Package 'qgg'

June 29, 2020

Type Package

Title Statistical Tools for Quantitative Genetic Analyses

Version 1.0.4

Date 2020-06-27

Maintainer Peter Soerensen <peter.sorensen@r-qgg.org>

Description Provides an infrastructure for efficient processing of large-scale genetic and phenotypic data including core functions for: 1) fitting linear mixed models, 2) constructing marker-based genomic relationship matrices, 3) estimating genetic parameters (heritability and correlation), 4) performing genomic prediction and genetic risk profiling, and 5) single or multi-marker association analyses.
Rohde et al. (2019) <doi:10.1101/503631>.

License GPL-3

Encoding UTF-8

Imports data.table, parallel, statmod, stats, MCMCpack, MASS

RoxygenNote 7.1.0

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 2020-06-29 16:40:02 UTC

R topics documented:

`adjLD` *LD pruning of summary statistics*

Description

Perform LD pruning of summary statistics before they are used in gene set enrichment analyses.

Usage

```
adjLD(
  stat = NULL,
  statistics = "p-value",
  Glist = NULL,
  r2 = 0.9,
  ldSets = NULL,
  threshold = 1,
  method = "pruning"
)
```

Arguments

`stat`	vector or matrix of single marker statistics (e.g. coefficients, t-statistics, p-values)
`statistics`	is the type of statistics used in stat (e.g. statistics="p-value")
`Glist`	list providing information about genotypes stored on disk
`r2`	threshold for r2 used in LD pruning
`ldSets`	list of marker sets - names corresponds to row names in stat
`threshold`	p-value threshold used in LD pruning
`method`	used including method="pruning" which is default or "clumping"

gbayes *Genomic prediction models implemented using Bayesian Methods (small data)*

Description

Genomic prediction models implemented using Bayesian Methods (small data). The models are implemented using empirical Bayesian methods. The hyperparameters of the dispersion parameters of the Bayesian model can be obtained from prior information or estimated by maximum likelihood, and conditional on these, the model is fitted using Markov chain Monte Carlo. These functions are currently under development and future release will be able to handle large data sets.

Usage

```
gbayes(
  y = NULL,
  W = NULL,
  sets = NULL,
  h2 = NULL,
  nsets = NULL,
  nsamp = 50,
  nburn = 10,
  nsave = 10000,
  tol = 0.001,
  method = "blasso",
  phi = c(0.999, 0.001)
)
```

Arguments

y	is a matrix of phenotypes
W	is a matrix of centered and scaled genotypes
sets	is a list of markers defining a group
h2	is the trait heritability
nsets	is a list of number of marker groups
nsamp	is the number of samples after burnin
nburn	is the number of burnin samples
nsave	is the number of samples to save
tol	is the tolerance
method	specifies the methods used (method="ssvs","blasso","blr")
phi	is the proportion of markers in each marker variance class (phi=c(0.999,0.001),used if method="ssvs")

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

gbayes(y=y, W=W, method="blasso", nsamp=50)
gbayes(y=y, W=W, method="ssvs", nsamp=50)
gbayes(y=y, W=W, method="blr", nsets=7, nsamp=50)
```

---

gblup *Compute Genomic BLUP values*

Description

Compute Genomic BLUP values based on linear mixed model fit output from greml

Usage

```
gblup(
  GRMlist = NULL,
  GRM = NULL,
  fit = NULL,
  ids = NULL,
  idsCLS = NULL,
  idsRWS = NULL
)
```

Arguments

GRMlist	list providing information about GRM matrix stored in binary files on disk
GRM	list of one or more genomic relationship matrices
fit	list object output from greml function
ids	vector of ids for which BLUP values is computed
idsCLS	vector of column ids in GRM for which BLUP values is computed
idsRWS	vector of row ids in GRM for which BLUP values is computed

getGRM *Extract elements from genomic relationship matrix (GRM) stored on disk*

Description

Extract elements from genomic relationship matrix (GRM) (whole or subset) stored on disk.

Usage

```
getGRM(
  GRMlist = NULL,
  ids = NULL,
  idsCLS = NULL,
  idsRWS = NULL,
  cls = NULL,
  rws = NULL
)
```

Arguments

GRMlist	list providing information about GRM matrix stored in binary files on disk
ids	vector of ids in GRM to be extracted
idsCLS	vector of column ids in GRM to be extracted
idsRWS	vector of row ids in GRM to be extracted
cls	vector of columns in GRM to be extracted
rws	vector of rows in GRM to be extracted

getW *Extract elements from genotype matrix (W) stored on disk*

Description

Extract elements from genotype matrix W (whole or subset) stored on disk.

Usage

```
getW(
  Glist = NULL,
  bedfiles = NULL,
  ids = NULL,
  rsids = NULL,
  rws = NULL,
  cls = NULL,
```

```
  impute = TRUE,
  scale = FALSE,
  allele = NULL
)
```

Arguments

Glist	only provided if task="summary" or task="sparseld"
bedfiles	vector of name for the PLINK bed-file
ids	vector of ids in W to be extracted
rsids	vector of rsids in W to be extracted
rws	vector of rows in W to be extracted
cls	vector of columns in W to be extracted
impute	logical if TRUE missing genotypes are set to its expected value (2*af where af is allele frequency)
scale	logical if TRUE the genotype markers have been scale to mean zero and variance one
allele	vector of alleles to be extracted

---

gprep *Prepare genotype data for all statistical analyses (initial step)*

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 (lma or mlma), 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,
  fnRAW = NULL,
  fnLD = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
  famfiles = NULL,
  ids = NULL,
  rsids = NULL,
  overwrite = FALSE,
  msize = 100,
  ncores = 1
)
```

Arguments

Glist	only provided if task="summary" or task="sparseld"
task	character specifying which task to perform ("prepare" is default, "summary", or "sparseld")
study	name of the study
fnRAW	path and filename of the binary file .raw or .bed used for storing genotypes on the disk
fnLD	path and filename of the binary files .ld for storing sparse ld matrix on the disk
bedfiles	vector of names for the PLINK bed-files
bimfiles	vector of names for the PLINK bim-files
famfiles	vector of names for the PLINK fam-files
ids	vector of individuals used in the study
rsids	vector of marker rsids used in the study
overwrite	logical if TRUE overwite binary genotype file
msize	number of markers used in compuation of sparseld
ncores	number of cores used to process the genotypes

Value

Returns a list structure (Glist) with information about genotypes

Author(s)

Peter Soerensen

Examples

```
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")

if(!grepl("^darwin", R.version$os)) {
  fnRAW <- tempfile(fileext=".raw")

  Glist <- gprep(study="1000G", fnRAW=fnRAW, bedfiles=bedfiles, bimfiles=bimfiles,
              famfiles=famfiles, overwrite=TRUE)

  file.remove(fnRAW)
}
```

`greml` *Genomic REML analysis*

Description

The greml function is used for 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 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 dataframe or list containing 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

```
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,
  makeplots = FALSE,
  interface = "R",
  fm = NULL,
  data = NULL
)
```

Arguments

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

Lee, S. H., & van Der Werf, J. H. (2006). An efficient variance component approach implementing an average information REML suitable for combined LD and linkage mapping with a general complex pedigree. Genetics Selection Evolution, 38(1), 25.

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)
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)

cvGB$accuracy
cvGF$accuracy
cvGT$accuracy
```

---

grm *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 = 0,
  task = "grm"
)
```

Arguments

Glist	list providing information about genotypes stored on disk
GRMlist	list providing information about GRM matrix stored in binary files on disk
ids	vector of individuals used for computing GRM
rsids	vector marker rsids used for computing GRM
rws	rows in genotype matrix used for computing GRM
cls	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=0 is default) used for missing values in the genotype data
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 prediction based on single marker summary statistics*

Description

The gscore function is used for genomic predictions based on single marker summary statistics (coefficients, log-odds ratios, z-scores) and observed genotypes.

Usage

```
gscore(
  Glist = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
  famfiles = NULL,
  stat = NULL,
  ids = NULL,
  scale = TRUE,
  impute = TRUE,
  msize = 100,
  ncores = 1
)
```

Arguments

Glist	list of information about genotype matrix
bedfiles	name of the PLINK bed-files
bimfiles	name of the PLINK bim-files
famfiles	name of the PLINK fam-files
stat	matrix of single marker effects
ids	vector of individuals used in the analysis
scale	logical if TRUE the genotype markers have been scale to mean zero and variance one
impute	logical if TRUE missing genotypes are set to its expected value (2*af where af is allele frequency)
msize	number of genotype markers used for batch processing
ncores	number of cores used in the analysis

Author(s)

Peter Soerensen

Examples

```
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")

fnRAW <- tempfile(fileext=".raw")

Glist <- gprep(study="1000G", fnRAW=fnRAW, bedfiles=bedfiles, bimfiles=bimfiles,
               famfiles=famfiles, overwrite=TRUE)

rsids <- Glist$rsids
stat <- data.frame(rsids=Glist$rsids,alleles=Glist$a2, af=Glist$af, effect=rnorm(Glist$m))

W <- getW(Glist=Glist,rsids=Glist$rsids)
pgs1 <- W%*%stat[,4]

pgs2 <- gscore(Glist = Glist, stat = stat)

pgs3 <- gscore(bedfiles=bedfiles, stat = stat)

pgs4 <- gscore(bedfiles=bedfiles,bimfiles=bimfiles,famfiles=famfiles, stat = stat)


cor(cbind(pgs1,pgs2,pgs3,pgs4))

file.remove(fnRAW)
```

---

gsea *Gene set enrichment analysis*

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 lma 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 liner 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='hyperg' (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)
fm <- y ~ 0 + mu
X <- model.matrix(fm, data = data)

# Single marker association analyses
ma <- lma(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
mma <- gsea(stat = ma[,"stat"]**2, sets = sets, method = "sum", nperm = 10000)
head(mma)

# Set test based on hyperG
mma <- gsea(stat = ma[,"p"], sets = sets, method = "hyperg", threshold = 0.05)
head(mma)


G <- grm(W=W)
fit <- greml(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)
```

---

gsolve *Genomic prediction based on a linear mixed model*

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,
  Glist = NULL,
  W = NULL,
  ids = NULL,
  rsids = NULL,
  sets = NULL,
  validate = 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
Glist	list of information about centered and scaled genotype matrix
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
validate	dataframe or list of individuals used in cross-validation (one column for each set)
scale	logical if TRUE the genotypes in Glist has been 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 reml to declare convergence
method	used in solver (currently only methods="gsru": gauss-seidel with resiudal update)
ncores	number of cores used in the analysis

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))
m <- ncol(W)
causal <- sample(1:ncol(W),50)
y <- rowSums(W[,causal]) + rnorm(nrow(W),sd=sqrt(50))

X <- model.matrix(y~1)

Sg <- 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)
```

`lma` *Single marker association analysis using linear models or linear mixed models*

Description

The function lma 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

```
lma(
  y = NULL,
  X = NULL,
  W = NULL,
  Glist = NULL,
  fit = NULL,
  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
fit	list of information about linear mixed model fit (output from greml)
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

Chen, W. M., & Abecasis, G. R. (2007). Family-based association tests for genomewide association scans. The American Journal of Human Genetics, 81(5), 913-926.

Loh, P. R., Tucker, G., Bulik-Sullivan, B. K., Vilhjalmsson, B. J., Finucane, H. K., Salem, R. M., ... & Patterson, N. (2015). Efficient Bayesian mixed-model analysis increases association power in large cohorts. Nature genetics, 47(3), 284-290.

Kang, H. M., Sul, J. H., Zaitlen, N. A., Kong, S. Y., Freimer, N. B., Sabatti, C., & Eskin, E. (2010). Variance component model to account for sample structure in genome-wide association studies. Nature genetics, 42(4), 348-354.

Lippert, C., Listgarten, J., Liu, Y., Kadie, C. M., Davidson, R. I., & Heckerman, D. (2011). FaST linear mixed models for genome-wide association studies. Nature methods, 8(10), 833-835.

Listgarten, J., Lippert, C., Kadie, C. M., Davidson, R. I., Eskin, E., & Heckerman, D. (2012). Improved linear mixed models for genome-wide association studies. Nature methods, 9(6), 525-526.

Listgarten, J., Lippert, C., & Heckerman, D. (2013). FaST-LMM-Select for addressing confounding from spatial structure and rare variants. Nature Genetics, 45(5), 470-471.

Lippert, C., Quon, G., Kang, E. Y., Kadie, C. M., Listgarten, J., & Heckerman, D. (2013). The benefits of selecting phenotype-specific variants for applications of mixed models in genomics. Scientific reports, 3.

Zhou, X., & Stephens, M. (2012). Genome-wide efficient mixed-model analysis for association studies. Nature genetics, 44(7), 821-824.

Svishcheva, G. R., Axenovich, T. I., Belonogova, N. M., van Duijn, C. M., & Aulchenko, Y. S. (2012). Rapid variance components-based method for whole-genome association analysis. Nature genetics, 44(10), 1166-1170.

Yang, J., Zaitlen, N. A., Goddard, M. E., Visscher, P. M., & Price, A. L. (2014). Advantages and pitfalls in the application of mixed-model association methods. Nature genetics, 46(2), 100-106.

Bulik-Sullivan, B. K., Loh, P. R., Finucane, H. K., Ripke, S., Yang, J., Patterson, N., ... & Schizophrenia Working Group of the Psychiatric Genomics Consortium. (2015). LD Score regression distinguishes confounding from polygenicity in genome-wide association studies. Nature genetics, 47(3), 291-295.

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

# Linear model analyses and single marker association test
maLM <- lma(y=y,X=X,W = W)

head(maLM)


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

# Estimate variance components using REML analysis
fit <- greml(y = y, X = X, GRM = list(GRM), verbose = TRUE)

# Single marker association test
maMLM <- lma(fit = fit, W = W)

head(maMLM)
```

`mergeGRM` *Merge multiple GRMlist objects*

Description

Merge multiple GRMlist objects each with information about a genomic rfelationship matrix stored on disk

Usage

```
mergeGRM(GRMlist = NULL)
```

Arguments

GRMlist	list providing information about GRM matrix stored in binary files on disk

---

qgg *Implements Genomic Feature Linear Mixed Models using Likelihood or Bayesian Methods*

Description

We have developed Genomic Feature Linear Mixed Models for predicting quantitative trait phenotypes from high resolution genomic polymorphism data. Genomic features are regions on the genome that are hypothesized to be enriched for causal variants affecting the trait. Several genomic feature classes can be formed based on previous studies and different sources of information including genes, chromosomes, biological pathways, gene ontologies, sequence annotation, prior QTL regions, or other types of external evidence. Using prior information on genomic features is important because prediction is difficult for populations of unrelated individuals when the number of causal variants is low relative to the total number of polymorphisms, and causal variants individually have small effects on the traits. The models were implemented using likelihood or Bayesian methods.

We have developed Genomic Feature Best Linear Unbiased Prediction (GFBLUP) models. We have extended these models to include multiple features and multiple traits. Different genetic models (e.g. additive, dominance, gene by gene and gene by environment interactions) can be specified.

We have developed Bayesian multiple Genomic Feature and Trait models. The models are implemented using an empirical Bayesian method that handles multiple features and multiple traits. The models were implemented using spectral decomposition that plays an important computational role in the Markov chain Monte Carlo strategy. This is a very flexible and formal statistical framework for using prior information to decompose genomic (co)variances and predict trait phenotypes.

The premise of the Genomic Feature models presented above is that genomic features are enriched for causal variants affecting the traits. However, in reality, the number, location and effect sizes of the true causal variants in the genomic feature are unknown. Therefore we have developed and evaluated a number of SNP set tests derived from a standard Genomic BLUP model. These approaches are computationally very fast allowing us to rapidly analyze different layers of genomic feature classes to discover genomic features potentially enriched for causal variants. Results from these analyses can be built into the above mentioned prediction models.

Details

Package:	qgg
Type:	Package
Version:	1.0
Date:	2015-10-21
License:	GPL-3

Author(s)

Maintainer: Peter Sørensen <pso@mbg.au.dk>

References

Mapping Variants to Gene Ontology Categories Improves Genomic Prediction for Quantitative Traits in Drosophila melanogaster. Under review Genetics (2016). Edwards SM, Sørensen IF, Sarup P, Mackay TF, Sørensen P.

Genomic BLUP Derived Set Tests Identify Genetic Variants Associated with Schizophrenia in Functionally Associated Biological Processes. Under review, Genetics (2015). Rohde PD, Demontis D, The GEMS Group, Børglum AD, Sørensen P.

Partitioning of Genomic Variance Reveals Biological Pathways Associated with Udder Health and Milk Production Traits in Dairy Cattle. GSE (2015) 47:60. Edwards SM, Thomsen B, Madsen P, Sørensen P.

Increased Prediction Accuracy using a Genomic Feature Model Including Prior Information on Quantitative Trait Locus Regions in Purebred Danish Duroc Pigs. BMC Genetics (2016) 17:11. Sarup P, Jensen J, OstersenT, Henryon M, Sørensen P.

Index