Package 'ssa'

January 14, 2019

Title Simultaneous Signal Analysis

Version 1.3.0

Description Procedures for analyzing simultaneous signals, e.g., features that are simultaneously significant in two different studies. Includes methods for detecting simultaneous signals, for identifying them under false discovery rate control, and for leveraging them to improve prediction.

Depends R (>= 3.2.2)

Imports stats, iterators, parallel

License GPL-3

LazyData true

RoxygenNote 6.1.1

NeedsCompilation yes

Author Dave Zhao [aut, cre]

Maintainer Dave Zhao <dave.zhao@gmail.com>

Repository CRAN

Date/Publication 2019-01-14 08:20:03 UTC

R topics documented:

`bi.npmle` *Bivariate NPMLE*

Description

General nonparametric maximum likelihood estimation for a bivariate mixing distribution, implemented using EM. Assumes that the observed data are tuples (X_{1i}, X_{2i}) with marginal likelihood

$$\int f_1(X_{1i}; u_1) f_2(X_{2i}; u_2) dG(u_1, u_2),$$

where G is the mixing distribution to be estimated. Suppose there are p observed tuples and G is to be estimated on a grid of d1 x d2 points.

Usage

```
bi.npmle(D1, D2, maxit = 200, tol = 1e-04, verbose = FALSE)
```

Arguments

`D1`	p x d1 matrix of conditional density values, where the ijth entry is $f_1(X_{1i}; u_{1j})$.
`D2`	p x d2 matrix of conditional density values, where the ijth entry is $f_2(X_{2i}; u_{2j})$.
`maxit`	maximum number of EM iterations
`tol`	error tolerance
`verbose`	TRUE to print the error attained by each EM iteration

Value

`g`	d1 x d2 matrix of probability masses at each grid point

Examples

```
## generate parameters from mixing distribution
p <- 1000;
set.seed(1); theta1 <- rnorm(p); theta2 <- -theta1+rnorm(p);
## generate observed variables
X1 <- rnorm(p,theta1,1); X2 <- rnorm(p,theta2,1);
## set grid points
d1 <- 25; d2 <- 30;
Theta1 <- seq(min(X1),max(X1),length=d1);
Theta2 <- seq(min(X2),max(X2),length=d2);
## calculate D matrices
D1 <- outer(X1,Theta1,function(x,y){
  dnorm(x,y,1);
});
D2 <- outer(X2,Theta2,function(x,y){
  dnorm(x,y,1);
});
## fit npmle
g <- bi.npmle(D1,D2);
contour(Theta1,Theta2,g);
points(theta1,theta2);
```

`expit` *Expit function*

Description

Expit function

Usage

```
expit(x)
```

Arguments

`x`	value between -infinity and infinity

Value

1/(1+exp(-x))

Examples

```
x <- 1;
expit(x);
```

ldd *Latent dependency detection*

Description

Given two sequences of paired test statistics, tests whether the latent indicators of significance are positively dependent.

Usage

```
ldd(T1, T2, m1 = 1000, m2 = 1000, perm = 0, p1 = TRUE, p2 = TRUE,
  jitter = NULL)
```

Arguments

T1, T2	paired vectors of test statistics, both must be the same length; can be p-values or otherwise; if not p-values, must be stochastically larger under the null
m1, m2	search only up the m1th (m2th) most significant test statistic in T1 (T2); NULL to search through all statistics
perm	the indices of T1 will be randomly permuted perm times; the permutation p-value will be calculated as a fraction of perm+1
p1, p2	TRUE if T1 (T2) is a vector of p-values
jitter	NULL if no jittering is desired to resolve ties, otherwise a jitter of runif(0,jitter) will be added to all entries of T1 and T2

Value

D	value of the test statistic
p.perm	permutation p-value
p.asymp	asymptotic approximate p-value

Examples

```
## generate paired test statistics

p <- 10^6; ## total number of pairs
X <- c(rep(0,p-30),rep(1,10),rep(2,10),rep(3,10));
## X=0: no signal in either sequence of tests
## X=1: signal in sequence 1 only
## X=2: signal in sequence 2 only
## X=3: simultaneous signal
set.seed(1);
Z1 <- rnorm(p,0,1); Z1[X==1|X==3] <- rnorm(20,3,1);
Z2 <- rnorm(p,0,1); Z2[X==2|X==3] <- rnorm(20,4,1);
## convert to p-value
P1 <- 2*pnorm(-abs(Z1));
P2 <- 2*pnorm(-abs(Z2));
```

```
## run different version of ldd()
out.pp <- ldd(P1,P2,perm=100);
out.zp <- ldd(abs(Z1),P2,p1=FALSE,perm=100);
out.pz <- ldd(P1,abs(Z2),p2=FALSE,perm=100);
out.zz <- ldd(abs(Z1),abs(Z2),p1=FALSE,p2=FALSE,perm=100);
```

`logit` *Logit function*

Description

Logit function

Usage

```
logit(x)
```

Arguments

`x`	value between 0 and 1

Value

log(x/(1-x))

Examples

```
x <- 0.3;
logit(x);
```

`maxtest` *Max test for detecting simultaneous signals*

Description

Given two sequences of paired test statistics, tests whether any simultaneous signals exist

Usage

```
maxtest(T1, T2)
```

Arguments

T1, T2	paired vectors of test statistics, both must be the same length; must be stochastically larger under the alternative than under the null; must contain only positive values

Value

p	p-value
M	value of max statistic

Examples

```
## generate paired test statistics
p <- 10^6; ## total number of pairs
X <- c(rep(0,p-30),rep(1,10),rep(2,10),rep(3,10));
## X=0: no signal in either sequence of tests
## X=1: signal in sequence 1 only
## X=2: signal in sequence 2 only
## X=3: simultaneous signal
set.seed(1);
Z1 <- rnorm(p,0,1); Z1[X==1|X==3] <- rnorm(20,3,1);
Z2 <- rnorm(p,0,1); Z2[X==2|X==3] <- rnorm(20,4,1);
maxtest(abs(Z1),abs(Z2));
```

---

neb.predict — *Nonparametric empirical Bayes classifier without annotations; prediction*

Description

Nonparametric empirical Bayes classifier without annotations; prediction

Usage

```
neb.predict(newX, neb, P = NULL, cores = 1)
```

Arguments

newX	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
neb	output of neb.train()
P	prevalence of cases in the testing set; if NULL, P is taken from the train object
cores	number of cores to use

Value

ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
neb <- neb.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,d=c(20,25));
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- neb.predict(newX,neb);
mean(abs(newY-Yhat$class));
```

---

neb.train *Nonparametric empirical Bayes classifier without annotations; training*

Description

Treats the control and case minor allele frequencies as random tuples from a bivariate prior distribution G and then estimates the optimal Bayesian classifier given G. Nonparametric maximum likelihood is used as a plug-in estimator for G.

Usage

```
neb.train(pi0, pi1, n0, n1, d = 25, maxit = 200, tol = 1e-04,
  verbose = FALSE)
```

Arguments

pi0, pi1	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
n0, n1	number of controls and number of cases, respectively

d	if a single number, G is estimated on a d x d grid; if a two-component vector (d0,d1), G is estimated on a d0 x d1 grid
maxit	maximum number of EM iterations
tol	error tolerance
verbose	TRUE to print the error attained by each EM iteration

Value

Pi0	grid points for estimating the distribution of the control minor allele frequencies
Pi1	grid points for estimating the distribution of the case minor allele frequencies
D0	conditional density matrix for controls
D1	conditional density matrix for cases
g	estimated mixing probability mass function
P	proportion of cases

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
neb <- neb.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,d=c(20,25));
contour(neb$Pi0,neb$Pi1,neb$g);
points(pi0,pi1);
```

---

nebula.bin.predict *Nonparametric empirical Bayes classifier using latent annotations: binary indicators; prediction*

Description

Nonparametric empirical Bayes classifier using latent annotations: binary indicators; prediction

Usage

```
nebula.bin.predict(newX, nebula, P = NULL, cores = 1)
```

Arguments

newX	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
nebula	output of nebula.chisq.train()
P	prevalence of cases in the testing set; if NULL, P is taken from the train object
cores	number of cores to use

Value

ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula <- nebula.bin.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,I,d=c(20,25));
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- nebula.bin.predict(newX,nebula);
mean(abs(newY-Yhat$class));
```

---

nebula.bin.train *Nonparametric empirical Bayes classifier using latent annotations: binary indicators; training*

Description

Assumes that binary indicators for each SNP are available; e.g. indicate whether the SNP is an eQTL. Treats the true control and case minor allele frequencies for SNPs with indicators equal to 0 and 1 as random triples from bivariate prior distributions G0 and G1, and estimates the optimal Bayesian classifier given G0 and G1. Nonparametric maximum likelihood is used as a plug-in estimator for G0 and G1.

Usage

```
nebula.bin.train(pi0, pi1, n0, n1, I, d = 25, maxit = 200,
  tol = 1e-04, verbose = FALSE)
```

Arguments

pi0, pi1	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
n0, n1	number of controls and number of cases, respectively
I	p x 1 vector of binary indicators
d	if a single number, G0 and G1 are estimated on d x d grids; if a two-component vector (d0,d1), G0 and G1 are estimated on d0 x d1 grids
maxit	maximum number of EM iterations
tol	error tolerance
verbose	TRUE to print the error attained by each EM iteration

Value

neb0	output of neb.train using only SNPs with I==0
neb1	output of neb.train using only SNPs with I==1
I	binary indicator

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula <- nebula.bin.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,I,d=c(20,25));
par(mfrow=c(1,2));
contour(nebula$neb0$Pi0,nebula$neb0$Pi1,apply(nebula$neb0$g,c(1,2),sum));
points(pi0[I==0],pi1[I==0]);
contour(nebula$neb1$Pi0,nebula$neb1$Pi1,apply(nebula$neb1$g,c(1,2),sum));
points(pi0[I==1],pi1[I==1]);
```

nebula.chisq.bin.predict

Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics and binary indicators; prediction

Description

Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics and binary indicators; prediction

Usage

```
nebula.chisq.bin.predict(newX, nebula, P = NULL, cores = 1)
```

Arguments

newX	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
nebula	output of nebula.chisq.train()
P	prevalence of cases in the testing set; if NULL, P is taken from the train object
cores	number of cores to use

Value

ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case
ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
```

```
nebula <- nebula.chisq.bin.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T,I,d=c(10,12,14));
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- nebula.chisq.bin.predict(newX,nebula);
mean(abs(newY-Yhat$class));
```

`nebula.chisq.bin.train` *Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics and binary indicators; training*

Description

Assumes that chi-square test statistics for each SNP are available from another study, and binary indicators for each SNP are available as well. Treats the true control and case minor allele frequencies and the chi-square non-centrality parameters as random triples from a bivariate prior distribution G0 for SNPs with indicators 0, and from G1 for SNPs with indicators equal to 1. Estimates the optimal Bayesian classifier given G. Nonparametric maximum likelihood is used as a plug-in estimator for G.

Usage

```
nebula.chisq.bin.train(pi0, pi1, n0, n1, T, I, d = 25, maxit = 200,
  tol = 1e-04, verbose = FALSE)
```

Arguments

`pi0, pi1`	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
`n0, n1`	number of controls and number of cases, respectively
`T`	p x 1 vector of chi-square test statistics
`I`	p x 1 vector of binary indicators
`d`	if a single number, G0 and G1 are estimated on d x d x d grids; if a three-component vector (d0,d1,dt), G0 and G1 are estimated on d0 x d1 x dt grids
`maxit`	maximum number of EM iterations
`tol`	error tolerance
`verbose`	TRUE to print the error attained by each EM iteration

Value

`nebula0`	output of nebula.chisq.train using only SNPs with I==0
`nebula1`	output of nebula.chisq.train using only SNPs with I==1
`I`	binary indicator

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula <- nebula.chisq.bin.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T,I,d=c(10,15,20));
par(mfrow=c(2,3));
contour(nebula$nebula0$Pi0,nebula$nebula0$Pi1,apply(nebula$nebula0$g,c(1,2),sum));
points(pi0[I==0],pi1[I==0]);
contour(nebula$nebula0$Pi0,nebula$nebula0$Lam,apply(nebula$nebula0$g,c(1,3),sum));
points(pi0[I==0],lam[I==0]);
contour(nebula$nebula0$Pi1,nebula$nebula0$Lam,apply(nebula$nebula0$g,c(2,3),sum));
points(pi1[I==0],lam[I==0]);
contour(nebula$nebula1$Pi0,nebula$nebula1$Pi1,apply(nebula$nebula1$g,c(1,2),sum));
points(pi0[I==1],pi1[I==1]);
contour(nebula$nebula1$Pi0,nebula$nebula1$Lam,apply(nebula$nebula1$g,c(1,3),sum));
points(pi0[I==1],lam[I==1]);
contour(nebula$nebula1$Pi1,nebula$nebula1$Lam,apply(nebula$nebula1$g,c(2,3),sum));
points(pi1[I==1],lam[I==1]);
```

`nebula.chisq.predict` *Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics; prediction*

Description

Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics; prediction

Usage

```
nebula.chisq.predict(newX, nebula, P = NULL, cores = 1)
```

Arguments

`newX`	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
`nebula`	output of nebula.chisq.train()
`P`	prevalence of cases in the testing set; if NULL, P is taken from the train object
`cores`	number of cores to use

Value

ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,50); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula <- nebula.chisq.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T,d=c(10,12,14));
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- nebula.chisq.predict(newX,nebula);
mean(abs(newY-Yhat$class));
```

---

nebula.chisq.train *Nonparametric empirical Bayes classifier using latent annotations: chi-square test statistics; training*

Description

Assumes that chi-square test statistics for each SNP are available from another study. Treats the true control and case minor allele frequencies and the chi-square non-centrality parameters as random triples from a bivariate prior distribution G, and estimates the optimal Bayesian classifier given G. Nonparametric maximum likelihood is used as a plug-in estimator for G.

Usage

```
nebula.chisq.train(pi0, pi1, n0, n1, T, d = 25, maxit = 200,
  tol = 1e-04, verbose = FALSE)
```

Arguments

pi0, pi1	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
n0, n1	number of controls and number of cases, respectively
T	p x 1 vector of chi-square test statistics
d	if a single number, G is estimated on a d x d x d grid; if a three-component vector (d0,d1,dt), G is estimated on a d0 x d1 x dt grid
maxit	maximum number of EM iterations
tol	error tolerance
verbose	TRUE to print the error attained by each EM iteration

Value

Pi0	grid points for estimating the distribution of the control minor allele frequencies
Pi1	grid points for estimating the distribution of the case minor allele frequencies
Lam	grid points for estimating the distribution of the non-centrality parameter
D0	conditional density matrix for controls
D1	conditional density matrix for cases
DT	conditional density matrix for test statistics
g	estimated mixing probability mass function
P	proportion of cases

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula <- nebula.chisq.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T,d=c(20,25,30));
par(mfrow=c(1,3));
contour(nebula$Pi0,nebula$Pi1,apply(nebula$g,c(1,2),sum));
points(pi0,pi1);
contour(nebula$Pi0,nebula$Lam,apply(nebula$g,c(1,3),sum));
points(pi0,lam);
contour(nebula$Pi1,nebula$Lam,apply(nebula$g,c(2,3),sum));
points(pi1,lam);
```

nebula.predict *Nonparametric empirical Bayes classifier using latent annotations: wrapper function; predict*

Description

Nonparametric empirical Bayes classifier using latent annotations: wrapper function; predict

Usage

```
nebula.predict(newX, nebula, P = NULL, cores = 1)
```

Arguments

newX	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
nebula	output of nebula.chisq.train()
P	prevalence of cases in the testing set; if NULL, P is taken from the train object
cores	number of cores to use

Value

ll	2 x p matrix of log-likelihoods, first row is from controls
score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,25); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula1 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,d=c(5,7));
nebula2 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T=T,d=c(5,7,9));
nebula3 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,I=I,d=c(5,7));
nebula4 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T=T,I=I,d=c(5,7,9));
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
```

```
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat1 <- nebula.predict(newX,nebula1);
Yhat2 <- nebula.predict(newX,nebula2);
Yhat3 <- nebula.predict(newX,nebula3);
Yhat4 <- nebula.predict(newX,nebula4);
mean(abs(newY-Yhat1$class));
mean(abs(newY-Yhat2$class));
mean(abs(newY-Yhat3$class));
mean(abs(newY-Yhat4$class));
```

`nebula.train` *Nonparametric empirical Bayes classifier using latent annotations: wrapper function; training*

Description

Nonparametric empirical Bayes classifier using latent annotations: wrapper function; training

Usage

```
nebula.train(pi0, pi1, n0, n1, T = NULL, I = NULL, d = 25,
  maxit = 200, tol = 1e-04, verbose = FALSE)
```

Arguments

`pi0, pi1`	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
`n0, n1`	number of controls and number of cases, respectively
`T`	p x 1 vector of chi-square test statistics
`I`	p x 1 vector of binary indicators
`d`	if a single number, G0 and G1 are estimated on d x d x d grids; if a three-component vector (d0,d1,dt), G0 and G1 are estimated on d0 x d1 x dt grids
`maxit`	maximum number of EM iterations
`tol`	error tolerance
`verbose`	TRUE to print the error attained by each EM iteration

Value

`type`	1=given neither T nor I; 2=given T but not I; 3=not given T but given I; 4=given both T and I
`nebula`	trained classifier

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
set.seed(1); lam <- rep(0,p); lam[I==1] <- rchisq(sum(I==1),1,50); ## ncps
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
T <- rchisq(p,1,lam); ## chi-square statistics
nebula1 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,d=c(10,15));
nebula2 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T=T,d=c(10,15,20));
nebula3 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,I=I,d=c(10,15));
nebula4 <- nebula.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1,T=T,I=I,d=c(10,15,20));
```

---

nfsdr *Nonparametric false simultaneous discovery rate control*

Description

Given D sequences of test statistics, returns the optimal square rejection that identifies the largest number of simultaneous signals while controlling the false discovery rate. Assumes a common threshold for each sequence.

Usage

```
nfsdr(T, alpha, rho = 0, m = 5000, rescale = TRUE)
```

Arguments

T	n x D matrix of test statistics that are stochastically larger under the null, where n is the number of features and D is the numberof sequences of test statistics
alpha	nominal false simultaneous discovery rate
rho	regularization parameter to guarantee asymptotic control of the false discovery rate; should be a small positive value, but rho = 0 works well in most simulations
m	search for the optimal threshold up to only the mth largest unique value of T; can speed up computation
rescale	apply rank transformation to the test statistics within each sequence such that they are of comparable scales

Value

indices of the features delcared to be simultaneous signals

Examples

```
## generate paired test statistics
p <- 10^6; ## total number of pairs
X <- c(rep(0,p-30),rep(1,10),rep(2,10),rep(3,10));
## X=0: no signal in either sequence of tests
## X=1: signal in sequence 1 only
## X=2: signal in sequence 2 only
## X=3: simultaneous signal
set.seed(1);
Z1 <- rnorm(p,0,1); Z1[X==1|X==3] <- rnorm(20,3,1);
Z2 <- rnorm(p,0,1); Z2[X==2|X==3] <- rnorm(20,4,1);
T <- cbind(Z1^2, Z2^2);
## rejected simultaneous signals
nfsdr(T, 0.05)
```

---

nfsdr2 *Nonparametric false simultaneous discovery rate control, two thresholds*

Description

Given two sequences of paired test statistics, returns the optimal rectangular rejection that identifies the largest number of simultaneous signals while controlling the false discovery rate. Allows different thresholds for each sequence.

Usage

```
nfsdr2(T1, T2, alpha, m1 = 10000, m2 = 10000, p1 = TRUE, p2 = TRUE,
  jitter = NULL)
```

Arguments

T1, T2	paired vectors of test statistics, both must be the same length; can be p-values or otherwise; if not p-values, must be stochastically larger under the null; must contain only positive values
alpha	nominal false simultaneous discovery rate
m1, m2	search only up the m1th (m2th) most significant test statistic in T1 (T2); NULL to search through all statistics
p1, p2	TRUE if T1 (T2) is a vector of p-values
jitter	NULL if no jittering is desired to resolve ties, otherwise a jitter of `runif(0,jitter)` will be added to all entries of T1 and T2

Value

two-component vector; the first component is the optimal threshold for T1 and the second is for T2

Examples

```
## generate paired test statistics
p <- 10^6; ## total number of pairs
X <- c(rep(0,p-30),rep(1,10),rep(2,10),rep(3,10));
## X=0: no signal in either sequence of tests
## X=1: signal in sequence 1 only
## X=2: signal in sequence 2 only
## X=3: simultaneous signal
set.seed(1);
Z1 <- rnorm(p,0,1); Z1[X==1|X==3] <- rnorm(20,3,1);
Z2 <- rnorm(p,0,1); Z2[X==2|X==3] <- rnorm(20,4,1);
## convert to p-value
P1 <- 2*pnorm(-abs(Z1));
P2 <- 2*pnorm(-abs(Z2));
## run different version of fsdr()
out.pp <- fsdr(P1,P2,alpha=0.05);
out.zp <- fsdr(abs(Z1),P2,p1=FALSE,alpha=0.05);
out.pz <- fsdr(P1,abs(Z2),p2=FALSE,alpha=0.05);
out.zz <- fsdr(abs(Z1),abs(Z2),p1=FALSE,p2=FALSE,alpha=0.05);
## discovered simultaneous features
R1 <- which(P1<=out.pp[1]&P2<=out.pp[2]);
R2 <- which(abs(Z1)>=out.zp[1]&P2<=out.zp[2]);
R3 <- which(P1<=out.pz[1]&abs(Z2)>=out.pz[2]);
R4 <- which(abs(Z1)>=out.zz[1]&abs(Z2)>=out.zz[2]);
```

`nfsdr2_all` *Nonparametric false simultaneous discovery rate control, two thresholds – report all thresholds*

Description

Given two sequences of paired test statistics, returns all rectangular rejections that identify the largest number of simultaneous signals while also controlling the false discovery rate. Allows different thresholds for each sequence.

Usage

```
nfsdr2_all(T1, T2, alpha, m1 = 5000, m2 = 5000)
```

Arguments

`T1, T2`	paired vectors of test statistics, both must be the same length; must be stochastically larger under the alternative than under the null; must contain only positive values
`alpha`	nominal false simultaneous discovery rate
`m1, m2`	search only up the m1th (m2th) most significant test statistic in T1 (T2)

Value

k x 2 matrix, where k is the number of rectangular regions found; the first column is the threshold for T1 and the second column is for T2

Examples

```
## generate paired test statistics
p <- 10^6; ## total number of pairs
X <- c(rep(0,p-30),rep(1,10),rep(2,10),rep(3,10));
## X=0: no signal in either sequence of tests
## X=1: signal in sequence 1 only
## X=2: signal in sequence 2 only
## X=3: simultaneous signal
set.seed(1);
Z1 <- rnorm(p,0,1); Z1[X==1|X==3] <- rnorm(20,3,1);
Z2 <- rnorm(p,0,1); Z2[X==2|X==3] <- rnorm(20,4,1);
## all rectangular rejection regions
out.zz <- fsdr_all(abs(Z1),abs(Z2),alpha=0.05,m1=1000,m2=1000);
## 10 sets of identified simultaneous signals
R <- apply(out.zz[1:10,],1,function(x){
             which(abs(Z1)>=x[1]&abs(Z2)>=x[2]);
           });
```

---

prs.predict *Polygenic risk score; prediction*

Description

Polygenic risk score; prediction

Usage

```
prs.predict(newX, prs, P = NULL)
```

Arguments

newX	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
prs	output of prs.train()
P	prevalence of cases in the testing set; if NULL, P is taken from the train object

Value

score	risk score
class	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
prs <- prs.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1);
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- prs.predict(newX,prs);
mean(abs(newY-Yhat$class));
```

`prs.predict.cv` *Polygenic risk score; prediction for classifier trained with CV*

Description

Polygenic risk score; prediction for classifier trained with CV

Usage

```
prs.predict.cv(newX, prs, P = NULL)
```

Arguments

`newX`	n x p matrix of additively coded genotypes to be predicted; IMPORTANT: must be coded relative to the same allele as in the cases and controls
`prs`	output of prs.train.cv()
`P`	prevalence of cases in the testing set; if NULL, P is taken from the train object

Value

`score`	risk score
`class`	predicted class, 0=control, 1=case

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
## training data
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
prs <- prs.train.cv(X0,X1,K=3,w=1,nlambda=100,verbose=TRUE);
## testing data
newX <- rbind(t(replicate(n0,rbinom(p,2,pi0))),
              t(replicate(n1,rbinom(p,2,pi1))));
newY <- c(rep(0,n0),rep(1,n1));
Yhat <- prs.predict.cv(newX,prs);
mean(abs(newY-Yhat$class));
```

`prs.train` *Polygenic risk score (given only allele frequencies); training*

Description

Equivalent to maximum likelihood naive Bayes classifier. The discriminant function is

$$\sum_j \hat{\beta}_j X_j,$$

where X_j is the additively coded genotype of SNP j.

Usage

```
prs.train(pi0, pi1, n0, n1)
```

Arguments

`pi0, pi1`	p x 1 vectors of control and case minor allele frequencies, respectively; IMPORTANT: must be relative to the same allele in both cases and controls
`n0, n1`	number of controls and number of cases, respectively

Value

`pi0`	minor allele frequencies in controls
`pi1`	minor allele frequencies in cases
`P`	proportion of cases

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
prs.train(colMeans(X0)/2,colMeans(X1)/2,n0,n1);
```

---

prs.train.cv *Polygenic risk score (given only allele frequencies); training with CV*

Description

Uses CV to select how many SNPs to include. SNPs are ordered by the magnitude of their estimated (ang possibly weighted) allelic log-odds ratio. The discriminant function is

$$\sum_j \hat{\beta}_j I\left(\left|\frac{\hat{\beta}_j}{w_j}\right| > \lambda\right) X_j,$$

where X_j is the additively coded genotype of SNP j.

Usage

```
prs.train.cv(X0, X1, K = 3, w = 1, nlambda = 100, verbose = FALSE)
```

Arguments

X0, X1	n x p vectors of control and case genotypes, additively coded; IMPORTANT: coding must be relative to the same allele in both cases and controls
K	number of folds for CV
w	p x 1 weight vector
nlambda	number of thresholds to tune over
verbose	if TRUE, report current fold of CV

Value

pi0	minor allele frequencies in controls of kept SNPs
pi1	minor allele frequencies in cases of kept SNPs
w	weight vector for kept SNPs
lambda	lambda cutoff
P	proportion of cases

Examples

```
p <- 1000; ## number of snps
I <- rep(0,p); I[1:10] <- 1; ## which snps are causal
set.seed(1); pi0 <- runif(p,0.1,0.5); ## control minor allele frequencies
set.seed(1); ors <- runif(sum(I),-1,1); ## odds ratios
pi1 <- pi0;
pi1[I==1] <- expit(ors+logit(pi0[I==1]));
n0 <- 100; ## number of controls
X0 <- t(replicate(n0,rbinom(p,2,pi0))); ## controls
n1 <- 50; ## number of cases
X1 <- t(replicate(n1,rbinom(p,2,pi1))); ## cases
prs.train.cv(X0,X1,K=3,w=1,nlambda=100,verbose=TRUE);
```

`tri.npmle` *Trivariate NPMLE*

Description

General nonparametric maximum likelihood estimation for a trivariate mixing distribution, implemented using EM. Assumes that the observed data are triples (X_{1i}, X_{2i}, X_{3i}) with marginal likelihood

$$\int f_1(X_{1i}; u_1) f_2(X_{2i}; u_2) f_3(X_{3i}; u_3) dG(u_1, u_2, u_3),$$

where G is the mixing distribution to be estimated. Suppose there are p observed tuples and G is to be estimated on a grid of d1 x d2 x d3 points.

Usage

```
tri.npmle(D1, D2, D3, maxit = 200, tol = 1e-04, verbose = FALSE)
```

Arguments

`D1`	p x d1 matrix of conditional density values, where the ijth entry is $f_1(X_{1i}; u_{1j})$.
`D2`	p x d2 matrix of conditional density values, where the ijth entry is $f_2(X_{2i}; u_{2j})$.
`D3`	p x d3 matrix of conditional density values, where the ijth entry is $f_3(X_{3i}; u_{3j})$.
`maxit`	maximum number of EM iterations
`tol`	error tolerance
`verbose`	TRUE to print the error attained by each EM iteration

Value

`g`	d1 x d2 x d3 array of probability masses at each grid point

Examples

```
## generate parameters from mixing distribution
p <- 1000;
set.seed(1);
theta1 <- rnorm(p);
theta2 <- -theta1+rnorm(p);
theta3 <- 0.5*theta1+theta2+rnorm(p);
## generate observed variables
X1 <- rnorm(p,theta1,1);
X2 <- rnorm(p,theta2,1);
X3 <- rnorm(p,theta3,1);
## set grid points
d1 <- 15; d2 <- 20; d3 <- 25;
Theta1 <- seq(min(X1),max(X1),length=d1);
Theta2 <- seq(min(X2),max(X2),length=d2);
Theta3 <- seq(min(X3),max(X3),length=d3);
## calculate D matrices
D1 <- outer(X1,Theta1,function(x,y){
  dnorm(x,y,1);
});
D2 <- outer(X2,Theta2,function(x,y){
  dnorm(x,y,1);
});
D3 <- outer(X3,Theta3,function(x,y){
  dnorm(x,y,1);
});
## fit npmle
g <- tri.npmle(D1,D2,D3);
par(mfrow=c(1,3));
contour(Theta1,Theta2,apply(g,c(1,2),sum));
points(theta1,theta2);
contour(Theta1,Theta3,apply(g,c(1,3),sum));
points(theta1,theta3);
contour(Theta2,Theta3,apply(g,c(2,3),sum));
points(theta2,theta3);
```

---

uni.npmle *Univariate NPMLE*

Description

General nonparametric maximum likelihood estimation for a univariate mixing distribution, implemented using EM. Assumes that the observed data are X_i with marginal likelihood

$$\int f(X_i; u) dG(u),$$

where G is the mixing distribution to be estimated. Suppose there are p observations and G is to be estimated on a grid of d points.

Usage

```
uni.npmle(D, maxit = 200, tol = 1e-04, verbose = FALSE)
```

Arguments

D	p x d matrix of conditional density values, where the ijth entry is $f(X_i; u_j)$
maxit	maximum number of EM iterations
tol	error tolerance
verbose	TRUE to print the error attained by each EM iteration

Value

g	d x 1 vector of probability masses at each grid point

Examples

```
## generate parameters from mixing distribution
p <- 1000;
set.seed(1); theta <- rnorm(p);
## generate observed variables
X <- rnorm(p,theta,1);
## set grid points
d <- 25;
Theta <- seq(min(X),max(X),length=d);
## calculate D matrix
D <- outer(X,Theta,function(x,y){
  dnorm(x,y,1);
});
## fit npmle
g <- uni.npmle(D);
plot(Theta,g/sum(g*(Theta[2]-Theta[1])),type="l");
lines(Theta,dnorm(Theta),col="red");
```

Index