Package ‘AllelicSeries’

October 6, 2023

Title Allelic Series Test

Version 0.0.4.1

Description Implementation of gene-level rare variant association tests targeting allelic series: genes where increasingly deleterious mutations have increasingly large phenotypic effects. The COding-variant Allelic Series Test (COAST) operates on the benign missense variants (BMVs), deleterious missense variants (DMVs), and protein truncating variants (PTVs) within a gene. COAST uses a set of adjustable weights that tailor the test towards rejecting the null hypothesis for genes where the average magnitude of effect increases monotonically from BMVs to DMVs to PTVs. See McCaw ZR, O'Dushlaine C, Somineni H, Bereket M, Klein C, Karaletsos T, Casale FP, Koller D, Soare TW. (2022) ``An allelic series rare variant association test for candidate gene discovery'' <doi:10.1101/2022.12.23.521658>.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Imports Rcpp, RNOmni, SKAT

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.0

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Zachary McCaw [aut, cre] (<https://orcid.org/0000-0002-2006-9828>),
Christoph Klein [ctb] (<https://orcid.org/0000-0002-1748-625X>),
insitro [cph]

Maintainer Zachary McCaw <zmccaw@insitro.com>

Repository CRAN

Date/Publication 2023-10-06 06:40:06 UTC

R topics documented:

`Aggregator` *Aggregator*

Description

Aggregates genotypes within annotation categories.

Usage

```
Aggregator(
  anno,
  geno,
  drop_empty = TRUE,
  indicator = FALSE,
  method = "none",
  weights = DEFAULT_WEIGHTS
)
```

Arguments

`anno` (snps x 1) annotation vector with values in c(0, 1, 2).

`geno` (n x snps) genotype matrix.

`drop_empty` Drop empty columns? Default: TRUE.

`indicator` Convert raw counts to indicators? Default: FALSE.

`method` Method for aggregating across categories: "none", "max", "sum". Default: "none".

`weights` Annotation category weights.

Value

(n x 3) Numeric matrix without weighting, (n x 1) numeric matrix with weighting.

ASBT *Allelic Series Burden Test*

Description

Burden test with allelic series weights.

Usage

```
ASBT(
  anno,
  geno,
  pheno,
  apply_int = TRUE,
  covar = NULL,
  indicator = FALSE,
  is_pheno_binary = FALSE,
  method = "none",
  score_test = FALSE,
  weights = DEFAULT_WEIGHTS
)
```

Arguments

`anno`	(snps x 1) annotation vector with values in c(0, 1, 2).
`geno`	(n x snps) genotype matrix.
`pheno`	(n x 1) phenotype vector.
`apply_int`	Apply rank-based inverse normal transform to the phenotype? Default: TRUE. Ignored if phenotype is binary.
`covar`	(n x p) covariate matrix. Defaults to an (n x 1) intercept.
`indicator`	Convert raw counts to indicators?
`is_pheno_binary`	Is the phenotype binary? Default: FALSE.
`method`	Method for aggregating across categories: "none", "max", "sum". Default: "none".
`score_test`	Run a score test? If FALSE, performs a Wald test.
`weights`	(3 x 1) annotation category weights.

Value

Numeric p-value.

Examples

```
# Generate data.
data <- DGP(n = 1e3, snps = 1e2)

# Run the Allelic Series Burden Test.
# Note: the output is a scalar p-value.
results <- ASBT(
  anno = data$anno,
  geno = data$geno,
  pheno = data$pheno,
  covar = data$covar
)
```

ASKAT *Allelic Series SKAT Test*

Description

Sequence kernel association test (SKAT) with allelic series weights.

Usage

```
ASKAT(
  anno,
  geno,
  pheno,
  apply_int = TRUE,
  covar = NULL,
  is_pheno_binary = FALSE,
  return_null_model = FALSE,
  weights = DEFAULT_WEIGHTS
)
```

Arguments

`anno`	(snps x 1) annotation vector with values in c(0, 1, 2).
`geno`	(n x snps) genotype matrix.
`pheno`	(n x 1) phenotype vector.
`apply_int`	Apply rank-based inverse normal transform to the phenotype? Default: TRUE. Ignored if phenotype is binary.
`covar`	(n x p) covariate matrix. Defaults to an (n x 1) intercept.
`is_pheno_binary`	Is the phenotype binary? Default: FALSE.
`return_null_model`	Return the null model in addition to the p-value? Useful if running additional SKAT tests. Default: FALSE.
`weights`	(3 x 1) annotation category weights.

Value

If `return_null_model`, a list containing the p-value and the SKAT null model. Otherwise, a numeric p-value.

Examples

```
# Generate data.
data <- DGP(n = 1e3, snps = 1e2)

# Run the Allelic Series SKAT Test.
# Note: the output is a scalar p-value.
results <- ASKAT(
  anno = data$anno,
  geno = data$geno,
  pheno = data$pheno,
  covar = data$covar
)
```

CalcRegParam *Calculate Regression Parameters*

Description

Calculate phenotypic regression coefficients and the residual variation based on proportion of variation explained (PVE) by each factor. Note that the proportion of variation explained by genotype is required, but genetic effects are not generated here.

Usage

```
CalcRegParam(pve_age = 0.1, pve_pcs = 0.2, pve_sex = 0.1)
```

Arguments

`pve_age`	PVE by age.
`pve_pcs`	PVE by PCs (collectively).
`pve_sex`	PVE by sex.

Value

List containing the (5 x 1) regression coefficient vector "coef" and the residual standard deviation "sd".

CheckInputs *Check Inputs*

Description

Check Inputs

Usage

```
CheckInputs(anno, covar, geno, is_pheno_binary, pheno, weights)
```

Arguments

anno	(snps x 1) annotation vector.
covar	(n x p) covariate matrix.
geno	(n x snps) genotype matrix.
is_pheno_binary	Is the phenotype binary?
pheno	(n x 1) phenotype vector.
weights	(3 x 1) annotation category weights.

Value

None.

COAST *COding-variant Allelic Series Test*

Description

Main allelic series test. Performs both Burden and SKAT type tests, then combines the results to calculate an omnibus p-value.

Usage

```
COAST(
  anno,
  geno,
  pheno,
  apply_int = TRUE,
  covar = NULL,
  include_orig_skato_all = FALSE,
  include_orig_skato_ptv = FALSE,
  is_pheno_binary = FALSE,
  return_omni_only = FALSE,
  score_test = FALSE,
  weights = DEFAULT_WEIGHTS
)
```

Arguments

`anno` (snps x 1) annotation vector with values in c(0, 1, 2).

`geno` (n x snps) genotype matrix.

`pheno` (n x 1) phenotype vector.

`apply_int` Apply rank-based inverse normal transform to the phenotype? Default: TRUE. Ignored if phenotype is binary.

`covar` (n x p) covariate matrix. Defaults to an (n x 1) intercept.

`include_orig_skato_all` Include the original version of SKAT-O applied to all variants in the omnibus test? Default: FALSE.

`include_orig_skato_ptv` Include the original version of SKAT-O applied to PTV variants only in the omnibus test? Default: FALSE.

`is_pheno_binary` Is the phenotype binary? Default: FALSE.

`return_omni_only` Return only the omnibus p-value? Default: FALSE.

`score_test` Use a score test for burden analysis? If FALSE, uses a Wald test.

`weights` (3 x 1) annotation category weights.

Value

Numeric p-value.

Examples

```
# Generate data.
data <- DGP(n = 1e3, snps = 1e2)

# Run the COding-variant Allelic Series Test.
results <- COAST(
  anno = data$anno,
  geno = data$geno,
  pheno = data$pheno,
  covar = data$covar
)
show(results)
```

Comparator *Comparator Test*

Description

Runs burden, SKAT, and SKAT-O, using default settings.

Usage

```
Comparator(covar, geno, pheno, apply_int = TRUE, is_pheno_binary = FALSE)
```

Arguments

covar	(n x p) covariate matrix.
geno	(n x snps) genotype matrix.
pheno	(n x 1) phenotype vector.
apply_int	Apply rank-based inverse normal transform to the phenotype? Default: TRUE. Ignored if phenotype is binary.
is_pheno_binary	Is the phenotype binary? Default: FALSE.

Value

Numeric vector of p-values.

Examples

```
# Generate data.
data <- DGP(n = 1e3, snps = 1e2)

# Run the comparators.
results <- Comparator(
  geno = data$geno,
  pheno = data$pheno,
  covar = data$covar
)
```

DGP *Data Generating Process*

Description

Generate a data set consisting of:

- "anno"A SNP-length annotation vector.
- "covar"A subject by 6 covariate matrix.
- "geno"A subject by SNP genotype matrix.
- "pheno"A subject-length phenotype vector.

Usage

```
DGP(
  anno = NULL,
  beta = c(0, 1, 2),
  binary = FALSE,
  geno = NULL,
  include_residual = TRUE,
  indicator = FALSE,
  maf_range = c(0.005, 0.01),
  method = "none",
  n = 100,
  p_dmv = 0.4,
  p_ptv = 0.1,
  prop_causal = 1,
  random_signs = FALSE,
  random_var = 0,
  snps = 100,
  weights = c(1, 2, 3)
)
```

Arguments

`anno` Annotation vector, if providing genotypes. Should match the number of columns in geno.

`beta` If method = "none", a (3 x 1) coefficient vector for bmvs, dmvs, and ptvs respectively. If method != "none", a scalar effect size.

`binary` Generate binary phenotype? Default: FALSE.

`geno` Genotype matrix, if providing genotypes.

`include_residual` Include residual? If FALSE, returns the expected value. Intended for testing.

`indicator` Convert raw counts to indicators? Default: FALSE.

`maf_range` Range of minor allele frequencies: c(MIN, MAX).

method Genotype aggregation method. Default: "none".

n Sample size.

p_dmv Frequency of deleterious missense variants. Default of 40% is based on the frequency of DMVs among rare coding variants in the UK Biobank.

p_ptv Frequency of protein truncating variants. Default of 10% is based on the frequency of PTVs among rare coding variants in the UK Biobank.

prop_causal Proportion of variants which are causal. Default: 1.0.

random_signs Randomize signs? FALSE for burden-type genetic architecture, TRUE for SKAT-type.

random_var Frailty variance in the case of random signs. Default: 0.

snps Number of SNP in the gene. Default: 100.

weights Aggregation weights.

Value

List containing: genotypes, annotations, covariates, phenotypes.

Examples

```
# Generate data.
data <- DGP(n = 100)

# View components.
table(data$anno)
head(data$covar)
head(data$geno[, 1:5])
hist(data$pheno)
```

FilterGenos *Filter Noncausal Variants*

Description

Remove a random fraction of variants, which are designated non-causal.

Usage

```
FilterGenos(anno, geno, prop_causal = 1)
```

Arguments

anno (snps x 1) annotation vector.

geno (n x snps) genotype matrix.

prop_causal Proportion of variants which are causal.

Value

List containing the (n x snps) genotype matrix "geno" and the (snps x 1) annotation vector "anno".

GenAnno *Generate Genotype Annotations*

Description

Returns a vector of length = the number of columns (SNPs) in the genotype matrix. Each SNP is classified as a benign missense variant (0), a deleterious missense variant (1), or a protein truncating variant (2).

Usage

```
GenAnno(snps, p_dmv = 0.33, p_ptv = 0.33)
```

Arguments

`snps`	Number of SNPs in the gene.
`p_dmv`	Frequency of deleterious missense variants.
`p_ptv`	Frequency of protein truncating variants.

Value

(snps x 1) integer vector.

GenCovar *Generate Covariates*

Description

Generate an (n x 6) covariate matrix with columns representing an intercept, age, sex, and 3 genetic PCs. Because these simulations address rare variant analysis, correlation between genotypes and the genetic PCs (based on common variants) is unnecessary.

Usage

```
GenCovar(n)
```

Arguments

`n`	Sample size.

Value

(n x 6) numeric matrix.

GenGeno *Generate Genotypes*

Description

Generate Genotypes

Usage

```
GenGeno(n, snps, maf_range = c(0.005, 0.01), p_dmv = 0.33, p_ptv = 0.33)
```

Arguments

`n`	Sample size.
`snps`	Number of SNP in the gene.
`maf_range`	Range of minor allele frequencies: c(MIN, MAX).
`p_dmv`	Frequency of deleterious missense variants.
`p_ptv`	Frequency of protein truncating variants.

Value

List containing the (n x snps) genotype matrix "geno" and the (snps x 1) annotation vector "anno".

GenGenoMat *Generate Genotype Matrix*

Description

Generate Genotype Matrix

Usage

```
GenGenoMat(n, snps, maf_range = c(0.005, 0.01))
```

Arguments

`n`	Sample size.
`snps`	Number of SNP in the gene.
`maf_range`	Range of minor allele frequencies: c(MIN, MAX).

Value

(n x snps) numeric matrix.

GenPheno *Generate Phenotypes*

Description

Generate Phenotypes

Usage

```
GenPheno(
  anno,
  beta,
  covar,
  geno,
  reg_param,
  binary = FALSE,
  include_residual = TRUE,
  indicator = FALSE,
  method = "none",
  prop_causal = 1,
  random_signs = FALSE,
  random_var = 0,
  weights = c(0, 1, 2)
)
```

Arguments

`anno` (snps x 1) annotation vector.

`beta` (3 x 1) coefficient vector for bmvs, dmvs, and ptvs respectively.

`covar` Covariate matrix.

`geno` (n x snps) genotype matrix.

`reg_param` Regression parameters.

`binary` Generate binary phenotype? Default: FALSE.

`include_residual` Include residual? If FALSE, returns the expected value. Intended for testing.

`indicator` Convert raw counts to indicators? Default: FALSE.

`method` Genotype aggregation method. Default: "none".

`prop_causal` Proportion of variants which are causal.

`random_signs` Randomize signs? FALSE for burden-type genetic architecture, TRUE for SKAT-type.

`random_var` Frailty variance in the case of random signs. Default: 0.

`weights` Aggregation weights.

Value

(n x 1) numeric vector.

`OLS` *Ordinary Least Squares*

Description

Fits the standard OLS model.

Usage

```
OLS(y, X)
```

Arguments

`y`	(n x 1) Numeric vector.
`X`	(n x p) Numeric matrix.

Value

List containing the following:

- BetaRegression coefficient.
- VOutcome variance.
- SEStandard errors.
- ZZ-scores.

`ResidVar` *Calculate Residual Variance*

Description

Calculate Residual Variance

Usage

```
ResidVar(y, X)
```

Arguments

`y`	(n x 1) Numeric phenotype vector.
`X`	(n x q) Numeric covariate matrix.

Value

Scalar residual variance.

`Score` *Calculate Score Statistic*

Description

Calculate Score Statistic

Usage

```
Score(y, G, X, v)
```

Arguments

`y`	(n x 1) Numeric phenotype vector.
`G`	(n x p) Numeric genotype matrix.
`X`	(n x q) Numeric covariate matrix.
`v`	Scalar residual variance.

Value

Scalar score statistic.

Index