Package ‘qgg’

September 7, 2023

Type Package
Title Statistical Tools for Quantitative Genetic Analyses
Version 1.1.2
Date 2023-09-06
Maintainer Peter Soerensen <peter.sorensen@r-qgg.org>
Description Provides an infrastructure for efficient processing of large-scale genetic and pheno-
typic data including core functions for: 1) fitting linear mixed models, 2) constructing marker-
based genomic relationship matrices, 3) estimating genetic parameters (heritability and correla-
tion), 4) performing genomic prediction and genetic risk profiling, and 5) single or multi-
marker association analyses.
License GPL-3
Encoding UTF-8
Depends R (>= 3.3.0)
Imports Rcpp (>= 1.0.5), data.table, parallel, statmod, stats,  
MCMCpack, MASS, coda, corpcor
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.2.3
URL https://github.com/psoerensen/qgg
BugReports https://github.com/psoerensen/qgg/issues
NeedsCompilation yes
Author Peter Soerensen [aut, cre],  
Palle Duun Rohde [aut],  
Izel Fourie Soerensen [aut]
Repository CRAN
Date/Publication 2023-09-07 11:20:02 UTC
**R topics documented:**

acc .................................................. 2
adjStat ............................................. 3
gbayes ............................................... 4
ggetG .................................................. 8
gfilter ............................................... 9
glma .................................................. 10
gmap .................................................. 13
gprep .................................................. 18
greml ................................................ 20
grm .................................................. 23
gscore ............................................... 25
gsea .................................................. 26
gsim ................................................ 29
gsolve ............................................... 30
ldsc .................................................. 32
mtadj ................................................ 34

**Index**

---

| acc | *Compute prediction accuracy for a quantitative or binary trait* |

**Description**

Compute prediction accuracy for a quantitative or binary trait

**Usage**

```r
acc(yobs = NULL, ypred = NULL, typeoftrait = "quantitative")
```

**Arguments**

- **yobs** is a vector of observed phenotypes
- **ypred** is a vector of predicted phenotypes
- **typeoftrait** is a character with possible values "binary" or "quantitative" (default)
adjStat

Adjustment of marker summary statistics using clumping and thresholding

Description

Adjust marker summary statistics using linkage disequilibrium information from Glist.

Usage

adjStat(
  stat = NULL,
  Glist = NULL,
  chr = NULL,
  statistics = "b",
  r2 = 0.9,
  ldSets = NULL,
  threshold = 1,
  header = NULL,
  method = "pruning"
)

Arguments

stat      A data frame with marker summary statistics (see required format above).
Glist     List of information about genotype matrix stored on disk.
chr       Chromosome(s) being processed.
statistics Specify what type of statistics ("b" or "z") is being processed. Default is "b".
r2        Threshold used in clumping/pruning procedure. Default is 0.9.
ldSets    List of marker sets - names correspond to row names in 'stat'.
threshold P-value threshold used in clumping procedure. Default is 1.
header    Character vector with column names to be excluded in the LD adjustment.
method    Method used in adjustment for linkage disequilibrium. Default is "clumping".

Details

Required input format for summary statistics:
stat can be a data.frame(rsids, chr, pos, ea, nea, eaf, b, seb, stat, p, n) (single trait)
stat can be a list(marker=(rsids, chr, pos, ea, nea, eaf), b, seb, stat, p, n) (multiple trait)
For details about the summary statistics format, see the main function description.

Author(s)

Peter Soerensen
Bayesian linear regression (BLR) models:
- unified mapping of genetic variants, estimation of genetic parameters (e.g. heritability) and prediction of disease risk
- handles different genetic architectures (few large, many small effects)
- scale to large data (e.g. sparse LD)

In the Bayesian multiple regression model the posterior density of the model parameters depend on the likelihood of the data given the parameters and a prior probability for the model parameters

The prior density of marker effects defines whether the model will induce variable selection and shrinkage or shrinkage only. Also, the choice of prior will define the extent and type of shrinkage induced. Ideally the choice of prior for the marker effect should reflect the genetic architecture of the trait, and will vary (perhaps a lot) across traits.

The following prior distributions are provided:

Bayes N: Assigning a Gaussian prior to marker effects implies that the posterior means are the BLUP estimates (same as Ridge Regression).
Bayes L: Assigning a double-exponential or Laplace prior is the density used in the Bayesian LASSO
Bayes A: similar to ridge regression but t-distribution prior (rather than Gaussian) for the marker effects; variance comes from an inverse-chi-square distribution instead of being fixed. Estimation via Gibbs sampling.
Bayes C: uses a “rounded spike” (low-variance Gaussian) at origin many small effects can contribute to polygenic component, reduces the dimensionality of the model (makes Gibbs sampling feasible).
Bayes R: Hierarchical Bayesian mixture model with 4 Gaussian components, with variances scaled by 0, 0.0001, 0.001, and 0.01.

Usage

gbayes(
  y = NULL,
  X = NULL,
  W = NULL,
  stat = NULL,
  covs = NULL,
  trait = NULL,
  fit = NULL,
  Glist = NULL,
  chr = NULL,
  rsids = NULL,
Arguments

y is a vector or matrix of phenotypes
X is a matrix of covariates
W is a matrix of centered and scaled genotypes
stat dataframe with marker summary statistics
covs is a list of summary statistics (output from internal cvs function)
trait is an integer used for selection traits in covs object
fit is a list of results from gbayes
Glist is a list of information about genotype matrix stored on disk
chr is the chromosome for which to fit BLR models
rsids is a character vector of rsids
b is a vector or matrix of marginal marker effects
bm is a vector or matrix of adjusted marker effects for the BLR model
seb is a vector or matrix of standard error of marginal effects
LD is a list with sparse LD matrices
n is a scalar or vector of number of observations for each trait
formatLD is a character specifying LD format (formatLD="dense" is default)
vG is a scalar or matrix of genetic (co)variances
vb is a scalar or matrix of marker (co)variances
ve is a scalar or matrix of residual (co)variances
ssg_prior is a scalar or matrix of prior genetic (co)variances
ssb_prior is a scalar or matrix of prior marker (co)variances
sse_prior is a scalar or matrix of prior residual (co)variances
lambda is a vector or matrix of lambda values
scaleY is a logical; if TRUE y is centered and scaled
h2 is the trait heritability
pi is the proportion of markers in each marker variance class (e.g. pi=c(0.999,0.001),used if method="ssvs")
updateB is a logical for updating marker (co)variances
updateG is a logical for updating genetic (co)variances
updateE is a logical for updating residual (co)variances
updatePi is a logical for updating pi
adjustE is a logical for adjusting residual variance
models is a list structure with models evaluated in bayesC
nug is a scalar or vector of prior degrees of freedom for prior genetic (co)variances
nub is a scalar or vector of prior degrees of freedom for marker (co)variances
nue is a scalar or vector of prior degrees of freedom for prior residual (co)variances
verbose is a logical; if TRUE it prints more details during iteration
msize number of markers used in computation of sparseld
mask is a vector or matrix of TRUE/FALSE specifying if marker should be ignored
GRMlist is a list providing information about GRM matrix stored in binary files on disk
ve_prior is a scalar or matrix of prior residual (co)variances
vg_prior is a scalar or matrix of prior genetic (co)variances
tol is tolerance, i.e. convergence criteria used in gbayes
nit is the number of iterations
nburn is the number of burnin iterations
nit_local is the number of local iterations
nit_global is the number of global iterations
method specifies the methods used (method=“bayesN”,“bayesA”,“bayesL”,“bayesC”,“bayesR”)
algorithm specifies the algorithm

Value

Returns a list structure including

b vector or matrix (mxt) of posterior means for marker effects
d vector or matrix (mxt) of posterior means for marker inclusion probabilities
vb scalar or vector (t) of posterior means for marker variances
g scalar or vector (t) of posterior means for genomic variances
ve scalar or vector (t) of posterior means for residual variances
rb matrix (txt) of posterior means for marker correlations
rg matrix (txt) of posterior means for genomic correlations
re matrix (txt) of posterior means for residual correlations
pi vector (1xnmodels) of posterior probabilities for models
h2 vector (1xt) of posterior means for model probability
param a list current parameters (same information as item listed above) used for restart of the analysis
stat matrix (mxt) of marker information and effects used for genomic risk scoring

Author(s)

Peter Sørensen

Examples

# Simulate data and test functions

W <- matrix(rnorm(100000),nrow=1000)
set1 <- sample(1:ncol(W),5)
set2 <- sample(1:ncol(W),5)
sets <- list(set1,set2)
g <- rowSums(W[,c(set1,set2)])
e <- rnorm(nrow(W),mean=0,sd=1)
y <- g + e
```r
fitM <- gbayes(y=y, W=W, method="bayesN")
fitA <- gbayes(y=y, W=W, method="bayesA")
fitL <- gbayes(y=y, W=W, method="bayesL")
fitC <- gbayes(y=y, W=W, method="bayesC")
```

---

### getG

*Get elements from genotype matrix stored in PLINK bedfiles*

#### Description

Extracts specific rows (based on ids or row numbers) and columns (based on rsids or column numbers) from a genotype matrix stored on disk. The extraction is based on provided arguments such as chromosome number, ids, rsids, etc. Genotypes can be optionally scaled and imputed.

#### Usage

```r
getG(
  Glist = NULL,
  chr = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
  famfiles = NULL,
  ids = NULL,
  rsids = NULL,
  rws = NULL,
  cls = NULL,
  impute = TRUE,
  scale = FALSE
)
```

#### Arguments

- **Glist**: A list structure containing information about genotypes stored on disk.
- **chr**: An integer representing the chromosome for which the genotype matrix is to be extracted. It is required.
- **bedfiles**: A vector of filenames for the PLINK bed-file.
- **bimfiles**: A vector of filenames for the PLINK bim-file.
- **famfiles**: A vector of filenames for the PLINK fam-file.
- **ids**: A vector of individual IDs for whom the genotype data needs to be extracted.
- **rsids**: A vector of SNP identifiers for which the genotype data needs to be extracted.
- **rws**: A vector of row numbers to be extracted from the genotype matrix.
- **cls**: A vector of column numbers to be extracted from the genotype matrix.
**gfilter**

impute  A logical or integer. If TRUE, missing genotypes are replaced with their expected values (2 times the allele frequency). If set to an integer, missing values are replaced by that integer.

scale   A logical. If TRUE, the genotype markers are scaled to have a mean of zero and variance of one.

**Details**

This function facilitates the extraction of specific genotype data from storage based on various criteria. The extracted genotype data can be optionally scaled or imputed. If rsids are provided that are not found in the 'Glist', a warning is raised.

**Value**

A matrix with extracted genotypic data. Rows correspond to individuals, and columns correspond to SNPs. Row names are set to individual IDs, and column names are set to rsids.

---

**gfilter**  
*Filter genetic marker data based on different quality measures*

**Description**

Quality control is a critical step for working with summary statistics (in particular for external). Processing and quality control of GWAS summary statistics includes:

- map marker ids (rsids/cpra (chr, pos, ref, alt)) to LD reference panel data
- check effect allele (flip EA, EAF, Effect)
- check effect allele frequency
- thresholds for MAF and HWE
- exclude INDELS, CG/AT and MHC region
- remove duplicated marker ids
- check which build version
- check for concordance between marker effect and LD data

External summary statistics format: marker, chr, pos, effect_allele, non_effect_allele, effect_allele_freq, effect, effect_se, stat, p, n

Internal summary statistics format: rsids, chr, pos, a1, a2, af, b, seb, stat, p, n

**Usage**

```r
  gfilter(
      Glist = NULL,
      excludeMAF = 0.01,
      excludeMISS = 0.05,
      excludeINFO = NULL,
      excludeCGAT = TRUE,
      excludeINDEL = TRUE,
      excludeDUPS = TRUE,
      excludeHWE = 1e-12,
      excludeMHC = FALSE,
      assembly = "GRCh37"
  )
```
Arguments

Glist A list containing information about the genotype matrix stored on disk.
excludeMAF A scalar threshold. Exclude markers with a minor allele frequency (MAF) below this threshold. Default is 0.01.
excludeMISS A scalar threshold. Exclude markers with missingness (MISS) above this threshold. Default is 0.05.
excludeINFO A scalar threshold. Exclude markers with an info score (INFO) below this threshold. Default is 0.8.
excludeCGAT A logical value; if TRUE exclude markers if the alleles are ambiguous (i.e., either CG or AT combinations).
excludeINDEL A logical value; if TRUE exclude markers that are insertions or deletions (INDELs).
excludeDUPS A logical value; if TRUE exclude markers if their identifiers are duplicated.
excludewHWE A scalar threshold. Exclude markers where the p-value for the Hardy-Weinberg Equilibrium test is below this threshold. Default is 0.01.
excludemHC A logical value; if TRUE exclude markers located within the MHC region.
assembly A character string indicating the name of the genome assembly (e.g., "GRCh38").

Author(s)

Peter Soerensen

---

glma Single marker association analysis using linear models or linear mixed models

Description

The function glma performs single marker association analysis between genotype markers and the phenotype either based on linear model analysis (LMA) or mixed linear model analysis (MLMA).
The basic MLMA approach involves 1) building a genetic relationship matrix (GRM) that models genome-wide sample structure, 2) estimating the contribution of the GRM to phenotypic variance using a random effects model (with or without additional fixed effects) and 3) computing association statistics that account for this component on phenotypic variance.
MLMA methods are the method of choice when conducting association mapping in the presence of sample structure, including geographic population structure, family relatedness and/or cryptic relatedness. MLMA methods prevent false positive associations and increase power. The general recommendation when using MLMA is to exclude candidate markers from the GRM. This can be efficiently implemented via a leave-one-chromosome-out analysis. Further, it is recommend that analyses of randomly ascertained quantitative traits should include all markers (except for the candidate marker and markers in LD with the candidate marker) in the GRM, except as follows. First, the set of markers included in the GRM can be pruned by LD to reduce running time (with association statistics still computed for all markers). Second, genome-wide significant markers of
large effect should be conditioned out as fixed effects or as an additional random effect (if a large number of associated markers). Third, when population stratification is less of a concern, it may be useful using the top associated markers selected based on the global maximum from out-of sample predictive accuracy.

Usage

```r
glma(
  y = NULL,
  X = NULL,
  W = NULL,
  Glist = NULL,
  chr = NULL,
  fit = NULL,
  verbose = FALSE,
  statistic = "mastor",
  ids = NULL,
  rsids = NULL,
  msize = 100,
  scale = TRUE
)
```

Arguments

- `y`: vector or matrix of phenotypes
- `X`: design matrix for factors modeled as fixed effects
- `W`: matrix of centered and scaled genotypes
- `Glist`: list of information about genotype matrix stored on disk
- `chr`: chromosome for which summary statistics are computed
- `fit`: list of information about linear mixed model fit (output from greml)
- `verbose`: is a logical; if TRUE it prints more details during optimization
- `statistic`: single marker test statistic used (currently based on the "mastor" statistics).
- `ids`: vector of individuals used in the analysis
- `rsids`: vector of marker rsids used in the analysis
- `msize`: number of genotype markers used for batch processing
- `scale`: logical if TRUE the genotypes have been scaled to mean zero and variance one

Value

Returns a dataframe (if number of traits = 1) else a list including

- `coef`: single marker coefficients
- `se`: standard error of coefficients
- `stat`: single marker test statistic
- `p`: p-value
Author(s)

Peter Soerensen

References


Examples

```r
# Simulate data
W <- matrix(rnorm(1000000), ncol = 1000)
colnames(W) <- as.character(1:ncol(W))
rownames(W) <- as.character(1:nrow(W))
y <- rowSums(W[, 1:10]) + rowSums(W[, 501:510]) + rnorm(nrow(W))

# Create model
data <- data.frame(y = y, mu = 1)
```
In the Bayesian multiple regression model, the posterior density of the model parameters depends on the likelihood of the data given the parameters and a prior probability for the model parameters. The choice of the prior for marker effects can influence the type and extent of shrinkage induced in the model.

Usage

gmap(
y = NULL,
X = NULL,
W = NULL,
stat = NULL,
trait = NULL,
sets = NULL,
fit = NULL,
Glist = NULL,
chr = NULL,
rsids = NULL,
ids = NULL,
b = NULL,
null, mask = NULL, LD = NULL, n = NULL, vg = NULL, vb = NULL, ve = NULL, ssg_prior = NULL, ssb_prior = NULL, sse_prior = NULL, lambda = NULL, scaleY = TRUE, shrinkLD = FALSE, shrinkCor = FALSE, formatLD = "dense", pruneLD = TRUE, r2 = 0.05, checkLD = TRUE, h2 = NULL, p1 = 0.001, updateB = TRUE, updateG = TRUE, updateE = TRUE, updatePi = TRUE, adjustE = TRUE, models = NULL, checkConvergence = FALSE, critVe = 3, critVg = 5, critVb = 5, critPi = 3, ntrial = 1, nug = 4, nub = 4, nue = 4, verbose = FALSE, msize = 100, threshold = NULL, ve_prior = NULL, vg_prior = NULL, tol = 0.001, nit = 100, nburn = 50, nit_local = NULL, nit_global = NULL, method = "bayesC", algorithm = "mcmc"
Arguments

y  A vector or matrix of phenotypes.
X  A matrix of covariates.
W  A matrix of centered and scaled genotypes.
stat Dataframe with marker summary statistics.
trait Integer used for selection traits in covs object.
sets A list of character vectors where each vector represents a set of items. If the names of the sets are not provided, they are named as "Set1", "Set2", etc.
fit List of results from gbayes.
Glist List of information about genotype matrix stored on disk.
chr Chromosome for which to fit BLR models.
rsids Character vector of rsids.
ids vector of individuals used in the study
b  Vector or matrix of marginal marker effects.
bm Vector or matrix of adjusted marker effects for the BLR model.
seb Vector or matrix of standard error of marginal effects.
mask Vector or matrix specifying if marker should be ignored.
LD List with sparse LD matrices.
n  Scalar or vector of number of observations for each trait.
vg Scalar or matrix of genetic (co)variances.
vb Scalar or matrix of marker (co)variances.
ve Scalar or matrix of residual (co)variances.
ssg_prior Scalar or matrix of prior genetic (co)variances.
ssb_prior Scalar or matrix of prior marker (co)variances.
sse_prior Scalar or matrix of prior residual (co)variances.
lambda Vector or matrix of lambda values
scaleY Logical indicating if y should be scaled.
shrinkLD Logical indicating if LD should be shrunk.
shrinkCor Logical indicating if cor should be shrunk.
formatLD Character specifying LD format (default is "dense").
pruneLD Logical indicating if LD pruning should be applied.
r2 Scalar providing value for r2 threshold used in pruning
checkLD Logical indicating if LD matches summary statistics.
h2 Trait heritability.
pi Proportion of markers in each marker variance class.
updateB Logical indicating if marker (co)variances should be updated.
updateG Logical indicating if genetic (co)variances should be updated.
updateE Logical indicating if residual (co)variances should be updated.
updatePi Logical indicating if pi should be updated.
adjustE Logical indicating if residual variance should be adjusted.
models List structure with models evaluated in bayesC.
checkConvergence Logical indicating if convergences should be checked.
critVe Scalar providing value for z-score threshold used in checking convergence for Ve

critVg Scalar providing value for z-score threshold used in checking convergence for Vg

critVb Scalar providing value for z-score threshold used in checking convergence for Vg

critPi Scalar providing value for z-score threshold used in checking convergence for Pi

ntrial Integer providing number of trials used if convergence is not obtaines
	nug Scalar or vector of prior degrees of freedom for genetic (co)variances.

nub Scalar or vector of prior degrees of freedom for marker (co)variances.

nue Scalar or vector of prior degrees of freedom for residual (co)variances.

verbose Logical; if TRUE, it prints more details during iteration.

msize Integer providing number of markers used in computation of sparseld

threshold Scalar providing value for threshold used in adjustment of B

ve_prior Scalar or matrix of prior residual (co)variances.

vg_prior Scalar or matrix of prior genetic (co)variances.

tol Convergence criteria used in gbayes.

nit Number of iterations.

nburn Number of burnin iterations.

nit_local Number of local iterations.

nit_global Number of global iterations.

method Method used (e.g. "bayesN","bayesA","bayesL","bayesC","bayesR").

algorithm Specifies the algorithm.

Details

This function implements Bayesian linear regression models to provide unified mapping of genetic variants, estimate genetic parameters (e.g. heritability), and predict disease risk. It is designed to handle various genetic architectures and scale efficiently with large datasets.
Value

A list containing:

- bm Vector or matrix of posterior means for marker effects.
- dm Vector or matrix of posterior means for marker inclusion probabilities.
- vb Scalar or vector of posterior means for marker variances.
- vg Scalar or vector of posterior means for genomic variances.
- ve Scalar or vector of posterior means for residual variances.
- rb Matrix of posterior means for marker correlations.
- rg Matrix of posterior means for genomic correlations.
- re Matrix of posterior means for residual correlations.
- pi Vector of posterior probabilities for models.
- h2 Vector of posterior means for model probability.
- param List of current parameters used for restarting the analysis.
- stat Matrix of marker information and effects used for genomic risk scoring.

Author(s)

Peter Sørensen

Examples

# Plink bed/bim/fam files
bedfiles <- system.file("extdata", paste0("sample_chr",1:2,".bed"), package = "qgg")
bimfiles <- system.file("extdata", paste0("sample_chr",1:2,".bim"), package = "qgg")
famfiles <- system.file("extdata", paste0("sample_chr",1:2,".fam"), package = "qgg")

# Prepare Glist
Glist <- gprep(study="Example", bedfiles=bedfiles, bimfiles=bimfiles, famfiles=famfiles)

# Simulate phenotype
sim <- gsim(Glist=Glist, chr=1, nt=1)

# Compute single marker summary statistics
stat <- glma(y=sim$y, Glist=Glist, scale=FALSE)
str(stat)

# Define fine-mapping regions
sets <- Glist$rsids
Glist$chr[[1]] <- gsub("21","1",Glist$chr[[1]])
Glist$chr[[2]] <- gsub("22","2",Glist$chr[[2]])

# Fine map
fit <- gmap(Glist=Glist, stat=stat, sets=sets, verbose=FALSE, method="bayesC", nit=1500, nburn=500, pi=0.001)
fit$post # Posterior inference for every fine-mapped region
fit$conv # Convergence statistics for every fine-mapped region

# Posterior inference for marker effect
head(fit$stat)

gprep

 Prepare genotype data for all statistical analyses

Description

All functions in qgg relies on a simple data infrastructure that takes five main input sources; phenotype data (y), covariate data (X), genotype data (G or Glist), a genomic relationship matrix (GRM or GRMlist) and genetic marker sets (sets). The genotypes are stored in a matrix (n x m (individuals x markers)) in memory (G) or in a binary file on disk (Glist).

It is only for small data sets that the genotype matrix (G) can stored in memory. For large data sets the genotype matrix has to stored in a binary file on disk (Glist). Glist is as a list structure that contains information about the genotypes in the binary file. The gprep function prepares the Glist, and is required for downstream analyses of large-scale genetic data. Typically, the Glist is prepared once, and saved as an *.Rdata-file.

The gprep function reads genotype information from binary PLINK files, and creates the Glist object that contains general information about the genotypes such as reference alleles, allele frequencies and missing genotypes, and construct a binary file on the disk that contains the genotypes as allele counts of the alternative allele (memory usage = (n x m)/4 bytes).

The gprep function can also be used to prepare sparse ld matrices. The r2 metric used is the pairwise correlation between markers (allele count alternative allele) in a specified region of the genome. The marker genotype is allele count of the alternative allele which is assumed to be centered and scaled. The Glist structure is used as input parameter for a number of qgg core functions including: 1) construction of genomic relationship matrices (grm), 2) construction of sparse ld matrices, 3) estimating genomic parameters (greml), 4) single marker association analyses (glma), 5) gene set enrichment analyses (gsea), and 6) genomic prediction from genotypes and phenotypes (gsolve) or genotypes and summary statistics (gscore).

Usage

gprep(
  Glist = NULL,
  task = "prepare",
  study = NULL,
  fnBED = NULL,
  ldfiles = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
famfiles = NULL,
mapfiles = NULL,
ids = NULL,
rsids = NULL,
assembly = NULL,
overwrite = FALSE,
msize = 100,
r2 = NULL,
kb = NULL,
km = NULL,
ncores = 1
)

Arguments

Glist A list containing information about the genotype matrix stored on disk.
task A character string specifying the task to perform. Possible tasks are "prepare" (default), "sparseld", "ldscores", "ldsets", and "geneticmap".
study The name of the study.
fnBED Path and filename of the .bed binary file used to store genotypes on disk.
ldfiles Path and filename of the .ld binary files used for storing the sparse LD matrix on disk.
bedfiles A vector of filenames for the PLINK bed-files.
bimfiles A vector of filenames for the PLINK bim-files.
famfiles A vector of filenames for the PLINK fam-files.
mapfiles A vector of filenames for the mapfiles.
ids A vector of individual identifiers used in the study.
rsids A vector of marker rsids used in the study.
assembly Character string indicating the name of the assembly.
overwrite A logical value; if TRUE, the binary genotype/LD file will be overwritten.
msize Number of markers used in the computation of sparseld.
r2 A threshold value (more context might be beneficial, e.g., threshold for what?).
kb Size of the genomic region in kilobases (kb).
km Size of the genomic region in centimorgans (cm).
ncores Number of processing cores to be used for genotype processing.

Value

Returns a list structure (Glist) with information about the genotypes.

Author(s)

Peter Soerensen
Examples

```r
bedfiles <- system.file("extdata", "sample_chr1.bed", package = "qgg")
bimfiles <- system.file("extdata", "sample_chr1.bim", package = "qgg")
famfiles <- system.file("extdata", "sample_chr1.fam", package = "qgg")

Glist <- gprep(study="Example", bedfiles=bedfiles, bimfiles=bimfiles, famfiles=famfiles)
```

Description

The `greml` function is used for the estimation of genomic parameters (co-variance, heritability and correlation) for linear mixed models using restricted maximum likelihood estimation (REML) and genomic prediction using best linear unbiased prediction (BLUP).

The linear mixed model can account for multiple genetic factors (fixed and random genetic marker effects), adjust for complex family relationships or population stratification and adjust for other non-genetic factors including lifestyle characteristics. Different genetic architectures (infinitesimal, few large and many small effects) is accounted for by modeling genetic markers in different sets as fixed or random effects and by specifying individual genetic marker weights. Different genetic models (e.g. additive and non-additive) can be specified by providing additive and non-additive genomic relationship matrices (GRMs) (constructed using `grm`). The GRMs can be accessed from the R environment or from binary files stored on disk facilitating the analyses of large-scale genetic data.

The output contains estimates of variance components, fixed and random effects, first and second derivatives of log-likelihood and the asymptotic standard deviation of parameter estimates.

Assessment of predictive accuracy (including correlation and R2, and AUC for binary phenotypes) can be obtained by providing `greml` with a data frame, or a list that contains sample IDs used in the validation (see examples for details).

Genomic parameters can also be estimated with DMU (http://www.dmu.agrsci.dk/DMU/) if interface = "DMU". This option requires DMU to be installed locally, and the path to the DMU binary files has to be specified (see examples below for details).

Usage

```r
greml(
  y = NULL,
  X = NULL,
  GRMlist = NULL,
  GRM = NULL,
  theta = NULL,
  ids = NULL,
  validate = NULL,
```
maxit = 100,
tol = 1e-05,
bin = NULL,
ncores = 1,
wkdir = getwd(),
verbose = FALSE,
interface = "R",
fm = NULL,
data = NULL
)

Arguments

y is a vector or matrix of phenotypes
X is a design matrix for factors modeled as fixed effects
GRMlist is a list providing information about GRM matrix stored in binary files on disk
GRM is a list of one or more genomic relationship matrices
theta is a vector of initial values of co-variance for REML estimation
ids is a vector of individuals used in the analysis
validate is a data frame or list of individuals used in cross-validation (one column/row for each validation set)
maxit is the maximum number of iterations used in REML analysis
tol is tolerance, i.e. convergence criteria used in REML
bin is the directory for fortran binaries (e.g. DMU binaries dmu1 and dmuai)
ncores is the number of cores used for the analysis
wkdir is the working directory used for REML
verbose is a logical; if TRUE it prints more details during optimization
interface is used for specifying whether to use R or Fortran implementations of REML
fm is a formula with model statement for the linear mixed model
data is a data frame containing the phenotypic observations and fixed factors specified in the model statements

Value

returns a list structure including:

llik log-likelihood at convergence
theta covariance estimates from REML
asd asymptotic standard deviation
b vector of fixed effect estimates
varb vector of variances of fixed effect estimates
g vector or matrix of random effect estimates
e vector or matrix of residual effects
accuracy matrix of prediction accuracies (only returned if [validate?] is provided)
Author(s)

Peter Soerensen

References


Examples

```r
# Simulate data
W <- matrix(rnorm(1000000), ncol = 1000)
colnames(W) <- as.character(1:ncol(W))
rownames(W) <- as.character(1:nrow(W))
y <- rowSums(W[, 1:10]) + rowSums(W[, 501:510]) + rnorm(nrow(W))

# Create model
data <- data.frame(y = y, mu = 1)
fm <- y ~ 0 + mu
X <- model.matrix(fm, data = data)

# Compute GRM
GRM <- grm(W = W)

# REML analyses
fitG <- greml(y = y, X = X, GRM = list(GRM))

# REML analyses and cross validation

# Create marker sets
setsGB <- list(A = colnames(W)) # gblup model
setsGF <- list(C1 = colnames(W)[1:500], C2 = colnames(W)[501:1000]) # gfblup model
setsGT <- list(C1 = colnames(W)[1:10], C2 = colnames(W)[501:510]) # true model
GB <- lapply(setsGB, function(x) {grm(W = W[, x])})
GF <- lapply(setsGF, function(x) {grm(W = W[, x])})
GT <- lapply(setsGT, function(x) {grm(W = W[, x])})

n <- length(y)
fold <- 10
nvalid <- 5

validate <- replicate(nvalid, sample(1:n, as.integer(n / fold)))
cvGB <- greml(y = y, X = X, GRM = GB, validate = validate)
cvGF <- greml(y = y, X = X, GRM = GF, validate = validate)
cvGT <- greml(y = y, X = X, GRM = GT, validate = validate)
```
Computing the genomic relationship matrix (GRM)

Description

The grm function is used to compute a genomic relationship matrix (GRM) based on all, or a subset of marker genotypes. GRM for additive, and non-additive (dominance and epistasis) genetic models can be constructed. The output of the grm function can either be a within-memory GRM object (n x n matrix), or a GRM-list which is a list structure that contains information about the GRM stored in a binary file on the disk.

Usage

grm(
  Glist = NULL,
  GRMlist = NULL,
  ids = NULL,
  rsids = NULL,
  rws = NULL,
  cls = NULL,
  W = NULL,
  method = "add",
  scale = TRUE,
  msize = 100,
  ncores = 1,
  fnG = NULL,
  overwrite = FALSE,
  returnGRM = FALSE,
  miss = NA,
  impute = TRUE,
  pedigree = NULL,
  task = "grm"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glist</td>
<td>list providing information about genotypes stored on disk</td>
</tr>
<tr>
<td>GRMlist</td>
<td>list providing information about GRM matrix stored in binary files on disk</td>
</tr>
<tr>
<td>ids</td>
<td>vector of individuals used for computing GRM</td>
</tr>
<tr>
<td>rsids</td>
<td>vector marker rsids used for computing GRM</td>
</tr>
</tbody>
</table>
grm

rows
rws

rows in genotype matrix used for computing GRM

columns in genotype matrix used for computing GRM

w

matrix of centered and scaled genotypes

method

indicator of method used for computing GRM: additive (add, default), dominance (dom) or epistasis (epi-pairs or epi-hadamard (all genotype markers))

scale

logical if TRUE the genotypes in Glist has been scaled to mean zero and variance one

msize

number of genotype markers used for batch processing

ncores

number of cores used to compute the GRM

fnG

name of the binary file used for storing the GRM on disk

overwrite

logical if TRUE the binary file fnG will be overwritten

returnGRM

logical if TRUE function returns the GRM matrix to the R environment

miss

the missing code (miss=NA is default) used for missing values in the genotype data

impute

if missing values in the genotype matrix W then mean impute

pedigree

is a dataframe with pedigree information

task

either computation of GRM (task="grm" which is default) or eigenvalue decomposition of GRM (task="eigen")

Value

Returns a genomic relationship matrix (GRM) if returnGRM=TRUE else a list structure (GRMlist) with information about the GRM stored on disk

Author(s)

Peter Soerensen

Examples

# Simulate data
W <- matrix(rnorm(1000000), ncol = 1000)
colnames(W) <- as.character(1:ncol(W))
rownames(W) <- as.character(1:nrow(W))

# Compute GRM
GRM <- grm(W = W)

# Eigen value decompostion GRM
eig <- grm(GRM=GRM, task="eigen")
gscore

Genomic scoring based on single marker summary statistics

Description

Computes genomic predictions using single marker summary statistics and observed genotypes.

Usage

```r
gscore(
  Glist = NULL,
  chr = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
  famfiles = NULL,
  stat = NULL,
  fit = NULL,
  ids = NULL,
  scaleMarker = TRUE,
  scaleGRS = TRUE,
  impute = TRUE,
  msize = 100,
  ncores = 1,
  verbose = FALSE
)
```

Arguments

- **Glist**: List of information about genotype matrix. Default is NULL.
- **chr**: Chromosome for which genomic scores is computed. Default is NULL.
- **bedfiles**: Names of the PLINK bed-files. Default is NULL.
- **bimfiles**: Names of the PLINK bim-files. Default is NULL.
- **famfiles**: Names of the PLINK fam-files. Default is NULL.
- **stat**: Matrix of single marker effects. Default is NULL.
- **fit**: Fit object output from gbayes. Default is NULL.
- **ids**: Vector of individuals used in the analysis. Default is NULL.
- **scaleMarker**: Logical; if TRUE the genotype markers are scaled to mean zero and variance one. Default is TRUE.
- **scaleGRS**: Logical; if TRUE the GRS are scaled to mean zero and variance one. Default is TRUE.
- **impute**: Logical; if TRUE, missing genotypes are set to its expected value (2*af where af is allele frequency). Default is TRUE.
- **msize**: Number of genotype markers used for batch processing. Default is 100.
ncores  Number of cores used in the analysis. Default is 1.
verbose Logical; if TRUE, more details are printed during optimization. Default is FALSE.

Value

Returns the genomic scores based on the provided parameters.

Author(s)

Peter Soerensen

Examples

```r
## Plink bed/bim/fam files
bedfiles <- system.file("extdata", paste0("sample_chr",1:2,".bed"), package = "qgg")
bimfiles <- system.file("extdata", paste0("sample_chr",1:2,".bim"), package = "qgg")
famfiles <- system.file("extdata", paste0("sample_chr",1:2,".fam"), package = "qgg")

# Summarize bed/bim/fam files
Glist <- gprep(study="Example", bedfiles=bedfiles, bimfiles=bimfiles, famfiles=famfiles)

# Simulate phenotype
sim <- gsim(Glist=Glist, chr=1, nt=1)

# Compute single marker summary statistics
stat <- glma(y=sim$y, Glist=Glist, scale=FALSE)

# Compute genomic scores
gsc <- gscore(Glist = Glist, stat = stat)
```

Description

The function `gsea` can perform several different gene set enrichment analyses. The general procedure is to obtain single marker statistics (e.g. summary statistics), from which it is possible to compute and evaluate a test statistic for a set of genetic markers that measures a joint degree of association between the marker set and the phenotype. The marker set is defined by a genomic feature such as genes, biological pathways, gene interactions, gene expression profiles etc.

Currently, four types of gene set enrichment analyses can be conducted with `gsea`; sum-based, count-based, score-based, and our own developed method, the covariance association test (CVAT). For details and comparisons of test statistics consult doi:10.1534/genetics.116.189498.

The sum test is based on the sum of all marker summary statistics located within the feature set. The single marker summary statistics can be obtained from linear model analyses (from PLINK or using
the qgg qlma approximation), or from single or multiple component REML analyses (GBLUP or GFBLUP) from the greml function. The sum test is powerful if the genomic feature harbors many genetic markers that have small to moderate effects.

The count-based method is based on counting the number of markers within a genomic feature that show association (or have single marker p-value below a certain threshold) with the phenotype. Under the null hypothesis (that the associated markers are picked at random from the total number of markers, thus, no enrichment of markers in any genomic feature) it is assumed that the observed count statistic is a realization from a hypergeometric distribution.

The score-based approach is based on the product between the scaled genotypes in a genomic feature and the residuals from the linear mixed model (obtained from greml).

The covariance association test (CVAT) is derived from the fit object from greml (GBLUP or GFBLUP), and measures the covariance between the total genomic effects for all markers and the genomic effects of the markers within the genomic feature.

The distribution of the test statistics obtained from the sum-based, score-based and CVAT is unknown, therefore a circular permutation approach is used to obtain an empirical distribution of test statistics.

Usage

gsea(
  stat = NULL,
  sets = NULL,
  Glist = NULL,
  W = NULL,
  fit = NULL,
  g = NULL,
  e = NULL,
  threshold = 0.05,
  method = "sum",
  nperm = 1000,
  ncores = 1
)

Arguments

stat vector or matrix of single marker statistics (e.g. coefficients, t-statistics, p-values)
sets list of marker sets - names corresponds to row names in stat
Glist list providing information about genotypes stored on disk
W matrix of centered and scaled genotypes (used if method = cvat or score)
fit list object obtained from a linear mixed model fit using the greml function
g vector (or matrix) of genetic effects obtained from a linear mixed model fit (GBLUP of GFBLUP)
e vector (or matrix) of residual effects obtained from a linear mixed model fit (GBLUP of GFBLUP)
threshold used if method='hyerp' (threshold=0.05 is default)
method including sum, cvat, hyperg, score
nperm number of permutations used for obtaining an empirical p-value
ncores number of cores used in the analysis

Value

Returns a dataframe or a list including

stat marker set test statistics
m number of markers in the set
p enrichment p-value for marker set

Author(s)

Peter Soerensen

Examples

# Simulate data
W <- matrix(rnorm(1000000), ncol = 1000)
colnames(W) <- as.character(1:ncol(W))
rownames(W) <- as.character(1:nrow(W))
y <- rowSums(W[, 1:10]) + rowSums(W[, 501:510]) + rnorm(nrow(W))

# Create model
data <- data.frame(y = y, mu = 1)
fmu <- y ~ 0 + mu
X <- model.matrix(fm, data = data)

# Single marker association analyses
stat <- glma(y=y,X=X,W=W)

# Create marker sets
f <- factor(rep(1:100,each=10), levels=1:100)
sets <- split(as.character(1:1000),f=f)

# Set test based on sums
b2 <- stat[,“stat”]**2
names(b2) <- rownames(stat)
mma <- gsea(stat = b2, sets = sets, method = “sum”, nperm = 100)
head(mma)

# Set test based on hyperG
p <- stat[,“p”]
names(p) <- rownames(stat)
mma <- gsea(stat = p, sets = sets, method = “hyperg”, threshold = 0.05)
head(mma)
G <- grm(W=W)
fit <- grem1(y=y, X=X, GRM=list(G=G), theta=c(10,1))

# Set test based on cvat
mma <- gsea(W=W, fit = fit, sets = sets, nperm = 1000, method="cvat")
head(mma)

# Set test based on score
mma <- gsea(W=W, fit = fit, sets = sets, nperm = 1000, method="score")
head(mma)

---

gsim

Genomic simulation

Description
Simulate Genotype and Phenotype Data

Usage

gsim(Glist = NULL, chr = 1, nt = 1, W = NULL, n = 1000, m = 1000, rsids = NULL)

Arguments

- **Glist**: A list of information about the genotype matrix. Default is ‘NULL’.
- **chr**: The chromosome(s) being used in the simulation. Default is 1.
- **nt**: Number of traits. Default is 1.
- **W**: Matrix of centered and scaled genotypes. Default is ‘NULL’.
- **n**: Number of individuals. Default is 1000.
- **m**: Number of markers. Default is 1000.
- **rsids**: A character vector of rsids. Default is ‘NULL’.

Details
This function simulates genotype and phenotype data based on the ‘Glist’, which is information about the genotype matrix.

Value
A list containing:

- **y**: Phenotypes.
- **W**: Matrix of centered and scaled genotypes.
- **e**: Errors.
gsolve

Solve linear mixed model equations

description

The gsolve function is used for solving of linear mixed model equations. The algorithm used to solve the equation system is based on a Gauss-Seidel (GS) method (matrix-free with residual updates) that handles large data sets.

The linear mixed model fitted can account for multiple traits, multiple genetic factors (fixed or random genetic marker effects), adjust for complex family relationships or population stratification, and adjust for other non-genetic factors including lifestyle characteristics. Different genetic architectures (infinitesimal, few large and many small effects) is accounted for by modeling genetic markers in different sets as fixed or random effects and by specifying individual genetic marker weights.

Usage

gsolve(
y = NULL,
X = NULL,
GRM = NULL,


va = NULL,
ve = NULL,
Glist = NULL,
W = NULL,
ids = NULL,
rsids = NULL,
sets = NULL,
scale = TRUE,
lambda = NULL,
weights = FALSE,
maxit = 500,
tol = 1e-05,
method = "gsru",
ncores = 1
)

Arguments

y      vector or matrix of phenotypes
X      design matrix of fixed effects
GRM    genetic relationship matrix
va     genetic variance
ve     residual variance
Glist  list of information about genotype matrix stored on disk
W      matrix of centered and scaled genotypes
ids    vector of individuals used in the analysis
rsids  vector of marker rsids used in the analysis
sets   list containing marker sets rsids
scale  logical if TRUE the genotypes in Glist will be scaled to mean zero and variance one
lambda overall shrinkage factor
weights vector of single marker weights used in BLUP
maxit  maximum number of iterations used in the Gauss-Seidel procedure
tol    tolerance, i.e. the maximum allowed difference between two consecutive iterations of the solver to declare convergence
method used in solver (currently only methods="gsru": gauss-seidel with residual update)
ncores number of cores used in the analysis

Author(s)

Peter Soerensen
Examples

```r
# Simulate data
W <- matrix(rnorm(1000000), ncol = 1000)
colnames(W) <- as.character(1:ncol(W))
rownames(W) <- as.character(1:nrow(W))
m <- ncol(W)
causal <- sample(1:ncol(W), 50)
y <- rowSums(W[, causal]) + rnorm(nrow(W), sd = sqrt(50))

X <- model.matrix(y - 1)

5g <- 50
Se <- 50
h2 <- Sg/(Sg + Se)
lambda <- Se/(Sg/m)
lambda <- m*(1-h2)/h2

# BLUP of single marker effects and total genomic effects based on Gauss-Seidel procedure
fit <- gsolve(y = y, X = X, W = W, lambda = lambda)
```

ldsc

** LD score regression **

Description

The ldsc function is used for LDSC analysis

Usage

```r
ldsc(
  Glist = NULL,
  ldscores = NULL,
  z = NULL,
  b = NULL,
  seb = NULL,
  af = NULL,
  stat = NULL,
  n = NULL,
  intercept = TRUE,
  what = "h2",
  SE.h2 = FALSE,
  SE.rg = FALSE,
  blk = 200
)
```
Arguments

- **Glist**: list of information about genotype matrix stored on disk
- **ldscores**: vector of LD scores (optional as LD scores are stored within Glist)
- **z**: matrix of z statistics for n traits
- **b**: matrix of marker effects for n traits if z matrix not is given
- **seb**: matrix of standard errors of marker effects for n traits if z matrix not is given
- **af**: vector of allele frequencies
- **stat**: dataframe with marker summary statistics
- **n**: vector of sample sizes for the traits (element i corresponds to column vector i in z matrix)
- **intercept**: logical if TRUE the LD score regression includes intercept
- **what**: either computation of heritability (what="h2") or genetic correlation between traits (what="rg")
- **SE.h2**: logical if TRUE standard errors and significance for the heritability estimates are computed using a block jackknife approach
- **SE.rg**: logical if TRUE standard errors and significance for the genetic correlations are computed using a block jackknife approach
- **blk**: numeric size of the blocks used in the jackknife estimation of standard error (default = 200)

Value

Returns a matrix of heritability estimates when what="h2", and if SE.h2=TRUE standard errors (SE) and significance levels (P) are returned. If what="rg" an n-by-n matrix of correlations is returned where the diagonal elements being h2 estimates. If SE.rg=TRUE a list is returned with n-by-n matrices of genetic correlations, estimated standard errors and significance levels.

Author(s)

- Peter Soerensen
- Palle Duun Rohde

Examples

```r
# Plink bed/bim/fam files
#bedfiles <- system.file("extdata", paste0("sample_chr",1:2,".bed"), package = "qgg")
bimfiles <- system.file("extdata", paste0("sample_chr",1:2,".bim"), package = "qgg")
famfiles <- system.file("extdata", paste0("sample_chr",1:2,".fam"), package = "qgg")
#
## Summarize bed/bim/fam files
#Glist <- gprep(study="Example", bedfiles=bedfiles, bimfiles=bimfiles, famfiles=famfiles)
#
## Filter rsids based on MAF, missingness, HWE
```
#rsids <- gfilter(Glist = Glist, excludeMAF=0.05, excludeMISS=0.05, excludeHWE=1e-12)
#
## Compute sparse LD (msize=size of LD window)
##ldfiles <- system.file("extdata", paste0("sample_chr",1:2,".ld"), package = "qgg")
##Glist <- gprep(Glist, task="sparseld", msize=200, rsids=rsids, ldfiles=ldfiles, overwrite=TRUE)
#
##Simulate data
##W1 <- getG(Glist, chr=1, scale=TRUE)
##W2 <- getG(Glist, chr=2, scale=TRUE)

##W <- cbind(W1,W2)
#causal <- sample(1:ncol(W),5)

#b1 <- rnorm(length(causal))
#b2 <- rnorm(length(causal))
#y1 <- W[, causal]%*%b1 + rnorm(nrow(W))
#y2 <- W[, causal]%*%b2 + rnorm(nrow(W))

#data1 <- data.frame(y = y1, mu = 1)
#data2 <- data.frame(y = y2, mu = 1)

#X1 <- model.matrix(y ~ 0 + mu, data = data1)
#X2 <- model.matrix(y ~ 0 + mu, data = data2)

## Linear model analyses and single marker association test
#maLM1 <- lma(y=y1, X=X1,W = W)
#maLM2 <- lma(y=y2,X=X2,W = W)
#
## Compute heritability and genetic correlations for trait 1 and 2
#z1 <- maLM1[,"stat"]
#z2 <- maLM2[,"stat"]

#z <- cbind(z1=z1,z2=z2)

#h2 <- ldsc(Glist, z=z, n=c(500,500), what="h2")
#rg <- ldsc(Glist, z=z, n=c(500,500), what="rg")

mtadj

Adjustment of marker effects using correlated trait information

Description

The ‘mtadj’ function uses selection index theory to determine the optimal weights across ‘n’ traits. These weights are then used to adjust marker effects by ‘n’ correlated traits. More details can be found [here](https://www.nature.com/articles/s41467-017-02769-6).
Usage

```r
mtadj(
  h2 = NULL,
  rg = NULL,
  stat = NULL,
  b = NULL,
  z = NULL,
  n = NULL,
  mtotal = NULL,
  meff = 60000,
  method = "ols",
  statistics = "z"
)
```

Arguments

- `h2`: A vector of heritability estimates.
- `rg`: An n-by-n matrix of genetic correlations.
- `stat`: A dataframe containing marker summary statistics.
- `b`: A matrix of marker effects.
- `z`: A matrix of z-scores.
- `n`: A vector indicating the sample size used to estimate marker effects for each trait.
- `mtotal`: Total number of markers.
- `meff`: Effective number of uncorrelated genomic segments (default = 60,000).
- `method`: Method to estimate marker effects. Can be "OLS" (ordinary least square, default) or "BLUP" (best linear unbiased prediction).
- `statistics`: Specifies which kind of statistics ("b" or "z") should be used in the analysis.

Value

A matrix of adjusted marker effects for each trait.

Author(s)

Palle Duun Rohde and Peter Soerensen

Examples

```r
#bedfiles <- system.file("extdata", "sample_22.bed", package = "qgg")
#bimfiles <- system.file("extdata", "sample_22.bim", package = "qgg")
#famfiles <- system.file("extdata", "sample_22.fam", package = "qgg")
#Glist <- gprep(study="1000G", bedfiles=bedfiles, bimfiles=bimfiles, famfiles=famfiles)
#Glist <- gprep(Glist, task="sparseld", msize=200)
#
##Simulate data
#set.seed(23)
```
# W <- getG(Glist, chr=1, scale=TRUE)
# causal <- sample(1:ncol(W), 50)
# set1 <- c(causal, sample(c(1:ncol(W))[-causal], 10))
# set2 <- c(causal, sample(c(1:ncol(W))[-set1], 10))
#
# b1 <- rnorm(length(set1))
# b2 <- rnorm(length(set2))
# y1 <- W[, set1] %*% b1 + rnorm(nrow(W))
# y2 <- W[, set2] %*% b2 + rnorm(nrow(W))
#
## Create model
# data1 <- data.frame(y = y1, mu = 1)
# data2 <- data.frame(y = y2, mu = 1)
# X1 <- model.matrix(y ~ 0 + mu, data = data1)
# X2 <- model.matrix(y ~ 0 + mu, data = data2)
#
## Linear model analyses and single marker association test
# maLM1 <- glma(y=y1, X=X1, W = W)
# maLM2 <- glma(y=y2, X=X2, W = W)
#
## Compute genetic parameters
# z1 <- maLM1[, "stat"]
# z2 <- maLM2[, "stat"]
# z <- cbind(z1, z2)
#
# h2 <- ldsc(Glist, z=z, n=c(500, 500), what="h2")
# rg <- ldsc(Glist, z=z, n=c(500, 500), what="rg")
#
## Adjust summary statistics using estimated genetic parameters
# b <- cbind(b1=maLM1[, "b"], b2=maLM2[, "b"])
# bm <- mtadj( h2=h2, rg=rg, b=b, n=c(500, 500), method="ols")
Index

acc, 2
adjStat, 3
gbayes, 4
gG, 8
gfilter, 9
glma, 10
gmap, 13
gprep, 18
greml, 20
grm, 23
gscore, 25
gsea, 26
gsim, 29
gsolve, 30
ldsc, 32
mtadj, 34