Package 'GPA'

May 2, 2026

Type Package

Version 1.25.0

Date 2020-02-24

Title GPA (Genetic analysis incorporating Pleiotropy and Annotation)

Author Dongjun Chung, Emma Kortemeier, Carter Allen

Maintainer Dongjun Chung <dongjun.chung@gmail.com>

Depends R (>= 4.0.0), methods, graphics, Rcpp

Imports parallel, ggplot2, ggrepel, plyr, vegan, DT, shiny, shinyBS, stats, utils, grDevices

Suggests gpaExample

LinkingTo Rcpp

Description
This package provides functions for fitting GPA, a statistical framework to prioritize GWAS results by integrating pleiotropy information and annotation data. In addition, it also includes ShinyGPA, an interactive visualization toolkit to investigate pleiotropic architecture.

License GPL (>= 2)

URL http://dongjunchung.github.io/GPA/

BugReports https://github.com/dongjunchung/GPA/issues

NeedsCompilation yes

biocViews Software, StatisticalMethod, Classification, GenomeWideAssociation, SNP, Genetics, Clustering, MultipleComparison, Preprocessing, GeneExpression, DifferentialExpression

SystemRequirements GNU make

LazyData FALSE

git_url https://git.bioconductor.org/packages/GPA

git_branch devel

git_last_commit f20fe5c

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Contents

GPA-package 2
assoc 4
aTest 6
fitAll 8
GPA 10
GPA-class 12
GPA-internal 14
pTest 14
shinyGPA 16

Index **18**

`GPA-package` *GPA (Genetic analysis incorporating Pleiotropy and Annotation)*

Description

This package provides functions for fitting GPA, a statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data, along with ShinyGPA, a visualization toolkit to investigate the pleiotropic architecture using GWAS results.

Details

Package:	GPA
Type:	Package
Version:	0.99.9
Date:	2020-04-14
License:	GPL (>= 2)
LazyLoad:	yes

This package contains a main class, `GPA`, which represents GPA model fit.

This package contains four main methods for the GPA framework (Chung et al., 2014), `GPA`, `assoc`, `pTest`, and `aTest`. `GPA` method fits the GPA model and `assoc` method implements association mapping. `pTest` and `aTest` methods implement hypothesis testing for pleiotropy and annotation enrichment, respectively.

This package contains two main methods for the ShinyGPA visualization toolkit (Kortemeier et al., 2017), `fitAll` and `shinyGPA`. `fitAll` function generates all the intermediate results needed to run `shinyGPA`. `shinyGPA` opens the ShinyGPA interface, which takes the results generated from `fitAll` as input.

Author(s)

Dongjun Chung, Emma Kortemeier

Maintainer: Dongjun Chung <dongjun.chung@gmail.com>

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

Kortemeier E, Ramos PS, Hunt KJ, Kim HJ, Hardiman G, and Chung D (2018), "ShinyGPA: An interactive and dynamic visualization toolkit for genetic studies," PLOS One, 13(1): e0190949.

See Also

GPA, assoc, pTest, aTest, GPA, fitAll, shinyGPA.

Examples

```
## Not run:
# simulation setting

nBin <- 1000
pi1 <- 0.2
common <- 0.5
betaAlpha <- c( 0.6, 0.6 )
annP <- c( 0.2, 0.4, 0.4, 0.4 )
seed <- 12345

# simulation setting

nCommon <- round( pi1 * common * nBin )
nUniq <- round( pi1 * ( 1 - common ) * nBin )
nBg <- nBin - 2 * nUniq - nCommon

# M * K matrix of GWAS p-value

set.seed( seed )

pvec1 <- c( rbeta( nCommon, betaAlpha[1], 1 ), rbeta( nUniq, betaAlpha[1], 1 ),
 runif( nUniq ), runif( nBg ) )
pvec2 <- c( rbeta( nCommon, betaAlpha[2], 1 ), runif( nUniq ),
 rbeta( nUniq, betaAlpha[2], 1 ), runif( nBg ) )
pmat <- cbind( pvec1, pvec2 )

# M * D matrix of annotation

ann <- c(
 sample( c(1,0), nCommon, replace=TRUE, prob = c( annP[4], 1 - annP[4] ) ),
 sample( c(1,0), nUniq, replace=TRUE, prob = c( annP[2], 1 - annP[2] ) ),
 sample( c(1,0), nUniq, replace=TRUE, prob = c( annP[3], 1 - annP[3] ) ),
 sample( c(1,0), nBg, replace=TRUE, prob = c( annP[1], 1 - annP[1] ) ) )

# GPA without annotation data

fit.GPA.noAnn <- GPA( pmat, NULL )
cov.GPA.noAnn <- cov( fit.GPA.noAnn )

# GPA with annotation data

fit.GPA.wAnn <- GPA( pmat, ann )
cov.GPA.wAnn <- cov( fit.GPA.wAnn )
```

```
# GPA under pleiotropy H0

fit.GPA.pleiotropy.H0 <- GPA( pmat, NULL, pleiotropyH0=TRUE )

# association mapping

assoc.GPA.wAnn <- assoc( fit.GPA.wAnn, FDR=0.05, fdrControl="global" )

# hypothesis testing for pleiotropy

test.pleiotropy <- pTest( fit.GPA.noAnn, fit.GPA.pleiotropy.H0 )

# hypothesis testing for annotation enrichment

test.annotation <- aTest( fit.GPA.noAnn, fit.GPA.wAnn )

# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.4
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

# Fit GPA for all possible pairs of GWAS datasets

out <- fitAll( pmat )

# Run the ShinyGPA app using the ouput from fitAll()

shinyGPA(out)

## End(Not run)
```

---

assoc *Association mapping*

Description

Association mapping.

Usage

```
assoc( object, ... )
## S4 method for signature 'GPA'
assoc( object, FDR=0.05, fdrControl="global", pattern=NULL )
```

Arguments

`object`	GPA model fit.
`FDR`	FDR level.
`fdrControl`	Method to control FDR. Possible values are "global" (global FDR control) and "local" (local FDR control). Default is "global".
`pattern`	Pattern for association mapping. By default (i.e., `pattern=NULL`), `assoc` returns a binary matrix indicating association of SNPs for each phenotypes. If a pattern is specified, a corresponding binary vector is provided. See the details about how users can specify the pattern.
`...`	Other parameters to be passed through to generic `assoc`.

Details

`assoc` uses the direct posterior probability approach of Newton et al. (2004) to control global FDR in association mapping.

Users can specify the pattern using 1 and * in `pattern` argument, where 1 and * indicate phenotypes of interest and phenotypes that are not of interest, respectively. For example, when there are three phenotypes, `pattern="111"` means a SNP associated with all of three phenotypes, while `pattern="11*"` means a SNP associated with the first two phenotypes (i.e., association with the third phenotype is ignored (averaged out)).

Value

If `pattern=NULL`, returns a binary matrix indicating association of SNPs for each phenotype, where its rows and columns match those of input p-value matrix for function `GPA`. Otherwise, returns a binary vector indicating association of SNPs for the phenotype combination of interest.

Author(s)

Dongjun Chung

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

Newton MA, Noueiry A, Sarkar D, and Ahlquist P (2004), "Detecting differential gene expression with a semiparametric hierarchical mixture method," *Biostatistics*, Vol. 5, pp. 155-176.

See Also

`GPA`, `GPA`.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

ann <- rbinom(n = nrow(pmat), size = 1, prob = 0.15)
ann <- as.matrix(ann,ncol = 1)

fit.GPA.wAnn <- GPA( pmat, ann , maxIter = 100 )
cov.GPA.wAnn <- cov( fit.GPA.wAnn )
assoc.GPA.wAnn <- assoc( fit.GPA.wAnn, FDR=0.05, fdrControl="global" )
```

---

aTest *Hypothesis testing for annotation enrichment*

Description

Hypothesis testing for annotation enrichment.

Usage

```
aTest( fitWithoutAnn, fitWithAnn, vDigit=1000 )
```

Arguments

fitWithoutAnn	GPA model fit without using annotation data.
fitWithAnn	GPA model fit with using annotation data.
vDigit	Number of digits for reporting parameter estimates and standard errors. For example, setting it to 1000 means printing out values up to three digits below zero.

Details

aTest implements the hypothesis testing for annotation enrichment. It requires two GPA model fits, one fitted with using annotation data and one fitted without using annotation data, and evaluates annotation enrichment for risk-associated SNPs using the likelihood ratio test.

Value

Returns a list with components:

q	q estimates.
statistics	Statistics of the test for annotation enrichment.
pvalue	p-value of the test for annotation enrichment.

Author(s)

Dongjun Chung

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

See Also

pTest, GPA, GPA.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )
```

```
ann <- rbinom(n = nrow(pmat), size = 1, prob = 0.15)
ann <- as.matrix(ann,ncol = 1)

# GPA without annotation data

fit.GPA.noAnn <- GPA( pmat, NULL, maxIter = 100 )

# GPA with annotation data

fit.GPA.wAnn <- GPA( pmat, ann, maxIter = 100 )

# hypothesis testing for annotation enrichment

test.annotation <- aTest( fit.GPA.noAnn, fit.GPA.wAnn )
```

`fitAll` *Fit GPA model for all possible pairs of GWAS datasets*

Description

Fit GPA model and the GPA model under H0 for all possible pairs of GWAS datasets.

Usage

```
fitAll( pmat,
  maxIter=2000, stopping="relative", epsStopLL=1e-10,
  parallel=FALSE, nCore=8 )
```

Arguments

`pmat`	p-value matrix from GWAS data, where row and column correspond to SNP and phenotype, respectively.
`maxIter`	Maximum number of EM iteration. Default is 2000.
`stopping`	Stopping rule for EM iteration. Possible values are `"absolute"` (based on absolute difference in log likelihood), `"relative"` (based on relative difference in log likelihood), or `"aitken"` (based on Aitken acceleration-based stopping rule). Default is `"relative"`.
`epsStopLL`	Threshold to stop the EM iteration. Default is 1e-100.
`parallel`	Utilize multiple CPUs for parallel computing using `"parallel"` package? Possible values are `TRUE` (utilize multiple CPUs) or `FALSE` (do not utilize multiple CPUs). Default is `FALSE` (do not utilize multiple CPUs).
`nCore`	Number of CPUs when parallel computing is utilized.

Details

`fitAll` function fits the GPA model and the GPA model under H0 for all possible pairs of GWAS datasets. Its output can be used as an input for the `shinyGPA` function.

Value

A list with 6 elements, including `pmat` (original GWAS p-value matrix), `combs` (a matrix of GWAS pair indices), `combList` (a matrix of GWAS pair indices), `pTestPval` (a matrix of pleiotropy test p-values), `fitGPA` (a list of the GPA fit for each pair), and `fitH0` (a list of the GPA fit under H0 for each pair).

Author(s)

Dongjun Chung, Emma Kortemeier

References

Kortemeier E, Ramos PS, Hunt KJ, Kim HJ, Hardiman G, and Chung D (2017), "ShinyGPA: An interactive and dynamic visualization toolkit for genetic studies."

See Also

`GPA`, `pTest`, and `shinyGPA`.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.4
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

# Fit GPA for all possible pairs of GWAS datasets

out <- fitAll( pmat, maxIter = 100 )
```

GPA *Fit GPA model*

Description

Fit GPA model.

Usage

```
GPA( gwasPval, annMat=NULL, pleiotropyH0=FALSE, empiricalNull=FALSE,
 maxIter=2000, stopping="relative", epsStopLL=1e-10,
 initBetaAlpha=0.1, initPi=0.1, initQ=0.75,
 lbPi=NA, lbBetaAlpha=0.001, lbQ=0.001, lbPval=1e-30,
 vDigit=1000, verbose=1 )
```

Arguments

gwasPval	p-value matrix from GWAS data, where row and column correspond to SNP and phenotype, respectively.
annMat	Binary matrix from annotation data, where row and column correspond to SNP and annotation, respectively.
pleiotropyH0	Fit GPA under the null hypothesis of pleiotropy test? Possible values are TRUE (under the null hypothesis of pleiotropy test) or FALSE (usual assumption for GPA). Default is FALSE.
empiricalNull	Empirically estimate null distribution for GPA? Possible values are TRUE (empirical null distribution) or FALSE (theoretical null distribution). Default is FALSE.
maxIter	Maximum number of EM iteration. Default is 2000.
stopping	Stopping rule for EM iteration. Possible values are "absolute" (based on absolute difference in log likelihood), "relative" (based on relative difference in log likelihood), or "aitken" (based on Aitken acceleration-based stopping rule). Default is "relative".
epsStopLL	Threshold to stop the EM iteration. Default is 1e-100.
initBetaAlpha	Initial value for alpha estimate. Default is 0.1.
initPi	Initial value for pi estimate. Default is 0.1.
initQ	Initial value for q estimate. Default is 0.75.
lbPi	Lower bound for pi estimate. If lbPi=NA, lower bound is set to 1 / [number of SNPs]. Default is NA.
lbBetaAlpha	Lower bound for alpha estimate. Default is 0.001.
lbQ	Lower bound for q estimate. Default is 0.001.
lbPval	Lower bound for GWAS p-value. Any GWAS p-values smaller than lbPval are set to lbPval. Default is 1e-30.
vDigit	Number of digits for reporting parameter estimates. For example, setting it to 1000 means printing out values up to three digits below zero.
verbose	Amount of progress report during the fitting procedure. Possible values are 0 (minimal output), 1, 2, or 3 (maximal output). Default is 1.

Details

GPA fits the GPA model. It requires to provide GWAS p-value to gwasPval, while users can also provide annotation data to annMat. It is assumed that number of rows of matrix provided to gwasPval equals to that provided to annMat.

pTest implements the hypothesis testing for pleiotropy. It requires two GPA model fits, one of interest and one under the null hypothesis, and they can be obtained by setting pleiotropyH0=FALSE and pleiotropyH0=TRUE, respectively.

aTest implements the hypothesis testing for annotation enrichment. It requires two GPA model fits, one fitted with using annotation data and one fitted without using annotation data, and they can be obtained by providing annotation data to annMat and not, respectively.

Value

Construct GPA class object.

Author(s)

Dongjun Chung

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

Kortemeier E, Ramos PS, Hunt KJ, Kim HJ, Hardiman G, and Chung D (2018), "ShinyGPA: An interactive and dynamic visualization toolkit for genetic studies," PLOS One, 13(1): e0190949.

See Also

assoc, pTest, aTest, GPA.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
```

```
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

ann <- rbinom(n = nrow(pmat), size = 1, prob = 0.15)
ann <- as.matrix(ann,ncol = 1)

# GPA without annotation data

fit.GPA.noAnn <- GPA( pmat, NULL, maxIter = 100 )
cov.GPA.noAnn <- cov( fit.GPA.noAnn )

# GPA with annotation data

fit.GPA.wAnn <- GPA( pmat, ann, maxIter = 100 )
cov.GPA.wAnn <- cov( fit.GPA.wAnn )

# GPA under the null hypothesis of pleiotropy test

fit.GPA.pleiotropy.H0 <- GPA( pmat, NULL, pleiotropyH0=TRUE, maxIter = 100 )
```

`GPA-class` *Class "GPA"*

Description

This class represents GPA model fit.

Details

When users use `fdr` method, users can specify the pattern using 1 and * in `pattern` argument, where 1 and * indicate phenotypes of interest and phenotypes that are not of interest, respectively. For example, when there are three phenotypes, `pattern="111"` means a SNP associated with all of three phenotypes, while `pattern="11*"` means a SNP associated with the first two phenotypes (i.e., association with the third phenotype is ignored (averaged out)).

Objects from the Class

Objects can be created by calls of the form `new("GPA", ...)`.

Slots

`fit`**:** Object of class `"list"`, representing the fitted GPA model.

`setting`**:** Object of class `"list"`, representing the setting for GPA model fitting.

`gwasPval`**:** Object of class `"matrix"`, representing the p-value matrix from GWAS data.

`annMat`**:** Object of class `"matrix"`, representing the annotation matrix.

Methods

show `signature(object = "GPA")`: provide brief summary of the object.

print `signature(x = "GPA")`: provide the matrix of posterior probability that a SNP belongs to each combination of association status.

fdr `signature(object = "GPA", pattern=NULL)`: provide local FDR. By default (i.e., `pattern=NULL`), it returns a matrix of local FDR that a SNP is not associated with each phenotype (i.e., marginal FDR), where the order of columns is same as that in input GWAS data. If a pattern is specified, a vector of corresponding local FDR is provided. See the details about how users can specify the pattern.

cov `signature(object = "GPA", silent=FALSE, vDigitEst=1000, vDigitSE=1000 )`: provide the covariance matrix for parameter estimates of GPA model. If `silent=TRUE`, it suppresses the summary output. `vDigitEst` and `vDigitSE` control number of digits for reporting parameter estimates and standard errors. For example, setting it to 1000 means printing out values up to three digits below zero.

estimates `signature(object = "GPA")`: extract parameter estimates from GPA model fit.

se `signature(object = "GPA")`: extract standard errors for parameter estimates from GPA model fit.

Author(s)

Dongjun Chung

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

See Also

GPA, pTest, aTest.

Examples

```
showClass("GPA")

# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
```

```
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

ann <- rbinom(n = nrow(pmat), size = 1, prob = 0.15)
ann <- as.matrix(ann,ncol = 1)

fit.GPA.wAnn <- GPA( pmat, ann )
fit.GPA.wAnn
pp.GPA.wAnn <- print( fit.GPA.wAnn )
fdr.GPA.wAnn <- fdr( fit.GPA.wAnn )
fdr11.GPA.wAnn <- fdr( fit.GPA.wAnn, pattern="11" )
fdr1..GPA.wAnn <- fdr( fit.GPA.wAnn, pattern="1*" )
cov.GPA.wAnn <- cov( fit.GPA.wAnn )
est.GPA.wAnn <- estimates( fit.GPA.wAnn )
se.GPA.wAnn <- se( fit.GPA.wAnn )
```

`GPA-internal` *Internal GPA objects*

Description

Internal GPA objects.

Details

These are not to be called by the user.

`pTest` *Hypothesis testing for pleiotropy*

Description

Hypothesis testing for pleiotropy.

Usage

```
pTest( fit, fitH0, vDigit=1000 )
```

Arguments

`fit`	Fit of the GPA model of interest.
`fitH0`	GPA model fit under the null hypothesis of pleiotropy test.
`vDigit`	Number of digits for reporting parameter estimates and standard errors. For example, setting it to 1000 means printing out values up to three digits below zero.

Details

pTest implements the hypothesis testing for pleiotropy. It requires two GPA model fits, one of interest and one under the null hypothesis (obtained by setting pleiotropyH0=TRUE when running GPA function), and evaluates genetical correlation among multiple phenotypes using the likelihood ratio test.

Value

Returns a list with components:

pi	pi estimates.
piSE	Standard errors for pi estimates.
statistics	Statistics of the pleiotropy test.
pvalue	p-value of the pleiotropy test.

Author(s)

Dongjun Chung

References

Chung D*, Yang C*, Li C, Gelernter J, and Zhao H (2014), "GPA: A statistical approach to prioritizing GWAS results by integrating pleiotropy information and annotation data," PLoS Genetics, 10: e1004787. (* joint first authors)

See Also

aTest, GPA, GPA.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
```

```
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

# GPA without annotation data

fit.GPA.noAnn <- GPA( pmat, NULL, maxIter = 100 )

# GPA under the null hypothesis of pleiotropy test

fit.GPA.pleiotropy.H0 <- GPA( pmat, NULL, pleiotropyH0=TRUE, maxIter = 100 )

# hypothesis testing for pleiotropy

test.pleiotropy <- pTest( fit.GPA.noAnn, fit.GPA.pleiotropy.H0 )
```

---

shinyGPA *Run ShinyGPA app*

Description

Run ShinyGPA app.

Usage

```
shinyGPA( out=NULL )
```

Arguments

out	output of `fitAll` function.

Details

`shinyGPA` runs the ShinyGPA app. It takes the output of the `fitAll` function, which fits the GPA model for all possible pairs of GWAS datasets, as input.

Value

Provides visualization to investigate pleiotropic architecture using GWAS results.

Author(s)

Dongjun Chung, Emma Kortemeier

References

Kortemeier E, Ramos PS, Hunt KJ, Kim HJ, Hardiman G, and Chung D (2018), "ShinyGPA: An interactive and dynamic visualization toolkit for genetic studies," PLOS One, 13(1): e0190949.

See Also

`fitAll`.

Examples

```
# simulator function

simulator <- function( risk.ind, nsnp=20000, alpha=0.6 ) {

  m <- length(risk.ind)

  p.sig <- rbeta( m, alpha, 1 )
  pvec <- runif(nsnp)
  pvec[ risk.ind ] <- p.sig

  return(pvec)
}

# run simulation

set.seed(12345)
nsnp <- 1000
alpha <- 0.3
pmat <- matrix( NA, nsnp, 5 )

pmat[,1] <- simulator( c(1:200), nsnp=nsnp, alpha=alpha )
pmat[,2] <- simulator( c(51:250), nsnp=nsnp, alpha=alpha )
pmat[,3] <- simulator( c(401:600), nsnp=nsnp, alpha=alpha )
pmat[,4] <- simulator( c(451:750), nsnp=nsnp, alpha=alpha )
pmat[,5] <- simulator( c(801:1000), nsnp=nsnp, alpha=alpha )

# Fit GPA for all possible pairs of GWAS datasets

out <- fitAll( pmat, maxIter = 100 )

# Run the ShinyGPA app using the ouput from fitAll()

# shinyGPA(out)
```

Index

∗ **classes**
GPA-class, 12
∗ **internal**
GPA-internal, 14
∗ **methods**
assoc, 4
aTest, 6
fitAll, 8
GPA, 10
pTest, 14
shinyGPA, 16
∗ **models**
assoc, 4
aTest, 6
fitAll, 8
GPA, 10
pTest, 14
shinyGPA, 16
∗ **package**
GPA-package, 2
.covGPA *(*GPA-internal*)*, 14
.ff_emStep *(*GPA-internal*)*, 14

assoc, *3*, 4, *11*
assoc,GPA-method *(*assoc*)*, 4
aTest, *3*, 6, *11*, *13*, *15*

cov *(*GPA-class*)*, 12
cov,GPA-method *(*GPA-class*)*, 12

estimates *(*GPA-class*)*, 12
estimates,GPA-method *(*GPA-class*)*, 12

fdr *(*GPA-class*)*, 12
fdr,GPA-method *(*GPA-class*)*, 12
fitAll, *3*, 8, *16*

GPA, *3*, *5*, *7*, *9*, 10, *11*, *13*, *15*
GPA-class, 12
GPA-internal, 14
GPA-package, 2

print,GPA-method *(*GPA-class*)*, 12
pTest, *3*, *7*, *9*, *11*, *13*, 14

se *(*GPA-class*)*, 12
se,GPA-method *(*GPA-class*)*, 12
shinyGPA, *3*, *9*, 16
show,GPA-method *(*GPA-class*)*, 12