Package ‘gap’

May 14, 2022

Version  1.2.3-6
Date  2022-5-10
Title  Genetic Analysis Package
URL  https://github.com/jinghuazhao/R
BugReports  https://github.com/jinghuazhao/R/issues
Depends  R (>= 2.10), gap.datasets
Imports  dplyr, ggplot2, plotly
Suggests  BradleyTerry2, MASS, Matrix, MCMCglmm, R2jags, bdsmatrix, calibrate, circlize, coda, cowplot, coxme, foreign, forestplot, genetics, grid, haplo.stats, htmlwidgets, jsonlite, kinship2, knitr, lattice, magic, matrixStats, meta, metafor, mets, nlme, pedigree, pedigreemm, plotrix, reshape, rmarkdown, rmeta, rms, survival
VignetteBuilder  knitr
Enhances  shiny
Description  As first reported [Zhao, J. H. 2007. "gap: Genetic Analysis Package". J Stat Soft 23(8):1-18. <doi:10.18637/jss.v023.i08>], it is designed as an integrated package for genetic data analysis of both population and family data. Currently, it contains functions for sample size calculations of both population-based and family-based designs, probability of familial disease aggregation, kinship calculation, statistics in linkage analysis, and association analysis involving genetic markers including haplotype analysis with or without environmental covariates. Over years, the package has been developed in-between many projects hence also in line with the name (gap).
License  GPL (>= 2)
LazyData  Yes
LazyLoad  Yes
NeedsCompilation  yes
Repository  CRAN
Encoding  UTF-8
Date/Publication 2022-05-13 22:10:02 UTC
RoxygenNote 7.1.2

Author Jing Hua Zhao [aut, cre] (<https://orcid.org/0000-0002-1463-5870>, 0000-0003-4930-3582), Kurt Hornik [ctb], Brian Ripley [ctb], Uwe Ligges [ctb]

Maintainer Jing Hua Zhao <jinghuazhao@hotmail.com>

R topics documented:

ab ......................................................... 4
AE3 .......................................................... 5
asplot ...................................................... 7
b2r .......................................................... 9
BFDP ......................................................... 10
bt ............................................................ 12
ccsize ....................................................... 14
chow.test .................................................. 17
circos.cis.vs.trans.plot ................................. 19
circos.cnvplot .......................................... 20
circos.mhtplot .......................................... 20
cis.vs.trans.classification ............................. 21
cnvplot .................................................... 22
comp.score ............................................... 22
cs .......................................................... 24
ESplot ...................................................... 25
fbsize ...................................................... 26
FPRP ......................................................... 28
gc.em ...................................................... 30
gc.lambda ............................................... 32
gcontrol ................................................... 33
gcontrol2 ............................................... 34
gcp ........................................................ 36
geneCounting ............................................ 37
get_b_se ................................................... 40
get_pve_se ................................................. 41
get_sdy .................................................... 41
gif ......................................................... 42
gsmr ......................................................... 44
h2.jags ..................................................... 45
h2G ........................................................ 46
h2GE ....................................................... 47
h2l ......................................................... 47
h2_mzdz ................................................... 48
hap ........................................................ 50
hap.em ..................................................... 52
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hap.score</td>
<td>54</td>
</tr>
<tr>
<td>htr</td>
<td>56</td>
</tr>
<tr>
<td>hwe</td>
<td>58</td>
</tr>
<tr>
<td>hwe.cc</td>
<td>59</td>
</tr>
<tr>
<td>hwe.hardy</td>
<td>61</td>
</tr>
<tr>
<td>hwe.jags</td>
<td>63</td>
</tr>
<tr>
<td>invnormal</td>
<td>66</td>
</tr>
<tr>
<td>kin.morgan</td>
<td>67</td>
</tr>
<tr>
<td>klem</td>
<td>69</td>
</tr>
<tr>
<td>LD22</td>
<td>70</td>
</tr>
<tr>
<td>LDk1</td>
<td></td>
</tr>
<tr>
<td>log10p</td>
<td>72</td>
</tr>
<tr>
<td>log10pvalue</td>
<td>73</td>
</tr>
<tr>
<td>logp</td>
<td>73</td>
</tr>
<tr>
<td>makeped</td>
<td>74</td>
</tr>
<tr>
<td>masize</td>
<td>75</td>
</tr>
<tr>
<td>MCMCgrm</td>
<td>80</td>
</tr>
<tr>
<td>METAL_forestplot</td>
<td>82</td>
</tr>
<tr>
<td>metap</td>
<td>83</td>
</tr>
<tr>
<td>metareg</td>
<td>85</td>
</tr>
<tr>
<td>mhtplot</td>
<td>87</td>
</tr>
<tr>
<td>mhtplot.trunc</td>
<td>89</td>
</tr>
<tr>
<td>mhtplot2</td>
<td>92</td>
</tr>
<tr>
<td>mia</td>
<td>94</td>
</tr>
<tr>
<td>miamiplot</td>
<td>96</td>
</tr>
<tr>
<td>mtdt</td>
<td>97</td>
</tr>
<tr>
<td>mtdt2</td>
<td>99</td>
</tr>
<tr>
<td>muvar</td>
<td>100</td>
</tr>
<tr>
<td>mvmeta</td>
<td>102</td>
</tr>
<tr>
<td>pbsize</td>
<td>104</td>
</tr>
<tr>
<td>pbsize2</td>
<td>105</td>
</tr>
<tr>
<td>pedtodot</td>
<td>107</td>
</tr>
<tr>
<td>pfc</td>
<td>110</td>
</tr>
<tr>
<td>pfc.sim</td>
<td>112</td>
</tr>
<tr>
<td>pgc</td>
<td>113</td>
</tr>
<tr>
<td>plot.hap.score</td>
<td>115</td>
</tr>
<tr>
<td>pqt12dplot</td>
<td>116</td>
</tr>
<tr>
<td>pqt12dplotly</td>
<td>117</td>
</tr>
<tr>
<td>pqt3dplotly</td>
<td>118</td>
</tr>
<tr>
<td>print.hap.score</td>
<td>119</td>
</tr>
<tr>
<td>pvalue</td>
<td>120</td>
</tr>
<tr>
<td>qqfun</td>
<td>120</td>
</tr>
<tr>
<td>qquNit</td>
<td>122</td>
</tr>
<tr>
<td>read.ms.output</td>
<td>124</td>
</tr>
<tr>
<td>revStrand</td>
<td>125</td>
</tr>
<tr>
<td>runshinygap</td>
<td>126</td>
</tr>
<tr>
<td>s2k</td>
<td>127</td>
</tr>
<tr>
<td>sentinels</td>
<td>128</td>
</tr>
</tbody>
</table>
Test/Power calculation for mediating effect

Description

This function tests for or obtains power of mediating effect based on estimates of two regression coefficients and their standard errors. Note that for binary outcome or mediator, one should use log-odds ratio and its standard error.

Usage

```r
ab(
  type = "power",
  n = 25000,
  a = 0.15,
  sa = 0.01,
  b = log(1.19),
  sb = 0.01,
  alpha = 0.05,
  fold = 1
)
```

Arguments

- `type` string option: "test", "power".
- `n` default sample size to be used for power calculation.
- `a` regression coefficient from independent variable to mediator.
- `sa` SE(a).
- `b` regression coefficient from mediator variable to outcome.
- `sb` SE(b).
- `alpha` size of significance test for power calculation.
- `fold` fold change for power calculation, as appropriate for a range of sample sizes.

Value

The returned value are z-test and significance level for significant testing or sample size/power for a given fold change of the default sample size.
Author(s)
Jing Hua Zhao

References


See Also
ccsize

Examples
```r
## Not run:
ab()
n <- power <- vector()
for (j in 1:10)
{
  z <- ab(fold=j*0.01)
n[j] <- z[1]
power[j] <- z[2]
}
plot(n,power,xlab="Sample size",ylab="Power")
title("SNP-BMI-T2D association in EPIC-Norfolk study")
## End(Not run)
```

AE3

AE model using nuclear family trios

Description
This function is adapted from example 7.1 of Rabe-Hesketh et al. (2008). It also provides heritability estimate and confidence intervals.

Usage
```r
AE3(model, random, data, seed = 1234, n.sim = 50000, verbose = TRUE)
```
Arguments

- **model**: a linear mixed model formula, see example below.
- **random**: random effect, see example below.
- **data**: data to be analyzed.
- **seed**: random number seed.
- **n.sim**: number of simulations.
- **verbose**: a flag for printing out results.

Value

The returned value is a list containing:

- **lme.result**: the linear mixed model result.
- **h2**: the heritability estimate.
- **CL**: confidence intervals.

Note

Adapted from f.mbf.R from the paper.

Author(s)

Jing Hua Zhao

References


Examples

```R
## Not run:
require(gap.datasets)
AE3(bwt ~ male + first + midage + highage + birthyr,
    list(familyid = pdIdent(~var1 + var2 + var3 -1)), mfblong)
## End(Not run)
```
asplot

Regional association plot

Description
This function obtains regional association plot for a particular locus, based on the information about recombinatino rates, linkage disequilibria between the SNP of interest and neighbouring ones, and single-point association tests p values.

Usage

\[
\text{asplot(}
\begin{align*}
\text{locus,} \\
\text{map,} \\
\text{genes,} \\
\text{flanking = 1000,} \\
\text{best.pval = NULL,} \\
\text{sf = c(4, 4),} \\
\text{logpmax = 10,} \\
\text{pch = 21}
\end{align*}
\)
\]

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>locus</td>
<td>Data frame with columns c(&quot;CHR&quot;, &quot;POS&quot;, &quot;NAME&quot;, &quot;PVAL&quot;, &quot;RSQR&quot;) containing association results.</td>
</tr>
<tr>
<td>map</td>
<td>Genetic map, i.e, c(&quot;POS&quot;,&quot;THETA&quot;,&quot;DIST&quot;).</td>
</tr>
<tr>
<td>genes</td>
<td>Gene annotation with columns c(&quot;START&quot;, &quot;STOP&quot;, &quot;STRAND&quot;, &quot;GENE&quot;).</td>
</tr>
<tr>
<td>flanking</td>
<td>Flanking length.</td>
</tr>
<tr>
<td>best.pval</td>
<td>Best p value for the locus of interest.</td>
</tr>
<tr>
<td>sf</td>
<td>scale factors for p values and recombination rates, smaller values are necessary for gene dense regions.</td>
</tr>
<tr>
<td>logpmax</td>
<td>Maximum value for -log10(p).</td>
</tr>
<tr>
<td>pch</td>
<td>Plotting character for the SNPs to be highlighted, e.g., 21 and 23 refer to circle and diamond.</td>
</tr>
</tbody>
</table>

Details
Note that the best p value is not necessarily within locus in the original design.

Author(s)
Paul de Bakker, Jing Hua Zhao, Shengxu Li
References

DGI. Whole-genome association analysis identifies novel loci for type 2 diabetes and triglyceride levels. Science 2007;316(5829):1331-6

Examples

```r
## Not run:
require(gap.datasets)
asplot(CDKNlocus, CDKNmap, CDKNgenes)
title("CDKN2A/CDKN2B Region")
asplot(CDKNlocus, CDKNmap, CDKNgenes, best.pval=5.4e-8, sf=c(3,6))

## NCBI2R

options(stringsAsFactors=FALSE)
p <- with(CDKNlocus,data.frame(SNP=NAME,PVAL))
hit <- subset(p,PVAL==min(PVAL,na.rm=TRUE))$SNP

library(NCBI2R)
# LD under build 36
chr_pos <- GetSNPInfo(with(p,SNP))[["chr","chrpos"]]
l <- with(chr_pos,min(as.numeric(chrpos),na.rm=TRUE))
u <- with(chr_pos,max(as.numeric(chrpos),na.rm=TRUE))
LD <- with(chr_pos,GetLDInfo(unique(chr),l,u))
# We have complaints; a possibility is to get around with
hit_LD <- subset(LD,SNPA==hit)
hit_LD <- within(hit_LD,{RSQR=r2})
info <- GetSNPInfo(p$SNP)
haldane <- function(x) 0.5*(1-exp(-2*x))
locus <- with(info, data.frame(CHR=chr,pos=chrpos,NAME=marker,
    DIST=(chrpos-min(chrpos))/1000000,
    THETA=haldane((chrpos-min(chrpos))/100000000)))
locus <- merge.data.frame(locus,hit_LD,by.x="NAME",by.y="SNPB",all=TRUE)
locus <- subset(locus,!is.na(pos))
ann <- AnnotateSNPList(p$SNP)
gen <- with(ann, data.frame(ID=locusID,CLASS=fxn_class,PATH=pathways,
    START=GeneLowPoint,STOP=GeneHighPoint,
    STRAND=ori,GENE=genesymbol,BUILD=build,CYTO=cyto))

attach(genes)
ugen <- unique(GENE)
ustart <- as.vector(as.table(by(START,GENE,min)[ugen]))
ustop <- as.vector(as.table(by(STOP,GENE,max)[ugen]))
ustrand <- as.vector(as.table(by(as.character(STRAND),GENE,max)[ugen]))
detach(genes)
gen <- data.frame(START=ustart,STOP=ustop,STRAND=ustrand,GENE=ugen)
gen <- subset(genesis,START!=0)
rm(l,u,ugen,ustart,ustop,ustrand)
# Assume we have the latest map as in CDKNmap
asplot(locus,CDKNmap,genes)
```
## b2r

Obtain correlation coefficients and their variance-covariances

**Description**

This function converts linear regression coefficients of phenotype on single nucleotide polymorphisms (SNPs) into Pearson correlation coefficients with their variance-covariance matrix. It is useful as a preliminary step for meta-analyze SNP-trait associations at a given region. Between-SNP correlations (e.g., from HapMap) are required as auxiliary information.

**Usage**

```r
b2r(b, s, rho, n)
```

**Arguments**

- `b`: the vector of linear regression coefficients.
- `s`: the corresponding vector of standard errors.
- `rho`: triangular array of between-SNP correlation.
- `n`: the sample size.

**Value**

The returned value is a list containing:

- `r`: the vector of correlation coefficients
- `V`: the variance-covariance matrix of correlations

**Author(s)**

Jing Hua Zhao

**References**


**See Also**

`mvmeta`, `LD22`
Examples

```r
## Not run:
n <- 10
r <- c(1, 0.2, 1, 0.4, 0.5, 1)
b <- c(0.1, 0.2, 0.3)
s <- c(0.4, 0.3, 0.2)
bs <- b2r(b, s, r, n)
## End(Not run)
```

BFDP

**Bayesian false-discovery probability**

Description

This function calculates BFDP, the approximate \( P(H_0|\dot{\theta}) \), given an estimate of the log relative risk, \( \dot{\theta} \), the variance of this estimate, \( V \), the prior variance, \( W \), and the prior probability of a non-null association. When \( \text{logscale} = \text{TRUE} \), the function accepts an estimate of the relative risk, \( \dot{RR} \), and the upper point of a 95% confidence interval \( \dot{RR}_{hi} \).

Usage

```r
BFDP(a, b, pi1, W, logscale = FALSE)
```

Arguments

- `a`: parameter value at which the power is to be evaluated.
- `b`: the variance for `a`, or the upper point \( (RR_{hi}) \) of a 95% CI if `logscale` = `FALSE`.
- `pi1`: the prior probability of a non-null association.
- `W`: the prior variance.
- `logscale`: `FALSE` = the original scale, `TRUE` = the log scale.

Value

The returned value is a list with the following components:

- \( \text{PH0} \): probability given `a,b`.
- \( \text{PH1} \): probability given `a,b,W`.
- \( \text{BF} \): Bayes factor, \( P_{H_0} / P_{H_1} \).
- \( \text{BFDP} \): Bayesian false-discovery probability.
- \( \text{ABF} \): approximate Bayes factor.
- \( \text{ABFDP} \): approximate Bayesian false-discovery probability.
Note
Adapted from BFDP functions by Jon Wakefield on 17th April, 2007.

Author(s)
Jon Wakefield, Jing Hua Zhao

References

See Also
FPRP

Examples
## Not run:
# Example from BDFP.xls by Jon Wakefield and Stephanie Monnier
# Step 1 - Pre-set an BFDP-level threshold for noteworthiness: BFDP values below this
# threshold are noteworthy
# The threshold is given by R/(1+R) where R is the ratio of the cost of a false
# non-discovery to the cost of a false discovery
T <- 0.8

# Step 2 - Enter up values for the prior that there is an association
pi0 <- c(0.7, 0.5, 0.01, 0.001, 0.00001, 0.6)

# Step 3 - Enter the value of the OR that is the 97.5% point of the prior, for example
# if we pick the value 1.5 we believe that the prior probability that the
# odds ratio is bigger than 1.5 is 0.025.
ORhi <- 3
W <- (log(ORhi)/1.96)^2
W

# Step 4 - Enter OR estimate and 95% confidence interval (CI) to obtain BFDP
OR <- 1.316
OR_L <- 1.10
OR_U <- 2.50
logOR <- log(OR)
selogOR <- (log(OR_U)-log(OR))/1.96
r <- W/(W+selogOR^2)

z <- logOR/selogOR
z
ABF <- exp(-z^2*r/2)/sqrt(1-r)
ABF
FF <- (1-pi0)/pi0
FF
BFDPex <- FF*ABF/(FF*ABF+1)
BFDPex
pi0[BFDPex>T]

## now turn to BFDP

pi0 <- c(0.7,0.5,0.01,0.001,0.00001,0.6)
ORhi <- 3
OR <- 1.316
OR_U <- 2.50
W <- (log(ORhi)/1.96)^2
z <- BFDP(OR,OR_U,pi0,W)
z

## End(Not run)

---

**bt**  
*Bradley-Terry model for contingency table*

**Description**
This function calculates statistics under Bradley-Terry model.

**Usage**
bt(x)

**Arguments**
x the data table.

**Value**
The returned value is a list containing:
y A column of 1
count the frequency count/weight
allele the design matrix
bt.glm a glm.fit object
etdt.dat a data table that can be used by ETDT

**Note**
Adapted from a SAS macro for data in the example section.
Author(s)

Jing Hua Zhao

References


See Also

mtdt

Examples

## Not run:
# Copeman JB, Cucca F, Hearne CM, Cornall RJ, Reed PW,
# (1995) Linkage disequilibrium mapping of a type 1 diabetes susceptibility gene (IDDM7) to chromosome 2q31-q33.
# Nat Genet 9: 80-5

x <- matrix(c(0,0, 0, 2, 0,0, 0, 0, 0, 0, 0, 0,
             0,1, 3, 0,0, 0, 2, 3, 0, 0, 0,
             2,3,26,35, 7,0,2,10,11, 3, 4, 1,
             2,3,22,26, 6,2,4, 4,10, 2, 2, 0,
             0,1, 7,10, 2,0, 0, 2, 2, 1, 1, 0,
             0,0, 1, 4, 0,1, 0, 1, 0, 0, 0,
             0,2, 5, 4, 1,1, 0, 0, 0, 2, 0, 0,
             0,0, 2, 6, 1,0, 2, 0, 2, 0, 0, 0,
             0,3, 6,19, 6,0,0, 2, 5, 3, 0, 0,
             0,0, 3, 1, 1,0, 0, 0, 1, 0, 0, 0,
             0,0, 0, 2, 0,0, 0, 0, 0, 0, 0, 0,
             0,0, 1, 0,0, 0, 0, 0, 0, 0, 0, 0),nrow=12)

# Bradley-Terry model, only deviance is available in glm
# (SAS gives score and Wald statistics as well)
bt.ex<-bt(x)
anova(bt.ex$bt.glm)
summary(bt.ex$bt.glm)

## End(Not run)
Power and sample size for case-cohort design

Description

The power of the test is according to

\[ \Phi \left( Z_\alpha + m^{1/2} \theta \sqrt{ \frac{p_1 p_2 p_D}{q + (1 - q)p_D} } \right) \]

where \( \alpha \) is the significance level, \( \theta \) is the log-hazard ratio for two groups, \( p_j, j=1, 2 \), are the proportion of the two groups in the population. \( m \) is the total number of subjects in the subcohort, \( p_D \) is the proportion of the failures in the full cohort, and \( q \) is the sampling fraction of the subcohort.

Usage

ccsize(n, q, pD, p1, theta, alpha, beta = 0.2, power = FALSE, verbose = FALSE)

Arguments

- \( n \): the total number of subjects in the cohort.
- \( q \): the sampling fraction of the subcohort.
- \( pD \): the proportion of the failures in the full cohort.
- \( p1 \): proportions of the two groups (p2=1-p1).
- \( \theta \): log-hazard ratio for two groups.
- \( \alpha \): type I error – significant level.
- \( \beta \): type II error.
- \( \text{power} \): if specified, the power for which sample size is calculated.
- \( \text{verbose} \): error messages are explicitly printed out.

Details

Alternatively, the sample size required for the subcohort is

\[ m = nBp_D/(n - B(1 - p_D)) \]

where \( B = (Z_{1-\alpha} + Z_\beta)^2/(\theta^2 p_1 p_2 p_D) \), and \( n \) is the size of cohort.

When infeasible configurations are specified, a sample size of -999 is returned.

Value

The returned value is a value indicating the power or required sample size.

Note

Programmed for EPIC study. keywords misc
Author(s)
Jing Hua Zhao

References

See Also
pbsize

Examples
## Not run:
# Table 1 of Cai & Zeng (2004).
outfile <- "table1.txt"
cat("n","pD","p1","theta","q","power\n",file=outfile,sep="\t")
alpha <- 0.05
n <- 1000
for(pD in c(0.10,0.05))
{
  for(p1 in c(0.3,0.5))
  {
    for(theta in c(0.5,1.0))
    {
      for(q in c(0.1,0.2))
      {
        power <- ccsize(n,q,pD,p1,alpha,theta)
cat(n,"\t",pD,"\t",p1,"\t",theta,"\t",q,"\t",signif(power,3),"\n",file=outfile,append=TRUE)
      }
    }
  }
}
n <- 5000
for(pD in c(0.05,0.01))
{
  for(p1 in c(0.3,0.5))
  {
    for(theta in c(0.5,1.0))
    {
      for(q in c(0.01,0.02))
      {
        power <- ccsize(n,q,pD,p1,alpha,theta)
cat(n,"\t",pD,"\t",p1,"\t",theta,"\t",q,"\t",signif(power,3),"\n",file=outfile,append=TRUE)
      }
    }
  }
}
table1<-read.table(outfile,header=TRUE,sep="\t")
```r
unlink(outfile)
# ARIC study
outfile <- "aric.txt"
q <- seq(1.35,1.48,1.50)
alpha <- 0.05
theta <- c(1.35,1.40,1.45)
beta1 <- 0.8
s_nb <- c(1463,722,468)
cat("n","pD","p1","hr","q","power","ssize\n",file=outfile,sep="t")
for(i in 1:3)
{
  q <- s_nb[i]/n
  power <- ccsize(n,q,pD,p1,alpha,log(theta[i]))
  ssize <- ccsize(n,q,pD,p1,alpha,log(theta[i]),beta1)
  cat(n,"pD",p1,"hr","q","power","ssize\n",file=outfile,sep="t")
}
aric<-read.table(outfile,header=TRUE,sep="t")
unlink(outfile)
# EPIC study
outfile <- "epic.txt"
q <- seq(0.0,0.5,by=0.1)
s_pD <- seq(0.3,0.2,0.1,0.05)
s_p1 <- seq(0.1,0.5,by=0.1)
s_hr <- seq(1.1,1.4,by=0.1)
cat("n","pD","p1","hr","alpha","ssize\n",file=outfile,sep="t")
# direct calculation
for(pD in s_pD)
{
  for(p1 in s_p1)
  {
    for(hr in s_hr)
    {
      ssize <- ccsize(n,q,pD,p1,alpha,log(hr),power)
      if (ssize>0) cat(n,"pD",p1,"hr","q","power","ssize\n",file=outfile,append=TRUE)
    }
  }
}
epic<-read.table(outfile,header=TRUE,sep="t")
unlink(outfile)
# exhaustive search
outfile <- "search.txt"
s_q <- seq(0.01,0.5,by=0.01)
cat("n","pD","p1","hr","nq","power\n",file=outfile,sep="t")
for(pD in s_pD)
{
  for(p1 in s_p1)
  {
    for(hr in s_hr)
    {
      ssize <- ccsize(n,q,pD,p1,alpha,log(hr),power)
      if (ssize>0) cat(n,"pD",p1,"hr","nq","power","ssize\n",file=outfile,append=TRUE)
    }
  }
}
```

for(hr in s_hr)
{
    for(q in s_q)
    {
        power <- ccsizex(n,q,pD,p1,alpha,log(hr))
        cat(n,"\t",pD,"\t",p1,"\t",hr,"\t","q",n,"\t",alpha,"\t",power,"\n",
            file=outfile,append=TRUE)
    }
}

search<-read.table(outfile,header=TRUE,sep="\t")
unlink(outfile)
## End(Not run)

chow.test  
Chow's test for heterogeneity in two regressions

Description
Chow's test is for differences between two or more regressions. Assuming that errors in regressions 1 and 2 are normally distributed with zero mean and homoscedastic variance, and they are independent of each other, the test of regressions from sample sizes \( n_1 \) and \( n_2 \) is then carried out using the following steps. 1. Run a regression on the combined sample with size \( n = n_1 + n_2 \) and obtain within group sum of squares called \( S_1 \). The number of degrees of freedom is \( n_1 + n_2 - k \), with \( k \) being the number of parameters estimated, including the intercept. 2. Run two regressions on the two individual samples with sizes \( n_1 \) and \( n_2 \), and obtain their within group sums of squares \( S_2 + S_3 \), with \( n_1 + n_2 - 2k \) degrees of freedom. 3. Conduct an \( F(k,n_1+n_2-2k) \) test defined by

\[
F = \frac{[S_1 - (S_2 + S_3)]/k}{[(S_2 + S_3)/(n_1 + n_2 - 2k)]}
\]

If the \( F \) statistic exceeds the critical \( F \), we reject the null hypothesis that the two regressions are equal.

Usage
chow.test(y1, x1, y2, x2, x = NULL)

Arguments
- \( y1 \): a vector of dependent variable.
- \( x1 \): a matrix of independent variables.
- \( y2 \): a vector of dependent variable.
- \( x2 \): a matrix of independent variables.
- \( x \): a known matrix of independent variables.
Details
In the case of haplotype trend regression, haplotype frequencies from combined data are known, so can be directly used.

Value
The returned value is a vector containing (please use subscript to access them):

- \( F \) the F statistic
- \( df1 \) the numerator degree(s) of freedom
- \( df2 \) the denominator degree(s) of freedom
- \( p \) the p value for the F test

Note
adapted from chow.R.

Author(s)
Shigenobu Aoki, Jing Hua Zhao

Source
http://aoki2.si.gunma-u.ac.jp/R/

References

See Also
htr

Examples
```r
## Not run:
dat1 <- matrix(c(
  1.2, 1.9, 0.9,
  1.6, 2.7, 1.3,
  3.5, 3.7, 2.0,
  4.0, 3.1, 1.8,
  5.6, 3.5, 2.2,
  5.7, 7.5, 3.5,
  6.7, 1.2, 1.9,
  7.5, 3.7, 2.7,
  8.5, 0.6, 2.1,
  9.7, 5.1, 3.6), byrow=TRUE, ncol=3)

dat2 <- matrix(c(
```
1.4, 1.3, 0.5, 1.5, 2.3, 1.3, 3.1, 3.2, 2.5, 4.4, 3.6, 1.1, 5.1, 3.1, 2.8, 5.2, 7.3, 3.3, 6.5, 1.5, 1.3, 7.8, 3.2, 2.2, 8.1, 0.1, 2.8, 9.5, 5.6, 3.9), byrow=TRUE, ncol=3)

y1<-dat1[,3]
y2<-dat2[,3]
x1<-dat1[,1:2]
x2<-dat2[,1:2]
chow.test.r<-chow.test(y1,x1,y2,x2)

## End(Not run)

---

circos.cis.vs.trans.plot

circos plot of cis/trans classification

Description
The function implements a circos plot at the early stage of SCALLOP-INF meta-analysis.

Usage

```
circos.cis.vs.trans.plot(hits, panel, id, radius = 1e+06)
```

Arguments

- **hits**: A text file as input data with variables named "CHR", "BP", "SNP", "prot".
- **panel**: Protein panel with prot(ein), uniprot (id) and "chr"."start","end","gene".
- **id**: Identifier.
- **radius**: The flanking distance as cis.

Value
None.

Examples

```
## Not run:
circos.cis.vs.trans.plot(hits="INF1.clumped", panel=inf1, id="uniprot")
```

## End(Not run)
circos.mhtplot

circos Manhattan plot with gene annotation

description
The function generates circos Manhattan plot with gene annotation.

Usage
circos.mhtplot(data, glist)

Arguments
- data: Data to be used.
- glist: A gene list.

Value
None.

circos.cnvplot

circos plot of CNVs.

description
The function plots frequency of CNVs.

Usage
circos.cnvplot(data)

Arguments
- data: CNV data containing chromosome, start, end and freq.

Value
None.

Examples
circos.cnvplot(cnv)
cis.vs.trans.classification

A cis/trans classifier

Description

The function classifies variants into cis/trans category according to a panel which contains id, chr, start, end, gene variables.

Usage

cis.vs.trans.classification(hits, panel, id, radius = 1e+06)

Arguments

hits Data to be used, which contains prot, Chr, bp, id and/or other information such as SNPId.
panel Panel data.
id Identifier.
radius The flanking distance for variants.

Value

The cis/trans classification.

Author(s)

James Peters

Examples

cis.vs.trans.classification(hits=jma.cojo, panel=inf1, id="uniprot")
## Not run:
INF <- Sys.getenv("INF")
f <- file.path(INF,"work","INF1.merge")
clumped <- read.delim(f,as.is=TRUE)
hits <- merge(clumped[c("CHR","POS","MarkerName","prot","log10p")],
inf1[c("prot","uniprot")],by="prot")
names(hits) <- c("prot","Chr","bp","SNP","log10p","uniprot")
cistrans <- cis.vs.trans.classification(hits,inf1,"uniprot")
cis.vs.trans <- with(cistrans,data)
knitr::kable(with(cistrans,table),caption="Table 1. cis/trans classification")
with(cistrans,total)
## End(Not run)
**cnvplot**

**Description**

The function generates a plot containing genomewide copy number variants (CNV) chr, start, end, freq(uencies).

**Usage**

```r
cnvplot(data)
```

**Arguments**

- `data`  
  Data to be used.

**Value**

None.

**Examples**

```r
knitr::kable(cnv, caption="A CNV dataset")
cnvplot(cnv)
```

---

**comp.score**

**Description**

The function empirically estimate the variance of the score functions. The variance-covariance matrix consists of two parts: the additive part and the part for the individual-specific environmental effect. Other reasonable decompositions are possible.

**Usage**

```r
comp.score(
  ibddata = "ibd_dist.out",
  phenotype = "pheno.dat",
  mean = 0,
  var = 1,
  h2 = 0.3
)
```
Arguments

**ibddata**
The output file from GENEHUNTER using command "dump ibd". The default file name is *ibd*.dist.out.

**phenotype**
The file of pedigree structure and trait value. The default file name is "pheno.dat". Columns (no headings) are: family ID, person ID, father ID, mother ID, gender, trait value, where Family ID and person ID must be numbers, not characters. Use character "NA" for missing phenotypes.

**mean**
(population) mean of the trait, with a default value of 0.

**var**
(population) variance of the trait, with a default value of 1.

**h2**
heritability of the trait, with a default value of 0.3.

Details

This program has the following improvement over "score.r":

1. It works with selected nuclear families
2. Trait data on parents (one parent or two parents), if available, are utilized.
3. Besides a statistic assuming no locus-specific dominance effect, it also computes a statistic that allows for such effect. It computes two statistics instead of one.

Function "merge" is used to merge the IBD data for a pair with the transformed trait data (i.e., $w_k w_l$).

Value

a matrix with each row containing the location and the statistics and their p-values.

Note

Adapt from score2.r.

Author(s)

Yingwei Peng, Kai Wang

References


Examples

```r
## Not run:
# An example based on GENEHUNTER version 2.1, with quantitative trait data in file
# "pheno.dat" generated from the standard normal distribution. The following
# example shows that it is possible to automatically call GENEHUNTER using R
# function "system".

cwd <- getwd()
if (!require(cs)) {
  stop("cs not installed", call. = FALSE)
}

if (exists("cs.dir", where = getwd())) {
  dir()
  dir()
}

# system("gh < gh.inp")

cs.dir <- file.path(path.package("gap"), "tests/comp.score")
setwd(cs.dir)

dir()

# system("gh < gh.inp")
cs.default <- comp.score()
setwd(cwd)

## End(Not run)
```

---

cs  

### Credible set

Description

The function implements credible set as in fine-mapping.

Usage

```r
cs(tbl, b = "Effect", se = "StdErr", log_p = NULL, cutoff = 0.95)
```

Arguments

- **tbl**: Input data.
- **b**: Effect size.
- **se**: Standard error.
- **log_p**: if not NULL it will be used to derive z-statistic
- **cutoff**: Threshold for inclusion.

Value

Credible set.
## Examples

```r
## Not run:
\preformatted{
  zcat METAL/4E.BP1-1.tbl.gz | \
  awk 'NR==1 || ($1==4 && $2 >= 187158034 - 1e6 && $2 < 187158034 + 1e6)' > 4E.BP1.z 
}
tbl <- within(read.delim("4E.BP1.z"),{logp <- logp(Effect/StdErr)})
z <- cs(tbl)
l <- cs(tbl,log_p="logp")
## End(Not run)
```

---

### Description

The function accepts parameter estimates and their standard errors for a range of models.

### Usage

```r
ESplot(ESdat, alpha = 0.05)
```

### Arguments

- **ESdat**: A data frame consisting of model id, parameter estimates and standard errors.
- **alpha**: Type-I error rate used to construct 100(1-alpha) confidence interval.

### Value

A high resolution plot object.

### Author(s)

Jing Hua Zhao

### Examples

```r
rs12075 <- data.frame(id=c("CCL2","CCL7","CCL8","CCL11","CCL13","CXCL6","Monocytes"),
  b=c(0.1694,-0.0899,-0.0973,0.0749,0.189,0.0816,0.0338387),
  se=c(0.0113,0.013,0.0116,0.0114,0.0114,0.0115,0.00713386))
ESplot(rs12075)
```

# The function replaces an older implementation.
```r
within(data.frame(
  id=c("Basic model","Adjusted","Moderately adjusted","Heavily adjusted","Other"),
  b=log(c(4.5,3.5,2.5,1.5,1)),
  se=c(0.2,0.1,0.2,0.3,0.2)
), {
```
lcl <- exp(b-1.96*se)
ucl <- exp(b+1.96*se)
x <- seq(-2,8,length=length(id))
y <- 1:length(id)
plot(x,y,type="n",xlab="",ylab="",axes=FALSE)
points((lcl+ucl)/2,y,pch=22,bg="black",cex=3)
segments(lcl,y,ucl,y,lwd=3,lty="solid")
axis(1,cex.axis=1.5,lwd=0.5)
abline(v=1)
axis(2,labels=id,at=y,lty="blank",hadj=0.2,cex.axis=1.5)
title("A fictitious plot")
}

---

### fbsize

**Sample size for family-based linkage and association design**

**Description**

This function implements Risch and Merikangas (1996) statistics evaluating power for family-based linkage (affected sib pairs, ASP) and association design. They are potentially useful in the prospect of genome-wide association studies.

**Usage**

```r
fbsize(
  gamma,
  p,
  alpha = c(1e-04, 1e-08, 1e-08),
  beta = 0.2,
  debug = 0,
  error = 0
)
```

**Arguments**

- **gamma**: genotype relative risk assuming multiplicative model.
- **p**: frequency of disease allele.
- **alpha**: Type I error rates for ASP linkage, TDT and ASP-TDT.
- **beta**: Type II error rate.
- **debug**: verbose output.
- **error**: 0=use the correct formula, 1=the original paper.

**Details**

The function calls auxiliary functions sn() and strlen; sn() contains the necessary thresholds for power calculation while strlen() evaluates length of a string (generic).
Value

The returned value is a list containing:

- **gamma**: input gamma.
- **p**: input p.
- **n1**: sample size for ASP.
- **n2**: sample size for TDT.
- **n3**: sample size for ASP-TDT.
- **lambda0**: lambda o.
- **lamdas**: lambda s.

Note

extracted from rm.c.

Author(s)

Jing Hua Zhao

References


See Also

- **pbsize**

Examples

```r
models <- matrix(c(
  4.0, 0.01,
  4.0, 0.10,
  4.0, 0.50,
  4.0, 0.80,
  2.0, 0.01,
  2.0, 0.10,
  2.0, 0.50,
  2.0, 0.80,
  1.5, 0.01,
  1.5, 0.10,
  1.5, 0.50,
  1.5, 0.80), ncol=2, byrow=TRUE)
outfile <- "fbsize.txt"
cat("gamma","p","Y","N_asp","P_A","H1","N_tdt","H2","N_asp/tdt","L_o","L_s
", sep="", file=outfile)
```
file=outfile,sep="\t")
for(i in 1:12) {
  g <- models[i,1]
  p <- models[i,2]
  z <- fbsize(g,p)
  cat(z$gamma,z$p,z$y,z$n1,z$pA,z$h1,z$n2,z$h2,z$n3,z$lambdao,z$lambda, file=outfile,
      append=TRUE,sep="\t")
  cat("\n",file=outfile,append=TRUE)
}
table1 <- read.table(outfile,header=TRUE,sep="\t")
nc <- c(4,7,9)
table1[,nc] <- ceiling(table1[,nc])
dc <- c(3,5,6,8,10,11)
table1[,dc] <- round(table1[,dc],2)
unlink(outfile)
# APOE-4, Scott WK, Pericak-Vance, MA & Haines JL
# Genetic analysis of complex diseases 1327
# note to replicate the Table we need set alpha=9.961139e-05,4.910638e-08 and
# beta=0.2004542 or reset the quantiles in fbsize.R

---

### FPRP

**False-positive report probability**

**Description**

The function calculates the false positive report probability (FPRP), the probability of no true association between a genetic variant and disease given a statistically significant finding, which depends not only on the observed P value but also on both the prior probability that the association is real and the statistical power of the test. An associate result is the false negative reported probability (FNRP). See example for the recommended steps.

**Usage**

FPRP(a, b, pi0, ORlist, logscale = FALSE)

**Arguments**

- **a**: parameter value at which the power is to be evaluated.
- **b**: the variance for a, or the upper point of a 95% CI if logscale=FALSE.
- **pi0**: the prior probability that \( H_0 \) is true.
- **ORlist**: a vector of ORs that is most likely.
- **logscale**: TRUE, a, b in log scale. FALSE, a, b in original scale.
Details

The FPRP and FNRP are derived as follows. Let \( H_0 \) = null hypothesis (no association), \( H_A \) = alternative hypothesis (association). Since classic frequentist theory considers they are fixed, one has to resort to Bayesian framework by introducing prior, \( \pi = P(H_0 = \text{TRUE}) = P(\text{association}) \). Let \( T \) = test statistic, and \( P(T > z_\alpha | H_0 = \text{TRUE}) = P(\text{rejecting } H_0 | H_0 = \text{TRUE}) = \alpha \), \( P(T > z_\alpha | H_0 = \text{FALSE}) = P(\text{rejecting } H_0 | H_A = \text{TRUE}) = 1 - \beta \). The joint probability of test and truth of hypothesis can be expressed by \( \alpha \), \( \beta \) and \( \pi \).

<table>
<thead>
<tr>
<th>Truth of ( H_A )</th>
<th>significant</th>
<th>nonsignificant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>(1 - ( \beta ))( \pi )</td>
<td>( \beta \pi )</td>
<td>( \pi )</td>
</tr>
<tr>
<td>FALSE</td>
<td>( \alpha (1 - \pi) )</td>
<td>( (1 - \alpha)(1 - \pi) )</td>
<td>1 - ( \pi )</td>
</tr>
<tr>
<td>Total</td>
<td>(1 - ( \beta ))( \pi ) + ( \alpha (1 - \pi) )</td>
<td>( \beta \pi + (1 - \alpha)(1 - \pi) )</td>
<td>1</td>
</tr>
</tbody>
</table>

We have \( FPRP = P(H_0 = \text{TRUE}|T > z_\alpha) = \alpha (1 - \pi)/[\alpha (1 - \pi) + (1 - \beta)\pi] = (1 + \pi/(1 - \pi)][(1 - \beta)/\alpha]^{-1} \) and similarly \( FNRP = (1 + [(1 - \alpha)/\beta][(1 - \pi)/\pi])^{-1} \).

Value

The returned value is a list with components,

- \( p \) p value corresponding to \( a,b \)
- \( \text{power} \) the power corresponding to the vector of ORs
- \( \text{FPRP} \) False-positive report probability
- \( \text{FNRP} \) False-negative report probability

Author(s)

Jing Hua Zhao

References


See Also

\( \text{BFDP} \)

Examples

```r
# Example by Laure El ghormli & Sholom Wacholder on 25-Feb-2004
# Step 1 - Pre-set an FPRP-level criterion for noteworthiness
T <- 0.2
```

# Step 2 - Enter values for the prior that there is an association
pi0 <- c(0.25,0.1,0.01,0.001,0.0001,0.00001)

# Step 3 - Enter values of odds ratios (OR) that are most likely, assuming that
# there is a non-null association

ORlist <- c(1.2,1.5,2.0)

# Step 4 - Enter OR estimate and 95% confidence interval (CI) to obtain FPRP

OR <- 1.316
ORlo <- 1.08
ORhi <- 1.60

logOR <- log(OR)
selogOR <- abs(logOR-log(ORhi))/1.96
p <- ifelse(logOR>0,2*(1-pnorm(logOR/selogOR)),2*pnorm(logOR/selogOR))
p
q <- qnorm(p/2)
POWER <- ifelse(log(ORlist)>0,1-pnorm(q-log(ORlist)/selogOR),
                pnorm(-q-log(ORlist)/selogOR))
POWER
FPRPex <- t(p*(1-pi0)/(p*(1-pi0)+POWER)
row.names(FPRPex) <- pi0
colnames(FPRPex) <- ORlist
FPRPex
FPRPex>T

## now turn to FPRP
OR <- 1.316
ORhi <- 1.60
ORlist <- c(1.2,1.5,2.0)
pi0 <- c(0.25,0.1,0.01,0.001,0.0001,0.00001)
z <- FPRP(OR,ORhi,pi0,ORlist,logscale=FALSE)
z

## End(Not run)
Arguments

data Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(data) = 2*K. Rows represent alleles for each subject.

locus.label Vector of labels for loci, of length K (see definition of data matrix).

converge.eps Convergence criterion, based on absolute change in log likelihood (lnlike).

maxiter Maximum number of iterations of EM.

handle.miss a flag for handling missing genotype data, 0=no, 1=yes.

miss.val missing value.

control a function, see genecounting.

Value

List with components:

converge Indicator of convergence of the EM algorithm (1=converged, 0 = failed).

niter Number of iterations completed in the EM algorithm.

locus.info A list with a component for each locus. Each component is also a list, and the items of a locus- specific list are the locus name and a vector for the unique alleles for the locus.

locus.label Vector of labels for loci, of length K (see definition of input values).

haplotype Matrix of unique haplotypes. Each row represents a unique haplotype, and the number of columns is the number of loci.

hap.prob Vector of mle’s of haplotype probabilities. The ith element of hap.prob corresponds to the ith row of haplotype.

hap.prob.noLD Similar to hap.prob, but assuming no linkage disequilibrium.

lnlike Value of lnlike at last EM iteration (maximum lnlike if converged).

lr Likelihood ratio statistic to test no linkage disequilibrium among all loci.

indx.subj Vector for index of subjects, after expanding to all possible pairs of haplotypes for each person. If indx=i, then i is the ith row of input matrix data. If the ith subject has n possible pairs of haplotypes that correspond to their marker phenotype, then i is repeated n times.

nreps Vector for the count of haplotype pairs that map to each subject’s marker genotypes.

hap1code Vector of codes for each subject’s first haplotype. The values in hap1code are the row numbers of the unique haplotypes in the returned matrix haplotype.

hap2code Similar to hap1code, but for each subject’s second haplotype.

post Vector of posterior probabilities of pairs of haplotypes for a person, given thier marker phenotypes.

htrtable A table which can be used in haplotype trend regression.
Note

Adapted from GENECOUNTING.

Author(s)

Jing Hua Zhao

References


See Also

genecounting, LDkl

Examples

## Not run:
data(hla)
gc.em(hla[,3:8],locus.label=c("DQR","DQA","DQB"),control=gc.control(assignment="t"))

## End(Not run)

gc.lambda

Estimation of the genomic control inflation statistic (lambda)

Description

Estimation of the genomic control inflation statistic (lambda)

Usage

gc.lambda(p)

Arguments

p

A vector of p values.

Value

Estimate of inflation factor.
**Examples**

```r
set.seed(12345)
p <- runif(100)
lambda <- gc.lambda(p)
```

**Description**

The Bayesian genomic control statistics with the following parameters,

**Usage**

```r
gcontrol(
data,  
zeta = 1000,  
kappa = 4,  
tau2 = 1,  
epsilon = 0.01,  
ngib = 500,  
burn = 50,  
idum = 2348
)
```

**Arguments**

- `data` the data matrix.
- `zeta` program constant with default value 1000.
- `kappa` multiplier in prior for mean with default value 4.
- `tau2` multiplier in prior for variance with default value 1.
- `epsilon` prior probability of marker association with default value 0.01.
- `ngib` number of Gibbs steps, with default value 500.
- `burn` number of burn-ins with default value 50.
- `idum` seed for pseudorandom number sequence.

**Details**

- `n` number of loci under consideration
- `lambdahat` median(of the n trend statistics)/0.46
- `kappa` multiplier in prior above, set at 1.6 * sqrt(log(n))
- `tau2` multiplier in prior above
- `epsilon` prior probability a marker is associated, set at 10/n
- `ngib` number of cycles for the Gibbs sampler after burn in
- `burn` number of cycles for the Gibbs sampler to burn in
Armitage’s trend test along with the posterior probability that each marker is associated with the disorder is given. The latter is not a p-value but any value greater than 0.5 (pout) suggests association.

Value

The returned value is a list containing:

- **deltot** the probability of being an outlier
- **x2** the $\chi^2$ statistic
- **A** the A vector

Note

Adapted from gcontrol by Bobby Jones and Kathryn Roeder, use -Dexecutable for standalone program, function getnum in the original code needs %*s to skip id string

Author(s)

Bobby Jones, Jing Hua Zhao

Source

https://www.cmu.edu/dietrich/statistics-datascience/index.html

References


Examples

```r
## Not run:
test<-c(1,2,3,4,5,6, 1,2,1,23,1,2, 100,1,2,12,1,1,
       1,2,3,4,5,61, 1,2,11,23,1,2, 10,11,2,12,1,11)
test<-matrix(test,nrow=6,byrow=T)
gcontrol(test)
## End(Not run)
```

Description

The function obtains 1-df $\chi^2$ statistics (observed) according to a vector of p values, and the inflation factor (lambda) according to medians of the observed and expected statistics. The latter is based on the empirical distribution function (EDF) of 1-df $\chi^2$ statistics.
Usage

gcontrol2(p, col = palette()[4], lcol = palette()[2], ...)  

Arguments

p         a vector of observed p values.
col       colour for points in the Q-Q plot.
lcol      colour for the diagonal line in the Q-Q plot.
...       other options for plot.

Details

It would be appropriate for genetic association analysis as of 1-df Armitage trend test for case-control data; for 1-df additive model with continuous outcome one has to consider the compatibility with p values based on z-/t- statistics.

Value

A list containing:

x  the expected $\chi^2$ statistics
y  the observed $\chi^2$ statistics
lambda  the inflation factor

Author(s)

Jing Hua Zhao

References


Examples

## Not run:
x2 <- rchisq(100,1,.1)
p <- pchisq(x2,1,lower.tail=FALSE)
r <- gcontrol2(p)
print(r$lambda)

## End(Not run)
gcp

Permutation tests using GENECOUNTING

Description

This function is a R port of the GENECOUNTING/PERMUTE program which generates EHPLUS-type statistics including z-tests for individual haplotypes.

Usage

```r
gcp(
  y,
  cc,
  g,
  handle.miss = 1,
  miss.val = 0,
  n.sim = 0,
  locus.label = NULL,
  quietly = FALSE
)
```

Arguments

- `y`: A column of 0/1 indicating cases and controls.
- `cc`: analysis indicator, 0 = marker-marker, 1 = case-control.
- `g`: the multilocus genotype data.
- `handle.miss`: a flag with value 1 indicating missing data are allowed.
- `miss.val`: missing value.
- `n.sim`: the number of permutations.
- `locus.label`: label of each locus.
- `quietly`: a flag if TRUE will suppress the screen output.

Value

The returned value is a list containing (p.sim and ph when n.sim > 0):

- `x2obs`: the observed chi-squared statistic
- `pobs`: the associated p value
- `zobs`: the observed z value for individual haplotypes
- `p.sim`: simulated p value for the global chi-squared statistic
- `ph`: simulated p values for individual haplotypes

Note

Built on gcp.c.
Author(s)

Jing Hua Zhao

References


See Also

genecounting

Examples

## Not run:
data(fsnps)
y<-fsnps$y
cc<-1
g<-fsnps[,3:10]
gcp(y,cc,g,miss.val="Z",n.sim=5)
hap.score(y,g,method="hap",miss.val="Z")
## End(Not run)

---

**genecounting**

Gene counting for haplotype analysis

Description

Gene counting for haplotype analysis with missing data

Usage

geneCounting(data, weight = NULL, loci = NULL, control = gc.control())
Arguments

data genotype table.
weight a column of frequency weights.
loci an array containing number of alleles at each locus.
control is a function with the following arguments:

1. xdata. a flag indicating if the data involves X chromosome, if so, the first column of data indicates sex of each subject: 1=male, 2=female. The marker data are no different from the autosomal version for females, but for males, two copies of the single allele present at a given locus.
2. convll. set convergence criteria according to log-likelihood, if its value set to 1
3. handle.miss. to handle missing data, if its value set to 1
4. eps. the actual convergence criteria, with default value 1e-5
5. tol. tolerance for genotype probabilities with default value 1e-8
6. maxit. maximum number of iterations, with default value 50
7. pl. criteria for trimming haplotypes according to posterior probabilities
8. assignment. filename containing haplotype assignment
9. verbose. If TRUE, yields print out from the C routine

Value

The returned value is a list containing:

- h haplotype frequency estimates under linkage disequilibrium (LD)
- h0 haplotype frequency estimates under linkage equilibrium (no LD)
- prob genotype probability estimates
- l0 log-likelihood under linkage equilibrium
- l1 log-likelihood under linkage disequilibrium
- hapid unique haplotype identifier (defunct, see gc.em)
- npusr number of parameters according user-given alleles
- npdat number of parameters according to observed
- htrtable design matrix for haplotype trend regression (defunct, see gc.em)
- iter number of iterations used in gene counting
- converge a flag indicating convergence status of gene counting
- di0 haplotype diversity under no LD, defined as $1 - \sum (h_0^2)$
- di1 haplotype diversity under LD, defined as $1 - \sum (h^2)$
- resid residuals in terms of frequency weights = o - e

Note

adapted from GENECOUNTING.
Author(s)

Jing Hua Zhao

References


See Also

gc.em, LDkl

Examples

```r
## Not run:
require(gap.datasets)
# HLA data
data(hla)
hl.gc <- genecounting(hla[,3:8])
summary(hla.gc)
hl.gc$l0
hl.gc$l1

# ALDH2 data
data(aldh2)
control <- gc.control(handle.miss=1,assignment="ALDH2.out")
aldh2.gc <- genecounting(aldh2[,3:6],control=control)
summary(aldh2.gc)
aldh2.gc$l0
aldh2.gc$l1

# Chromosome X data
# assuming allelic data have been extracted in columns 3-13
# and column 3 is sex
filespec <- system.file("tests/genecounting/mao.dat")
mao2 <- read.table(filespec)
dat <- mao2[,3:13]
loci <- c(12,9,6,5,3)
contr <- gc.control(xdata=TRUE,handle.miss=1)
mao.gc <- genecounting(dat,loci=loci,control=contr)
mao.gc$npusr
mao.gc$npdat

## End(Not run)
```
**get_b_se**

*Get b and se from AF, n, and z*

**Description**

The function obtains effect size and its standard error.

**Usage**

```r
get_b_se(f, n, z)
```

**Arguments**

- `f`: Allele frequency.
- `n`: Sample size.
- `z`: z-statistics.

**Value**

`b` and `se`.

**Examples**

```r
## Not run:
library(dplyr)
# eQTLGen
cis_pQTL <- merge(read.delim('eQTLGen.lz') %>%
  filter(GeneSymbol=='LTBR'),read.delim('eQTLGen.AF'),by="SNP") %>%
  mutate(data.frame(get_b_se(AlleleB_all,NrSamples,Zscore)))
head(cis_pQTL,1)
SNP  Pvalue SNPChr SNPPos AssessedAllele OtherAllele Zscore
rs1003563 2.308e-06 12 6424577 A G 4.7245
Gene GeneSymbol GeneChr GenePos NrCohorts NrSamples FDR
ENSG0000011321 LTBR 12 6492472 34 23991 0.006278872
BonferroniP hg19_chr hg19_pos AlleleA AlleleB allA_total allAB_total
1 12 6424577 A G 2574 8483
allB_total AlleleB_all b se
7859 0.6396966 0.04490488 0.009504684
## End(Not run)
```
get_pve_se

Get pve and its standard error from n, z

Description

This function obtains proportion of explained variance of a continuous outcome.

Usage

get_pve_se(n, z, correction = TRUE)

Arguments

n  Sample size.
z  z-statistic, i.e., b/se when they are available instead.
correction  if TRUE an correction based on t-statistic is applied.

Value

pve and its se.

get_sdy

Get sd(y) from AF, n, b, se

Description

This function obtains standard error of a continuous outcome.

Usage

get_sdy(f, n, b, se, method = "mean", ...)

Arguments

f  Allele frequency.
n  Sample size.
b  effect size.
se  standard error.
method  method of averaging: "mean" or "median".
...  argument(s) passed to method

Value

sd(y).
Examples

```r
## Not run:
set.seed(1)
X1 <- matrix(rbinom(1200,1,0.4),ncol=2)
X2 <- matrix(rbinom(1000,1,0.6),ncol=2)
colnames(X1) <- colnames(X2) <- c("f1","f2")
Y1 <- rnorm(600,apply(X1,1,sum),2)
Y2 <- rnorm(500,2*apply(X2,1,sum),5)
summary(lm1 <- lm(Y1~f1+f2,data=as.data.frame(X1)))
summary(lm2 <- lm(Y2~f1+f2,data=as.data.frame(X2)))
b1 <- coef(lm1)
b2 <- coef(lm2)
v1 <- vcov(lm1)
v2 <- vcov(lm2)
require(coloc)
## Bayesian approach, esp. when only p values are available
abf <- coloc.abf(list(beta=b1, varbeta=diag(v1), N=nrow(X1), sdY=sd(Y1), type="quant"),
                 list(beta=b2, varbeta=diag(v2), N=nrow(X2), sdY=sd(Y2), type="quant"))
abf
```

```
# sdY
cat("sd(Y)="sd(Y1),"==> Estimates: ",sqrt(diag(var(X1)*b1[-1]^2+var(X1)*v1[-1,-1]*nrow(X1)))),"\n")
for(k in 1:2)
  {
    k1 <- k + 1
    cat("Based on b",k," sd(Y1) = ",sqrt(var(X1[k,k])*b1[k1]^2+nrow(X1)*v1[k1,k1])),"\n",sep="")
  }
cat("sd(Y)="sd(Y2),"==> Estimates: ",sqrt(diag(var(X2)*b2[-1]^2+var(X2)*v2[-1,-1]*nrow(X2)))),"\n")
for(k in 1:2)
  {
    k1 <- k + 1
    cat("Based on b",k," sd(Y2) = ",sqrt(var(X2[k,k])*b2[k1]^2+nrow(X2)*v2[k1,k1])),"\n",sep="")
  }
get_sdy(0.6396966,23991,0.04490488,0.009504684)
## End(Not run)
```

---

**Kinship coefficient and genetic index of familiality**

### Description

The genetic index of famillality is defined as the mean kinship between all pairs of individuals in a set multiplied by 100,000. Formally, it is defined as

\[
100,000 \times \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} k_{ij}
\]

where \( n \) is the number of individuals in the set and \( k_{ij} \) is the kinship coefficient between individuals \( i \) and \( j \).
Usage

gif(data, gifset)

Arguments

data the trio data of a pedigree.
gifset a subgroup of pedigree members.

Details

The scaling is purely for convenience of presentation.

Value

The returned value is a list containing:

*gifval* the genetic index of familiarity.

Note

Adapted from gif.c, testable with -Dexecutable as standalone program, which can be use for any pair of individuals

Author(s)

Alun Thomas, Jing Hua Zhao

References


See Also

pfc

Examples

```r
## Not run:
test<-c(
  5, 0, 0,
  1, 0, 0,
  9, 5, 1,
  6, 0, 0,
  10, 9, 6,
  15, 9, 6,
  21, 10, 15,
  3, 0, 0,
  18, 3, 15,
  23, 21, 18,
  2, 0, 0,
)```
The function initially intends to rework on GSMR outputs, but it would be appropriate for general use.

Usage

```r
gsmr(data, X, Y, alpha = 0.05, other_plots = FALSE)
```

Arguments

data: Data to be used.

X: Exposure.

Y: Outcome.

alpha: Type I error rate for confidence intervals.

other_plots: To add funnel and forest plots.

Value

The result and plots.
**Examples**

```r
library(cowplot)
library(ggplot2)
library(gap)
r <- gsmr(mr, "LIF.R", c("CAD","FEV1"))
```

---

**h2.jags**  
*Heritability estimation based on genomic relationship matrix using JAGS*

---

**Description**

Heritability estimation based on genomic relationship matrix using JAGS.

**Usage**

```r
h2.jags(
  y,
  x,
  G,
  eps = 1e-04,
  sigma.p = 0,
  sigma.r = 1,
  parms = c("b", "p", "r", "h2"),
  ...
)
```

**Arguments**

- `y`  
  outcome vector.
- `x`  
  covariate matrix.
- `G`  
  genomic relationship matrix.
- `eps`  
  a positive diagonal perturbation to G.
- `sigma.p`  
  initial parameter values.
- `sigma.r`  
  initial parameter values.
- `parms`  
  monitored parameters.
- `...`  
  parameters passed to jags, e.g., n.chains, n.burnin, n.iter.

**Details**

This function performs Bayesian heritability estimation using genomic relationship matrix.

**Value**

The returned value is a fitted model from jags().
Author(s)

Jing Hua Zhao keywords htest

References


Examples

```r
## Not run:
require(gap.datasets)
set.seed(1234567)
meyer <- within(meyer,{
  y[is.na(y)] <- rnorm(length(y[is.na(y)]),mean(y,na.rm=TRUE),sd(y,na.rm=TRUE))
  g1 <- ifelse(generation==1,1,0)
  g2 <- ifelse(generation==2,1,0)
  id <- animal
  animal <- ifelse(!is.na(animal),animal,0)
  dam <- ifelse(!is.na(dam),dam,0)
  sire <- ifelse(!is.na(sire),sire,0)
})
G <- kin.morgan(meyer)$kin.matrix*2
library(regress)
r <- regress(y~-1+g1+g2,~G,data=meyer)
r
with(r,h2G(sigma,sigma.cov))
eps <- 0.001
y <- with(meyer,y)
x <- with(meyer,cbind(g1,g2))
ex <- h2.jags(y,x,G,sigma.p=0.03,sigma.r=0.014)
print(ex)
## End(Not run)
```

---

**h2G**

*Heritability and its variance*

**Description**

Heritability and its variance

**Usage**

```r
h2G(V, VCOV, verbose = TRUE)
```
**Arguments**

- V: Variance estimates.
- VCOV: Variance-covariance matrix.
- verbose: Detailed output.

**Value**

A list of phenotypic variance/heritability estimates and their variances.

---

**h2GE**

*Heritability and its variance when there is an environment component*

**Description**

Heritability and its variance when there is an environment component

**Usage**

```r
h2GE(V, VCOV, verbose = TRUE)
```

**Arguments**

- V: Variance estimates.
- VCOV: Variance-covariance matrix.
- verbose: Detailed output.

**Value**

A list of phenotypic variance/heritability/GxE interaction estimates and their variances.

---

**h2l**

*Heritability under the liability threshold model*

**Description**

Heritability under the liability threshold model

**Usage**

```r
h2l(K = 0.05, P = 0.5, h2, se, verbose = TRUE)
```
Arguments

- **K**
  - Disease prevalence.
- **P**
  - Phenotypeic variance.
- **h2**
  - Heritability estimate.
- **se**
  - Standard error.
- **verbose**
  - Detailed output.

Value

A list of the input heritability estimate/standard error and their counterpart under liability threshold model, the normal deviate.

---

**h2_mzdz**

*Heritability estimation according to twin correlations*

Description

Heritability and variance estimation according to twin pair correlations.

Usage

```r
h2_mzdz(
  mzDat = NULL,
  dzDat = NULL,
  rmz = NULL,
  rdz = NULL,
  nmz = NULL,
  ndz = NULL,
  selV = NULL
)
```

Arguments

- **mzDat**
  - a data frame for monzygotic twins (MZ).
- **dzDat**
  - a data frame for dizygotic twins (DZ).
- **rmz**
  - correlation for MZ twins.
- **rdz**
  - correlation for DZ twins.
- **nmz**
  - sample size for MZ twins.
- **ndz**
  - sample size for DZ twins.
- **selV**
  - names of variables for twin and cotwin.

Details

The example section shows how to obtain bootstrap 95% CI.
Value

The returned value is a matrix containing heritability and their variance estimations for "h2","c2","e2","vh","vc","ve".

Author(s)

Jing Hua Zhao

References

Keeping ES. Introduction to Statistical Inference, Dover Publications, Inc. 1995

Examples

```r
# Not run:

ACE_CI <- function(mzData,dzData,n.sim=5,selV=NULL,verbose=TRUE)
{
    ACEr_twinData <- h2(mzDat=mzData,dzDat=dzData,selV=selV)
    print(ACEr_twinData)

    nmz <- dim(mzData)[1]
    ndz <- dim(dzData)[1]
    a <- ar <- vector()
    set.seed(12345)
    for(i in 1:n.sim)
    {
        cat("\nRunning # ",i,"/", n.sim,"\r",sep="")
        sampled_mz <- sample(1:nmz, replace=TRUE)
        sampled_dz <- sample(1:ndz, replace=TRUE)
        mzDat <- mzData[sampled_mz,]
        dzDat <- dzData[sampled_dz,]
        ACEr_i <- h2(mzDat=mzDat,dzDat=dzDat,selV=selV)
        if(verbose) print(ACEr_i)
        ar <- rbind(ar,ACEr_i)
    }

    cat("\n\nheritability according to correlations\n\n")
    ar <- as.data.frame(ar)
    m <- mean(ar,na.rm=TRUE)
    s <- sd(ar,na.rm=TRUE)
    allr <- data.frame(mean=m,sd=s,lcl=m-1.96*s,ucl=m+1.96*s)
    print(allr)
}

selVars <- c('bmi1','bmi2')

library(mvtnorm)
n.sim <- 500

cat("\nThe first study\n")
mzm <- as.data.frame(rmvnorm(195, c(22.75,22.75),
matrix(2.66^2*c(1, 0.67, 0.67, 1), 2)))
dzm <- as.data.frame(rmvnorm(130, c(23.44,23.44),
matrix(2.75^2*c(1, 0.32, 0.32, 1), 2)))
```

hap <- as.data.frame(rmvnorm(384, c(21.44, 21.44),
    matrix(3.08^2*c(1, 0.72, 0.72, 1), 2)))
dzw <- as.data.frame(rmvnorm(243, c(21.72, 21.72),
    matrix(3.12^2*c(1, 0.33, 0.33, 1), 2)))
names(mzm) <- names(dzm) <- names(mzw) <- names(dzw) <- c("bmi1", "bmi2")
ACE_CI(mzm,dzm,n.sim,selV=selVars,verbose=FALSE)
ACE_CI(mzw,dzw,n.sim,selV=selVars,verbose=FALSE)

## End(Not run)

**Haplotype reconstruction**

**Description**

Haplotype reconstruction using sorting and trimming algorithms.

**Usage**

```r
hap(
  id,
  data,
  nloci,
  loci = rep(2, nloci),
  names = paste("loci", 1:nloci, sep = ","),
  control = hap.control()
)
```

**Arguments**

- `id` a column of subject id.
- `data` genotype table.
- `nloci` number of loci.
- `loci` number of alleles at all loci.
- `names` locus names.
- `control` is a function with the following arguments,
  1. `mb` Maximum dynamic storage to be allocated, in Mb
  2. `pr` Prior (ie population) probability threshold
  3. `po` Posterior probability threshold
  4. `to` Log-likelihood convergence tolerance
  5. `th` Posterior probability threshold for output
  6. `maxit` Maximum EM iteration
  7. `n` Force numeric allele coding (1/2) on output (off)
  8. `ss` Tab-delimited spreadsheet file output (off)
9. rs Random starting points for each EM iteration (off)
10. rp Restart from random prior probabilities
11. ro Loci added in random order (off)
12. rv Loci added in reverse order (off)
13. sd Set seed for random number generator (use date+time)
14. mm Repeat final maximization multiple times
15. mi Create multiple imputed datasets. If set >0
16. mc Number of MCMC steps between samples
17. ds Starting value of Dirichlet prior parameter
18. de Finishing value of Dirichlet prior parameter
19. q Quiet operation (off)
20. hapfile a file for haplotype frequencies
21. assignfile a file for haplotype assignment

Details

The package can handle much larger number of multiallelic loci. For large sample size with relatively small number of multiallelic loci, genecounting should be used.

Value

The returned value is a list containing:

- **l1**  log-likelihood assuming linkage disequilibrium
- **converge**  convergence status, 0=failed, 1=succeeded
- **niter**  number of iterations

Note

adapted from hap.

References


See Also

genecounting
Examples

```r
## Not run:
require(gap.datasets)
# 4 SNP example, to generate hap.out and assign.out alone
data(fsnps)
hap(id=fsnps[,1],data=fsnps[,3:10],nloci=4)
dir()

# to generate results of imputations
control <- hap.control(ss=1,mi=5,hapfile="h",assignfile="a")
hap(id=fsnps[,1],data=fsnps[,3:10],nloci=4,control=control)
dir()

## End(Not run)
```

---

**hap.em**

*Gene counting for haplotype analysis*

**Description**

Gene counting for haplotype analysis with missing data, adapted for hap.score.

**Usage**

```r
hap.em(id, data, locus.label = NA, converge.eps = 1e-06, maxiter = 500, miss.val = 0)
```

**Arguments**

- **id**
  - a vector of individual IDs.
- **data**
  - Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(data) = 2*K. Rows represent alleles for each subject.
- **locus.label**
  - Vector of labels for loci, of length K (see definition of data matrix).
- **converge.eps**
  - Convergence criterion, based on absolute change in log likelihood (lnlike).
- **maxiter**
  - Maximum number of iterations of EM.
- **miss.val**
  - missing value.
Value

List with components:

**converge**  Indicator of convergence of the EM algorithm (1=converged, 0 = failed).

**niter**  Number of iterations completed in the EM algorithm.

**locus.info**  A list with a component for each locus. Each component is also a list, and the items of a locus- specific list are the locus name and a vector for the unique alleles for the locus.

**locus.label**  Vector of labels for loci, of length K (see definition of input values).

**haplotype**  Matrix of unique haplotypes. Each row represents a unique haplotype, and the number of columns is the number of loci.

**hap.prob**  Vector of mle’s of haplotype probabilities. The ith element of hap.prob corresponds to the ith row of haplotype.

**lnlike**  Value of lnlike at last EM iteration (maximum lnlike if converged).

**indx.subj**  Vector for index of subjects, after expanding to all possible pairs of haplotypes for each person. If indx=i, then i is the ith row of input matrix data. If the ith subject has n possible pairs of haplotypes that correspond to their marker phenotype, then i is repeated n times.

**nreps**  Vector for the count of haplotype pairs that map to each subject’s marker genotypes.

**hap1code**  Vector of codes for each subject’s first haplotype. The values in hap1code are the row numbers of the unique haplotypes in the returned matrix haplotype.

**hap2code**  Similar to hap1code, but for each subject’s second haplotype.

**post**  Vector of posterior probabilities of pairs of haplotypes for a person, given thier marker phenotypes.

Note

Adapted from HAP.

Author(s)

Jing Hua Zhao

See Also

hap, LDk1

Examples

```r
## Not run:
data(hla)
hap.em(id=1:length(hla[,1]),data=hla[,3:8],locus.label=c("DQR","DQA","DQB"))
## End(Not run)
```
hap.score

Score statistics for association of traits with haplotypes

Description

Compute score statistics to evaluate the association of a trait with haplotypes, when linkage phase is unknown and diploid marker phenotypes are observed among unrelated subjects. For now, only autosomal loci are considered. This package haplo.score which this function is based is greatly acknowledged.

Usage

```r
hap.score(
  y, 
  geno, 
  trait.type = "gaussian", 
  offset = NA, 
  x.adj = NA, 
  skip.haplo = 0.005, 
  locus.label = NA, 
  miss.val = 0, 
  n.sim = 0, 
  method = "gc", 
  id = NA, 
  handle.miss = 0, 
  mloci = NA, 
  sexid = NA
)
```

Arguments

- **y** Vector of trait values. For trait.type = "binomial", y must have values of 1 for event, 0 for no event.
- **geno** Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.
- **trait.type** Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal".
- **offset** Vector of offset when trait.type = "poisson".
- **x.adj** Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should not be included, as it will be added in this function.
- **skip.haplo** Skip score statistics for haplotypes with frequencies < skip.haplo.
- **locus.label** Vector of labels for loci, of length K (see definition of geno matrix).
- **miss.val** Vector of codes for missing values of alleles.
`hap.score`  

- `n.sim` Number of simulations for empirical p-values. If `n.sim`=0, no empirical p-values are computed.
- `method` method of haplotype frequency estimation, "gc" or "hap".
- `id` an added option which contains the individual IDs.
- `handle.miss` flag to handle missing genotype data, 0=no, 1=yes.
- `mloci` maximum number of loci/sites with missing data to be allowed in the analysis.
- `sexid` flag to indicator sex for data from X chromosome, i=male, 2=female.

**Details**

This is a version which substitutes `haplo.em`.

**Value**

List with the following components:

- `score.global` Global statistic to test association of trait with haplotypes that have frequencies $\geq$ `skip.haplo`.
- `df` Degrees of freedom for `score.global`.
- `score.global.p` P-value of `score.global` based on chi-square distribution, with degrees of freedom equal to `df`.
- `score.global.p.sim` P-value of `score.global` based on simulations (set equal to `NA` when `n.sim`=0).
- `score.haplo` Vector of score statistics for individual haplotypes that have frequencies $\geq$ `skip.haplo`.
- `score.haplo.p` Vector of p-values for `score.haplo`, based on a chi-square distribution with 1 df.
- `score.haplo.p.sim` Vector of p-values for `score.haplo`, based on simulations (set equal to `NA` when `n.sim`=0).
- `score.max.p.sim` P-value of maximum `score.haplo`, based on simulations (set equal to `NA` when `n.sim`=0).
- `haplotype` Matrix of haplotypes analyzed. The ith row of `haplotype` corresponds to the ith item of `score.haplo`, `score.haplo.p`, and `score.haplo.p.sim`.
- `hap.prob` Vector of haplotype probabilities, corresponding to the haplotypes in the matrix `haplotype`.
- `locus.label` Vector of labels for loci, of length K (same as input argument).
- `n.sim` Number of simulations.
- `n.val.global` Number of valid simulated global statistics.
- `n.val.haplo` Number of valid simulated score statistics (score.haplo) for individual haplotypes.

**References**

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA (2002) Score tests for association of traits with haplotypes when linkage phase is ambiguous. Amer J Hum Genet 70:425-34
Examples

```r
## Not run:
data(hla)
y <- hla[,2]
genoc <- hla[,3:8]
# complete data
hap.score(y, geno, locus.label = c("DRB", "DQA", "DQB"))
# incomplete genotype data
hap.score(y, geno, locus.label = c("DRB", "DQA", "DQB"), handle.miss = 1, mloci = 1)
unlink("assign.dat")

### note the differences in p values in the following runs

data(aldh2)
# to subset the data since hap doesn't handle one allele missing
deleted <- c(40, 239, 256)
aldh2[deleted,]
aldh2 <- aldh2[-deleted,]
y <- aldh2[,2]
genoc <- aldh2[,3:18]
# only one missing locus
hap.score(y, geno, handle.miss = 1, mloci = 1, method = "hap")
# up to seven missing loci and with 10,000 permutations
hap.score(y, geno, handle.miss = 1, mloci = 7, method = "hap", n.sim = 10000)

# hap.score takes considerably longer time and does not handle missing data
hap.score(y, geno, n.sim = 10000)

## End(Not run)
```

---

### htr

**Haplotype trend regression**

#### Description

Haplotype trend regression (with permutation)

#### Usage

```r
htr(y, x, n.sim = 0)
```

#### Arguments

- **y**: a vector of phenotype.
- **x**: a haplotype table.
- **n.sim**: the number of permutations.
The returned value is a list containing:

- \( f \) the F statistic for overall association
- \( p \) the p value for overall association
- \( f_v \) the F statistics for individual haplotypes
- \( p_i \) the p values for individual haplotypes

Note
adapted from emgi.cpp, a pseudorandom number seed will be added on.

Author(s)
Dimitri Zaykin, Jing Hua Zhao

References


See Also
hap.score

Examples
```r
## Not run:
# 26-10-03
# this is now part of demo
test2<-read.table("test2.dat")
y<-test2[,1]
x<-test2[,-1]
y<-as.matrix(y)
x<-as.matrix(x)
htr.test2<-htr(y,x)
htr.test2
htr.test2<-htr(y,x,n.sim=10)
htr.test2

# 13-11-2003
require(gap.datasets)
data(apoeapoc)
apeapoc.gc<-gc.em(apeapoc[,5:8])
y<-apeapoc$y
for(i in 1:length(y)) if(y[i]==2) y[i]<-1
htr(y,apeapoc.gc$htrtable)
```
HaploEM <- haplo.em(Geno, locus.label=SNPnames)
HapMat <- HapDesign(HaploEM)
m1 <- lm(Trait~HapMat)
m2 <- lm(Trait~1)
anova(m2,m1)

## End(Not run)

---

**hwe**

*Hardy-Weinberg equilibrium test for a multiallelic marker*

**Description**

Hardy-Weinberg equilibrium test.

**Usage**

```r
hwe(data, data.type = "allele", yates.correct = FALSE, miss.val = 0)
```

**Arguments**

- `data`: A rectangular data containing the genotype, or an array of genotype counts.
- `data.type`: An option taking values "allele", "genotype", "count" if data is alleles, genotype or genotype count.
- `yates.correct`: A flag indicating if Yates' correction is used for Pearson $\chi^2$ statistic.
- `miss.val`: A list of missing values.

**Details**

This function obtains Hardy-Weinberg equilibrium test statistics. It can handle data coded as allele numbers (default), genotype identifiers (by setting data.type="genotype") and counts corresponding to individual genotypes (by setting data.type="count") which requires that genotype counts for all $n(n+1)$ possible genotypes, with $n$ being the number of alleles.

For highly polymorphic markers when asymptotic results do not hold, please resort to hwe.hardy.

**Value**

The returned value is a list containing:

- `allele.freq`: Frequencies of alleles
- `x2`: Pearson $\chi^2$
- `p.x2`: p value for $\chi^2$
hwe.cc

A likelihood ratio test of population Hardy-Weinberg equilibrium for case-control studies

Description

A likelihood ratio test of population Hardy-Weinberg equilibrium for case-control studies

Usage

hwe.cc(model, case, ctrl, k0, initial1, initial2)
Arguments

- **model**: model specification, dominant, recessive.
- **case**: a vector of genotype counts in cases.
- **ctrl**: a vector of genotype counts in controls.
- **k0**: prevalence of disease in the population.
- **initial1**: initial values for beta, gamma, and q.
- **initial2**: initial values for logit(p) and log(gamma).

Details

This is a collection of utility functions. The null hypothesis declares that the proportions of genotypes are according to Hardy-Weinberg law, while under the alternative hypothesis, the expected genotype counts are according to the probabilities that particular genotypes are obtained conditional on the prevalence of disease in the population. In so doing, Hardy-Weinberg equilibrium is considered using both case and control samples but pending on the disease model such that 2-parameter multiplicative model is built on baseline genotype $\alpha$, $\alpha\beta$ and $\alpha\gamma$.

Value

The returned value is a list with the following components.

- **Cox**: statistics under a general model
- **t2par**: under the null hypothesis
- **t3par**: under the alternative hypothesis
- **lrt.stat**: the log-likelihood ratio statistic
- **pval**: the corresponding p value

Author(s)

Chang Yu, Li Wang, Jing Hua Zhao

References


See Also

hwe

Examples

```r
## Not run:

### Saba Sile, email of Jan 26, 2007, data always in order of GG AG AA, p=Pr(G),
### q=1-p=Pr(A)
### case=c(155,27,4)
```
ctrl=c(408,55,15)
k0=.2
initial1=c(1.0,0.94,0.0904)
initial2=c(logit(1-0.0904),log(0.94))
hwe.cc("recessive",case,ctrl,k0, initial1, initial2)

### John Phillips III, TGFb1 data codon 10: TT CT CC, CC is abnormal and increasing
### TGFb1 activity
case=c(29,78,13)
ctrl=c(17,28,6)
k0 <- 1e-5
initial1 <- c(2.45,2.45,0.34)
initial2 <- c(logit(1-0.34),log(2.45))
hwe.cc("dominant",case,ctrl,k0,initial1,initial2)

## End(Not run)

---

**hwe.hardy**

*Hardy-Weinberg equilibrium test using MCMC*

**Description**

Hardy-Weinberg equilibrium test by MCMC

**Usage**

hwe.hardy(a, alleles = 3, seed = 3000, sample = c(1000, 1000, 5000))

**Arguments**

- **a**: an array containing the genotype counts, as integer.
- **alleles**: number of allele at the locus, greater than or equal to 3, as integer.
- **seed**: pseudo-random number seed, as integer.
- **sample**: optional, parameters for MCMC containing number of chunks, size of a chunk and burn-in steps, as integer.

**Value**

The returned value is a list containing:

- **method**: Hardy-Weinberg equilibrium test using MCMC
- **data.name**: name of used data if x is given
- **p.value**: Monte Carlo p value
- **p.value.se**: standard error of Monte Carlo p value
- **switches**: percentage of switches (partial, full and altogether)
Note

Codes are commented for taking x a genotype object, as genotype to prepare a and alleles on the fly.
Adapted from HARDY, testable with -Dexecutable as standalone program.

keywords htest

Author(s)

Sun-Wei Guo, Jing Hua Zhao, Gregor Gorjanc

Source

https://sites.stat.washington.edu/thompson/Genepi/pangaea.shtml

References


See Also

hwe, HWE.test, genotype

Examples

```r
## Not run:
# example 2 from hwe.doc:
a<-c(3,
   4, 2,
   2, 2, 2,
   3, 3, 2, 1,
   0, 1, 0, 0,
   0, 0, 0, 0, 1,
   0, 0, 1, 0, 0, 0,
   0, 0, 2, 1, 0, 0, 0)
ex2 <- hwe.hardy(a=a, alleles=8)

# example using HLA
data(hla)
x <- hla[,3:4]
y <- pgc(x, handle.miss=0, with.id=1)
n.alleles <- max(x, na.rm=TRUE)
z <- vector("numeric", n.alleles*(n.alleles+1)/2)
z[y$idsave] <- y$wt
hwe.hardy(a=z, alleles=n.alleles)

# with use of class 'genotype'
# this is to be fixed
library(genetics)
hlagen <- genotype(a1=x$DQR.a1, a2=x$DQR.a2,
```
alleles=sort(unique(c(x$DQR.a1, x$DQR.a2))))

hwe.hardy(hlagen)

# comparison with hwe
hwe(z, data.type="count")

# to create input file for HARDY
print.tri <- function(xx, n) {
  cat(n, "\n")
  for(i in 1:n) {
    for(j in 1:i) {
      cat(xx[i, j], " ")
    }
    cat("\n")
  }
  cat("100 170 1000\n")
}

xx <- matrix(0, n.alleles, n.alleles)
xxx <- lower.tri(xx, diag=TRUE)
xx[xxx] <- z
sink("z.dat")
print.tri(xx, n.alleles)
sink()
# now call as: hwe z.dat z.out

## End(Not run)
Arguments

k
   number of alleles.

n
   a vector of k(k+1)/2 genotype counts.

delta
   initial parameter values.

lambda
   initial parameter values.

lambdamu
   initial parameter values.

lambdasd
   initial parameter values.

parms
   monitored parameters.

... parameters passed to jags, e.g., n.chains, n.burnin, n.iter.

Details

This function performs Bayesian Hardy-Weinberg equilibrium test, which mirrors hwe.hardy, another implementation for highly polymorphic markers when asymptotic results do not hold.

Value

The returned value is a fitted model from jags().

Author(s)

Jing Hua Zhao, Jon Wakefield

References


See Also

hwe.hardy

Examples

```r
## Not run:
ex1 <- hwe.jags(4,c(5,6,1,7,11,2,8,19,26,15))
print(ex1)
ex2 <- hwe.jags(2,c(49,45,6))
print(ex2)
ex3 <- hwe.jags(4,c(0,3,1,5,18,1,3,7,5,2),lambda=0.5,lambdamu=-2.95,lambdasd=1.07)
print(ex3)
ex4 <- hwe.jags(9,c(1236,120,3,18,0,0,982,55,7,249,32,1,0,12,0,2582,132,20,1162,29,1312,6,0,0,4,0,4,0,2,0,0,0,0,0,0,0,0,0,0,0,0,115,5,2,53,1,149,0,0,4),
   delta=c(1,1,1,1,1,1,1,1,1),lambdamu=-4.65,lambdasd=0.21)
print(ex4)
ex5 <- hwe.jags(8,n=c(3,4,2),
```

hwe.jags

\[
2, 2, 2,
3, 3, 2, 1,
0, 1, 0, 0, 0,
0, 0, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 2, 1, 0, 0, 0)
\]

print(ex5)

# Data and code according to the following URL,
# http://darwin.eeb.uconn.edu/eeb348-notes/testing-hardy-weinberg.pdf

hwe.jags.ABO <- function(n,...)
{
  hwe <- function()
  {
    # likelihood
    pi[1] <- p.a*p.a + 2*p.a*p.o
    pi[2] <- 2*p.a*p.b
    n[1:4] ~ dmulti(pi[],N)
    # priors
    a1 ~ dexp(1)
    b1 ~ dexp(1)
    o1 ~ dexp(1)
    p.a <- a1/(a1 + b1 + o1)
    p.b <- b1/(a1 + b1 + o1)
    p.o <- o1/(a1 + b1 + o1)
  }
  hwd <- function()
  {
    # likelihood
    pi[1] <- p.a*p.a + f*p.a*(1-p.a) + 2*p.a*p.o*(1-f)
    n[1:4] ~ dmulti(pi[],N)
    # priors
    a1 ~ dexp(1)
    b1 ~ dexp(1)
    o1 ~ dexp(1)
    p.a <- a1/(a1 + b1 + o1)
    p.b <- b1/(a1 + b1 + o1)
    p.o <- o1/(a1 + b1 + o1)
    f ~ dunif(0,1)
  }
  N <- sum(n)
  ABO.hwe <- R2jags::jags(list(n=n,N=N),c("pi","p.a","p.b","p.o"),hwe,...)
  ABO.hwd <- R2jags::jags(list(n=n,N=N),c("pi","p.a","p.b","p.o","f"),hwd,...)
  invisible(list(hwe=ABO.hwe,hwd=ABO.hwd))
}

hwe.jags.ABO.results <- hwe.jags.ABO(n=c(862, 131, 365, 702))

hwe.jags.ABO.results

## End(Not run)
**invnormal**  
*Inverse normal transformation*

**Description**  
Inverse normal transformation

**Usage**  
```r
invnormal(x)
```

**Arguments**

- `x`  
  Data with missing values.

**Value**  
Transformed value.

**Examples**

```r
x <- 1:10
z <- invnormal(x)
plot(z,x,type="b")
```

---

**kin.morgan**  
*kinship matrix for simple pedigree*

**Description**  
kinship matrix according to Morgan v2.1.

**Usage**  
```r
kin.morgan(ped, verbose = FALSE)
```

**Arguments**

- `ped`  
  individual's id, father's id and mother's id.
- `verbose`  
  an option to print out the original pedigree.

**Value**  
The returned value is a list containing:

- **kin**  
  the kinship matrix in vector form
- **kin.matrix**  
  the kinship matrix
**Note**

The input data is required to be sorted so that parents precede their children.

**Author(s)**

Morgan development team, Jing Hua Zhao

**References**

Morgan V2.1 [https://sites.stat.washington.edu/thompson/Genepi/MORGAN/Morgan.shtml](https://sites.stat.washington.edu/thompson/Genepi/MORGAN/Morgan.shtml)

**See Also**

gif

**Examples**

```r
## Not run:
# Werner syndrome pedigree
werner<-c(
  1, 0, 0, 1,
  2, 0, 0, 2,
  3, 0, 0, 2,
  4, 1, 2, 1,
  5, 0, 0, 1,
  6, 1, 2, 2,
  7, 1, 2, 2,
  8, 0, 0, 1,
  9, 4, 3, 2,
 10, 5, 6, 1,
 11, 5, 6, 2,
 12, 8, 7, 1,
 13,10, 9, 2,
 14,12, 11, 1,
 15,14, 13, 1)
werner<-t(matrix(werner,nrow=4))
kin.morgan(werner[,1:3])
## End(Not run)
```

---

**klem**

*Haplotype frequency estimation based on a genotype table of two multiallelic markers*

**Description**

Haplotype frequency estimation using expectation-maximization algorithm based on a table of genotypes of two multiallelic markers.
Usage

klem(obs, k=2, l=2)

Arguments

obs a table of genotype counts
k number of alleles at marker 1
l number of alleles at marker 2

Details

The dimension of the genotype table should be \( k(k+1)/2 \times l(l+1)/2 \).
Modified from 2ld.c.

Value

The returned value is a list containing:

h haplotype Frequencies
l0 log-likelihood under linkage equilibrium
l1 log-likelihood under linkage disequilibrium

Author(s)

Jing Hua Zhao

See Also

genecounting

Examples

## Not run:
# an example with known genotype counts
z <- klem(obs=1:9)
# an example with imputed genotypes at SH2B1
cwd <- getwd()
cs.dir <- file.path(path.package("gap"),"tests/klem")
setwd(cs.dir)
dir()
source("SH2B1.R",echo=TRUE)
setwd(cwd)

## End(Not run)
LD22

LD statistics for two diallelic markers

Description
LD statistics for two SNPs.

Usage
LD22(h, n)

Arguments
h a vector of haplotype frequencies.
n number of haplotypes.

Details
It is possible to perform permutation test of $r^2$ by re-ordering the genotype through R’s sample function, obtaining the haplotype frequencies by gc.em or genecounting, supplying the estimated haplotype frequencies to the current function and record x2, and comparing the observed x2 and that from the replicates.

Value
The returned value is a list containing:

- h the original haplotype frequency vector
- n the number of haplotypes
- D the linkage disequilibrium parameter
- VarD the variance of D
- Dmax the maximum of D
- VarDmax the variance of Dmax
- Dprime the scaled disequilibrium parameter
- VarDprime the variance of Dprime
- x2 the Chi-squared statistic
- lor the log(OR) statistic
- vlor the var[log(OR)] statistic

Note
extracted from 2ld.c.


Author(s)

Jing Hua Zhao

References

Zabetian CP, Buxbaum SG, Elston RC, Kohnke MD, Anderson GM, Gelernter J, Cubells JF. The structure of linkage disequilibrium at the DBH locus strongly influences the magnitude of association between diallelic markers and plasma dopamine beta-hydroxylase activity Am J Hum Genet 72: 1389-1400


See Also

LDk1

Examples

```r
## Not run:
h <- c(0.442356, 0.291532, 0.245794, 0.020319)
n <- 481*2
t <- LD22(h, n)
## End(Not run)
```

LDk1

*LD statistics for two multiallelic markers*

Description

LD statistics for two multiallelic loci. For two diallelic makers, the familiar $r^2$ has standard error $\text{se}X2$.

Usage

`LDk1(n1 = 2, n2 = 2, h, n, optrho = 2, verbose = FALSE)`

Arguments

- `n1`: number of alleles at marker 1.
- `n2`: number of alleles at marker 2.
- `h`: a vector of haplotype frequencies.
- `n`: number of haplotypes.
- `optrho`: type of contingency table association, 0=Pearson, 1=Tschuprow, 2=Cramer (default).
- `verbose`: detailed output of individual statistics.
Value

The returned value is a list containing:

- **n1** the number of alleles at marker 1
- **n2** the number of alleles at marker 2
- **h** the haplotype frequency vector
- **n** the number of haplotypes
- **Dp** $D'$
- **VarDp** variance of $D'$
- **Dijtable** table of Dij
- **VarDijtable** table of variances for Dij
- **Dmaxtable** table of Dmax
- **Dijptable** table of $Dij'$
- **VarDijptable** table of variances for $Dij'$
- **X2table** table of Chi-squares (based on Dij)
- **ptable** table of p values
- **x2** the Chi-squared statistic
- **seX2** the standard error of $x2/n$
- **rho** the measure of association
- **seR** the standard error of rho
- **optrho** the method for calculating rho
- **klinfo** the Kullback-Leibler information

Note

adapted from 2ld.c.

Author(s)

Jing Hua Zhao

References

log10p

Description

log10(p) for a normal deviate z

Usage

log10p(z)

Arguments

z  
normal deviate.

Value

log10(P)

Author(s)

James Peters
**log10pvalue**

*log10(p) for a P value including its scientific format*

**Description**

log10(p) for a P value including its scientific format

**Usage**

```r
log10pvalue(p = NULL, base = NULL, exponent = NULL)
```

**Arguments**

- `p`: value.
- `base`: base part in scientific format.
- `exponent`: exponent part in scientific format.

**Value**

log10(P)

**Examples**

```r
log10pvalue(1e-323)
log10pvalue(base=1, exponent=-323)
```

---

**logp**

*log(p) for a normal deviate z*

**Description**

log(p) for a normal deviate z

**Usage**

```r
logp(z)
```

**Arguments**

- `z`: normal deviate.
makeped

Value

\[ \log_{10}(P) \]

Examples

\[ \log_{10}(100) \]

---

**Description**

Many computer programs for genetic data analysis require pedigree data to be in the so-called “post-MAKEPED” format. This function performs this translation and allows for some inconsistencies to be detected.

**Usage**

```r
makeped(
  pifile = "pedfile.pre",
  pofile = "pedfile.ped",
  auto.select = 1,
  with.loop = 0,
  loop.file = NA,
  auto.proband = 1,
  proband.file = NA
)
```

**Arguments**

- `pifile`: input filename.
- `pofile`: output filename.
- `auto.select`: no loops in pedigrees and probands are selected automatically? 0=no, 1=yes.
- `with.loop`: input data with loops? 0=no, 1=yes.
- `loop.file`: filename containing pedigree id and an individual id for each loop, set if with.loop=1.
- `auto.proband`: probands are selected automatically? 0=no, 1=yes.
- `proband.file`: filename containing pedigree id and proband id, set if auto.proband=0 (not implemented).

Before invoking makeped, input file, loop file and proband file have to be prepared.  
By default, auto.select=1, so translation proceeds without considering loops and proband statuses. If there are loops in the pedigrees, then set auto.select=0, with.loop=1, loop.file=“filespec”.

There may be several versions of makeped available, but their differences with this port should be minor.
Details

The first four columns of the input file contain the following information:
- Pedigree ID, individual ID, father’s ID, mother’s ID, sex
Either father’s or mother’s id is set to 0 for founders, i.e., individuals with no parents. Numeric coding for sex is 0=unknown, 1=male, 2=female. These can be followed by satellite information such as disease phenotype and marker information.
The output file has extra information extracted from data above.

Note

adapted from makeped.c by W Li and others. keywords datagen

Source

https://lab.rockefeller.edu/ott/

Examples

```r
## Not run:
cwd <- getwd()
cs.dir <- file.path(path.package("gap.datasets"),"tests","kinship")
setwd(cs.dir)
dir()
makeped("ped7.pre","ped7.ped",0,1,"ped7.lop")
setwd(cwd)

## End(Not run)
```

---

**masize**

Sample size calculation for mediation analysis

**Description**

The function computes sample size for regression problems where the goal is to assess mediation of the effects of a primary predictor by an intermediate variable or mediator.

**Usage**

`masize(model, opts, alpha = 0.025, gamma = 0.2)`

**Arguments**

- `model` "linear", "logistic", "poisson", "cox", where i,j,k,l range from 1 to 4,5,9,9, respectively.
- `opts` A list specific to the model
b1 regression coefficient for the primary predictor X1
b2 regression coefficient for the mediator X2
rho correlation between X1 and X2
sdx1, sdx2 standard deviations (SDs) of X1 and X2
f1, f2 prevalence of binary X1 and X2
sdx residual SD of the outcome for the linear model
p marginal prevalence of the binary outcome in the logistic model
m marginal mean of the count outcome in a Poisson model
f proportion of uncensored observations for the Cox model
fc proportion of observations censored early
alpha one-sided type-I error rate
gamma type-II error rate
ns number of observations to be simulated
seed random number seed

For linear model, the arguments are b2, rho, sdx2, sdy, alpha, and gamma. For cases CpBm and BpBm, set sdx2 = √f2(1−f2). Three alternative functions are included for the linear model. These functions make it possible to supply other combinations of input parameters affecting mediation:

b1* coefficient for the primary predictor in the reduced model excluding the mediator (b1star)
b1 coefficient for the primary predictor in the full model including the mediator
PTE proportion of the effect of the primary predictor explained by the mediator, defined as (b1*-b1)/b1*

These alternative functions for the linear model require specification of an extra parameter, but are provided for convenience, along with two utility files for computing PTE and b1* from the other parameters. The required arguments are explained in comments within the R code.

alpha Type-I error rate, one-sided.
gamma Type-II error rate.

Details

Mediation has been thought of in terms of the proportion of effect explained, or the relative attenuation of b1, the coefficient for the primary predictor X1, when the mediator, X2, is added to the model. The goal is to show that b1*, the coefficient for X1 in the reduced model (i.e., the model with only X1, differs from b1, its coefficient in the full model (i.e., the model with both X1 and the mediator X2. If X1 and X2 are correlated, then showing that b2, the coefficient for X2, differs from zero is equivalent to showing b1* differs from b1. Thus the problem reduces to detecting an effect of X2, controlling for X1. In short, it amounts to the more familiar problem of inflating sample size to account for loss of precision due to adjustment for X1.

The approach here is to approximate the expected information matrix from the regression model including both X1 and X2, to obtain the expected standard error of the estimate of b2, evaluated at the
MLE. The sample size follows from comparing the Wald test statistic (i.e., the ratio of the estimate of b2 to its SE) to the standard normal distribution, with the expected value of the numerator and denominator of the statistic computed under the alternative hypothesis. This reflects the Wald test for the statistical significance of a coefficient implemented in most regression packages.

The function provides methods to calculate sample sizes for the mediation problem for linear, logistic, Poisson, and Cox regression models in four cases for each model:

- **CpCm**: continuous primary predictor, continuous mediator
- **BpCm**: binary primary predictor, continuous mediator
- **CpBm**: continuous primary predictor, binary mediator
- **BpBm**: binary primary predictor, binary mediator

The function is also generally applicable to the analogous problem of calculating sample size adequate to detect the effect of a primary predictor in the presence of confounding. Simply treat X2 as the primary predictor and consider X1 the confounder.

For linear model, a single function, `linear`, implements the analytic solution for all four cases, based on Hsieh et al., is to inflate sample size by a variance inflation factor, \(1/(1 - \rho^2)\), where \(\rho\) is the correlation of X1 and X2. This also turns out to be the analytic solution in cases CpCm and BpCm for the Poisson model, and underlies approximate solutions for the logistic and Cox models. An analytic solution is also given for cases CpBm and BpBm for the Poisson model. Since analytic solutions are not available for the logistic and Cox models, a simulation approach is used to obtain the expected information matrix instead.

For logistic model, the approximate solution due to Hsieh is implemented in the function `logistic.approx`, and can be used for all four cases. Arguments are p, b2, rho, sdx2, alpha, and gamma. For a binary mediator with prevalence f2, sdx2 should be reset to \(\sqrt{f_2(1-f_2)}\). Simulating the information matrix of the logistic model provides somewhat more accurate sample size estimates than the Hsieh approximation. The functions for cases CpCm, BpCm, CpBm, and BpBm are respectively `logistic.ccs`, `logistic.bcs`, `logistic.cbs`, and `logistic.bbs`, as for the Poisson and Cox models. Arguments for these functions include p, b1, sdx1 or f1, b2, sdx2 or f2, rho, alpha, gamma, and ns. As in other functions, sdx1, sdx2, alpha, and gamma are set to the defaults listed above. These four functions call two utility functions, `getb0` (to calculate the intercept parameter from the others) and `antilogit`, which are supplied.

For Poisson model, The function implementing the approximate solution based on the variance inflation factor is `poisson.approx`, and can be used for all four cases. Arguments are EY (the marginal mean of the Poisson outcome), b2, sdx2, rho, alpha and gamma, with sdx2, alpha and gamma set to the usual defaults; use sdx2=\(\sqrt{f_2(1-f_2)}\) for a binary mediator with prevalence f2 (cases CpBm and BpBm). For cases CpCm and BpCm (continuous mediators), the approximate formula is also the analytic solution. For these cases, we supply redundant functions `poisson.cc` and `poisson.bc`, with the same arguments and defaults as for `poisson.approx` (it’s the same function). For the two cases with binary mediators, the functions are `poisson.ch` and `poisson.bb`. In addition to m, b2, f2, rho, alpha, and gamma, b1 and sdx1 or f1 must be specified. Defaults are as usual. Functions using simulation for the Poisson model are available: `poisson.ccs`, `poisson.bcs`, `poisson.cbs`, and `poisson.bbs`. As in the logistic case, these require arguments b1 and sdx1 or f1. For this case, however, the analytic functions are faster, avoid simulation error, and should be used. We include these functions as templates that could be adapted to other joint predictor distributions. For Cox model, the function implementing the approximate solution, using the variance inflation factor and derived by Schmoor et al., is `cox.approx`, and can be used for all four cases. Arguments are b2, sdx2, rho,
alpha, gamma, and f. For binary X2 set sdX2 = \sqrt{f^2(1 - f^2)}. The approximation works very well for cases CpCm and BpCm (continuous mediators), but is a bit less accurate for cases CpBm and BpBm (binary mediators). We get some improvement for those cases using the simulation approach. This approach is implemented for all four, as functions cox.ccs, cox.bcs, cox.cbs, and cox.bbs. Arguments are b1, sdx1 or f1, b2, sdx2 or f2, rho, alpha, gamma, f, and ns, with defaults as described above. Slight variants of these functions, cox.ccs2, cox.bcs2, cox.cbs2, and cox.bbs2, make it possible to allow for early censoring of a fraction fc of observations; but in our experience this has virtually no effect, even with values of fc of 0.5. The default for fc is 0.

A summary of the arguments is as follows, noting that additional parameter seed can be supplied for simulation-based method.

<table>
<thead>
<tr>
<th>model</th>
<th>arguments</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear1</td>
<td>b2, rho, sdx2, sdy</td>
<td>linear</td>
</tr>
<tr>
<td>linear2</td>
<td>b1star, PTE, rho, sdx1, sdy</td>
<td>lineara</td>
</tr>
<tr>
<td>linear3</td>
<td>b1star, b2, PTE, sdx1, sdx2, sdy</td>
<td>linearb</td>
</tr>
<tr>
<td>linear4</td>
<td>b1star, b1, b2, sdx1, sdx2, sdy</td>
<td>linearc</td>
</tr>
<tr>
<td>logistic1</td>
<td>p, b2, rho, sdx2</td>
<td>logistic.approx</td>
</tr>
<tr>
<td>logistic2</td>
<td>p, b1, b2, rho, sdx1, sdx2, ns</td>
<td>logistic.ccs</td>
</tr>
<tr>
<td>logistic3</td>
<td>p, b1, f1, b2, rho, sdx2, ns</td>
<td>logistic.bcs</td>
</tr>
<tr>
<td>logistic4</td>
<td>p, b1, b2, f2, rho, sdx1, ns</td>
<td>logistic.cbs</td>
</tr>
<tr>
<td>logistic5</td>
<td>p, b1, f1, b2, f2, rho, ns</td>
<td>logistic.bbs</td>
</tr>
<tr>
<td>poisson1</td>
<td>m, b2, rho, sdx2</td>
<td>poisson.approx</td>
</tr>
<tr>
<td>poisson2</td>
<td>m, b2, rho, sdx2</td>
<td>poisson.cc</td>
</tr>
<tr>
<td>poisson3</td>
<td>m, b2, rho, sdx2</td>
<td>poisson.bc</td>
</tr>
<tr>
<td>poisson4</td>
<td>m, b1, b2, f2, rho, sdx1</td>
<td>poisson.cb</td>
</tr>
<tr>
<td>poisson5</td>
<td>m, b1, f1, b2, f2, rho</td>
<td>poisson.bb</td>
</tr>
<tr>
<td>poisson6</td>
<td>m, b1, b2, rho, sdx1, sdx2, ns</td>
<td>poisson.ccs</td>
</tr>
<tr>
<td>poisson7</td>
<td>m, b1, f1, b2, rho, sdx2, ns</td>
<td>poisson.bcs</td>
</tr>
<tr>
<td>poisson8</td>
<td>m, b1, b2, f2, rho, sdx1, ns</td>
<td>poisson.cbs</td>
</tr>
<tr>
<td>poisson9</td>
<td>m, b1, f1, b2, f2, rho, ns</td>
<td>poisson.bbs</td>
</tr>
<tr>
<td>cox1</td>
<td>b2, rho, f, sdx2</td>
<td>cox.approx</td>
</tr>
<tr>
<td>cox2</td>
<td>b1, b2, rho, f, sdx1, sdx2, ns</td>
<td>cox.ccs</td>
</tr>
<tr>
<td>cox3</td>
<td>b1, f1, b2, rho, f, sdx2, ns</td>
<td>cox.bcs</td>
</tr>
<tr>
<td>cox4</td>
<td>b1, b2, f2, rho, f, sdx1, ns</td>
<td>cox.cbs</td>
</tr>
<tr>
<td>cox5</td>
<td>b1, f1, b2, f2, rho, f, ns</td>
<td>cox.bbs</td>
</tr>
<tr>
<td>cox6</td>
<td>b1, b2, rho, f, fc, sdx1, sdx2, ns</td>
<td>cox.ccs2</td>
</tr>
<tr>
<td>cox7</td>
<td>b1, f1, b2, rho, f, fc, sdx2, ns</td>
<td>cox.bcs2</td>
</tr>
<tr>
<td>cox8</td>
<td>b1, b2, f2, rho, f, fc, sdx1, ns</td>
<td>cox.cbs2</td>
</tr>
<tr>
<td>cox9</td>
<td>b1, f1, b2, f2, rho, f, fc, ns</td>
<td>cox.bbs2</td>
</tr>
</tbody>
</table>

Value

A short description of model (desc, b=binary, c=continuous, s=simulation) and sample size (n). In the case of Cox model, number of events (d) is also indicated.
References


See Also

ab

Examples

## Not run:
## linear model
# CpCm
opts <- list(b2=0.5, rho=0.3, sdx2=1, sdy=1)
masize("linear1",opts)
# BpBm
opts <- list(b2=0.75, rho=0.3, f2=0.25, sdx2=sqrt(0.25*0.75), sdy=3)
masize("linear1",opts,gamma=0.1)

## logistic model
# CpBm
opts <- list(p=0.25, b2=log(0.5), rho=0.5, sdx2=0.5)
masize("logistic1",opts)
opts <- list(p=0.25, b1=log(1.5), sdx1=1, b2=log(0.5), f2=0.5, rho=0.5, ns=10000,
seed=1234)
masize("logistic4",opts)
opts <- list(p=0.25, b1=log(1.5), sdx1=4.5, b2=log(0.5), f2=0.5, rho=0.5, ns=50000,
seed=1234)
masize("logistic4",opts)

## Poisson model
# BpBm
opts <- list(m=0.5, b2=log(1.25), rho=0.3, sdx2=sqrt(0.25*0.75))
masize("poisson1",opts)
opts <- list(m=0.5, b1=log(1.4), f1=0.25, b2=log(1.25), f2=0.25, rho=0.3)
masize("poisson5",opts)
 opts <- c(opts,ns=10000, seed=1234)
masize("poisson9",opts)

## Cox model
# BpBm
opts <- list(b2=log(1.5), rho=0.45, f=0.2, sdx2=sqrt(0.25*0.75))
masize("cox1",opts)
 opts <- list(b1=log(2), f1=0.5, b2=log(1.5), f2=0.25, rho=0.45, f=0.2, seed=1234)
MCMCgrm

Mixed modeling with genetic relationship matrices

Description

Mixed modeling with genomic relationship matrix. This is appropriate with relationship matrix derived from family structures or unrelated individuals based on whole genome data.

Usage

MCMCgrm(
  model,  # statistical model.
  prior,  # a list of priors for parameters in the model above.
  data,   # a data.frame containing outcome and covariates.
  GRM,    # a relationship matrix.
  eps = 0,  # a small number added to the diagonal of the a nonpositive definite GRM.
  n.thin = 10,  # thinning parameter in the MCMC.
  n.burnin = 3000,  # the number of burn-in's.
  n.iter = 13000,  # the number of iterations.
  ...  # other options as appropriate for MCMCglmm.
)

Arguments

table

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>statistical model.</td>
</tr>
<tr>
<td>prior</td>
<td>a list of priors for parameters in the model above.</td>
</tr>
<tr>
<td>data</td>
<td>a data.frame containing outcome and covariates.</td>
</tr>
<tr>
<td>GRM</td>
<td>a relationship matrix.</td>
</tr>
<tr>
<td>eps</td>
<td>a small number added to the diagonal of the a nonpositive definite GRM.</td>
</tr>
<tr>
<td>n.thin</td>
<td>thinning parameter in the MCMC.</td>
</tr>
<tr>
<td>n.burnin</td>
<td>the number of burn-in's.</td>
</tr>
<tr>
<td>n.iter</td>
<td>the number of iterations.</td>
</tr>
<tr>
<td>...</td>
<td>other options as appropriate for MCMCglmm.</td>
</tr>
</tbody>
</table>
Details

The function was created to address a number of issues involving mixed modelling with family data or population sample with whole genome data. First, the implementation will shed light on the uncertainty involved with polygenic effect in that posterior distributions can be obtained. Second, while the model can be used with the MCMCglmm package there is often issues with the specification of pedigree structures but this is less of a problem with genetic relationship matrices. We can use established algorithms to generate kinship or genomic relationship matrix as input to the MCMCglmm function. Third, it is more intuitive to specify function arguments in line with other packages such as R2OpenBUGS, R2jags or glmmBUGS. In addition, our experiences of tuning the model would help to reset the input and default values.

Value

The returned value is an object as generated by MCMCglmm.

Author(s)

Jing Hua Zhao

References


Examples

```r
## Not run:
### with kinship
# library(kinship)
# fam <- with(l51,makefamid(id,fid,mid))
# s <-with(l51, makekinship(fam, id, fid, mid))
# K <- as.matrix(s)*2

### with gap
s <- kin.morgan(l51)
K <- with(s,kin.matrix*2)
prior <- list(R=list(V=1, nu=0.002), G=list(G1=list(V=1, nu=0.002)))
m <- MCMCgrm(qt~1,prior,l51,K)
save(m,file="l51.m")
pdf("l51.pdf")
plot(m)
dev.off()

# A real analysis on bats
## data
bianfu.GRM <- read.table("bianfu.GRM.txt", header = TRUE)
bianfu.GRM[1:5,1:6]
Data <- read.table(file = "PHONE.txt", header = TRUE,
colClasses=c(rep("factor",3),rep("numeric",7)))
```
library("MCMCglmm")
GRM <- as.matrix(bianfu.GRM[, -1])
colnames(GRM) <- rownames(GRM) <- bianfu.GRM[, 1]
library(gap)
names(Data)[1] <- "id"
prior <- list(G = list(G1 = list(V = 1, nu = 0.002)), R = list(V = 1, nu = 0.002))
model1.1 <- MCMCglmm(WEIGHT ~ 1, prior, Data, GRM, n.burnin=100, n.iter=1000, verbose=FALSE)
## an alternative
names(Data)[1] <- "animal"
N <- nrow(Data)
i <- rep(1:N, rep(N, N))
j <- rep(1:N, N)
s <- Matrix::spMatrix(N, N, i, j, as.vector(GRM))
Ginv <- Matrix::solve(s)
class(Ginv) <- "dgCMatrix"
rownames(Ginv) <- Ginv@Dimnames[[1]] <- with(Data, animal)
model1.2 <- MCMCglmm(WEIGHT ~ 1, random= ~ animal, data = Data,
  ginverse=list(animal=Ginv), prior = prior, burnin=100, nitt=1000, verbose=FALSE)
## without missing data
model1.3 <- MCMCglmm(Peak_Freq ~ WEIGHT, random = ~ animal,
  data=subset(Data,!is.na(Peak_Freq)&!is.na(WEIGHT)),
  ginverse=list(animal=Ginv), prior = prior, burnin=100, nitt=1000, verbose=FALSE)

## End(Not run)

METAL_forestplot function as R/meta's forest for METAL outputs

Description

This function takes a meta-data from METAL (tbl) and data from contributing studies (all) for forest plot. It also takes a SNPID-rsid mapping (rsid) as contributing studies often involve discrepancies in rsid so it is appropriate to use SNPID, i.e., chr:pos_A1_A2 (A1<=A2).

Usage

METAL_forestplot(tbl, all, rsid, package = "meta", split = FALSE, ...)

Arguments

tbl Meta-analysis summary statistics.
all statistics from all contributing studies.
rsid SNPID-rsid mapping file.
package style of plot as in meta, rmeta or forestplot.
split when TRUE, individual prot-MarkerName.pdf will be generated.
... options to use for the individual pdf device.
**Details**

The study-specific and total sample sizes (N) can be customised from METAL commands

```
CUSTOMVARIABLE N
LABEL N as N
WEIGHTLABEL N
```

**Value**

It will generate a forest plot specified by pdf for direction-adjusted effect sizes.

**Author(s)**

Jing Hua Zhao

**References**


**See Also**

`METAL_forestplot`

**Examples**

```r
## Not run:
require(gap.datasets)
data(OPG)
METAL_forestplot(OPGtbl,OPGall,OPGrsid,width=8.75,height=5)
## End(Not run)
```

---

`metap`  

*Meta-analysis of p values*

**Description**

This function is the method of meta-analysis used in the Genetic Investigation of ANThropometric Traits (GIANT) consortium, which is based on normal approximation of p values and weighted by sample sizes from individual studies.

**Usage**

```
metap(data, N, verbose = "Y", prefixp = "p", prefixn = "n")
```
Arguments

- **data**: data frame.
- **N**: Number of studies.
- **verbose**: Control of detailed output.
- **prefixp**: Prefix of p value, with default value "p".
- **prefixn**: Prefix of sample size, with default value "n".

Value

- **x2**: Fisher's chi-squared statistics.
- **p**: P values from Fisher's method according to chi-squared distribution with 2*N degree(s) of freedom.
- **z**: Combined z value.
- **p1**: One-sided p value.
- **p2**: Two-sided p value.

Author(s)

Jing Hua Zhao

See Also

- `metareg`

Examples

```r
## Not run:
s <- data.frame(p1=0.1^rep(8:2,each=7,times=1),n1=rep(32000,49),
p2=0.1^rep(8:2,each=1,times=7),n2=rep(8000,49))
cbind(s,metap(s,2))
```

```r
# Speliotes, Elizabeth K., M.D. [ESPELIOTES@PARTNERS.ORG]
# 22-2-2008 MRC-Epid JHZ

np <- 7
p <- 0.1^((np+1):2)
z <- qnorm(1-p/2)
n <- c(32000,8000)
n1 <- n[1]
s1 <- s2 <- vector("numeric")

for (i in 1:np)
{
a <- z[i]
  for (j in 1:np)
  {
    b <- z[j]
    ```
metaz1 <- (sqrt(n1)*a+sqrt(n1)*b)/sqrt(n1+n1)
metap1 <- pnorm(-abs(metaz1))
metaz2 <- (sqrt(n1)*a+sqrt(n2)*b)/sqrt(n1+n2)
metap2 <- pnorm(-abs(metaz2))
k <- (i-1)*np+j
cat(k, "\t", p1, "\t", p2, "\t", metap1, metaz1, "\t", metap2, metaz2, "\n")
s1[k] <- metap1
s2[k] <- metap2

q <- -log10(sort(p, decreasing=TRUE))
t1 <- matrix(-log10(sort(s1, decreasing=TRUE)), np, np)
t2 <- matrix(-log10(sort(s2, decreasing=TRUE)), np, np)
par(mfrow=c(1,2), bg="white", mar=c(4.2,3.8,0.2,0.2))
persp(q, q, t1)
persp(q, q, t2)
## End(Not run)

metareg

Fixed and random effects model for meta-analysis

Description

Given \( k = n \) studies with \( b_1, \ldots, b_N \) being \( \beta \)'s and \( se_1, \ldots, se_N \) standard errors from regression, the fixed effects model uses inverse variance weighting such that \( w_1 = 1/se^2_1, \ldots, w_N = 1/se^2_N \) and the combined \( \beta \) as the weighted average, \( \beta_f = (b_1 * w_1 + \ldots + b_N * w_N)/w, \) with \( w = w_1 + \ldots + w_N \) being the total weight, the se for this estimate is \( se_f = \sqrt{1/w}. \) A normal z-statistic is obtained as \( z_f = \beta_f/se_f, \) and the corresponding p value \( p_f = 2 * pnorm(-abs(z_f)). \) For the random effects model, denote \( q_w = w_1 * (b_1 - \beta_f)^2 + \ldots + w_N * (b_N - \beta_f)^2 \) and \( dl = max(0, q_w - (k - 1))/(w - (w_1^2 + \ldots + w_N^2)/w)) \), corrected weights are obtained such that \( w_1c = 1/(1/w_1 + dl), \ldots, w_Nc = 1/(1/w_N + dl), \) totaling \( w_c = w_1c + \ldots + w_Nc. \) The combined \( \beta \) and se are then \( \beta_r = (b_1 * w_1 + \ldots + b_N * w_N)/w_c \) and \( se_r = \sqrt{(1/w_c)}, \) leading to a z-statistic \( z_r = \beta_r/se_r \) and a p-value \( p_r = 2 * pnorm(-abs(z_r)). \) Moreover, a p-value testing for heterogeneity is \( p_{heter} = pchisq(q_w, k - 1, lower.tail = FALSE). \)

Usage

metareg(data, N, verbose="Y", prefixb="b", prefixse="se")

Arguments

data Data frame to be used
N Number of studies
verbose A control for screen output
prefixb Prefix of estimate; default value is "b"
prefixse Prefix of standard error; default value is "se" The function accepts a wide format data with estimates as $b_1, \ldots, b_N$ and standard errors as $se_1, \ldots, se_N$. More generally, they can be specified by prefixes in the function argument.

Value

The returned value is a data frame with the following variables:

- p_f P value (fixed effects model)
- p_r P value (random effects model)
- beta_f regression coefficient
- beta_r regression coefficient
- se_f standard error
- se_r standard error
- z_f z value
- z_r z value
- p_heter heterogeneity test p value
- $I^2$ statistic
- k No of tests used
- eps smallest double-precision number

References


Note

Adapted from a SAS macro

Author(s)

Shengxu Li, Jing Hua Zhao

Examples

```r
## Not run:
abc <- data.frame(chromosome=1,rsn='abcd',startpos=1234,
                   b1=1,se1=2,p1=0.1,b2=2,se2=6,p2=0,b3=3,se3=8,p3=0.5)
metareg(abc,3)
abc2 <- data.frame(b1=c(1,2),se1=c(2,4),b2=c(2,3),se2=c(4,6),b3=c(3,4),se3=c(6,8))
print(metareg(abc2,3))
## End(Not run)
```
mhtplot  Manhattan plot

Description
To generate Manhattan plot, e.g., of genomewide significance (p values) and a random variable that is uniformly distributed. By default, a log10-transformation is applied. Note that with real chromosomal positions, it is also appropriate to plot and some but not all chromosomes.

Usage
mhtplot(data, control = mht.control(), hcontrol = hmht.control(), ...)

Arguments
data a data frame with three columns representing chromosome, position and p values
control A control function named mht.control() with the following arguments,
  • type a flag with value "p" or "l" indicating if points or lines are to be drawn.
  • usepos a flag to use real chromosomal positions as composed to ordinal positions with default value FALSE
  • logscale a flag to indicate if p value is to be log-transformed with default value TRUE
  • base the base of the logarithm with default value 10
  • cutoffs the cut-offs where horizontal line(s) are drawn with default value NULL
  • colors the color for different chromosome(s), and random if unspecified with default values NULL
  • labels labels for the ticks on x-axis with default value NULL
  • srt degree to which labels are rotated with default value of 45
  • gap gap between chromosomes with default value NULL
  • cex cex for the data points
  • yline Margin line position
  • xline Margin line position
hcontrol A control function named hmht.control() with the following arguments,
  • data. chunk of data to be highlighted with default value NULL
  • colors. colors for annotated genes
  • yoffset. offset above the data point showing most significant p value with default value 0.5
  • cex shrinkage factor for data points with default value 1.5
  • boxed if the label for the highlighted region with default value FALSE

... other options in compatible with the R plot function.
Details

It is possible to specify options such as xlab and ylim when the plot is requested for data in other context.

Value

The plot is shown on or saved to the appropriate device.

Author(s)

Jing Hua Zhao

See Also

qqunif

Examples

```r
## Not run:
# foo example
test <- matrix(c(1,1,4,1,1,6,1,10,3,2,1,5,2,2,6,2,4,8),byrow=TRUE,6)
mhtplot(test)
mhtplot(test,mht.control(logscale=FALSE))

# fake example with Affy500k data
affy <-c(40220, 41400, 33801, 32334, 32056, 31470, 25835, 27457, 22864, 28501, 26273,
        24954, 19188, 15721, 14356, 15309, 11281, 14881, 6399, 12400, 7125, 6207)
CM <- cumsum(affy)
n.markers <- sum(affy)
n.chr <- length(affy)
test <- data.frame(chr=rep(1:n.chr,affy),pos=1:n.markers,p=runif(n.markers))

# to reduce size of the plot
# bitmap("mhtplot.bmp",res=72*5)
oldpar <- par()
par(cex=0.6)
colors <- rep(c("blue","green"),11)
# other colors, e.g.
# colors <- c("red","blue","green","cyan","yellow","gray","magenta","red","blue","green",
#            "cyan","yellow","gray","magenta","red","blue","green","cyan","yellow","gray",
#            "magenta","red")
mhtplot(test,control=mht.control(colors=colors),pch=19,srt=0)
title("A simulated example according to EPIC-Norfolk QCed SNPs")
axis(2)
axis(1,pos=0,labels=FALSE,tick=FALSE)
abline(0,0)
# dev.off()
par(oldpar)

mhtplot(test,control=mht.control(usepos=TRUE,colors=colors,gap=10000),pch=19,bg=colors)
title("Real positions with a gap of 10000 bp between chromosomes")
box()
```
mhtplot.trunc

Truncated Manhattan plot

Description

To generate truncated Manhattan plot, e.g., of genomewide significance (P values) or a random variable that is uniformly distributed.

Usage

mhtplot.trunc(
  x,
  chr = "CHR",
)
bp = "BP",
p = NULL,
log10p = NULL,
z = NULL,
snp = "SNP",
col = c("gray10", "gray60"),
chrlabs = NULL,
suggestiveline = -log10(1e-05),
genomewideline = -log10(5e-08),
highlight = NULL,
annotatelog10P = NULL,
annotateTop = FALSE,
cex.mtext = 1.5,
cex.text = 0.7,
mtext.line = 2,
y.ax.space = 5,
y.brk1,
y.brk2,
trunc.yaxis = TRUE,
cex.axis = 1.2,
delta = 0.05,
...)

Arguments

x A data.frame.
chr Chromosome.
bp Position.
p p values, e.g., "1.23e-600".
log10p log10(p).
z z statistic, i.e., BETA/SE.
snp SNP. Pending on the setup it could either of variant or gene ID(s).
col Colours.
chrlabs Chromosome labels, 1,2,...22,23,24,25.
suggestiveline Suggestive line.
genomewideline Genomewide line.
highlight A list of SNPs to be highlighted.
annotatelog10P Threshold of -log10(P) to annotate.
annotateTop Annotate top.
cex.mtext axis label extension factor.
cex.text SNP label extension factor.
mtext.line position of the y lab.
y.ax.space interval of ticks of the y axis.
mhtplot.trunc

y.brk1    lower -log10(P) break point.
y.brk2    upper -log10(P) break point.
trunc.yaxis do not truncate y-axis when FALSE.
cex.axis  extension factor for y-axis.
delta     a value to enable column(s) of red points.

Details
The rationale of this function is to extend mhtplot() to handle extremely small p values as often seen
from a protein GWAS; for R will break down when p <= 1e-324.

Value
The plot is shown on or saved to the appropriate device.

Author(s)
James Peters, Jing Hua Zhao

See Also
mhtplot.

Examples
## Not run:
options(width=120)
require(gap.datasets)
mhtdata <- within(mhtdata, (z=qnorm(p/2, lower.tail=FALSE)))
mhtplot.trunc(mhtdata, chr = "chr", bp = "pos", z = "z", snp = "rsn",
y.brk1=6, y.brk2=10, y.ax.space=1, mtext.line=2.5)
# https://portals.broadinstitute.org/collaboration/
# giant/images/c/c8/1Meta-analysis_Locke_et_al%2BUKBiobank_2018_UPDATED.txt.gz
gz <- gzfile("work/1Meta-analysis_Locke_et_al\+UKBiobank_2018_UPDATED.txt.gz")
BMI <- within(read.delim(gz, as.is=TRUE), (Z <- BETA/SE))
print(subset(BMI[,c("CHR","POS","SNP","P")],CHR!=16 & P<=1e-150))
library(Rmpfr)
print(within(subset(BMI, P==0, select=c("CHR","POS","SNP","Z")),
(P <- format(2*pnorm(mpfr(abs(Z),100),lower.tail=FALSE));
Pvalue <- pvalue(Z); log10P <- -log10p(Z))))
png("BMI.png", res=300, units="in", width=9, height=6)
par oma=c(0,0,0,0), mar=c(5,6.5,1,1))
mhtplot.trunc(BMI, chr="CHR", bp="POS", z="Z", snp="SNP",
suggestiveline=FALSE, genomewideline=-log10(1e-8),
cex.mtext=1.2, cex.text=1.2,
annotate.log10P=156, annotateTop = FALSE,
highlight=c("rs13021737","rs17817449","rs6567160"),
mtext.line=3, y.brk1=200, y.brk2=280, trunc.yaxis=TRUE,
cex.axis=1.2, cex=0.5,
mhtplot2

Manhattan plot with annotations

Description
To generate Manhattan plot with annotations. The function is generic and for instance could be used for genomewide p values or any random variable that is uniformly distributed. By default, a log10-transformation is applied. Note that with real chromosomal positions, it is also appropriate to plot and some but not all chromosomes.

Usage
mhtplot2(data, control = mht.control(), hcontrol = hmht.control(), ...)

Arguments
- data: a data frame with three columns representing chromosome, position and p values.
- control: A control function named mht.control() with the following arguments.
  - type: a flag with value "p" or "l" indicating if points or lines are to be drawn.
  - usepos: a flag to use real chromosomal positions as composed to ordinal positions with default value FALSE
  - logscale: a flag to indicate if p value is to be log-transformed with default value TRUE
  - base: the base of the logarithm with default value 10
  - cutoffs: the cut-offs where horizontal line(s) are drawn with default value NULL
  - colors: the color for different chromosome(s), and random if unspecified with default values NULL
  - labels: labels for the ticks on x-axis with default value NULL
  - srt: degree to which labels are rotated with default value of 45
  - gap: gap between chromosomes with default value NULL
  - cex: cex for the data points
  - yline: Margin line position
  - xline: Margin line position
- hcontrol: A control function named hmht.control() with the following arguments.
  - data: chunk of data to be highlighted with default value NULL
• colors colors for annotated genes
• yoffset offset above the data point showing most significant p value with default value 0.5
• cex shrinkage factor for data points with default value 1.5
• boxed if the label for the highlighted region with default value FALSE

... other options in compatible with the R plot function.

Details

It is possible to specify options such as xlab, ylim and font family when the plot is requested for data in other context.

To maintain back compatibility options as in mhtplot are used. The positions of the horizontal labels are now in the middle rather than at the beginning of their bands in the plot.

Value

The plot is shown on or saved to the appropriate device.

Author(s)

Jing Hua Zhao

References


Examples

## Not run:
The following example uses only chromosomes 14 and 20 of the Nat Genet paper.

```r
mdata <- within(hr1420, {
  c1 <- colour == 1
  c2 <- colour == 2
  c3 <- colour == 3
  colour[c1] <- 62
  colour[c2] <- 73
  colour[c3] <- 552
})
mdata <- mdata[, c("CHR", "POS", "P", "gene", "colour")]
ops <- mht.control(colors = rep(c("lightgray", "gray"), 11), yline = 1.5, xline = 2, srt = 0)
hops <- hmht.control(data = subset(mdata, !is.na(gene)))
v <- "Verdana"
ifelse(Sys.info()
  ["sysname"] == "Windows", windowsFonts(family = windowsFont(v)),
  family <- v)
tiff("mh.tiff", width = .03937 * 189, height = .03937 * 189 / 2, units = "in", res = 1200,
  compress = "lzw")
par(las = 2, xpd = TRUE, cex.axis = 1.8, cex = 0.4)
mhtplot2(with(mdata, cbind(CHR, POS, P, colour)), ops, hops, pch = 19,
```
mia

Multiple imputation analysis for hap

Description

This command reads outputs from hap session that uses multiple imputations, i.e. -mi# option. To simplify matters it assumes -ss option is specified together with -mi option there.

Usage

```
mia(
    hapfile = "hap.out",
    assfile = "assign.out",
    miafile = "mia.out",
    so = 0,
    ns = 0,
    mi = 0,
    allsnps = 0,
    sas = 0
)
```

Arguments

- **hapfile** hap haplotype output file name.
- **assfile** hap assignment output file name.
- **miafile** mia output file name.
- **so** to generate results according to subject order.
- **ns** do not sort in subject order.
- **mi** number of multiple imputations used in hap.
- **allsnps** all loci are SNPs.
- **sas** produce SAS data step program.

# To exemplify the use of chr, pos and p without gene annotation
# in response to query from Vallejo, Roger <Roger.Vallejo@ARS.USDA.GOV>
opar <- par()
par(cex=0.4)
ops <- mht.control(colors=rep(c("lightgray","lightblue"),11),srt=0,yline=2.5,xline=2)
mhtplot2(data.frame(mhtdata[,c("chr","pos","p")],gene=NA,color=NA),ops,xlab="",ylab="",srt=0)
axis(2,at=1:16)
title("data in mhtplot used by mhtplot2")
par(opar)
```
Details

This is a very naive version of MIANALYZE, but can produce results for PROC MIANALYZE of SAS.

It simply extracts outputs from hap.

Value

The returned value is a list.

Note

adapted from hap, in fact cline.c and cline.h are not used. keywords utilities

References


See Also

hap

Examples

## Not run:
# 4 SNP example, to generate hap.out and assign.out alone
data(fsnps)
hap(id=fsnps[,1],data=fsnps[,3:10],nloci=4)

# to generate results of imputations
control <- hap.control(ss=1,mi=5)
hap(id=fsnps[,1],data=fsnps[,3:10],nloci=4,control=control)

# to extract information from the second run above
mia(so=1,ns=1,mi=5)
file.show("mia.out")

## commands to check out where the output files are as follows:
## Windows
# system("command.com")
## Unix
# system("csh")

## End(Not run)
Description

The function allows for contrast of genomewide P values from two GWASs. It is conceptually simpler than at the first sight since it involves only one set of chromosomal positions.

Usage

```r
miamiplot(
  x,
  chr = "CHR",
  bp = "BP",
  p = "P",
  pr = "PR",
  snp = "SNP",
  col = c("midnightblue", "chartreuse4"),
  col2 = c("royalblue1", "seagreen1"),
  ymax = NULL,
  highlight = NULL,
  highlight.add = NULL,
  pch = 19,
  cex = 0.75,
  cex.lab = 1,
  xlab = "Chromosome",
  ylab = "-log10(P) [y>0]; log10(P) [y<0]",
  lcols = c("red", "black"),
  lwds = c(5, 2),
  ltys = c(1, 2),
  main = "",
  ...
)
```

Arguments

- **x**: Input data.
- **chr**: Chromosome.
- **bp**: Position.
- **p**: P value.
- **pr**: P value of the other GWAS.
- **snp**: Marker.
- **col**: Colors.
- **col2**: Colors.
- **ymax**: Max y.
**mtdt**  

**Description**  

This function calculates transmission-disequilibrium statistics involving multiallelic marker. Inside the function are tril and triu used to obtain lower and upper triangular matrices.

**Usage**  

```r  
mtdt(x, n.sim=0)  
```

**Arguments**  

- `x`  
  the data table
  
- `n.sim`  
  the number of simulations
Value

It returned list contains the following components:

- SE: Spielman-Ewens Chi-square from the observed data
- ST: Stuart or score Statistic from the observed data
- pSE: the simulated p value
- sSE: standard error of the simulated p value
- pST: the simulated p value
- sST: standard error of the simulated p value

References


Author(s)

Mike Miller, Jing Hua Zhao

See Also

bt

Examples

```r
## Not run:

x <- matrix(c(0,0, 0, 2, 0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0,0, 0, 0, 2, 3, 0, 0, 0, 2,3,26,35, 7,0, 2,10,11, 3, 4, 1, 2,3,22,26, 6,2, 4, 4,10, 2, 2, 0, 0,1, 7,10, 2,0, 0, 2, 2, 1, 1, 0, 0,0, 1, 4, 0,1, 0, 1, 0, 0, 0, 0, 0,2, 5, 4,1,1, 0, 0, 0,2, 0, 0, 0,0, 0, 0, 0, 2,6,1,0, 2, 0, 2, 0, 0, 0, 0, 0,3, 6,19, 6,0, 0, 2, 5, 3, 0, 0, 0,0, 3, 1, 1,0, 0, 0, 1, 0, 0, 0, 0,0, 0, 2,0,0, 0, 0, 0, 0, 0, 0, 0, 0,0, 1, 0, 0,0, 0, 0, 0, 0, 0, 0,nrow=12)

# See note to bt for the score test obtained by SAS
```
mtdt(x)

## End(Not run)

mtdt2  
*Transmission/disequilibrium test of a multiallelic marker by Bradley-Terry model*

**Description**

This function calculates transmission-disequilibrium statistics involving multiallelic marker according to Bradley-Terry model.

**Usage**

```r
mtdt2(x, verbose = TRUE, n.sim = NULL, ...)
```

**Arguments**

- `x`  
  the data table.
- `verbose`  
  To print out test statistics if TRUE.
- `n.sim`  
  Number of simulations.
- `...`  
  other options compatible with the BTm function.

**Value**

It returned list contains the following components:

- `c2b`  
  A data frame in four-column format showing transmitted vs nontransmitted counts.
- `BTm`  
  A fitted Bradley-Terry model object.
- `X2`  
  Allele-wise, genotype-wise and goodness-of-fit Chi-squared statistics.
- `df`  
  Degrees of freedom.
- `p`  
  P value.
- `pn`  
  Monte Carlo p values when n.sim is specified.

**Author(s)**

Jing Hua Zhao keywords models keywords htest
muvar

Means and variances under 1- and 2- locus (biallelic) QTL model

Description

Function muvar() gives means and variances under 1-locus and 2-locus QTL model (simple); in the latter case it gives results from different avenues. This function is included for experimental purpose and yet to be generalized.
Usage

\[
\text{muvar}( \\
\quad \text{n.loci} = 1, \\
\quad y1 = c(0, 1, 1), \\
\quad y12 = c(1, 1, 1, 1, 0, 0, 0, 0), \\
\quad p1 = 0.99, \\
\quad p2 = 0.9 \\
\)
\]

Arguments

- **n.loci**: number of loci, 1=single locus, 2=two loci.
- **y1**: the genotypic means of aa, Aa and AA.
- **y12**: the genotypic means of aa, Aa and AA at the first locus and bb, Bb and BB at the second locus.
- **p1**: the frequency of the lower allele, or the that for the first locus under a 2-locus model.
- **p2**: the frequency of the lower allele at the second locus.

Value

Currently it does not return any value except screen output; the results can be kept via R’s sink() command or via modifying the C/R codes.

Note

Adapted from an earlier C program written for the above book.

Author(s)

Jing Hua Zhao

References


Examples

```r
## Not run:
# the default 1-locus model
muvar(n.loci=1,y1=c(0,1,1),p1=0.5)

# the default 2-locus model
muvar(n.loci=2,y12=c(1,1,1,1,0,0,0,0),p1=0.99,p2=0.9)

## End(Not run)
```
mvmeta

Multivariate meta-analysis based on generalized least squares

Description

This function accepts a data matrix of parameter estimates and their variance-covariance matrix from individual studies and obtain a generalized least squares (GLS) estimate and heterogeneity statistic.

Usage

mvmeta(b, V)

Arguments

b
the parameter estimates.
V
the triangular variance-covariance matrix.

Details

For instance, this would be appropriate for combining linear correlation coefficients of single nucleotide polymorphisms (SNPs) for a given region.

Value

The returned value is a list containing:

d the compact parameter estimates
Psi the compact covariance-covariance matrix
X the design matrix
beta the pooled parameter estimates
cov.beta the pooled variance-covariance matrix
X2 the Chi-squared statistic for heterogeneity
df the degrees(s) of freedom
p the p value

Author(s)

Jing Hua Zhao

References


See Also

metareg
Examples

```
## Not run:
# example 11.3 from Hartung et al.
#
# b <- matrix(c(
#  0.808, 1.308, 1.379, NA, NA, 
#  NA, 1.266, 1.828, 1.962, NA, 
#  NA, 1.835, NA, 2.568, NA, 
#  NA, 1.272, NA, NA, 2.038, 
#  1.171, 2.024, 2.423, 3.159, NA, 
#  0.681, NA, NA, NA, NA),ncol=5, byrow=TRUE)

psi1 <- psi2 <- psi3 <- psi4 <- psi5 <- psi6 <- matrix(0,5,5)
psi1[1,1] <- 0.0985
psi1[1,2] <- 0.0611
psi1[1,3] <- 0.0623
psi1[2,2] <- 0.1142
psi1[2,3] <- 0.0761
psi1[3,3] <- 0.1215

psi2[2,2] <- 0.0713
psi2[2,3] <- 0.0539
psi2[2,4] <- 0.0561
psi2[3,3] <- 0.0938
psi2[3,4] <- 0.0698
psi2[4,4] <- 0.0981

psi3[2,2] <- 0.1228
psi3[2,4] <- 0.1119
psi3[4,4] <- 0.1790

psi4[2,2] <- 0.0562
psi4[2,5] <- 0.0459
psi4[5,5] <- 0.0815

psi5[1,1] <- 0.0895
psi5[1,2] <- 0.0729
psi5[1,3] <- 0.0806
psi5[1,4] <- 0.0950
psi5[2,2] <- 0.1350
psi5[2,3] <- 0.1151
psi5[2,4] <- 0.1394
psi5[3,3] <- 0.1669
psi5[3,4] <- 0.1609
psi5[4,4] <- 0.2381

psi6[1,1] <- 0.0223

V <- rbind(psi1[upper.tri(psi1,diag=TRUE)],psi2[upper.tri(psi2,diag=TRUE)],
psi3[upper.tri(psi3,diag=TRUE)],psi4[upper.tri(psi4,diag=TRUE)],
psi5[upper.tri(psi5,diag=TRUE)],psi6[upper.tri(psi6,diag=TRUE)])
```
Description
This function implements Long et al. (1997) statistics for population-based association design. This is based on a contingency table test and accurate level of significance can be obtained by Fisher’s exact test.

Usage
pbsize(kp, gamma = 4.5, p = 0.15, alpha = 5e-08, beta = 0.2)

Arguments
kp population disease prevalence.
gamma genotype relative risk assuming multiplicative model.
p frequency of disease allele.
alpha type I error rate.
beta type II error rate.

Value
The returned value is scaler containing the required sample size.

Author(s)
Jing Hua Zhao extracted from rm.c.

References

See Also
fbsize
Examples

```r
kp <- c(0.01,0.05,0.10,0.2)
models <- matrix(c(
  4.0, 0.01,
  4.0, 0.10,
  4.0, 0.50,
  4.0, 0.80,
  2.0, 0.01,
  2.0, 0.10,
  2.0, 0.50,
  2.0, 0.80,
  1.5, 0.01,
  1.5, 0.10,
  1.5, 0.50,
  1.5, 0.80), ncol=2, byrow=TRUE)
outfile <- "pbsize.txt"
cat("gamma","p","p1","p5","p10","p20\n",sep="\t",file=outfile)
for(i in 1:dim(models)[1]) {
  g <- models[i,1]
  p <- models[i,2]
  n <- vector()
  for(k in kp) n <- c(n,ceiling(pbsize(k,g,p)))
  cat(models[i,1:2],n,sep="\t",file=outfile,append=TRUE)
  cat("\n",file=outfile,append=TRUE)
}
table5 <- read.table(outfile,header=TRUE,sep="\t")
unlink(outfile)
```

# Alzheimer's disease

g <- 4.5
p <- 0.15
alpha <- 5e-8
beta <- 0.2
zalpha <- qnorm(1-alpha/2)  # 5.45
zbeta <- qnorm(1-beta)
q <- 1-p
pi <- 0.065  # 0.07 and zbeta generate 163
k <- pi*(g*p+q)^2
s <- (1-pi*g^2)*p^2+(1-pi*g)*2*p*q+(1-pi)*q^2
# LGL formula
lambda <- pi*(g^2*p+q-(g*p+q)^2)/(1-pi*(g*p+q)^2)
# mine
lambda <- pi*p*q*(g-1)^2/(1-k)
n <- (zalpha+zbeta)^2/lambda
cat("\nPopulation-based result: Kp =",k, "Kq =",s, "n =",ceiling(n),"\n")
Description

This is a revised version of `pbsize` which is appropriate for a case-control design under a range of disease models. Essentially, for given sample size(s), a proportion of which (fc) being cases, the function calculates power estimate for a given type I error (alpha), genotype relative risk (gamma), frequency of the risk allele (p), the prevalence of disease in the population (kp) and optionally a disease model (model). A major difference would be the consideration of case/control ascertainment in `pbsize`.

Usage

```r
pbsize2(
  N,
  fc = 0.5,
  alpha = 0.05,
  gamma = 4.5,
  p = 0.15,
  kp = 0.1,
  model = "additive"
)
```

Arguments

- **N**: The sample size.
- **fc**: The proportion of cases in the sample.
- **alpha**: Type I error rate.
- **gamma**: The genotype relative risk (GRR).
- **p**: Frequency of the risk allele.
- **kp**: The prevalence of disease in the population.
- **model**: Disease model, i.e., "multiplicative", "additive", "dominant", "recessive", "overdominant".

Details

Internally, the function obtains a baseline risk to make the disease model consistent with kp as in `tscc` and should produce accurate power estimate. Note it provides power estimates for given sample size(s) only.

Value

The returned value is the power for the specified design.

Note

Why is the comparison with power.casectrl so bad?

Author(s)

Jing Hua Zhao
See Also

The design follows that of `pbsize`.

Examples

```r
## Not run:
# single calc
m <- c("multiplicative","recessive","dominant","additive","overdominant")
for(i in 1:5) print(pbsize2(N=50, alpha=5e-2, gamma=1.1, p=0.1, kp=0.1, model=m[i]))

# for a range of sample sizes
pbsize2(p=0.1, N=c(25,50,100,200,500), gamma=1.1, kp=0.1, alpha=5e-2, model='r')

# create a power table
f <- function(p)
  pbsize2(p=p, N=seq(100,1000,by=100), gamma=1.1, kp=0.1, alpha=5e-2, model='recessive')
m <- sapply(X=seq(0.1,0.9, by=0.1), f)
colnames(m) <- seq(0.1,0.9, by=0.1)
rownames(m) <- seq(100,1000,by=100)
print(round(m,2))

library(genetics)
m <- c("multiplicative","recessive","dominant","partialrecessive")
for(i in 1:4) print(power.casectrl(p=0.1, N=50, gamma=1.1, kp=0.1, alpha=5e-2, minh=m[i]))
power.casectrl(p=0.1, N=c(25,50,100,200,500), gamma=1.1, kp=0.1, alpha=5e-2, minh='r')
f <- function(p)
  power.casectrl(p=p, N=seq(100,1000,by=100), gamma=1.1, kp=0.1, alpha=5e-2, minh='recessive')
m <- sapply(X=seq(0.1,0.9, by=0.1), f)
colnames(m) <- seq(0.1,0.9, by=0.1)
rownames(m) <- seq(100,1000,by=100)
print(round(m,2))

## End(Not run)
```

---

pedtodot

*Converting pedigree(s) to dot file(s)*

**Description**

This function converts GAS or LINKAGE formatted pedigree(s) into .dot file for each pedigree to be used by dot in graphviz, which is a flexible package for graphics freely available.
Usage

```r
pedtodot(
  pedfile,
  makeped = FALSE,
  sink = TRUE,
  page = "B5",
  url = "https://jinghuazhao.github.io/",
  height = 0.5,
  width = 0.75,
  rotate = 0,
  dir = "none"
)
```

Arguments

- **pedfile**: a pedigree file in GAS or LINKAGE format, note if individual's ID is character then it is necessary to specify as.is=T in the read.table command.
- **makeped**: a logical variable indicating if the pedigree file is post-makeped.
- **sink**: a logical variable indicating if .dot file(s) are created.
- **page**: a string indicating the page size, e.g., A4, A5, B5, Legal, Letter, Executive, "x,y", where x, y is the customized page size.
- **url**: Unified Resource Locator (URL) associated with the diagram(s).
- **height**: the height of node(s).
- **width**: the width of node(s).
- **rotate**: if set to 90, the diagram is in landscape.
- **dir**: direction of edges, i.e., "none", "forward","back","both". This will be useful if the diagram is viewed by inetera.

Details

Note that a single PostScript (PDF) file can be obtained by dot, fdp, or neato.

```
  dot -Tps <dot file> -o <ps file>
```
or

```
  fdp -Tps <dot file> -o <ps file>
```
or

```
  neato -Tps <dot file> -o <ps file>
```
See relevant documentations for other formats.

To preserve the original order of pedigree(s) in the data, you can examine the examples at the end of this document.

Under Cygwin/Linux/Unix, the PostScript file can be converted to Portable Document Format (PDF) default to Acrobat.

```
  ps2pdf <ps file>
```
Use ps2pdf12, ps2pdf13, or ps2pdf14 for appropriate versions of Acrobat according to information given on the headline of <ps file>.

Under Linux, you can also visualize the .dot file directly via command,
dotty <dot file> &

We can extract the code below (or within pedtodot.Rd) to pedtodot and then use command:
sh pedtodot <pedigree file>

Value

For each pedigree, the function generates a .dot file to be used by dot. The collection of all pedigrees (*.dot) can also be put together.

Note

This is based on the gawk script program pedtodot by David Duffy with minor changes.

Author(s)

David Duffy, Jing Hua Zhao

See Also


Examples

```r
## Not run:
# example as in R News and Bioinformatics (see also plot.pedigree in package kinship)
# it works from screen paste only
p1 <- scan(nlines=16,what=list(0,0,0,0,0,""))
  1  2  3  2  2  7/7  7/10
  2  0  0  1  1  -/-  -/-
  3  0  0  2  2  7/9  3/10
  4  2  3  2  2  7/9  3/7
  5  2  3  2  1  7/7  7/10
  6  2  3  1  1  7/7  7/10
  7  2  3  2  1  7/7  7/10
  8  0  0  1  1  -/-  -/-
  9  8  4  1  1  7/9  3/10
 10  0  0  2  1  -/-  -/-
 11  2 10  2  1  7/7  7/7
 12  2 10  2  2  6/7  7/7
 13  0  0  1  1  -/-  -/-
 14 13 11  1  1  7/8  7/8
 15  0  0  1  1  -/-  -/-
 16 15 12  2  1  6/6  7/7
p2 <- as.data.frame(p1)
names(p2) <-c("id","fid","mid","sex","aff","GABRB1","D4S1645")
p3 <- data.frame(pid=10081,p2)
```
# Three examples of pedigree-drawing
# assuming pre-MakePed LINKAGE file in which IDs are characters
pre<-read.table("pheno.pre",as.is=TRUE)[,1:6]
pedtodot(pre)
dir()
# for post-MakePed LINKAGE file in which IDs are integers
ped <- read.table("pheno.ped")[,1:10]
pedtodot(ped,makeped=TRUE)
dir()
# for a single file with a list of pedigrees ordered data
sink("gawl4.dot")
pedtodot(ped,sink=FALSE)
sink()
file.show("gawl4.dot")
# more details
pedtodot(ped,sink=FALSE,page="B5",url="https://jinghuazhao.github.io/"

# An example from Richard Mott and in the demo
filespec <- system.file("tests/ped.1.3.pre")
pre <- read.table(filespec,as.is=TRUE)
pre
pedtodot(pre,dir="forward")

## End(Not run)

---

**pfc**

*Probability of familial clustering of disease*

**Description**

To calculate exact probability of familial clustering of disease

**Usage**

```r
pfc(famdata, enum = 0)
```

**Arguments**

- `famdata` collective information of sib size, number of affected sibs and their frequencies.
- `enum` a switch taking value 1 if all possible tables are to be enumerated.

**Value**

The returned value is a list containing (tailp,sump,enum) are only available if enum=1:

- `p` the probability of familial clustering
stat the deviances, chi-squares based on binomial and hypergeometric distributions, the degrees of freedom should take into account the number of marginals used

tailp the exact statistical significance

sump sum of the probabilities used for error checking

nenum the total number of tables enumerated

Note
Adapted from family.for by Dani Zelterman, 25/7/03

Author(s)
Dani Zelterman, Jing Hua Zhao

References

See Also
kin.morgan

Examples

## Not run:
# IPF among 203 siblings of 100 COPD patients from Liang KY, SL Zeger, # Qaquis B. Multivariate regression analyses for categorical data # (with discussion). J Roy Stat Soc B 1992, 54:3-40

# the degrees of freedom is 15
famtest<-c(
1, 0, 36,
1, 1, 12,
2, 0, 15,
2, 1, 7,
2, 2, 1,
3, 0, 5,
3, 1, 7,
3, 2, 3,
3, 3, 2,
4, 0, 3,
4, 1, 3,
4, 2, 1,
6, 0, 1,
6, 2, 1,
6, 3, 1,
6, 4, 1,
6, 6, 1)
test<-t(matrix(famtest,nrow=3))
famp<-pfc(test)

## End(Not run)

---

### pfc.sim

#### Probability of familial clustering of disease

**Description**

To calculate probability of familial clustering of disease using Monte Carlo simulation.

**Usage**

```r
pfc.sim(famdata, n.sim = 1e+06, n.loop = 1)
```

**Arguments**

- `famdata`: collective information of sib size, number of affected sibs and their frequencies.
- `n.sim`: number of simulations in a single Monte Carlo run.
- `n.loop`: total number of Monte Carlo runs.

**Value**

The returned value is a list containing:

- `n.sim`: a copy of the number of simulations in a single Monte Carlo run.
- `n.loop`: the total number of Monte Carlo runs.
- `p`: the observed p value.
- `tailpl`: accumulated probabilities at the lower tails.
- `tailpu`: simulated p values.

**Note**

Adapted from runi.for from Change Yu, 5/6/4

**Author(s)**

Chang Yu, Dani Zelterman

**References**

## Not run:

```r
# Li FP, Fraumeni JF Jr, Mulvihill JJ, Blattner WA, Dreyfus MG, Tucker MA,
# Miller RW. A cancer family syndrome in twenty-four kindreds.

# family_size  #_of_affected frequency

famtest<-c(1, 0, 2, 
           1, 1, 0, 
           2, 0, 1, 
           2, 1, 4, 
           2, 2, 3, 
           3, 0, 0, 
           3, 1, 2, 
           3, 2, 1, 
           3, 3, 1, 
           4, 0, 0, 
           4, 1, 2, 
           5, 0, 0, 
           5, 1, 1, 
           6, 0, 0, 
           6, 1, 1, 
           7, 0, 0, 
           7, 1, 1, 
           8, 0, 0, 
           8, 1, 1, 
           8, 2, 1, 
           8, 3, 1, 
           9, 3, 1)

test<-matrix(famtest,byrow=T,ncol=3)
famp<-pfc.sim(test)
```

## End(Not run)

---

### Description

This function is a R port of the GENECOUNTING/prepare program which takes an array of genotypic data and collapses individuals with the same multilocus genotype. This function can also be used to prepare for the genotype table in testing Hardy-Weinberg equilibrium.
Usage

pgc(data, handle.miss = 1, is.genotype = 0, with.id = 0)

Arguments

data the multilocus genotype data for a set of individuals.
handle.miss a flag to indicate if missing data is kept, 0 = no, 1 = yes.
is.genotype a flag to indicate if the data is already in the form of genotype identifiers.
with.id a flag to indicate if the unique multilocus genotype identifier is generated.

Value

The returned value is a list containing:
cdata the collapsed genotype data
wt the frequency weight
obscom the observed number of combinations or genotypes
idsave optional, available only if with.id = 1

Note

Built on pgc.c.

Author(s)

Jing Hua Zhao

References

Zhao JH, Sham PC (2003). Generic number system and haplotype analysis. Comp Prog Meth Biomed 70:1-9

See Also

genecounting, hwe.hardy

Examples

## Not run:
require(gap.datasets)
data(hla)
x <- hla[,3:8]

# do not handle missing data
y<-pgc(x,handle.miss=0,with.id=1)

hla.gc<-genecounting(y$data,y$wt,handle.miss=0)

# handle missing but with multilocus genotype identifier
pgc(x,handle.miss=1,with.id=1)
plot.hap.score

# handle missing data with no identifier
pgc(x, handle.miss=1, with.id=0)

## End(Not run)

plot.hap.score  

Plot haplotype frequencies versus haplotype score statistics

Description

Method function to plot a class of type hap.score

Usage

## S3 method for class 'hap.score'
plot(x, ...)

Arguments

x  The object returned from hap.score (which has class hap.score).
...
Optional arguments

Details

This is a plot method function used to plot haplotype frequencies on the x-axis and haplotype-specific scores on the y-axis. Because hap.score is a class, the generic plot function can be used, which in turn calls this plot.hap.score function.

Value

Nothing is returned.

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA (2002) Score tests for association of traits with haplotypes when linkage phase is ambiguous. Amer J Hum Genet 70:425-34

See Also

hap.score
Examples

```r
## Not run:
save <- hap.score(y, geno, trait.type = "gaussian")

# Example illustrating generic plot function:
plot(save)

# Example illustrating specific method plot function:
plot.hap.score(save)

## End(Not run)
```

---

**pqtl2dplot**

---

## Description

2D pQTL plot

## Usage

```r
pqtl2dplot(
  d,              # Data to be used.
  chrlen = gap::hg19,    # lengths of chromosomes for specific build: hg18, hg19, hg38.
  snp_name = "SNP",  # variant name.
  snp_chr = "Chr",    # variant chromosome.
  snp_pos = "bp",     # variant position.
  gene_chr = "p.chr", # gene chromosome.
  gene_start = "p.start", # gene start.
  gene_end = "p.end", # gene end.
  protein = "p.target.short", # protein.
  lp = "log10p",      # log10(p).
  cis = "cis",        # cis common.
  plot = TRUE,        # plot.
  cex = 0.6           # character expansion.
)
```

## Arguments

- `d`: Data to be used.
- `chrlen`: lengths of chromosomes for specific build: hg18, hg19, hg38.
- `snp_name`: variant name.
- `snp_chr`: variant chromosome.
- `snp_pos`: variant position.
- `gene_chr`: gene chromosome.
pqt12dplotly

**Description**

2D pQTL plotly

**Usage**

```r
pqt12dplotly(d, chrlen = gap::hg19)
```

**Arguments**

- **d**: Data in pqt12dplot() format.
- **chrlen**: Lengths of chromosomes for specific build: hg18, hg19, hg38.

**Value**

A plotly figure.

---

```
## Not run:
INF <- Sys.getenv("INF")
d <- read.csv(file.path(INF,"work","INF1.merge.cis.vs.trans"),as.is=TRUE)
r <- pqt12dplot(d)
## End(Not run)
```
Examples

```r
## Not run:
INF <- Sys.getenv("INF")
d <- read.csv(file.path(INF,"work","INF1.merge.cis.vs.trans"),as.is=TRUE)
r <- pqtl3dplotly(d)
htmlwidgets::saveWidget(r,file=file.path(INF,"INF1.pqtl3dplotly.html"))
r
## End(Not run)
```

pqtl3dplotly 3D pQTL plot

Description

3D pQTL plot

Usage

`pqtl3dplotly(d, chrlen = gap::hg19, zmax = 300)`

Arguments

- `d` Data in `pqtl2d()` format.
- `chrlen` Lengths of chromosomes for specific build: hg18, hg19, hg38.
- `zmax` Maximum -log10p to truncate, above which they would be set to this value.

Value

A plotly figure.

Examples

```r
## Not run:
INF <- Sys.getenv("INF")
d <- read.csv(file.path(INF,"work","INF1.merge.cis.vs.trans"),as.is=TRUE)
r <- pqtl3dplotly(d, chrlen = gap::hg19, zmax = 300)
htmlwidgets::saveWidget(r,file=file.path(INF,"INF1.pqtl3dplotly.html"))
r
## End(Not run)
```
print.hap.score  Print a hap.score object

Description
Method function to print a class of type hap.score

Usage
## S3 method for class 'hap.score'
print(x, ...)

Arguments
x The object returned from hap.score (which has class hap.score).
... Optional arguments.

Details
This is a print method function used to print information from hap.score class, with haplotype-specific information given in a table. Because hap.score is a class, the generic print function can be used, which in turn calls this print.hap.score function.

Value
Nothing is returned.

References
Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA (2002) Score tests for association of traits with haplotypes when linkage phase is ambiguous. Amer J Hum Genet 70:425-34

See Also
hap.score

Examples
## Not run:
save <- hap.score(y, geno, trait.type = "gaussian")

# Example illustrating generic print function:
print(save)

# Example illustrating specific method print function:
print.hap.score(save)

## End(Not run)
pvalue  

P value for a normal deviate

Description

P value for a normal deviate

Usage

pvalue(z, decimals = 2)

Arguments

z normal deviate.

decimals number of decimal places.

Value

P value as a string variable.

Examples

pvalue(-1.96)

qqfun  

Quantile-comparison plots

Description

Plots empirical quantiles of a variable against theoretical quantiles of a comparison distribution.

Usage

qqfun(
  x,
  distribution = "norm",
  ylab = deparse(substitute(x)),
  xlab = paste(distribution, "quantiles"),
  main = NULL,
  las = par("las"),
  envelope = 0.95,
  labels = FALSE,
  col = palette()[4],
  lcol = palette()[2],
  xlim = NULL,
  ylim = NULL,
lwd = 1,
pch = 1,
bg = palette()[4],
cex = 0.4,
line = c("quartiles", "robust", "none"),
...
)

Arguments

x vector of numeric values.
distribution root name of comparison distribution – e.g., norm for the normal distribution; t for the t-distribution.
ylab label for vertical (empirical quantiles) axis.
xlab label for horizontal (comparison quantiles) axis.
main label for plot.
las if 0, ticks labels are drawn parallel to the axis; set to 1 for horizontal labels (see par).
envelope confidence level for point-wise confidence envelope, or FALSE for no envelope.
lables vector of point labels for interactive point identification, or FALSE for no labels.
col color for points; the default is the fourth entry in the current color palette (see palette and par).
lcol color for lines; the default is the second entry as above.
xlim the x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a reversed axis.
ylim the y limits of the plot.
lwd line width; default is 1 (see par). Confidence envelopes are drawn at half this line width.
pch plotting character for points; default is 1 (a circle, see par).
bg background color of points.
cex factor for expanding the size of plotted symbols; the default is .4.
line "quartiles" to pass a line through the quartile-pairs, or "robust" for a robust-regression line; the latter uses the rlm function in the MASS package. Specifying line = "none" suppresses the line.
...
arguments such as df to be passed to the appropriate quantile function.

details

Draws theoretical quantile-comparison plots for variables and for studentized residuals from a linear model. A comparison line is drawn on the plot either through the quartiles of the two distributions, or by robust regression.

Any distribution for which quantile and density functions exist in R (with prefixes q and d, respectively) may be used. Studentized residuals are plotted against the appropriate t-distribution.

This is adapted from qq.plot of package car with different values for points and lines, more options, more transparent code and examples in the current setting. Another similar but sophisticated function is qqmath of package lattice.
Value

NULL. These functions are used only for their side effect (to make a graph).

Author(s)

John Fox, Jing Hua Zhao

References


See Also

qqnorm, qqunif, gcontrol2

Examples

```r
## Not run:
p <- runif(100)
alpha <- 1/log(10)
qqfun(p,dist="unif")
qqfun(-log10(p),dist="exp",rate=alpha,pch=21)
#library(car)
#qq.plot(p,dist="unif")
#qq.plot(-log10(p),dist="exp",rate=alpha)

#library(lattice)
#qqmath(~ -log10(p), distribution = function(p) qexp(p,rate=alpha))

## End(Not run)
```

qunif

*Q-Q plot for uniformly distributed random variable*

Description

This function produces Q-Q plot for a random variable following uniform distribution with or without using log-scale. Note that the log-scale is by default for type "exp", which is a plot based on exponential order statistics. This appears to be more appropriate than the commonly used procedure whereby the expected value of uniform order statistics is directly log-transformed.
qqunif

Usage

qqunif(
u,  
type = "unif",  
logscale = TRUE,  
base = 10,  
col = palette()[4],  
lcol = palette()[2],  
Ci = FALSE,  
alpha = 0.05,  
...  
)

Arguments

  u  
a vector of uniformly distributed random variables.
  type  
string option to specify distribution: "unif"=uniform, "exp"=exponential.
  logscale  
to use logscale.
  base  
the base of the log function.
  col  
color for points.
  lcol  
color for the diagonal line.
  ci  
logical option to show confidence interval.
  alpha  
1-confidence level, e.g., 0.05.
  ...  
other options as appropriate for the qqplot function.

Value

The returned value is a list with components of a qqplot:

  x  expected value for uniform order statistics or its -log(base) counterpart
  y  observed value or its -log(base) counterpart

Author(s)

Jing Hua Zhao

References


See Also

qqfun
Examples

## Not run:
# Q-Q Plot for 1000 U(0,1) r.v., marking those <= 1e-5
u_obs <- runif(1000)
r <- qqunif(u_obs,pch=21,bg="blue",bty="n")
u_exp <- r$y
hits <- u_exp >= 2.30103
points(r$x[hits],u_exp[hits],pch=21,bg="green")
legend("topleft",sprintf("GC.lambda="
## End(Not run)

---

**read.ms.output**

*A utility function to read ms output*

Description

This function reads in the output of the program ms, a program to generate samples under a variety of neutral models.

Usage

```r
read.ms.output(
  msout,
  is.file = TRUE,
  xpose = TRUE,
  verbose = TRUE,
  outfile = NULL,
  outfileonly = FALSE
)
```

Arguments

- **msout**: an ms output.
- **is.file**: a flag indicating ms output as a system file or an R object.
- **xpose**: a flag to obtain the tranposed format as it is (when TRUE).
- **verbose**: when TRUE, display on screen every 1000 for large nsam.
- **outfile**: to save the haplotypes in a tab-delimited ASCII file.
- **outfileonly**: to reset gametes to NA when nsam/nreps is very large and is useful with outfile.

The returned value is a list storing the results.

**call**: system call to ms

**seed**: random number seed to ms

**nsam**: number of copies of the locus in each sample

**nreps**: the number of independent samples to generate
revStrand

**segsites** a vector of the numbers of segregating sites
**times** vectors of time to most recent ancestor (TMRCA) and total tree lengths
**positions** positions of polymorphic sites on a scale of (0,1)
**gametes** a list of haplotype arrays
**probs** the probability of the specified number of segregating sites given the genealogical history of the sample and the value to \(-t\) option

### Details

The argument indicates either a file name or a vector of character strings, one string for each line of the output of ms. As with the second case, it is appropriate with system(,intern=TRUE), see example below.

### Author(s)

D Davison, RR Hudson, JH Zhao

### References


### Examples

```r
## Not run:
# Assuming ms is on the path

system("ms 5 4 -s 5 > ms.out")
msout1 <- read.ms.output("ms.out")

system("ms 50 4 -s 5 > ms.out")
msout2 <- read.ms.output("ms.out",outfile="out",outfileonly=TRUE)

msout <- system("ms 5 4 -s 5 -L", intern=TRUE)
msout3 <- read.ms.output(msout,FALSE)

## End(Not run)
```

### Description

The function obtains allele on the reverse strand.
Usage

revStrand(allele)

Arguments

allele Allele to reverse.

Value

Allele on the reverse strand.

Examples

## Not run:
alleles <- c("a", "c", "G", "t")
reverse_strand(alleles)

## End(Not run)

Description

This function starts the interactive 'shinygap' shiny web application that allows for flexible model specification.

Usage

runshinygap(...)

Arguments

... Additional arguments passed to the 'runApp' function from the 'shiny' package.

Details

The 'shiny' based web application allows for flexible model specification for the implemented study designs.

Value

These are design specific.
Description

This function calculates one-to-others and maximum accumulated chi-squared statistics for a 2 by K contingency table.

Usage

\[ s2k(y1, y2) \]

Arguments

- **y1**: a vector containing the first row of a 2 by K contingency table.
- **y2**: a vector containing the second row of a 2 by K contingency table.

Value

The returned value is a list containing:

- **x2a**: the one-to-other chi-square.
- **x2b**: the maximum accumulated chi-square.
- **col1**: the column index for x2a.
- **col2**: the column index for x2b.
- **p**: the corresponding p value.

Note

The lengths of y1 and y2 should be the same.

Author(s)

Chihiro Hirotsu, Jing Hua Zhao

References

## Not run:
# an example from Mike Neale
# termed 'ugly' contingency table by Patrick Sullivan
y1 <- c(2,15,16,35,132,30,25,7,12,24,10,10,0)
y2 <- c(0, 6,31,49,120,27,15,8,14,25, 3, 9,3)

result <- s2k(y1,y2)

## End(Not run)

sentinels

Sentinel identification from GWAS summary statistics

Description

This function accepts an object containing GWAS summary statistics for signal identification as defined by flanking regions. As the associate P value could be extremely small, the effect size and its standard error are used.

Usage

sentinels(
  p,
  pid,
  st,
  debug = FALSE,
  flanking = 1e+06,
  chr = "Chrom",
  pos = "End",
  b = "Effect",
  se = "StdErr",
  log_p = NULL,
  snp = "MarkerName",
  sep = ","
)

Arguments

- **p**: an object containing GWAS summary statistics.
- **pid**: a phenotype (e.g., protein) name in pGWAS.
- **st**: row number as in p.
- **debug**: a flag to show the actual data.
- **flanking**: the width of flanking region.
- **chr**: Chromosome name.
- **pos**: Position.
Details

A distance-based approach was consequently used and reframed as an algorithm here. It takes as input signals multiple correlated variants in particular region(s) which reach genomewide significance and output three types of sentinels in a region-based manner. For a given protein and a chromosome, the algorithm proceeds as follows:

Algorithm sentinels

Step 1. for a particular collection of genomewide significant variants on a chromosome, the width of the region is calculated according to the start and end chromosomal positions and if it is smaller than the flanking distance, the variant with the smallest P value is taken as sentinel (I) otherwise goes to step 2.

Step 2. The variant at step 1 is only a candidate and a flanking region is generated. If such a region contains no variant the candidate is recorded as sentinel (II) and a new iteration starts from the variant next to the flanking region.

Step 3. When the flanking is possible at step 2 but the P value is still larger than the candidate at step 2, the candidate is again recorded as sentinel (III) but next iteration starts from the variant just after the variant at the end position; otherwise the variant is updated as a new candidate where the next iteration starts.

Note Type I signals are often seen from variants in strong LD at a cis region, type II results seen when a chromosome contains two trans signals, type III results seen if there are multiple trans signals.

Typically, input to the function are variants reaching certain level of significance and the function identifies minimum p value at the flanking interval; in the case of another variant in the flanking window has smaller p value it will be used instead.

For now key variables in p are "MarkerName", "End", "Effect", "StdErr", "P.value", where "End" is as in a bed file indicating marker position, and the function is set up such that row names are numbered as 1:nrow(p); see example below. When log_p is specified, log(P) is used instead, which is appropriate with output from METAL with LOGPVALUE ON. In this case, the column named log(P) in the output is actually log10(P).

Value

The function give screen output.

Examples

```r
## Not run:
## OPG as a positive control in our pGWAS
require(gap.datasets)
data(OPG)
```
p <- reshape::rename(OPGtbl, c(Chromosome="Chrom", Position="End"))
chrs <- with(p, unique(Chrom))
for(chr in chrs)
{
  ps <- subset(p[c("Chrom","End","MarkerName","Effect","StdErr")], Chrom==chr)
  row.names(ps) <- 1:nrow(ps)
  sentinels(ps, "OPG", 1)
}
subset(OPGrsid,MarkerName=="chr8:120081031_C_T")
subset(OPGrsid,MarkerName=="chr17:26694861_A_G")
## log(P)
p <- within(p, {logp <- log(P.value)})
for(chr in chrs)
{
  ps <- subset(p[c("Chrom","End","MarkerName","logp")], Chrom==chr)
  row.names(ps) <- 1:nrow(ps)
  sentinels(ps, "OPG", 1, log_p="logp")
}
## to obtain variance explained
tbl <- within(OPGtbl, chi2n <- (Effect/StdErr)^2/N)
s <- with(tbl, aggregate(chi2n,list(prot),sum))
names(s) <- c("prot", "h2")
sd <- with(tbl, aggregate(chi2n,list(prot),sd))
names(sd) <- c("p1","sd")
m <- with(tbl, aggregate(chi2n,list(prot),length))
names(m) <- c("p2","m")
h2 <- cbind(s,sd,m)
ord <- with(h2, order(h2))
sink("h2.dat")
print(h2[ord,c("prot","h2","sd","m")], row.names=FALSE)
sink()
png("h2.png", res=300, units="in", width=12, height=8)
np <- nrow(h2)
with(h2[ord,], {
  plot(h2, cex=0.4, pch=16, xaxt="n", xlab="protein", ylab=expression(h^2))
  xtick <- seq(1, np, by=1)
  axis(side=1, at=xtick, labels = FALSE)
  text(x=xtick, par("usr")[3],labels = prot, srt = 75, pos = 1, xpd = TRUE, cex=0.5)
})
dev.off()
write.csv(tbl,file="INF1.csv",quote=FALSE,row.names=FALSE)

## End(Not run)
Description

Eventually, this will be a set of functions specifically for single nucleotide polymorphisms (SNPs), which are biallelic markers. This is particularly relevant to the genomewide association studies (GWAS) using GeneChips and in line with the classic generalised single-locus model. snp.HWE is from Abecasis’s website and yet to be adapted for chromosome X.

Usage

snp.ES(beta, SE, N)

snp.HWE(g)

PARn(p, RRlist)

snp.PAR(RR, MAF, unit = 2)

Arguments

beta Regression coefficient.
SE Standard error for beta.
N Sample size.
g Observed genotype vector.
p genotype frequencies.
RRlist A list of RRs.
RR Relative risk.
MAF Minor allele frequency.
unit Unit to exponentiate for homozygote.

Details

snp.ES provides effect size estimates based on the linear regression coefficient and standard error. For logistic regression, we can have similar idea for \log(OR) and \log(\text{SE}(OR)).

snp.HWE gives an exact Hardy-Weinberg Equilibrium (HWE) test and it return -1 in the case of misspecification of genotype counts.

snp.PAR calculates the the population attributable risk (PAR) for a particular SNP. Internally, it calls for an internal function PARn, given a set of frequencies and associate relative risks (RR). Other 2x2 table statistics familiar to epidemiologists can be added when necessary.

Author(s)

Jing Hua Zhao, Shengxu Li
**Description**
A utility to generate SNPTEST sample file

**Usage**
```
snptest_sample(  
data,
sample_file = "snptest.sample",
ID_1 = "ID_1",
ID_2 = "ID_2",
missing = "missing",
C = NULL,
D = NULL,
P = NULL  
)
```

**Arguments**
- **data**: Data to be used.
- **sample_file**: Output filename.
- **ID_1**: ID_1 as in the sample file.
- **ID_2**: ID_2 as in the sample file.
- **missing**: Missing data column.
- **C**: Continuous variables.
- **D**: Discrete variables.
- **P**: Phenotypic variables.

**Value**
Output file in SNPTEST’s sample format.

**Examples**
```r
## Not run:
d <- data.frame(ID_1=1, ID_2=1, missing=0, PC1=1, PC2=2, D1=1, P1=10)
snptest_sample(d, C=paste0("PC",1:2), D=paste0("D",1:1), P=paste0("P",1:1))
## End(Not run)
```
Power calculation for two-stage case-control design

Description

This function gives power estimates for two-stage case-control design for genetic association.

Usage

```
tsc(model, GRR, p1, n1, n2, M, alpha.genome, pi.samples, pi.markers, K)
```

Arguments

- `model`: any in c("multiplicative","additive","dominant","recessive").
- `GRR`: genotype relative risk.
- `p1`: the estimated risk allele frequency in cases.
- `n1`: total number of cases.
- `n2`: total number of controls.
- `M`: total number of markers.
- `alpha.genome`: false positive rate at genome level.
- `pi.samples`: sample% to be genotyped at stage 1.
- `pi.markers`: markers% to be selected (also used as the false positive rate at stage 1).
- `K`: the population prevalence.

Details

The false positive rates are calculated as follows,

\[
P(|z_1| > C_1)P(|z_2| > C_2, \text{sign}(z_1) = \text{sign}(z_2))
\]

and

\[
P(|z_1| > C_1)P(|z_j| > C_j ||z_1| > C_1)
\]

for replication-based and joint analyses, respectively; where C1, C2, and Cj are thresholds at stages 1, 2 replication and joint analysis,

\[
z_1 = z(p1, p2, n1, n2, pi.samples)
\]

\[
z_2 = z(p1, p2, n1, n2, 1 - pi.samples)
\]

\[
z_j = \sqrt{\pi.samples} \star z1 + \sqrt{1 - pi.samples} \star z2
\]
Value

The returned value is a list containing a copy of the input plus output as follows,

- **model** any in c("multiplicative","additive","dominant","recessive").
- **GRR** genotype relative risk.
- **p1** the estimated risk allele frequency in cases.
- **pprime** expected risk allele frequency in cases.
- **p** expected risk allele frequency in controls.
- **n1** total number of cases.
- **n2** total number of controls.
- **M** total number of markers.
- **alpha.genome** false positive rate at genome level.
- **pi.samples** sample% to be genotyped at stage 1.
- **pi.markers** markers% to be selected (also used as the false positive rate at stage 1).
- **K** the population prevalence.
- **C** thresholds for no stage, stage 1, stage 2, joint analysis.
- **power** power corresponding to C.

Note

solve.skol is adapted from CaTS.

Author(s)

Jing Hua Zhao

References


Examples

```r
## Not run:
K <- 0.1
p1 <- 0.4
n1 <- 1000
n2 <- 1000
M <- 300000
alpha.genome <- 0.05
GRR <- 1.4
p1 <- 0.4
pi.samples <- 0.2
pi.markers <- 0.1

options(echo=FALSE)
```
whscore

```r
cat("sample\n for(GRR in c(1.3,1.35,1.40))
{
  cat("\n")
  for(pi.samples in c(1.0,0.5,0.4,0.3,0.2))
  {
    if(pi.samples==1.0) s <- 1.0
    else s <- c(0.1,0.05,0.01)
    for(pi.markers in s)
    {
      x <- tscc("multiplicative",GRR,p1,n1,n2,M,alpha.genome,
      pi.samples,pi.markers,K)
      l <- c(pi.samples,pi.markers,GRR,x$C,x$power)
      l <- sprintf("\n      l[1],l[2],l[3],l[4],l[5],l[6],l[7],l[8],l[9],l[10],l[11])
      cat(l,"\n")
    }
    cat("\n")
  }
} options(echo=TRUE)
```

## End(Not run)

### whscore

*Whittemore-Halpern scores for allele-sharing*

**Description**

Allele sharing score statistics.

**Usage**

```r
whscore(allele, type)
```

**Arguments**

- `allele`: a matrix of alleles of affected pedigree members.
- `type`: 0 = pairs, 1 = all.

**Value**

The returned value is the value of score statistic.

**Note**

adapted from GENEHUNTER.
Author(s)

Leonid Kruglyak, Jing Hua Zhao

References


Examples

```r
## Not run:
c <- matrix(c(1,1,1,2,2,2),ncol=2)
whscore(c,type=1)
whscore(c,type=2)
## End(Not run)
```
Index

* datagen
  b2r, 9
  gif, 42
  kin.morgan, 66
  mvmeta, 102
* distribution
  METAL_forestplot, 82
  qqfun, 120
  qqunif, 122
* dplot
  pedtodor, 107
* hplot
  mhtplot.trunc, 89
* hplot
  asplot, 7
  ESplot, 25
  METAL_forestplot, 82
  mhtplot, 87
  mhtplot2, 92
  plot.hap.score, 115
  qqunif, 122
* htest
  ab, 4
  AE3, 5
  chow.test, 17
  comp.score, 22
  gcp, 36
  h2.mzdz, 48
  hwe, 58
  hwe.cc, 59
  hwe.jags, 63
  klem, 67
  MCMCgrm, 80
  metap, 83
* misc
  fbsize, 26
  masize, 75
  pbsize, 104
  pbsize2, 105
  tscc, 133
* models
  AE3, 5
  BFDP, 10
  bt, 12
  FPRP, 28
  gc.em, 30
  gcontrol, 33
  gcontrol2, 34
  gcp, 36
  genecounting, 37
  hap, 50
  hap.em, 52
  hap.score, 54
  LD22, 69
  LDkl, 70
  metareg, 85
  mtdt, 97
  muvar, 100
  pfc, 110
  pfc.sim, 112
  s2k, 127
* print
  print.hap.score, 119
* regression
  hap.score, 54
  htr, 56
  qqfun, 120
* univar
  qqfun, 120
  qqunif, 122
* utilities
  pgc, 113
  read.ms.output, 124
  sentinels, 128
  snp.ES, 130
  whscore, 135
* utilities
  muvar, 100
ab, 4, 79
AE3, 5
asplot, 7
b2r, 9
BFDP, 10, 29
bt, 12, 98
ccsize, 5, 14
chow.test, 17
cirros.cis.vs.trans.plot, 19
cirros.cnvplot, 20
cirros.mhtplot, 20
cis.vs.trans.classification, 21
cnvplot, 22
comp.score, 22
cs, 24
ESplot, 25
fbsize, 26, 104
FPRP, 11, 28
gc.em, 30, 39, 69
gc.lambda, 32
gcontrol, 33
gcontrol2, 34, 122
gcp, 36
geneCounting, 31, 32, 37, 37, 51, 68, 69, 114
genotype, 62
get_b_se, 40
get_pve_se, 41
get_sdy, 41
gif, 42, 67
gsmr, 44
h2.jags, 45
h2.mdzdz, 48
h2G, 46
h2GE, 47
h21, 47
hap, 50, 53, 95
hap.em, 52
hap.score, 54, 57, 115, 119
htr, 18, 56
hwe, 58, 60, 62
hwe.cc, 59
hwe.hardy, 59, 61, 64, 114
hwe.jags, 63
HWE.test, 62
invnormal, 66
kin.morgan, 66, 111
klem, 67
LD22, 9, 69, 72
LDkl, 32, 39, 53, 70, 70
log10p, 72
log10pvalue, 73
logp, 73
makeped, 74
masize, 75
MCMCgrm, 80
METAL_forestplot, 82, 83
metap, 83
metareg, 84, 85, 102
mhtplot, 87, 91, 93
mhtplot.trunc, 89
mhtplot2, 92
mia, 94
miamiplot, 96
mttd, 13, 97, 100
mttd2, 99
muvar, 100
mvmeta, 9, 102
palette, 121
par, 121
PARn(snp.ES), 130
pbsize, 15, 27, 104, 106, 107
pbsize2, 105
pedtotodot, 107
pfc, 43, 110, 113
pfc.sim, 112
pgc, 113
plot.hap.score, 115
pqtl2dplot, 116
pqtl2dplotly, 117
pqtl3dplotly, 118
print.hap.score, 119
pvalue, 120
qqfun, 120, 123
qqnorm, 122
qqunif, 88, 122, 122
read.ms.output, 124
revStrand, 125
runshinygap, 126
INDEX

s2k, 127
sentinels, 128
snp.ES, 130
snp.HWE (snp.ES), 130
snp.PAR (snp.ES), 130
snptest_sample, 132

tsc, 106, 133

whscore, 135