Package ‘qgg’

August 8, 2022

Type Package

Title Statistical Tools for Quantitative Genetic Analyses

Version 1.1.1

Date 2022-08-07

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

Depends R (>= 3.3.0)

Imports Rcpp (>= 1.0.5), data.table, parallel, statmod, stats,
MCMCpack, MASS

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.0

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 2022-08-08 13:40:15 UTC

R topics documented:

---

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

Description

Compute prediction accuracy for a quantitative or binary trait

Usage

```
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 *LD adjustment of marker summary statistics*

Description

Adjust marker summary statistics using linkage disequilibrium information from Glist

Required input format for summary statistics:

stat can be a data.frame(rsids, chr, pos, a1, a2, af, b, seb, stat, p, n) (single trait)

stat can be a list(marker=(rsids, chr, pos, a1, a2, af), b, seb, stat, p, n) (multiple trait)

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`	specificy 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 corresponds 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

stat can be a data.frame(rsids, chr, pos, a1, a2, af, b, seb, stat, p, n) (single trait)

stat can be a list(marker=(rsids, chr, pos, a1, a2, af), b, seb, stat, p, n) (multiple trait)

Author(s)

Peter Soerensen

gbayes *Bayesian linear regression models*

Description

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,
  b = NULL,
  bm = NULL,
  seb = NULL,
  LD = NULL,
  n = NULL,
  vg = NULL,
  vb = NULL,
  ve = NULL,
  ssg_prior = NULL,
  ssb_prior = NULL,
  sse_prior = NULL,
  lambda = NULL,
  scaleY = TRUE,
  h2 = NULL,
  pi = 0.001,
  updateB = TRUE,
  updateG = TRUE,
  updateE = TRUE,
  updatePi = TRUE,
  adjustE = TRUE,
  models = NULL,
  nug = 4,
  nub = 4,
  nue = 4,
  verbose = FALSE,
  msize = 100,
  GRMlist = NULL,
  ve_prior = NULL,
  vg_prior = NULL,
  tol = 0.001,
  nit = 100,
  nburn = 0,
  nit_local = NULL,
  nit_global = NULL,
  method = "mixed",
  algorithm = "default"
)
```

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`	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
`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 compuation of sparseld
`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

`vg` 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


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 *Extract elements from genotype matrix stored on disk*

Description

Extract elements from genotype matrix (rows/columns, ids/rsids) stored on disk.

Usage

```
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	list structure with information about genotypes stored on disk
chr	chromosome for which W is to be extracted
bedfiles	vector of name for the PLINK bed-file
bimfiles	vector of name for the PLINK bim-file
famfiles	vector of name for the PLINK fam-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

gfilter *Quality control of marker summary statistics*

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

```
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`	list of information about genotype matrix stored on disk
`excludeMAF`	exclude marker if minor allele frequency (MAF) is below threshold (0.01 is default)
`excludeMISS`	exclude marker if missingness (MISS) is above threshold (0.05 is default)
`excludeINFO`	exclude marker if info score (INFO) is below threshold (0.8 is default)
`excludeCGAT`	exclude marker if alleles are ambigous (CG or AT)
`excludeINDEL`	exclude marker if it an insertion/deletion
`excludeDUPS`	exclude marker id if duplicated
`excludeHWE`	exclude marker if p-value for Hardy Weinberg Equilibrium test is below threshold (0.01 is default)
`excludeMHC`	exclude marker if located in MHC region
`assembly`	character name of assembly

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

```
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

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
stat <- glma(y=y,X=X,W = W)

head(stat)


# 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
stat <- glma(fit = fit, W = W)

head(stat)
```

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 (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,
  cm = NULL,
  ncores = 1
)
```

Arguments

`Glist` list of information about genotype matrix stored on disk - 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

`fnBED` path and filename of the binary file .bed used for storing genotypes on the disk

`ldfiles` 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

`mapfiles` vector of names for the mapfiles

`ids` vector of individuals used in the study

`rsids` vector of marker rsids used in the study

`assembly` character name of assembly

`overwrite` logical if TRUE overwite binary genotype/ld file

`msize` number of markers used in compuation of sparseld

`r2` threshold

`kb` size of genomic region in kb

`cm` size of genomic region in cm

`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_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)
```

greml *GREML analysis*

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

```
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

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 = NA,
  impute = TRUE,
  pedigree = NULL,
  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=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

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,
  chr = NULL,
  bedfiles = NULL,
  bimfiles = NULL,
  famfiles = NULL,
  stat = NULL,
  fit = NULL,
  ids = NULL,
  scale = TRUE,
  impute = TRUE,
  msize = 100,
  ncores = 1,
  verbose = FALSE
)
```

Arguments

`Glist`	list of information about genotype matrix
`chr`	chromosome for which genomic scores is computed
`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
`fit`	fit object output from gbayes
`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
`verbose`	is a logical; if TRUE it prints more details during optimization

Author(s)

Peter Soerensen

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")

# 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)
```

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 glma 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
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 <- 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)
```

gsim *Genomic simulation*

Description

The gsim function is used for simulating genotype and phenotype data based Glist. It is currently under active development.

Usage

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

Arguments

Glist	list of information about genotype matrix
chr	is the chromosome(s) being used in the simulation
nt	number of traits
W	matrix of centered and scaled genotypes
n	number of individuals
m	number of markers

Author(s)

Peter Soerensen

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")

# 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)
head(sim$y)
head(sim$e)
head(sim$causal)
```

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 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)
```

ldsc *LD score regression*

Description

The ldsc function is used for LDSC analysis

Usage

```
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

```
# 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` *Adjust marker effects based on correlated information*

Description

The mtadj function use selection index theory to find the optimal weights across n traits, which is used to adjust marker effects by n correlated traits. (https://www.nature.com/articles/s41467-017-02769-6)

Usage

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

Arguments

`h2`	vector of heritability estimates
`rg`	n-by-n matrix of genetic correlations
`stat`	dataframe with marker summary statistics
`b`	matrix of marker effects
`z`	matrix of z-scores
`n`	vector of 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 used to estimate marker effects; OLS: ordinary least square (default), or BLUP: best linear unbiased prediction

`statistics` which kind of statistics ("b" or "z") used in the analysis

Value

Matrix of adjusted marker effects for each trait

Author(s)

Palle Duun Rohde and 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”)
#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=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")
#
## 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")
```

qcStat *Quality control of marker summary statistics*

Description

Quality control is a critical step for working with GWAS summary statistics. Processing and quality control of 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

Required headers for external summary statistics: marker, chr, pos, effect_allele, non_effect_allele, effect_allele_freq, effect, effect_se, stat, p, n

Required headers for internal summary statistics: rsids, chr, pos, a1, a2, af, b, seb, stat, p, n

Usage

```
qcStat(
  Glist = NULL,
  stat = NULL,
  excludeMAF = 0.01,
  excludeMAFDIFF = 0.05,
  excludeINFO = 0.8,
  excludeCGAT = TRUE,
  excludeINDEL = TRUE,
  excludeDUPS = TRUE,
  excludeMHC = FALSE,
  excludeMISS = 0.05,
  excludeHWE = 1e-12
)
```

Arguments

`Glist` list of information about genotype matrix stored on disk

`stat` data frame with marker summary statistics (see required format above)

`excludeMAF` exclude marker if minor allele frequency (MAF) is below threshold (0.01 is default)

`excludeMAFDIFF` exclude marker if minor allele frequency difference (MAFDIFF) between Glist$af and stat$af is above threshold (0.05 is default)

`excludeINFO` exclude marker if info score (INFO) is below threshold (0.8 is default)

`excludeCGAT` exclude marker if alleles are ambiguous (CG or AT)

`excludeINDEL` exclude marker if it an insertion/deletion

`excludeDUPS` exclude marker id if duplicated

`excludeMHC` exclude marker if located in MHC region

`excludeMISS` exclude marker if sample missingness (MISS) is above threshold (0.05 is default)

`excludeHWE` exclude marker if p-value for Hardy Weinberg Equilibrium test is below threshold (0.01 is default)

Author(s)

Peter Soerensen

`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