	when option weight="freq" used, you can provide fixed allele frequencies
n.snpfreq	when option weight="freq" used, you can provide number of people used to estimate allele frequencies

**Details**

Without "freq" option, homozygosity is measured as proportion of homozygous genotypes observed in a person.

With "freq" option, for person  $i$  inbreeding is estimated with

$$f_i = \frac{(O_i - E_i)}{(L_i - E_i)}$$

where  $O_i$  is observed homozygosity,  $L_i$  is the number of SNPs measured in individual  $i$  and

$$E_i = \sum_{j=1}^{L_i} (1 - 2p_j(1 - p_j)) \frac{T_{Aj}}{T_{Aj} - 1}$$

where  $T_{Aj}$  is the number of measured genotypes at locus  $j$ ;  $T_{Aj}$  is either estimated from data or provided by "n.snpfreq" parameter (vector). Allele frequencies are either estimated from data or provided by the "snpfreq" vector.

Only polymorphic loci with number of measured genotypes >1 are used with this option.

This measure is the same as used by PLINK (see reference).

You should use as many people and markers as possible when estimating inbreeding from marker data.

**Value**

With option `weight="no"`: A matrix with rows corresponding to the ID names and columns showing the number of genotypes measured (NoMeasured) and homozygosity (Hom).

With option `weight="freq"`: the same as above + expected homozygosity (E(Hom)) and the estimate of inbreeding, F.

**Author(s)**

Yurii Aulchenko

**References**

Purcell S. et al, (2007) PLINK: a toolset for whole genome association and population-based linkage analyses. Am. J. Hum. Genet.

**See Also**

[ibs](#), [gwaa.data-class](#), [snp.data-class](#)

**Examples**

```
data(ge03d2)
h <- hom(ge03d2[,c(1:100)])
homsem <- h[, "Hom"] * (1-h[, "Hom"]) / h[, "NoMeasured"]
plot(h[, "Hom"], homsem)
# wrong analysis: one should use all people (for right frequency) and markers (for right F)
h <- hom(ge03d2[,c(1:10)], weight="freq")
h
```

---

HWE.show

*show HWE tables*


---

**Description**

This function displays HWE tables and shows Chi2 and exact HWE P-values

**Usage**

```
HWE.show(data, idsubset = c(1:data@gtdata@nids),
          snpsubset = c(1:data@gtdata@nsnps))
```

**Arguments**

<code>data</code>	object of class <a href="#">gwaa.data-class</a> or <a href="#">snp.data-class</a>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data/cc</code> are used for analysis.

**Value**

Only screen output

**Author(s)**

Yurii Aulchenko

**See Also**

[check.marker](#)

**Examples**

```
data(srdta)
mc <- check.marker(srdta, p.lev=0.01, ibs.mrk=0)
mc$nohwe
HWE.show(data=srdta, snps=mc$nohwe)
```

---

ibs.old

*Computes (average) Identity-by-State for a set of people and markers*

---

**Description**

Given a set of SNPs, computes a matrix of average IBS for a group of people

**Usage**

```
ibs.old(data, snpsubset, idsubset, weight="no")
```

**Arguments**

data	object of <a href="#">snp.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data are used for analysis.
weight	"no" for direct IBS computations, "freq" to weight by allelic frequency



## Details

This function facilitates quality control of genomic data. E.g. people with extremely high (close to 1) IBS may indicate duplicated samples (or twins), simply high values of IBS may indicate relatives.

When weight "freq" is used, IBS for a pair of people  $i$  and  $j$  is computed as

$$f_{i,j} = \sum_k \frac{(x_{i,k} - p_k) * (x_{j,k} - p_k)}{(p_k * (1 - p_k))}$$

where  $k$  changes from 1 to  $N$  = number of SNPs GW,  $x_{i,k}$  is a genotype of  $i$ th person at the  $k$ th SNP, coded as 0, 1/2, 1 and  $p_k$  is the frequency of the "+" allele. This apparently provides an unbiased estimate of the kinship coefficient.

Only with "freq" option monomorphic SNPs are regarded as non-informative.

ibs() operation may be very lengthy for a large number of people.

## Value

A (Npeople X Npeople) matrix giving average IBS (kinship) values between a pair below the diagonal and number of SNP genotype measured for both members of the pair above the diagonal.

On the diagonal, homozygosity (0.5+inbreeding) is provided.

## Author(s)

Yurii Aulchenko

## See Also

[check.marker](#), [summary.snp.data](#), [snp.data-class](#)

## Examples

```
data(ge03d2c)
a <- ibs(data=ge03d2c, ids=c(1:10), snps=c(1:1000))
a
# compute IBS based on a random sample of 1000 autosomal marker
a <- ibs(ge03d2c, snps=sample(ge03d2c@gtdata@snpnames[ge03d2c@gtdata@chromosome!="X"], 1000, replace=TRUE))
mds <- cmdscale(as.dist(1-a))
plot(mds)
# identify smaller cluster of outliers
km <- kmeans(mds, centers=2, nstart=1000)
cl1 <- names(which(km$cluster==1))
cl2 <- names(which(km$cluster==2))
if (length(cl1) > length(cl2)) cl1 <- cl2;
cl1
# PAINT THE OUTLIERS IN RED
points(mds[cl1,], pch=19, col="red")
```

ibs

*Computes (average) Identity-by-State for a set of people and markers***Description**

Given a set of SNPs, computes a matrix of average IBS for a group of people

**Usage**

```
ibs(data, snpsubset, idsubset, cross.idsubset, weight="no", snpfreq)
```

**Arguments**

data	object of <code>snp.data-class</code>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	IDs of people to be analysed. If missing, all people from data are used for analysis.
cross.idsubset	Parameter allowing parallel implementation. Not to be used normally. If supplied together with idsubset, the ibs/kinship for all pairs between idsubset and cross.idsubset computed.
weight	"no" for direct IBS computations, "freq" to weight by allelic frequency
snpfreq	when option weight="freq" used, you can provide fixed allele frequencies

**Details**

This function facilitates quality control of genomic data. E.g. people with extremely high (close to 1) IBS may indicate duplicated samples (or twins), simply high values of IBS may indicate relatives.

When weight "freq" is used, IBS for a pair of people i and j is computed as

$$f_{i,j} = \sum_k \frac{(x_{i,k} - p_k) * (x_{j,k} - p_k)}{(p_k * (1 - p_k))}$$

where k changes from 1 to N = number of SNPs GW,  $x_{i,k}$  is a genotype of ith person at the kth SNP, coded as 0, 1/2, 1 and  $p_k$  is the frequency of the "+" allele. This apparently provides an unbiased estimate of the kinship coefficient.

Only with "freq" option monomorphic SNPs are regarded as non-informative.

ibs() operation may be very lengthy for a large number of people.

**Value**

A (Npeople X Npeople) matrix giving average IBS (kinship) values between a pair below the diagonal and number of SNP genotype measured for both members of the pair above the diagonal.

On the diagonal, homozygosity (0.5+inbreeding) is provided.

**Author(s)**

Yurii Aulchenko

**See Also**[check.marker](#), [summary.snp.data](#), [snp.data-class](#)**Examples**

```

data(ge03d2c)
a <- ibs(data=ge03d2c,ids=c(1:10),snps=c(1:1000))
a
# compute IBS based on a random sample of 1000 autosomal marker
a <- ibs(ge03d2c,snps=sample(ge03d2c@gtdata@snpnames[ge03d2c@gtdata@chromosome!="X"],1000,replace=TRUE))
mds <- cmdscale(as.dist(1-a))
plot(mds)
# identify smaller cluster of outliers
km <- kmeans(mds,centers=2,nstart=1000)
cl1 <- names(which(km$cluster==1))
cl2 <- names(which(km$cluster==2))
if (length(cl1) > length(cl2)) cl1 <- cl2;
cl1
# PAINT THE OUTLIERS IN RED
points(mds[cl1,],pch=19,col="red")

```

---

load.gwaa.data	<i>function to load GWAA data</i>
----------------	-----------------------------------

---

**Description**

Load data (genotypes and phenotypes) from files to gwaa.data object

**Usage**

```

load.gwaa.data(phenofile = "pheno.dat", genofile = "geno.raw",
               force = TRUE, makemap = FALSE, sort = TRUE)

```

**Arguments**

phenofile	data table with phenotypes
genofile	internally formatted genotypic data file (see <a href="#">convert.snp.text</a> to convert data)
force	Force loading the data if heterozygous X-chromosome genotypes are found in male
makemap	Make a consecutive map in case if map is provided chromosome-specifically
sort	Should SNPs be sorted in ascending order according to chromosome and position?

**Details**

The genofile must be the one resulting from `convert.snp.text`, `convert.snp.ped`, `convert.snp.tped`, or `convert.snp.illumina` (see documentation for these functions for the file formats).

The phenotype file relates study subjects with their covariate and outcome values. In the phenotypic data file, the first line gives a description of the data contained in a particular column; the names should be unique, otherwise R will change them. The first column of the phenotype file **MUST** contain the subjects' unique ID, named "id"; there should also be a column named "sex" and giving sex information (0 = female, 1 = male). Other columns in the file should contain phenotypic information. Missing values should be coded with "NA"; binary traits should have values 0 or 1. An example of few first lines of a phenotype file is as follows:

```
id sex age bt1 qt qt1
"289982" 0 30.33 NA NA 3.93
"325286" 0 36.514 1 0.49 3.61
"357273" 1 37.811 0 1.65 5.30
"872422" 1 20.393 0 1.95 4.07
"1005389" 1 28.21 1 0.35 3.90
```

This file tells us that, for example, person 325286 is female (0 in second column), and she has "1" (usually this means a "case") value for the trait "bt1", so on. Person 289982 has measurements only for sex, age and qt1, while other measurements are missing (NA, Not Available).

IDs are better kept in quotation (this would keep away the problem of e.g., leading zeros).

**Value**

Object of class gwaa.data

**Author(s)**

Yurii Aulchenko

**See Also**

`save.gwaa.data`, `convert.snp.text`, `convert.snp.ped`, `convert.snp.tped`, `convert.snp.illumina`

---

`merge.gwaa.data`      *function to merge objects of gwaa.data-class*

---

**Description**

function to merge two objects of gwaa.data-class

**Usage**

```
merge.gwaa.data(x, y, ... )
```

**Arguments**

x	the first object of <code>gwaa.data-class</code>
y	the second object of <code>gwaa.data-class</code>
...	arguments passed to <code>merge.snp.data</code>

**Details**

This function calls `merge.snp.data` to merge gtdata slots of the incoming objects; the phenotypic data contained in phdata slots are merged using standard function for data frames with arguments `by="id"` and `all=TRUE`. The merged object is filtered and sorted according to order of ids presented in the merged `snp.data` object.

**Value**

Merged object of `gwaa.data-class`

**Author(s)**

Maksim Struchalin, Yurii Aulchenko

**See Also**

`merge.snp.data`, `add.phdata`

**Examples**

```
data(srdta)
x1 <- srdta[c(1,3,5,6),c(2,4,5,6)]
x2 <- srdta[c(2,4,5,6),c(1,3,5,6)]
x3 <- merge(x1,x2)
x1
as.character(x1)
x2
as.character(x2)
x3
as.character(x3)
srdta[1:6,1:6]
as.character(srdta[1:6,1:6])
```

---

<code>merge.snp.data</code>	<i>function to merge objects of <code>snp.data-class</code></i>
-----------------------------	---

---

**Description**

function to merge two objects of `snp.data-class`

**Usage**

```
merge.snp.data(x, y, ... , error_amount = 1e+06, replacena = TRUE, forcestranduse =
```

**Arguments**

<code>x</code>	the first object of <code>snp.data-class</code>
<code>y</code>	the second object of <code>snp.data-class</code>
<code>...</code>	additional arguments (not used or passed)
<code>error_amount</code>	if this amount of errors is exceeded, only error table is returned
<code>replacena</code>	Some genotypes may be missing in set 1, but measured in set 2. If <code>replacena=TRUE</code> , genotypes from the set 2 will appear in the merged data.
<code>forcestranduse</code>	if <code>TRUE</code> , forces use of strand information even if coding information is sufficient for merging
<code>sort</code>	if <code>TRUE</code> , sorts the object according to chromosome and SNP position

**Details**

By default, when a genotype for a person is measured in both set "x" and set "y", and these are not equal, the value specified by set "x" is returned in the merged set. In case when genotype is NA in the first set, the behaviour depends on the value of the "replacena" parameter – if set to `TRUE` (default), these are replaced with the non-NA values from set "y".

When "forcestranduse" is set to `FALSE`, strand information is not used unless the coding is not sufficient for merging the data (i.e. strand information is used only to merge A/T and G/C polymorphisms).

SNP error is returned to "snp" table when SNP coding is incompatible between the two sets. For such SNPs, only the data provided by set "x" are used in the merged data.

ID error is returned to "id" table when genotypes of the same person at the same SNP are different between set "x" and set "y". For such genotypes, the data provided by set "x" are used in the merged data.

**Value**

A list is returned

<code>id</code>	This table summarises individual genotype inconsistencies. These may occur when some person is present and genotyped for the same marker in both sets, but these genotypes are inconsistent. The table's first column, "id", contains personal ID, the second, "snpnames", contain SNP name, and third (set "x") and fourth (set "y") contain the genotypes for this person at this SNP in sets 1 and 2.
<code>snp</code>	This table summarises coding errors. These occur when for some SNPs coding in the set 1 is not compatible with set 2. The table's first column ("snpnames") provides SNP name, and second (set "x") and third (set "y") report coding used in respective sets.
<code>data</code>	merged object of <code>snp.data-class</code>

**Author(s)**

Maksim Struchalin, Yurii Aulchenko

**See Also**

[merge.gwaa.data](#), [add.phdata](#)

**Examples**

```
data(srdta)
x1 <- srdta[c(1,3,5,6),c(2,4,5,6)]@gtdata
x2 <- srdta[c(2,4,5,6),c(1,3,5,6)]@gtdata
x3 <- merge(x1,x2)
as.character(x1)
as.character(x2)
as.character(x3$data)
as.character(srdta[1:6,1:6])
```

---

mlreg.p

*EXPERIMENTAL Linear and logistic regression and Cox models for genome-wide SNP data*

---

**Description**

Linear and logistic regression and Cox models for genome-wide SNP data

**Usage**

```
mlreg.p(formula, data, snpsubset, idsubset, gtmode = "additive", trait.type = "guess")
```

**Arguments**

formula	Standard formula object
data	an object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
gtmode	Either "additive", "dominant", "recessive" or "overdominant". Specifies the analysis model.
trait.type	Either "gaussian", "binomial" or "survival", corresponding to analysis using linear regression, logistic regression, and Cox proportional hazards models, respectively. When default value "guess" is used, the program tries to guess the type

**Details**

Linear regression is performed using standard approach; logisitic regression is implemented using IRLS; Cox model makes use of code contributed by Thomas Lumley (survival package).

For logistic and Cox, `exp(effB)` gives Odds Ratios and Hazard Ratios, respectively.

**Value**

An object of `scan.gwaa-class`

**Author(s)**

Yurii Aulchenko

**See Also**

`GASurv`, `qtscore`

**Examples**

```
data(ge03d2)
dta <- ge03d2[,1:100]
# analysis using linear model
xq <- mlreg.p(bmi~sex,dta)
# logistic regression, type guessed automatically
xb <- mlreg.p(dm2~sex,dta)
# Cox proportional hazards model, assuming that age is the follow-up time
# generally this does not make sense (could be ok if age is age at onset)
xs <- mlreg.p(GASurv(age,dm2)~sex,dta)
```

---

mlreg

*Linear and logistic regression and Cox models for genome-wide SNP data*

---

**Description**

Linear and logistic regression and Cox models for genome-wide SNP data

**Usage**

```
mlreg(formula, data, gtmode = "additive", trait.type = "guess", propPs = 1)
```



**Arguments**

<code>formula</code>	Standard formula object
<code>data</code>	an object of <code>gwaa.data-class</code>
<code>gtmode</code>	Either "additive", "dominant", "recessive" or "overdominant". Specifies the analysis model.
<code>trait.type</code>	Either "gaussian", "binomial" or "survival", corresponding to analysis using linear regression, logistic regression, and Cox proportional hazards models, respectively. When default vale "guess" is used, the program tries to guess the type
<code>propPs</code>	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <code>estlambda</code>

**Details**

Linear regression is performed using standard approach; logisitic regression is implemented using IRLS; Cox model makes use of code contributed by Thomas Lumley (survival package).

For logistic and Cox, `exp(effB)` gives Odds Ratios and Hazard Ratios, respectively.

**Value**

An object of `scan.gwaa-class`

**Author(s)**

Yurii Aulchenko

**See Also**

`GASurv`, `qtscore`

**Examples**

```
data(ge03d2)
dta <- ge03d2[,1:100]
# analysis using linear model
xq <- mlreg(bmi~sex,dta)
# logistic regression, type guessed automatically
xb <- mlreg(dm2~sex,dta)
# Cox proportional hazards model, assuming that age is the follow-up time
# generally this does not make sense (could be ok if age is age at onset)
xs <- mlreg(GASurv(age,dm2)~sex,dta)
```

mmscore

*Score test for association in related people***Description**

Score test for association between a trait and genetic polymorphism, in samples of related individuals

**Usage**

```
mmscore(h2object, data, snpsubset, idsubset, strata, times=1, quiet=FALSE, bcast=10, clambda
```

**Arguments**

h2object	An object returned by <a href="#">polygenic</a> polygenic mixed model analysis routine. The sub-objects used are measuredIDs, residualY, and InvSigma. One can supply mmscore with a fake h2object, containing these list elements.
data	An object of <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
strata	Stratification variable. If provided, scores are computed within strata and then added up.
times	If more then one, the number of replicas to be used in derivation of empirical genome-wide significance. NOTE: The structure of the data is not exchangable, therefore do not use times > 1 unless you are really sure you understand what you are doing!
quiet	do not print warning messages
bcast	If the argument times > 1, progress is reported once in bcast replicas
clambda	If inflation facot Lambda is estimated as lower then one, this parameter controls if the original P1df (clambda=TRUE) to be reported in Pc1df, or the original 1df statistics is to be multiplied onto this "deflation" factor (clambda=FALSE). If a numeric value is provided, it is used as a correction factor.
propPs	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <a href="#">estlambda</a>

**Details**

Score test is performed using the formula

$$\frac{((G - E[G])V^{-1}residualY)^2}{(G - E[G])V^{-1}(G - E[G])}$$

where  $G$  is the vector of genotypes (coded 0, 1, 2) and  $E[G]$  is a vector of (strata-specific) mean genotypic values;  $V^{-1}$  is the InvSigma and *residualY* are residuals from the trait analysis with [polygenic](#) procedure.

This test is similar to that implemented by Abecasis et al. (see reference).

## Value

Object of class [scan.gwaa-class](#); only 1 d.f. test is implemented currently.

## Author(s)

Yurii Aulchenko

## References

Chen WM, Abecasis GR. Family-based association tests for genome-wide association scans. *Am J Hum Genet.* 2007 Nov;81(5):913-26.

## See Also

[grammar](#), [qtscore](#), [egscore](#), [plot.scan.gwaa](#), [scan.gwaa-class](#)

## Examples

```
# ge03d2 is rather bad data set to demonstrate grammar,
# because this is a population-based study
data(ge03d2.clean)
#take half for speed
ge03d2.clean <- ge03d2.clean[1:450,]
gkin <- ibs(ge03d2.clean,w="freq")
h2ht <- polygenic(height ~ sex + age,kin=gkin,ge03d2.clean)
h2ht$est
mm <- mmscore(h2ht,data=ge03d2.clean)
# compute grammar
gr <- qtscore(h2ht$pgres,data=ge03d2.clean,clam=FALSE)
#compute GC
gc <- qtscore(height ~ sex + age,data=ge03d2.clean)
#compare
plot(mm,df="Pc1df",cex=0.5)
add.plot(gc,df="Pc1df",col="red")
add.plot(gr,df="Pc1df",col="lightgreen",cex=1.1)
# can see that mmscore and grammar are quite the same... in contrast to GC
```

---

npsubreated	<i>non-parametric trait "imputations" in treated people</i>
-------------	---

---

## Description

For people on treatment, the algorithm substitutes the value of the trait using non-parametric algorithm described in Tobin et al., 2005. This algorithm assumes that the measurement in treated subject is a right-censored trait. Essentially, the algorithm substitutes the QT for a person on treatment with the mean of the above-ranked substituted QT value.

## Usage

```
npsubreated(trait, medication, increase = FALSE)
```

## Arguments

trait	vector if trait values
medication	medication indicator (0/1)
increase	if medication INCREASE the value of the trait (should never be true for e.g. blood pressure, LDL cholesterol, etc.)

## Details

Should put the formulas here...

## Value

Vector of trait values, where the values for treated people are substituted with average of the above-ranked substituted trait value.

## Author(s)

Yurii Aulchenko

## References

Levy D, DeStefano AL, Larson MG, O'Donnell CJ, Lifton RP, Gavras H, Cupples LA, Myers RH. Evidence for a gene influencing blood pressure on chromosome 17. Genome scan linkage results for longitudinal blood pressure phenotypes in subjects from the framingham heart study. Hypertension. 2000 Oct;36(4):477-83.

Tobin MD, Sheehan NA, Scurrah KJ, Burton PR. Adjusting for treatment effects in studies of quantitative traits: antihypertensive therapy and systolic blood pressure. Stat Med. 2005 Oct 15;24(19):2911-35.

**Examples**

```

# simulate SBP data
simmeddat <- function(mu=144,bage=0.5,bsex=4.,bg=2.,pB=0.3,rvar=21^2,N=1000) {
  ageb <- c(25,74)
  pmale <- .5
  htthresh <- 160
  trprob <- .5
  mutreff <- (-15.)
  trvar <- 4^2
  age <- runif(N,min=ageb[1],max=ageb[2])
  sex <- 1*(runif(N)<=pmale)
  gt <- rbinom(N,size=2,prob=pB)
  y.true <- rnorm(N,mu,sqrt(rvar)) + bage*age + bsex*sex + bg*gt
  d.true <- (y.true>=htthresh)
  medication <- 1*d.true
  medication[d.true] <- 1*(runif(sum(d.true))<=trprob)
  treatm <- rnorm(sum(medication),mutreff,sqrt(trvar))
  treatm[treatm>0] <- 0
  treff <- rep(0,N)
  treff[medication==1] <- treatm
  y.obs <- y.true + treff
  out <- data.frame(age,sex,gt,y.true,d.true,medication,treff,y.obs)
  out
}
x <- simmeddat(bg=2.0,N=3000)
x[1:15,]

# substitute value of treated people
imptra <- npsubtreated(x$y.obs,x$medication)
imptra[1:15]

# Almost always, correlation should be higher for the "imputed" trait
cor(x$y.true,x$y.obs)
cor(x$y.true,imptra)

# see what comes out of regression
# analysis of true value
summary(lm(y.true~sex+age+gt,data=x))
# ignore treatment (as a rule, betas are underestimated; on average, power goes down)
summary(lm(y.obs~sex+age+gt,data=x))
# treatment as covariate (as a rule, betas are underestimated; on average, power goes down)
summary(lm(y.obs~sex+age+gt+medication,data=x))
# analyse "imputed" trait (as a rule betas are better; power approaches that of analysis of
summary(lm(imptra~sex+age+gt,data=x))

```

**Description**

Produces call rate and heterozygosity per person

**Usage**

```
perid.summary(data, snpsubset, idsubset, ... )
```

**Arguments**

<code>data</code>	object of <code>snp.data-class</code>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data</code> are used for analysis.
<code>...</code>	additional parameters to be passed to <code>hom</code>

**Details**

This function facilitates quality control of genomic data. E.g. extreme outliers for heterozygosity indicate possibly contaminated DNA samples, while low call rate of a person may indicate poor DNA quality.

**Value**

A matrix, giving per person (row) its' average heterozygosity ("Het" column) and call rate ("CallPP"), over all SNPs

**Author(s)**

Yurii Aulchenko

**See Also**

`check.marker`, `summary.snp.data`, `snp.data-class`

**Examples**

```
data(ge03d2c)
a <- perid.summary(data=ge03d2c, snps=c(1:100), ids=c(1:10))
a
a <- perid.summary(data=ge03d2c)
hist(a[, "CallPP"])
hist(a[, "Het"])
```

---

`plot.check.marker` *plots "check.marker" object*

---

## Description

Plots "check.marker" object, as returned by `check.marker`

## Usage

```
plot.check.marker(x, y, ...)
```

## Arguments

<code>x</code>	Object of class "check.marker", as returned by <code>check.marker</code> or <code>snp.subset</code>
<code>y</code>	this argument is not used
<code>...</code>	other arguments to be passed to plot

## Details

In this plot, along the X axes, you can see colour representation of markers which did not pass (pass – black) the QC. The diagonal shows redundant markers. If for some marker there exist markers, which show exactly the same (or some minimum concordance) genotypic distribution, such markers are depicted as crosses an solid line is dropped on the X axes from it. Other solid line connects the original SNP with the redundant ones (depicted as circles). From each redundant SNP, a dashed line is dropped on X. Normally, one expects that redundant markers are positioned very closely and redundancy appears because of linkage disequilibrium.

## Value

No value returned. Explanatory note is shown on the screen.

## Author(s)

Yurii Aulchenko

## See Also

`check.marker`, `snp.subset`

## Examples

```
data(srdta)
mc <- check.marker(data=srdta@gtdata[,1:100], redundant="all", maf=0.01, minconcordance=0.9, fdr=0.01)
mc <- check.marker(data=srdta@gtdata[,1:100], maf=0.01, fdr=.1, ibs.mrk=0)
plot(mc)
mc1 <- snp.subset(mc, snps=srdta@gtdata@snpnames[20:40])
plot(mc1)
```

---

plot.scan.gwaa.2D *function to plot 2D scan results*

---

## Description

Plots results of 2D analysis produced by `scan.glm.2D` or `scan.haplo.2D`

## Usage

```
plot.scan.gwaa.2D(x, y, ..., df=1)
```

## Arguments

x	object of type <code>scan.gwaa.2D-class</code> , as returned by <code>scan.glm.2D</code> or <code>scan.haplo.2D</code>
y	this argument is not used
...	additional arguments to be passed to plot
df	Whether 1, 2, or "all" d.f.s should be plotted. Note that for <code>scan.haplo.2D</code> 1 and 2 d.f. list the same values.

## Details

Now plots only "allelic" results. This is fine for `scan.haplo.2D` as only allelic tests are produced; however, `scan.glm.2D` also produces "genotypic" results.

## Value

No value returned.

## Author(s)

Yurii Aulchenko

## See Also

`scan.gwaa.2D-class`, `scan.glm.2D`, `scan.haplo.2D`

## Examples

```
data(srdta)
a <- scan.glm.2D("qt3~CRSNP", data=srdta, snps=c(1:10))
# "allelic" results
plot(a)
# to plot "genotypic" results:
filled.contour(x=a$map, y=a$map, z=-log10(a$P2df))
```



---

plot.scan.gwaa	<i>function to plot GWAA results</i>
----------------	--------------------------------------

---

**Description**

Plots results of GWA analysis

**Usage**

```
plot.scan.gwaa(x, y, ..., df=1, ystart=0, col=c("blue","green"), sort=TRUE, ylim, c
```

**Arguments**

x	object of type <code>scan.gwaa-class</code> , as returned by <code>scan.glm</code> , <code>qtscore</code> , <code>ccfast</code> , <code>emp.ccfast</code> , <code>emp.qtscore</code> , or <code>scan.haplo</code>
y	this argument is not used
...	additional arguments to be passed to plot
df	Plot results of 1 or 2-df test (1, 2). Could be also "Pc1df" (for GC corrected P-values) and "Pc2df" (for robust genomic control of the 2 df test)
ystart	truncate lower value of Y at this point (can help avoiding plotting too many points)
ylim	ylim, same as in the standard plot function
col	which colors to use to depict consecutive chromosomes
sort	whether results should be plotted after sorting by chromosome and position
delta	gap width between chromosomes

**Value**

No value returned.

**Author(s)**

Yurii Aulchenko

**See Also**

`scan.gwaa-class`, `add.plot`, `snp.subset`, `scan.glm`, `qtscore`, `ccfast`, `emp.qtscore`, `emp.ccfast`, `scan.haplo`

**Examples**

```
data(srdta)
a <- qtscore(bt, srdta, snps=c(1:250))
plot(a)
add.plot(a, df="Pc1df", col="green")
```

polygenic

*Estimation of polygenic model***Description**

Estimates linear mixed (polygenic) model based on trait and covariates data and kinship matrix

**Usage**

```
polygenic(formula, kinship.matrix, data, fixh2, starth2=0.3, trait.type="gaussian", opt.m
```

**Arguments**

<code>formula</code>	Formula describing fixed effects to be used in analysis, e.g. <code>y ~ a + b</code> means that outcome (y) depends on two covariates, a and b. If no covariates used in analysis, skip the right-hand side of the equation.
<code>kinship.matrix</code>	Kinship matrix, as provided by e.g. <code>ibs(,weight="freq")</code> , or estimated outside of GenABEL from pedigree data.
<code>data</code>	An (optional) object of <code>gwaa.data-class</code> or a data frame with outcome and covariates
<code>fixh2</code>	Optional value of heritability to be used, instead of maximisation. The uses of this option are two-fold: (a) testing significance of heritability and (b) using a priori known heritability to derive the rest of MLEs and var.-cov. matrix.
<code>starth2</code>	Starting value for h2 estimate
<code>trait.type</code>	"gaussian" or "binomial"
<code>opt.method</code>	"nlm" or "optim". These two use different optimisation functions. <code>optim</code> is slower than <code>nlm</code> , but may give better results.
<code>scaleh2</code>	Only relevant when "nlm" optimisation function is used. "scaleh2" is the heritability scaling parameter, regulating how "big" are parameter changes in h2 with the respect to changes in other parameters. As other parameters are estimated from previous regression, these are expected to change little from the initial estimate. The default value of 1000 proved to work rather well under a range of conditions.
<code>quiet</code>	If FALSE (default), details of optimisation process are reported.
<code>...</code>	Optional arguments to be passed to <code>nlm(optim)</code> minimisation function.

**Details**

This function maximises the likelihood of the data under polygenic model with covariates and reports the maximum likelihood estimates and the inverse of variance-covariance matrix at the point of ML.

One of the major use of this function is to estimate residuals of the trait and the inverse of the variance-covariance matrix for further use in analysis with `mmscore` and `grammar`.

Also, it can be used for a variant of GRAMMAR analysis, which allows for permutations for GW significance by use of environmental residuals as an analysis trait with `qtscore`.

"Environmental residuals" (not to be mistaken with just "residuals") are the residual where both the effect of covariates AND the estimated polygenic effect (breeding values) are factored out. This thus provides an estimate of the trait value contributed by environment (or, turning this other way around, the part of trait not explained by covariates and by the polygene). Polygenic residuals are estimated as

$$\sigma^2 V^{-1} (Y - (\hat{\mu} + \hat{\beta} C_1 + \dots))$$

where  $\sigma^2$  is the residual variance,  $V^{-1}$  is the InvSigma (inverse of the var-cov matrix at the maximum of polygenic model) and  $(Y - (\hat{\mu} + \hat{\beta} C_1 + \dots))$  is the trait values adjusted for covariates (also at at the maximum of polygenic model likelihood).

It can also be used for heritability analysis. If you want to test significance of heritability, estimate the model and write down the function minimum reported at "h2an" element of the output (this is  $-2 \times \text{MaxLikelihood}$ ). Then do next round of estimation, but set `fixh2=0`. The difference between you function minima gives you one-sided test distributed as chi-squared with 1 d.f.

The way to compute the likelihood is partly based on the paper of Thompson (see refs), namely instead of taking inverse of var-cov matrix every time, eigenvectors of the inverse of G (taken only once) are used.

## Value

A list with values

<code>h2an</code>	A list supplied by the <code>nlm</code> minimisation routine. Of particular interest are elements "estimate" containing parameter maximal likelihood estimates (MLEs) (order: mean, betas for covariates, heritability, (polygenic + residual variance))
<code>residualY</code>	Residuals from analysis, based on covariate effects only; NOTE: these are NOT grammar "environmental residuals"!
<code>esth2</code>	Estimate (or fixed value) of heritability
<code>pgresidualY</code>	Environmental residuals from analysis, based on covariate effects and predicted breeding value.
<code>InvSigma</code>	Inverse of the variance-covariance matrix, computed at the MLEs – these are used in <code>mmscore</code> and <code>grammar</code> functions.
<code>call</code>	The details of call
<code>measuredIDs</code>	Logical values for IDs who were used in analysis (traits and all covariates measured) == TRUE

## Note

Presence of twins may screw up your analysis. Check kinship matrix for singularities, or rather use `check.marker` for identification and exclusion of twin samples.

If a trait (no covarites) is used, make sure that order of IDs in `kinship.matrix` is exactly the same as in the outcome

**Author(s)**

Yurii Aulchenko

**References**

Thompson EA, Shaw RG (1990) Pedigree analysis for quantitative traits: variance components without matrix inversion. *Biometrics* 46, 399-413.

Aulchenko YS, de Koning DJ, Haley C. Genomewide rapid association using mixed model and regression: a fast and simple method for genome-wide pedigree-based quantitative trait loci association analysis. *Genetics*. 2007 177(1):577-85.

Amin N, van Duijn CM, Aulchenko YS. A genomic background based method for association analysis in related individuals. *PLoS ONE*. 2007 Dec 5;2(12):e1274.

**See Also**

[mmscore](#), [grammar](#)

**Examples**

```
# note that procedure runs on CLEAN data
data(ge03d2ex.clean)
gkin <- ibs(ge03d2ex.clean,w="freq")
h2ht <- polygenic(height ~ sex + age, kin=gkin, ge03d2ex.clean)
# estimate of heritability
h2ht$esth2
# other parameters
h2ht$h2an
# minimum negative log-likelihood
h2ht$h2an$minimum
# maximum log-likelihood
-h2ht$h2an$minimum

#for binary trait (experimental)
h2dm <- polygenic(dm2 ~ sex + age, kin=gkin, ge03d2ex.clean, trait="binomial")
# estimated parameters
h2dm$h2an
```

---

qtscore

*Fast score test for association*

---

**Description**

Fast score test for association between a trait and genetic polymorphism

**Usage**

```
qtscore(formula, data, snpsubset, idsubset, strata, trait.type="gaussian", times=1, quiet=
```

## Arguments

<code>formula</code>	Formula describing fixed effects to be used in analysis, e.g. $y = a + b$ means that outcome (y) depends on two covariates, a and b. If no covariates used in analysis, skip the right-hand side of the equation.
<code>data</code>	An object of <code>gwaa.data-class</code>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idssubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data/cc</code> are used for analysis.
<code>strata</code>	Stratification variable. If provided, scores are computed within strata and then added up.
<code>trait.type</code>	"gaussian" or "binomial" or "guess" (later option guesses trait type)
<code>times</code>	If more than one, the number of replicas to be used in derivation of empirical genome-wide significance. See <code>emp.qtscore</code> , which calls <code>qtscor</code> with <code>times&gt;1</code> for details
<code>quiet</code>	do not print warning messages
<code>bcast</code>	If the argument <code>times &gt; 1</code> , progress is reported once in <code>bcast</code> replicas
<code>clambda</code>	If inflation factor Lambda is estimated as lower than one, this parameter controls if the original P1df ( <code>clambda=TRUE</code> ) to be reported in <code>Pc1df</code> , or the original 1df statistics is to be multiplied onto this "deflation" factor ( <code>clambda=FALSE</code> ). If a numeric value is provided, it is used as a correction factor.
<code>propPs</code>	proportion of non-corrected P-values used to estimate the inflation factor Lambda, passed directly to the <code>estlambda</code>
<code>details</code>	when <code>FALSE</code> , SNP and ID names are not reported in the returned object (saves some memory). This is experimental and will be not maintained anymore as soon as we achieve better memory efficiency for storage of SNP and ID names (currently default R character data type used)

## Details

When `formula` contains covariates, the traits is analysed using GLM and later residuals used when score test is computed for each of the SNPs in analysis. Coefficients of regression are reported for the quantitative trait.

For binary traits, odds ratios (ORs) are reported. When adjustment is performed, first, "response" residuals are estimated after adjustment for covariates and scaled to [0,1]. Reported effects are approximately equal to ORs expected in logistic regression model.

With no adjustment for binary traits, 1 d.f., the test is equivalent to the Armitage test.

This is a valid function to analyse GWA data, including X chromosome. For X chromosome, stratified analysis is performed (`strata=sex`).

## Value

Object of class `scan.gwaa-class`

**Author(s)**

Yurii Aulchenko

**References**

Aulchenko YS, de Koning DJ, Haley C. Genomewide rapid association using mixed model and regression: a fast and simple method for genome-wide pedigree-based quantitative trait loci association analysis. *Genetics*. 2007 177(1):577-85.

Amin N, van Duijn CM, Aulchenko YS. A genomic background based method for association analysis in related individuals. *PLoS ONE*. 2007 Dec 5;2(12):e1274.

**See Also**

[mlreg](#), [mmscore](#), [egscore](#), [emp.qtscore](#), [plot.scan.gwaa](#), [scan.gwaa-class](#)

**Examples**

```
data(srdta)
#qtscore with stratification
a <- qtscore(qt3~sex,data=srdta)
plot(a)
b <- qtscore(qt3, strata=srdta@phdata$sex, data=srdta)
add.plot(b, col="green", cex=2)
# qtscore with extra adjustment
a <- qtscore(qt3~sex+age, data=srdta)
a
plot(a)
# compare results of score and chi-square test for binary trait
a1 <- ccfast("bt", data=srdta, snps=c(1:100))
a2 <- qtscore(bt, data=srdta, snps=c(1:100), trait.type="binomial")
plot(a1, ylim=c(0, 2))
add.plot(a2, col="red", cex=1.5)
# the good thing about score test is that we can do adjustment...
a2 <- qtscore(bt~age+sex, data=srdta, snps=c(1:100), trait.type="binomial")
points(a2$map, -log10(a2$P1df), col="green")
```

---

qvaluebh95

---

*Computes Benjamini-Hochberg (95) q-value*


---

**Description**

Computes Benjamini-Hochberg (95) q-value

**Usage**

```
qvaluebh95(p, fdrate=0.1)
```

**Arguments**

`p` vector containing p-values  
`fdrate` desired FRD

**Value**

List

**0.1 Names**

normal-bracket16bracket-normal

`pass` Is true if this P-value passed specified FDR

`qvalue` qvalue

normal-bracket16bracket-normal

**Names**

`pass` Is true if this P-value passed specified FDR

`qvalue` qvalue

**Author(s)**

Yurii Aulchenko

**Examples**

```
data(srdta)
a<-qtscore(qt2,data=srdta)
qv <- qvaluebh95(a$Pldf)
plot(a$map,-log10(qv$qvalue))
```

---

`r2fast.old`

*Estimates r2 between multiple markers*

---

**Description**

Given a set of SNPs, computes a matrix of r2

**Usage**

```
r2fast.old(data, snpsubset, idsubset)
```

**Arguments**

<code>data</code>	object of <code>snp.data-class</code>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idssubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data</code> are used for analysis.

**Details**

The function is based on slightly modified code of Hao et al.

**Value**

A (Nsnps X Nsnps) matrix giving  $r^2$  values between a pairs of SNPs above the diagonal and number of SNP genotype measured for both SNPs below the diagonal

**Author(s)**

Yurii Aulchenko

**References**

Hao K, Di X, Cawley S. (2006) LdCompare: rapid computation of single- and multiple-marker  $r^2$  and genetic coverage. *Bioinformatics*, 23: 252-254.

**See Also**

[rhofast](#)

**Examples**

```
data(ge03d2)
# r2s using r2fast.old
a <- r2fast.old(ge03d2, snps=c(1:10))
# r2s using package genetics
b <- LD(as.genotype(ge03d2[,1:10]))$"R^2"
# see that the r2s are not exactly the same
cor(a[upper.tri(a)], b[upper.tri(b)])
plot(a[upper.tri(a)], b[upper.tri(b)])
```



---

`r2fast`*Estimates r2 between multiple markers*

---

**Description**

Given a set of SNPs, computes a matrix of r2

**Usage**

```
r2fast(data, snpsubset, cross.snpsubset, idsubset)
```

**Arguments**

<code>data</code>	object of <code>snp.data-class</code>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>cross.snpsubset</code>	Parameter allowing parallel implementation. Not to be used normally. If supplied together with <code>snpsubset</code> , the r2 for all pairs between <code>snpsubset</code> and <code>cross.snpsubset</code> computed.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data</code> are used for analysis.

**Details**

The function is based on slightly modified code of Hao et al.

**Value**

A (Nsnps X Nsnps) matrix giving r2 values between a pairs of SNPs above the diagonal and number of SNP genotype measured for both SNPs below the diagonal

**Author(s)**

Yurii Aulchenko

**References**

Hao K, Di X, Cawley S. (2006) LdCompare: rapid computation of single- and multiple-marker r2 and genetic coverage. *Bioinformatics*, 23: 252-254.

**See Also**

[rhofast](#)

## Examples

```
data(ge03d2)
# r2s using r2fast
a <- r2fast(ge03d2, snps=c(1:10))
# r2s using package genetics
b <- LD(as.genotype(ge03d2[,1:10]))$"R^2"
# see that the r2s are not exactly the same
cor(a[upper.tri(a)], b[upper.tri(b)])
plot(a[upper.tri(a)], b[upper.tri(b)])
```

---

redundant	<i>function to do redundancy check</i>
-----------	--

---

## Description

Checks marker redundancy, understood as comcordance between genotypic distributions (including missing values)

## Usage

```
redundant(data, pairs = "bychrom", minconcordance = 2.0)
```

## Arguments

data	gwaa.data or snp.data object
pairs	"bychrom" or "all" to check pairs within chromosome only or genome-wide
minconcordance	find "redundant" pairs of markers with concordance $\geq$ "minconcordance". If "minconcordance" is more then 1.0, only pairs of markers which are exactly the same (independent of coding), including NA pattern, are considered as redundant. If "minconcordance" is $\leq 1$ , the concordance rate is computed as percent of genotypes which are the same, including the genotypes with NA. I.e. if both genotypes are NA, this is counted as a match, if one is NA and other is measured, this is counted as mismatch. Note that option with "minconcordance" $\leq 1$ takes much longer time to run.

## Value

A list containing reference SNP as a name and all SNPs which has "the same" genotypic distribution as values:

"refSNP1"	SNP11, SNP12, ...
"refSNP2"	SNP21, SNP22, ...
...	
"refSNPlast"	SNPlast1, SNPlast2, ...
"all"	list of all redundant SNPs, which can be dropped from consideration

**Author(s)**

Yurii Aulchenko

**See Also**[check.marker](#)**Examples**

```
data(srdta)
redundant(srdta@gtdata)
redundant(srdta@gtdata[,1:50],minconcordance=0.8)
```

---

`refresh.gwaa.data`    *Updates an object from old to new GenABEL format*

---

**Description**

Attempts to update an object of gwaa.data-class from old to new format

**Usage**

```
refresh.gwaa.data(data, force=FALSE)
```

**Arguments**

<code>data</code>	An object of <a href="#">gwaa.data-class</a> in pre-1.2-6 (data version 0) format.
<code>force</code>	When TRUE, the refreshing is forced, with any data in <code>@strand</code> and <code>@coding</code> replaced by default data (0/1 coding, u-strand)

**Details**

Takes old-style gwaa.data object and sets `@coding` and `@strand` attributes to SNPs. All coding is set to 1/2 and strand is set to "u" (unknown).

**Value**

Object of [gwaa.data-class](#) in new (GenABEL v > 1.2-6, raw data format version 0.1) format.

**Author(s)**

Yurii Aulchenko

**See Also**[load.gwaa.data](#)

---

rhofast*Estimates rho between multiple markers*

---

**Description**

Given a set of SNPs, computes a matrix of rho

**Usage**

```
rhofast(data, snpsubset, idsubset)
```

**Arguments**

data	object of <code>snp.data-class</code>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data are used for analysis.

**Details**

Rho is the measure of association described by N. Morton and A. Collins (see reference). The function is based on slightly modified code of Hao et al.

**Value**

A (Nsnps X Nsnps) matrix giving rho values between a pairs of SNPs above the diagonal and Kij below the diagonal.

**Author(s)**

Yurii Aulchenko

**References**

Collins A, Morton NE. (1998) Mapping a disease locus by allelic association. PNAS, 17:1741-1745.

Hao K, Di X, Cawley S. (2006) LdCompare: rapid computation of single- and multiple-marker rho and genetic coverage. Bioinformatics, 23: 252-254.

**See Also**

[r2fast](#)

**Examples**

```
data(ge03d2)
# rhos using rhofast
a <- rhofast(ge03d2,snps=c(1:10))
# rhos using package genetics
b <- LD(as.genotype(ge03d2[,1:10]))$"R^2"
# see that the rhos are not exactly the same
cor(a[upper.tri(a)],b[upper.tri(b)])
plot(a[upper.tri(a)],b[upper.tri(b)])
```

---

rntransform	<i>Rank-transformation to normality</i>
-------------	---

---

**Description**

Rank-transformation to normality of a variable or residuals from GLM analysis.

**Usage**

```
rntransform(formula, data, family=gaussian)
```

**Arguments**

formula	GLM formula for the variable to be transformed, or just the variable
data	data.frame or gwaa.data object containing the data
family	GLM family

**Details**

Rank-transformation to normality generates perfectly normal distribution from ANY distribution, unless many/heavy ties are present in variable (or residuals, if formula is used).

When formula is supplied, this procedure first calls [ztransform](#), and then applies rank transformation to residuals.

**Value**

Vector containing transformed variable, distributed as standard normal.

**Author(s)**

Yurii Aulchenko

**See Also**

[ztransform](#)

## Examples

```
# uniformly distributed variable
x <- round(runif(200)*100)
# get 7 missing values
x[round(runif(7,min=1,max=100))] <- NA
# Z-transform
y0 <- ztransform(x)
# Rank-transform to normality
y1 <- rntransform(x)
# test normality of the original and transformed var
shapiro.test(x)
shapiro.test(y0)
shapiro.test(y1)
# plot histogram
par(mfcol=c(3,1))
hist(x)
hist(y0)
hist(y1)
```

---

save.gwaa.data	<i>function to save gwaa.data object</i>
----------------	--

---

## Description

Saves GenABEL data in internal format

## Usage

```
save.gwaa.data(data, phenofile = "pheno.dat", genofile = "geno.raw",
               human = FALSE)
```

## Arguments

data	gwaa.data object
phenofile	name of file where the phenotypes will be saved to
genofile	name of file where the genotypes will be saved to
human	if human=TRUE, saves in human-readable format (to be converted to internal format later)

## Details

When running with human=TRUE, a lot of memory (and time to complete the operation) is required. Probably, this option would not work because of memory limitations in a GWA scan iwth more then few hundreds of people. This is possible to fix; drop me a message if you need that.

**Value**

No value returned

**Author(s)**

Yurii Aulchenko

**See Also**

[load.gwaa.data](#)

---

scan.glm.2D

*Scans regional data allowing for gene-gene interaction using glm*

---

**Description**

Scans regional data allowing for gene-gene interaction using glm

**Usage**

```
scan.glm.2D(formula, family = gaussian(), data, snpsubset, idsubset,  
            bcast = 50)
```

**Arguments**

formula	character string containing formula to be used in <a href="#">glm</a> . You should put CRSNP argument in the formula, to arrange how the SNP from the list would be treated. This allows to put in an interaction term.
family	family to be passed to <a href="#">glm</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
data	object of class "gwaa.data"
bcast	show progress every bcast SNPs

**Value**

Object of class [scan.gwaa.2D-class](#)

**Author(s)**

Yurii Aulchenko

**See Also**

[scan.gwaa.2D-class](#), [scan.haplo.2D](#)

**Examples**

```
data(srdta)
a <- scan.glm.2D("bt~sex+age+CRSNP", family=binomial(), data=srdta, snps=(1:10), bcast=2)
plot(a)
```

---

scan.glm

*Scan GWA data using glm*

---

**Description**

Scan GWA data using glm

**Usage**

```
scan.glm(formula, family = gaussian(), data, snpsubset, idsubset,
          bcast = 50)
```

**Arguments**

formula	character string containing formula to be used in <a href="#">glm</a> . You should put CRSNP argument in the formula, to arrange how the SNP from the list would be treated. This allows to put in an interaction term.
family	family to be passed to <a href="#">glm</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
data	object of class "gwaa.data"
bcast	show progress every bcast SNPs

**Value**

Object of class [scan.gwaa-class](#)

**Author(s)**

Yurii Aulchenko

**See Also**

[ccfast](#), [qtscore](#), [scan.gwaa-class](#)



## Examples

```
data(srdta)
a <- scan.glm("bt~sex+age+CRSNP", family=binomial(), data=srdta, snps=(1:10), bcast=2)
plot(a)

osnp <- "rs4934"
maposnp <- srdta@gtdata@map[osnp]
maposnp
reg <- snp.names(srdta, begin=maposnp-100000, end=maposnp+100000, chrom="1")
a <- scan.glm("qt3~sex+age+CRSNP", data=srdta, snps=reg)
plot(a)
plot(a, df=1)
add.plot(a, df=2)

# interaction with sex
a <- scan.glm("qt3~age+sex*CRSNP", data=srdta, snps=reg)
plot(a, df=1)
add.plot(a, df=2)
# you can do interaction with a selected polymorphisms in the same way
```

---

scan.gwaa-class	Class "scan.gwaa"
-----------------	-------------------

---

## Description

This class contains results of GWA analysis. This is an list object, generated by `scan.glm`, `scan.haplo`, `ccfast`, `qtscore`, `emp.ccfast`, or `emp.qtscore`.

## Names

snpnames list of names of SNPs tested

P1df corresponding list of P-values of 1-d.f. (additive or allelic) test for association between SNP and trait

P2df corresponding list of P-values of 2-d.f. (genotypic) test for association between SNP and trait

Pc1df P-values from the 1-d.f. test for association between SNP and trait; the statistics is corrected for possible inflation

effB Effect of the B allele in allelic test (OR for `ccfast`, difference from the mean for `qtscore` and beta from the `scan.glm`)

effAB Effect of the AB genotype in genotypic test

effBB Effect of the BB genotype in genotypic test

map list of map positions of the SNPs

chromosome list of chromosomes the SNPs belong to

idnames list of people used in analysis

lambda list with elements "estimate" (inflation factor estimate, as computed using lower 90 percents of the distribution) and "se" (standard error of the estimate)

formula which formula/function call was used to comput P-values

family family of the link function / nature of the test

## Methods

**plot** signature(object = "scan.gwaa"): Plots summary of GWAA

## Author(s)

Yurii Aulchenko

## See Also

[ccfast](#), [qtsscore](#), [scan.glm](#), [scan.haplo](#), [emp.ccfast](#), [emp.qtsscore](#), [estlambda](#), [plot.scan.gwaa](#)

## Examples

```
data(srdta)
sc <- scan.glm("qt3~CRSNP", data=srdta, snps=c(1:10))
class(sc)
sc$P1df
sc$P2df
sc
plot(sc)
```

---

scan.gwaa.2D-class *Class "scan.gwaa.2D"*

---

## Description

This class contains results of 2D analysis. This is an list object, generated by [scan.glm.2D](#) or [scan.haplo.2D](#).

**Names**

snpnames list of names of SNPs tested

P1df corresponding list of P-values of allelic test for association between SNP and trait.

Pint1df corresponding list of P-values of significance of the interactions between SNPs, for the allelic model

P2df corresponding list of P-values of genotypic test for association between SNP and trait For `link{scan.haplo}` and `link{scan.haplo.2D}` this is equal to P1df and has nothing to do with the actual degrees of freedom of the test

Pint1df corresponding list of P-values of significance of the interactions between SNPs for the genotypic test

medChi1df Median Chi-square for allelic test

medChi2df Median Chi-square on genotypic test

map list of map positions of the SNPs

chromosome list of chromosomes the SNPs belong to

formula which formula/function call was used to compute P-values

family family of the link function / nature if the test

idnames list of people used in analysis

**Methods**

**plot** signature(object = "scan.gwaa.2D"): Plots summary of 2D scan, using lsit element P1df

**Author(s)**

Yurii Aulchenko

**See Also**

[scan.gwaa.2D-class](#), [scan.glm.2D](#), [scan.haplo.2D](#), [plot.scan.gwaa.2D](#)

**Examples**

```
data(srdta)
sc <- scan.glm.2D("qt3~CRSNP", data=srdta, snps=c(1:10))
class(sc)
sc$P1df
sc$P2df
sc
plot(sc)
```

---

scan.haplo.2D	<i>runs haplo.score.slide with all pairs of markers in a region</i>
---------------	---

---

## Description

Runs [haplo.score.slide](#) from the package `haplo.stats` on all pairs of markers in a region and presents output as `scan.gwaa.2D-class` object

## Usage

```
scan.haplo.2D(formula, data, snpsubset, idsubset, bcast = 10, simulate=FALSE, trait)
```

## Arguments

<code>formula</code>	Formula to be used in analysis. It should be a character string following standard notation. On the left-hand side, there should be outcome. On the right-hand side, covariates are listed, with "+" separating the covariates (additive action). The left- and right-hand sides are separated by " ". You should put CRSNP argument in the formula. For example "qt3 CRSNP" would analyse association between SNPs and trait "qt3", without any adjustment. To adjust for age and sex, use "qt3 age+sex+CRSNP". Currently, only additive effects ("+") are allowed.
<code>data</code>	object of class <a href="#">gwaa.data-class</a>
<code>snpsubset</code>	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from <code>data</code> are used for analysis.
<code>idsubset</code>	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from <code>data/cc</code> are used for analysis.
<code>bcast</code>	show progress every <code>bcast</code> percents of progress
<code>simulate</code>	if simulated P-values should be generated
<code>trait.type</code>	Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal" (see help for <a href="#">haplo.score.slide</a> for details). If not specified, the routine picks up "gaussian" or "binomial" (two levels of trait).
<code>...</code>	other arguments to be passed to <a href="#">haplo.score.slide</a>

## Details

List element `P2df` is set equal to `P1df`, as only allelic results are returned. This has nothing to do with actual degrees of freedom of the test.

## Value

Object of class `scan.gwaa.2D-class`

## Author(s)

Yurii Aulchenko

References

For haplo.stats (scan.haplo, scan.haplo.2D), please cite:  
Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. (2002) Score tests for association between traits and haplotypes when linkage phase is ambiguous. Am J Hum Genet, 70: 425-434.

See Also

[scan.gwaa.2D-class](#), [scan.haplo](#), [scan.glm.2D](#), [haplo.score.slide](#)

Examples

```
data(srdta)
c <- scan.haplo.2D("bt~sex+age+CRSNP",data=srdta,snps=(717:733),
  ids=(srdta@phdata$age<40))
plot(c)
```

---

scan.haplo	<i>scan.haplo</i>
------------	-------------------

---

Description

Runs [haplo.score.slide](#) from the package haplo.stats and represents output as [scan.gwaa-class](#) data object

Usage

```
scan.haplo(formula, data, snpsubset, idsubset, n.slide = 2, bcast = 10, simulate=FALSE)
```

Arguments

formula	Formula to be used in analysis. It should be a character string following standard notation. On the left-hand side, there should be outcome. On the right-hand side, covariates are listed, with "+" separating the covariates (additive action). The left- and right-hand sides are separated by " ". You should put CRSNP argument in the formula. For example "qt3 CRSNP" would analyse association between SNPs and trait "qt3", without any adjustment. To adjust for age and sex, use "qt3 age+sex+CRSNP". Currently, only additive effects ("+") are allowed.
data	object of class <a href="#">gwaa.data-class</a>
snpsubset	Index, character or logical vector with subset of SNPs to run analysis on. If missing, all SNPs from data are used for analysis.
idsubset	Index, character or logical vector with subset of IDs to run analysis on. If missing, all people from data/cc are used for analysis.
n.slide	Default = 2. 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.

bcast	show progress every bcast SNPs
simulate	if simulated P-values should be generated
trait.type	Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal" (see help for <a href="#">haplo.score.slide</a> for details). If not specified, the routine picks up "gaussian" or "binomial" (two levels of trait).
...	other arguments to be passed to <a href="#">haplo.score.slide</a>

## Details

List element P2df is set equal to P1df, as only allelic results are returned. This has nothing to do with degrees of freedom.

## Value

Object of class [scan.gwaa-class](#)

## Author(s)

Yurii Aulchenko

## References

For haplo.stats (scan.haplo, scan.haplo.2D), please cite:

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. (2002) Score tests for association between traits and haplotypes when linkage phase is ambiguous. Am J Hum Genet, 70: 425-434.

## See Also

[scan.gwaa-class](#), [haplo.score.slide](#)

## Examples

```
data(srdta)
a <- ccfast("bt", srdta, snps=(717:733), ids=(srdta@phdata$age<40))
b <- scan.haplo("bt~sex+CRSNP", srdta, snps=(717:733),
               ids=(srdta@phdata$age<40))
c <- scan.haplo("bt~sex+CRSNP", srdta, snps=(717:733),
               ids=(srdta@phdata$age<40), n.slide=3)

plot(a)
add.plot(b, col="red", type="l")
add.plot(c, col="darkgreen", type="l")
```

---

show.ncbi	<i>Shows the region on NCBI map</i>
-----------	-------------------------------------

---

**Description**

This function calls web browser and direct it to NCBI MapViewer, to show the region of interest.

**Usage**

```
show.ncbi(region)
```

**Arguments**

region            a vector containing regional landmarks

**Details**

The elements of input vector could be SNP rs-names

**Author(s)**

Yurii Aulchenko

**Examples**

```
## Not run:
show.ncbi(c("rs7926624", "rs11564708"))

## End(Not run)
```

---

snp.coding-class	<i>Class "snp.coding"</i>
------------------	---------------------------

---

**Description**

This class contains the actual nucleotide codes for the typed SNPs

**Slots**

**.Data:** nucleotide coding data

**Methods**

`[ signature(x = "snp.coding", i = "ANY", j = "missing", drop = "missing"):`  
 subset operations. `x[i]` will show coding for SNPs selected in `i`.

`coerce signature(from = "snp.coding", to = "character"):` converts SNP coding from internal (raw) to human-readable character.

`show signature(object = "snp.coding"):` shows the object. Take care that this is internal representation

**Author(s)**

Yurii Aulchenko

**See Also**

[snp.strand-class](#), [gwaa.data-class](#), [snp.data-class](#)

**Examples**

```
data(srdta)
srdta@gtdata@coding[1:10]
as.character(srdta@gtdata@coding[1:10])
```

---

snp.data-class	Class "snp.data"
----------------	------------------

---

**Description**

This class contains objects holding large arrays of single nucleotide polymorphism (SNP) genotypes

**Slots**

**nbytes:** number of bytes used to store data on a SNP

**nids:** number of people

**male:** male code

**idnames:** ID names

**nsnps:** number of SNPs

**nsnpnames:** list of SNP names

**chromosome:** list chromosomes corresponding to SNPs

**coding:** list of nucleotide coding for the SNPs

**strand:** strands of the SNPs

**map:** list SNPs' positions

**gtps:** [snp.mx-class](#) object used to store genotypes



## Methods

[ signature(x = "snp.data", i = "ANY", j = "ANY", drop = "ANY"): subset operations. x[i,j] will select people listed in i and SNPs listed in j.

**coerce** signature(from = "snp.data", to = "numeric"): map to codes 0, 1, 2, or NA

**coerce** signature(from = "snp.data", to = "character"): map to actual nucleotide codes, e.g. "A/A", "A/G", "G/G", ""

**coerce** signature(from = "snp.data", to = "genotype"): map to data frame with [genotype-class](#) data, for later use with package `genetics`

**coerce** signature(from = "snp.data", to = "hsgeno"): map to data frame with allelic data frame, for later use with package `haplo.stats`

**show** signature(object = "snp.data"): shows the object. Take care that the objects are usually very large!

**summary** signature(object = "snp.data"): calculate allele frequencies, genotype frequencies, and chi-square tests for Hardy-Weinberg equilibrium. Results are returned as a dataframe

## Author(s)

Yurii Aulchenko

## See Also

[gwaa.data-class](#), [snp.data](#), [snp.mx-class](#)

## Examples

```
data(srdta)
class(srdta)
x <- srdta@gtdata
class(x)
x@nids
x@nsnps
x@idnames[1:12]
x@male[1:12]
x@male[c("p1", "p2", "p3", "p4")]
x@snpnames[1:4]
x@chromosome[1:4]
x@map[1:4]
n4 <- c("rs18", "rs655")
n4
x@map[n4]
n4 <- c("rs18", "rs65")
n4
x@map[n4]
x@chromosome[n4]
x[1:12, 1:4]
summary(x[, 1:10])
as.numeric(x[1:12, 1:4])
```

```

as.numeric(x[c("p1", "p3", "p4"), c("rs18", "rs65")])
as.character(x[c("p1", "p3", "p4"), c("rs18", "rs65")])
as.genotype(x[c("p1", "p3", "p4"), c("rs18", "rs65")])
as.hsgeno(x[c("p1", "p3", "p4"), c("rs18", "rs65")])

```

---

snp.data	<i>creates an snp.data object</i>
----------	-----------------------------------

---

## Description

Creates object of class `snp.data-class`

## Usage

```

snp.data(nids, rawdata, idnames = as.character(c(1:nids)),
         snpnames = as.character(c(1:(length(rawdata)/ceiling(nids/4)))),
         chromosome = as.factor(rep(1, (length(rawdata)/ceiling(nids/4)))),
         map = as.double(seq(1, (length(rawdata)/ceiling(nids/4)))),
         coding=as.raw(rep(1, length(rawdata)/ceiling(nids/4))),
         strand=as.raw(rep(0, length(rawdata)/ceiling(nids/4))),
         male = rep(0, nids))

```

## Arguments

nids	number of people
idnames	list of IDs
male	male indicator for IDs
snpnames	list of SNP names
chromosome	list of chromosomes SNPs belong to
coding	list of nucleotide coding for the SNPs
strand	strands of the SNPs
map	map position of SNPs
rawdata	genotypes presented in raw data format

## Value

Object of class `snp.data-class`

## Author(s)

Yurii Aulchenko

## See Also

`snp.data-class`

---

snp.mx-class	Class "snp.mx"
--------------	----------------

---

**Description**

This low-level class contains objects holding large arrays of single nucleotide polymorphism (SNP) genotypes

**Slots**

**.Data:** object used to store genotypes

**Methods**

`[ signature(x = "snp.mx", i = "ANY", j = "ANY", drop = "ANY"): subset operations. x[i,j] will select people listed in i and SNPs listed in j.`

`coerce signature(from = "raw", to = "snp.mx"): makes an snp.mx object out of raw data`

`show signature(object = "snp.mx"): shows the object. Take care that (a) this is internal representation and (b) the objects are usually very large!`

**Note**

User is not supposed to work with this class. Use [snp.data-class](#).

**Author(s)**

Yurii Aulchenko

**See Also**

[gwaa.data-class](#), [snp.data-class](#)

---

snp.names	<i>extracts names of SNPs in a region</i>
-----------	---

---

**Description**

Based on boundary conditions specified and (or) chromosome selects SNP names in the region

**Usage**

```
snp.names(data, begin, end, chromosome)
```

**Arguments**

data	object of class <code>gwaa.data-class</code> , <code>snp.data-class</code> , <code>scan.gwaa-class</code> or <code>check.marker-class</code>
begin	Start position (or name of the first SNP)
end	End-position or name of last SNP
chromosome	Chromosome code

**Details**

Any of the arguments, except the `data` can be missing

**Value**

A vector of names of SNPs located in the region

**Author(s)**

Yurii Aulchenko

**See Also**

`snp.data-class`

**Examples**

```
data(srdta)
snp.names(srdta, begin = 50000, end = 100000)
snp.names(srdta, begin = 50000, end = 100000, chromosome = "1")

# does not make sense with these data:
snp.names(srdta, begin = 50000, end = 100000, chromosome = "X")

# again makes sense:
snp.names(srdta, end = 100000)
snp.names(srdta, begin = 2200000)

# show summary for SNPs in region between 50,000 and 100,000
a <- snp.names(srdta, begin = 50000, end = 100000)
summary(srdta@gtdata[,a])
```

---

snp.strand-class      *Class "snp.strand"*

---

## Description

This class contains the strands of the typed SNPs

## Slots

**.Data:** nucleotide strand data

## Methods

[ signature(x = "snp.strand", i = "ANY", j = "missing", drop = "missing"): subset operations. x[i] will show strand for SNPs selected in i.

**coerce** signature(from = "snp.strand", to = "character"): converts SNP strand from internal (raw) to human-readable character.

**show** signature(object = "snp.strand"): shows the object. Take care that this is internal representation

## Author(s)

Yurii Aulchenko

## See Also

[snp.coding-class](#), [gwaa.data-class](#), [snp.data-class](#)

## Examples

```
data(srdta)
srdta@gtdata@strand[1:10]
as.character(srdta@gtdata@strand[1:10])
```

---

snp.subset      *function to subset objects of class scan.gwaa and check.marker*

---

## Description

Computing objects of class scan.gwaa may take long, especially when haplotypic analysis is performed. Therefore this function helps substracting results on some region (indicated by list of SNPs)

**Usage**

```
snp.subset(data, snpsubset)
```

**Arguments**

data                    object of class `scan.gwaa-class` or `check.marker-class`  
 snpsubset              character vector of snps to select

**Value**

Object of class `scan.gwaa-class` or `check.marker-class`

**Author(s)**

Yurii Aulchenko

**See Also**

`scan.gwaa-class`, `check.marker-class`

**Examples**

```
data(srdta)
# processing check.marker object
#mc <- check.marker(data=srdta@gtdata[,1:100],redundant="all",maf=0.01,minconcordance=0.9,fdr=0.01)
mc <- check.marker(data=srdta@gtdata[,1:100],maf=0.01,fdr=.1,ibs.mrk=0)
summary(mc)
#plot(mc)
mc1 <- snp.subset(mc,snps=srdta@gtdata@snpsnames[20:50])
summary(mc1)
#plot(mc1)
# processing scan.gwaa object
a <- qtscore(qt3~sex+age,data=srdta)
plot(a)
a1 <- snp.subset(a,snps=srdta@gtdata@snpsnames[10:20])
plot(a1)
```

---

snps.cell-class	<i>Class "snps.cell"</i>
-----------------	--------------------------

---

**Description**

This is a lowest-level class based on which `snp.mx-class` is build

**Note**

User is not supposed to work with this class. Use `snp.data-class`.

**Author(s)**

Yurii Aulchenko

**See Also**[snp.mx-class](#), [gwaa.data-class](#), [snp.data-class](#)

---

srda*GWA-type data on small region*

---

**Description**

srda contains gwaa.data object with results on a small region of about 2.5 Mb. 833 SNPs are typed on 2500 people. NA rate is 95%. Sex, age, two quantitative (qt1 and qt2) and one binary (bt) traits are available for analysis. Run demo(srda) and check tut-srda.pdf to see examples of work with this data set. Original data files used for this set are located at YOUR\_R\_LIB\_LOCATION/exdata/srphenos.dat (pehnotypes), srphenos.dat (human-readable genotypes) and srphenos.raw (genotypes in internal format)

**Usage**

```
data(srda)
```

**Format**

Standard object of class [gwaa.data-class](#)

**Examples**

```
## Not run:
demo(srda)
## End(Not run)

# load and work with srda
data(srda)
mc <- check.marker(data=srda@gtdata[,1:100], redundant="all", maf=0.01, minconcordance=0.9, fdr=0.01)
plot(mc)
check.trait(names(srda@phdata), srda)
```

---

`sset`*Internal use function for class `snp.mx-class`*

---

**Description**

Interface to C function `sset` subsetting genotypes from `snp.mx-class`

**Usage**

```
sset(data, nsnp, nids, list)
```

**Arguments**

<code>data</code>	genotypic data in internal format
<code>nsnp</code>	no. snps
<code>nids</code>	no. people
<code>list</code>	something internal...

**Details**

Rather simple function which I wrote before discovering R's `setdiff`, etc. functions.

**Value**

Sub-set from `snp.mx-class` object

**Author(s)**

Yurii Aulchenko

**See Also**

`snp.mx-class`



---

`summary.check.marker`*Summary of check.marker object*

---

**Description**

Provides cross-tabulation summarising number of marker which did not pass this or that criteria

**Usage**

```
summary.check.marker(object, ...)
```

**Arguments**

<code>object</code>	object of class <code>check.marker-class</code>
<code>...</code>	additional arguments (not used)

**Value**

A list containing 2 tables: per-marker and per-person inconsistencies

**Author(s)**

Yurii Aulchenko

**See Also**

`check.marker`, `check.marker-class`

**Examples**

```
data(srdta)
mc <- check.marker(srdta, ids=c(1:500))
summary(mc)
```

---

summary.gwaa.data *function to summarise GWAA data*

---

## Description

Summary of phenotypic and genotypic parts of GWAA data

## Usage

```
summary.gwaa.data(object, ...)
```

## Arguments

object	object of class <code>gwaa.data-class</code>
...	additional arguments (not used)

## Value

Returns list with two elements:

pheno	Summary for phenotypic part of gwaa.data object
geno	Summary for genotypic part of gwaa.data object

## Author(s)

Yurii Aulchenko

## See Also

[summary.snp.data](#)

## Examples

```
data(srdta)
# be prepared : long output!
summary(srdta)
```

---

summary.snp.data    *function to summary GWAA data*

---

## Description

Provides summary of an object of class `snp.data-class`. Number of observed genotypes, allelic frequency, genotypic distribution, P-value of the exact test for HWE and chromosome are listed

## Usage

```
summary.snp.data(object, ...)
```

## Arguments

object	snp.data object
...	additional arguments (not used)

## Value

Data frame summary for snp.data object

## Note

The P-values reported for X-chromosome are based on analysis of female data, but other statistics (frequencies, calls, ...) are based on all data. Statistics for Y-chromosome are based on male-only. P-HWE is not defined for mt- and Y- markers (set to 1.0).

## Author(s)

Yurii Aulchenko

## References

Wigginton, JE, Cutler, DJ, and Abecasis, GR (2005) A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics. 76: 887-93.

## See Also

`summary.gwaa.data`, `snp.data-class`

## Examples

```
data(srdta)
summary(srdta@gtdata[,1:20])
```

---

`Xfix`*function to set impossible genotypes as missing*

---

**Description**

Sets impossible genotypes (e.g. heterozygous male X-linked genotypes) to missing

**Usage**

```
Xfix(data)
```

**Arguments**

`data`                      Object of [gwaa.data-class](#)

**Details**

Sets to missing genotypes in the following situations: (1) heterozygous male X-genotypes (2) heterozygous Y- and mtDNA genotypes (3) any Y-genotypes in females. Should only be used after [check.marker](#), which identifies systematic sex errors.

**Value**

The same object of [gwaa.data-class](#), with fixed genotypes

**Author(s)**

Yurii Aulchenko

**See Also**

[check.marker](#)

**Examples**

```
data(ge03d2c)
# many errors
mc0 <- check.marker(ge03d2c)
# take only people and markers passing QC
fixed0 <- ge03d2c[mc0$idok,mc0$snpok]
# major errors fixed, still few males are heterozygous for X-chromosome markers
mc1 <- check.marker(fixed0)
# fix minor X-chromosome problems
fixed1 <- Xfix(fixed0)
# no errors
mc2 <- check.marker(fixed1)
summary(mc2)
```

---

`ztransform`*Transformation to standard Normal*

---

**Description**

Transformation of a variable or residuals from GLM analysis to standard Normal.

**Usage**

```
ztransform(formula, data, family=gaussian)
```

**Arguments**

<code>formula</code>	GLM formula for the variable to be transformed, or just the variable
<code>data</code>	data.frame or gwaa.data object containing the data
<code>family</code>	GLM family

**Details**

Transformation to normality generates a variable which has mean zero and variance of one. If formula used, residuals from regression model are scaled to standard Normal.

**Value**

Vector containing transformed variable, distributed as standard normal.

**Author(s)**

Yurii Aulchenko

**See Also**

[ztransform](#)

**Examples**

```
# uniformly distributed variable
x <- round(runif(200)*100)
# get 7 missing values
x[round(runif(7,min=1,max=100)))] <- NA
# Z-transform
y0 <- ztransform(x)
# Rank-transform to normality
y1 <- rntransform(x)
# test normality of the original and transformed var
shapiro.test(x)
shapiro.test(y0)
```

```
shapiro.test(y1)
# plot histogram
par(mfcol=c(3,1))
hist(x)
hist(y0)
hist(y1)
```