Package ‘RAINBOW’

October 19, 2019

Type Package

Title Perform genome-wide asscoiation study (GWAS) by kernel-based methods

Version 0.1.5

Author Kosuke Hamazaki and Hiroyoshi Iwata

Maintainer Kosuke Hamazaki <hamazaki@ut-biomet.org>

Description Users can test multiple SNPs simultaneously by kernel-based methods. Users can test not only additive effects but also dominance and epistatic effects.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports Rcpp, RcppEigen, rrBLUP, rgl, tcltk, Matrix, cluster, MASS, pbmcapply

LinkingTo Rcpp, RcppEigen

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`CalcThreshold` *Function to calculate threshold for GWAS*

Description

Calculate thresholds for the given GWAS result by the Benjamini-Hochberg method or Bonferroni method.

Usage

```
CalcThreshold(input, sig.level = 0.05, method = "BH")
```

Arguments

`input`	Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.
`sig.level`	Significance level for the threshold. The default is 0.05. You can also assign vector of sinificance levels.
`method`	Two methods are offered: "BH" : Benjamini-Hochberg method. To control FDR, use this method. "Bonf" : Bonferroni method. To perform simple correction of multiple testing, use this method. You can also assign both of them by 'method = c("BH", "Bonf")'

Value

the value of the threshold. If there is no threshold, it returns NA.

References

Benjamini, Y. and Hochberg, Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc. 57(1): 289-300.

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

`cumsum.pos` *Function to calculate cumulative position (beyond chromosome)*

Description

Function to calculate cumulative position (beyond chromosome)

Usage

```
## S3 method for class 'pos'
cumsum(map)
```

Arguments

`map`	Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position.

Value

Cumulative position (beyond chromosome) will be returned.

`design.Z` *Function to generate design matrix (Z)*

Description

Function to generate design matrix (Z)

Usage

```
design.Z(pheno.labels, geno.names)
```

Arguments

`pheno.labels`	A vector of genotype (line; accesion; variety) names which correpond to phenotypic values.
`geno.names`	A vector of genotype (line; accesion; variety) names for marker genotype data (duplication is not recommended).

Value

Z of $y = X\beta + Zu + e$. Design matrix, which is useful for GS or GWAS.

EM3.cpp *Equation of mixed model for multi-kernel (slow, general version)*

Description

This function solves the following multi-kernel linear mixed effects model.

$y = X\beta + \sum_{l=1}^{L} Z_l u_l + \epsilon$

where $Var[y] = \sum_{l=1}^{L} Z_l K_l Z_l' \sigma_l^2 + I\sigma_e^2$.

Usage

```
EM3.cpp(y, X0 = NULL, ZETA, eigen.G = NULL, eigen.SGS = NULL,
  tol = NULL, optimizer = "nlminb", traceInside = 0, n.thres = 450,
  REML = TRUE, pred = TRUE)
```

Arguments

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X0` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA` A list of variance matrices and its design matrices of random effects. You can use more than one kernel matrix. For example, ZETA = list(A = list(Z = Z.A, K = K.A), D = list(Z = Z.D, K = K.D)) (A for additive, D for dominance) Please set names of lists "Z" and "K"!

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`traceInside` Perform trace for the optimzation if traceInside >= 1, and this argument shows the frequency of reports.

`n.thres`	If $n >= n.thres$, perform EMM1.cpp. Else perform EMM2.cpp.
`REML`	You can choose which method you will use, "REML" or "ML". If REML = TRUE, you will perform "REML", and if REML = FALSE, you will perform "ML".
`pred`	If TRUE, the fitting values of y is returned.

Value

$y.pred the fitting values of y $y = X\beta + Zu$

$Vu estimator for σ^2_u, all of the genetic variance

$Ve estimator for σ^2_e

$beta BLUE($\beta$)

$u BLUP($u$)

$weights the proportion of each genetic variance (corresponding to each kernel of ZETA) to Vu

$LL maximized log-likelihood (full or restricted, depending on method)

$Vinv the inverse of $V = Vu \times ZKZ' + Ve \times I$

$Hinv the inverse of $H = ZKZ' + \lambda I$

References

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate additive genetic relationship matrix & epistatic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")
K.AA <- K.A * K.A   ### additive x additive epistatic effects
```

```
### Modify data
Z <- design.Z(pheno.labels = rownames(y),
              geno.names = rownames(K.A))  ### design matrix for random effects
pheno.mat <- y[rownames(Z), , drop = FALSE]
ZETA <- list(A = list(Z = Z, K = K.A),
             AA = list(Z = Z, K = K.AA))


### Solve multi-kernel linear mixed effects model (2 random efects)
EM3.res <- EM3.cpp(y = pheno.mat, X = NULL, ZETA = ZETA)
(Vu <- EM3.res$Vu)   ### estimated genetic variance
(Ve <- EM3.res$Ve)   ### estimated residual variance
(weights <- EM3.res$weights)   ### estimated proportion of two genetic variances
(herit <- Vu * weights / (Vu + Ve))  ### genomic heritability (additive, additive x additive)

(beta <- EM3.res$beta)   ### Here, this is an intercept.
u <- EM3.res$u   ### estimated genotypic values (additive, additive x additive)
See(u)
```

EM3.linker.cpp *Equation of mixed model for multi-kernel (fast, for limited cases)*

Description

This function solves multi-kernel mixed model using fastlmm.snpset approach (Lippert et al., 2014). This function can be used only when the kernels other than genomic relationship matrix are linear kernels.

Usage

```
EM3.linker.cpp(y0, X0 = NULL, ZETA = NULL, Zs0 = NULL, Ws0,
  Gammas0 = lapply(Ws0, function(x) diag(ncol(x))), gammas.diag = TRUE,
  X.fix = TRUE, eigen.SGS = NULL, eigen.G = NULL, tol = NULL,
  bounds = c(1e-06, 1e+06), optimizer = "nlminb", traceInside = 0,
  n.thres = 450, spectral.method = NULL, REML = TRUE, pred = TRUE)
```

Arguments

y0	$n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
X0	$n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.
ZETA	A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!
Zs0	A list of design matrices (Z; $n \times m$ matrix) for Ws. For example, Zs0 = list(A.part = Z.A.part, D.part = Z.D.part)
Ws0	A list of low rank matrices (W; $m \times k$ matrix). This forms linear kernel $K = W \Gamma W'$. For example, Ws0 = list(A.part = W.A, D.part = W.D)

`Gammas0` A list of matrices for weighting SNPs (Gamma; $k \times k$ matrix). This forms linear kernel $K = W \Gamma W'$. For example, if there is no weighting, Gammas0 = lapply(Ws0, function(x) diag(ncol(x)))

`gammas.diag` If each Gamma is the diagonal matrix, please set this argument TRUE. The calculationtime can be saved.

`X.fix` If you repeat this function and when X0 is fixed during iterations, please set this argument TRUE.

`eigen.SGS` A list with $values : eigen values and $vectors : eigen vectors. The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`bounds` Lower and upper bounds for weights.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`traceInside` Perform trace for the optimzation if traceInside >= 1, and this argument shows the frequency of reports.

`n.thres` If $n >= n.thres$, perform EMM1.cpp. Else perform EMM2.cpp.

`REML` You can choose which method you will use, "REML" or "ML". If REML = TRUE, you will perform "REML", and if REML = FALSE, you will perform "ML".

`pred` If TRUE, the fitting values of y is returned.

Value

$y.pred the fitting values of y $y = X\beta + Zu$

$Vu estimator for σ^2_u, all of the genetic variance

$Ve estimator for σ^2_e

$beta BLUE($\beta$)

$u BLUP($u$)

$weights the proportion of each genetic variance (corresponding to each kernel of ZETA) to Vu

$LL maximized log-likelihood (full or restricted, depending on method)

$Vinv the inverse of $V = Vu \times ZKZ' + Ve \times I$

$Hinv the inverse of $H = ZKZ' + \lambda I$

References

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

EMM.cpp *Equation of mixed model for one kernel, a wrapper of two methods*

Description

This function estimates maximum-likelihood (ML/REML) solutions for the following mixed model.

$$y = X\beta + Zu + \epsilon$$

where β is a vector of fixed effects and u is a vector of random effects with $Var[u] = K\sigma_u^2$. The residual variance is $Var[\epsilon] = I\sigma_e^2$.

Usage

```
EMM.cpp(y, X = NULL, ZETA, eigen.G = NULL, eigen.SGS = NULL,
  n.thres = 450, reestimation = FALSE, lam.len = 4,
  init.range = c(1e-06, 100), init.one = 0.5, conv.param = 1e-06,
  count.max = 20, bounds = c(1e-06, 1e+06), tol = NULL,
  optimizer = "nlminb", traceInside = 0, REML = TRUE,
  silent = TRUE, plot.l = FALSE, SE = FALSE, return.Hinv = TRUE)
```

Arguments

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`n.thres` If $n >= n.thres$, perform EMM1.cpp. Else perform EMM2.cpp.

`reestimation` If TRUE, EMM2.cpp is performed when the estimation by EMM1.cpp may not be accurate.

`lam.len` The number of initial values you set. If this number is large, the estimation will be more accurate, but computational cost will be large. We recommend setting this value 3 <= lam.len <= 6.

`init.range` The range of the initial parameters. For example, if lam.len = 5 and init.range = c(1e-06, 1e02), corresponding initial heritabilities will be calculated as seq(1e-06, 1 - 1e-02, length = 5), and then initial lambdas will be set.

`init.one` The initial parameter if lam.len = 1.

`conv.param` The convergence parameter. If the diffrence of log-likelihood by updating the parameter "lambda" is smaller than this conv.param, the iteration steps will be stopped.

`count.max` Sometimes algorithms won't converge for some initial parameters. So if the iteration steps reache to this argument, you can stop the calculation even if algorithm doesn't converge.

`bounds` Lower and Upper bounds of the parameter lambda. If the updated parameter goes out of this range, the parameter is reset to the value in this range.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`traceInside` Perform trace for the optimzation if traceInside >= 1, and this argument shows the frequency of reports.

`REML` You can choose which method you will use, "REML" or "ML". If REML = TRUE, you will perform "REML", and if REML = FALSE, you will perform "ML".

`silent` If this argument is TRUE, warning messages will be shown when estimation is not accurate.

`plot.l` If you want to plot log-likelihood, please set plot.l = TRUE. We don't recommend plot.l = TRUE when lam.len >= 2.

`SE` If TRUE, standard errors are calculated.

`return.Hinv` If TRUE, the function returns the inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma_e^2 / \sigma_u^2$. This is useful for GWAS.

Value

$Vu estimator for σ_u^2

$Ve estimator for σ_e^2

$beta BLUE($\beta$)

$u BLUP($u$)

$LL maximized log-likelihood (full or restricted, depending on method)

$beta.SE standard error for β (If SE = TRUE)

$u.SE standard error for $u^* - u$ (If SE = TRUE)

$Hinv the inverse of $H = ZKZ' + \lambda I$ (If return.Hinv = TRUE)

$Hinv2 the inverse of $H2 = ZKZ'/\lambda + I$ (If return.Hinv = TRUE)

$lambda estimators for $\lambda = \sigma^2_e/\sigma^2_u$ (If $n >= n.thres$)

$lambdas lambdas for each initial values (If $n >= n.thres$)

$reest If parameter estimation may not be accurate, reest = 1, else reest = 0 (If $n >= n.thres$)

$counts the number of iterations until convergence for each initial values (If $n >= n.thres$)

References

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")

### Modify data
modify.res <- modify.data(pheno.mat = y, geno.mat = x, return.ZETA = T)
pheno.mat <- modify.res$pheno.modi
ZETA <- modify.res$ZETA


### Solve linear mixed effects model
EMM.res <- EMM.cpp(y = pheno.mat, X = NULL, ZETA = ZETA)
(Vu <- EMM.res$Vu)   ### estimated genetic variance
```

```
(Ve <- EMM.res$Ve)   ### estimated residual variance
(herit <- Vu / (Vu + Ve))   ### genomic heritability

(beta <- EMM.res$beta)   ### Here, this is an intercept.
u <- EMM.res$u   ### estimated genotypic values
See(u)
```

EMM1.cpp — *Equation of mixed model for one kernel, GEMMA-based method (inplemented by Rcpp)*

Description

This function solves the single-kernel linear mixed effects model by GEMMA (genome wide efficient mixed model association; Zhou et al., 2012) approach.

Usage

```
EMM1.cpp(y, X = NULL, ZETA, eigen.G = NULL, lam.len = 4,
  init.range = c(1e-04, 100), init.one = 0.5, conv.param = 1e-06,
  count.max = 15, bounds = c(1e-06, 1e+06), tol = NULL,
  REML = TRUE, silent = TRUE, plot.l = FALSE, SE = FALSE,
  return.Hinv = TRUE)
```

Arguments

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`lam.len` The number of initial values you set. If this number is large, the estimation will be more accurate, but computational cost will be large. We recommend setting this value 3 <= lam.len <= 6.

`init.range` The range of the initial parameters. For example, if lam.len = 5 and init.range = c(1e-06, 1e02), corresponding initial heritabilities will be calculated as seq(1e-06, 1 - 1e-02, length = 5), and then initial lambdas will be set.

`init.one` The initial parameter if lam.len = 1.

`conv.param` The convergence parameter. If the diffrence of log-likelihood by updating the parameter "lambda" is smaller than this conv.param, the iteration steps will be stopped.

`count.max` Sometimes algorithms won't converge for some initial parameters. So if the iteration steps reache to this argument, you can stop the calculation even if algorithm doesn't converge.

`bounds` Lower and Upper bounds of the parameter 1 / lambda. If the updated parameter goes out of this range, the parameter is reset to the value in this range.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`REML` You can choose which method you will use, "REML" or "ML". If REML = TRUE, you will perform "REML", and if REML = FALSE, you will perform "ML".

`silent` If this argument is TRUE, warning messages will be shown when estimation is not accurate.

`plot.l` If you want to plot log-likelihood, please set plot.l = TRUE. We don't recommend plot.l = TRUE when lam.len >= 2.

`SE` If TRUE, standard errors are calculated.

`return.Hinv` If TRUE, the function returns the inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma^2_e / \sigma^2_u$. This is useful for GWAS.

Value

$Vu estimator for σ^2_u

$Ve estimator for σ^2_e

$beta BLUE($\beta$)

$u BLUP($u$)

$LL maximized log-likelihood (full or restricted, depending on method)

$beta.SE standard error for β (If SE = TRUE)

$u.SE standard error for $u^* - u$ (If SE = TRUE)

$Hinv the inverse of $H = ZKZ' + \lambda I$ (If return.Hinv = TRUE)

$Hinv2 the inverse of $H2 = ZKZ'/\lambda + I$ (If return.Hinv = TRUE)

$lambda estimators for $\lambda = \sigma^2_e / \sigma^2_u$

$lambdas lambdas for each initial values

$reest If parameter estimation may not be accurate, reest = 1, else reest = 0

$counts the number of iterations until convergence for each initial values

References

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

EMM2.cpp *Equation of mixed model for one kernel, EMMA-based method (inplemented by Rcpp)*

Description

This function solves single-kernel linear mixed model by EMMA (efficient mixed model association; Kang et al., 2008) approach.

Usage

```
EMM2.cpp(y, X = NULL, ZETA, eigen.G = NULL, eigen.SGS = NULL,
  tol = NULL, optimizer = "nlminb", traceInside = 0, REML = TRUE,
  bounds = c(1e-09, 1e+09), SE = FALSE, return.Hinv = FALSE)
```

Arguments

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`traceInside` Perform trace for the optimzation if traceInside >= 1, and this argument shows the frequency of reports.

`REML` You can choose which method you will use, "REML" or "ML". If REML = TRUE, you will perform "REML", and if REML = FALSE, you will perform "ML".

`bounds`	Lower and Upper bounds of the parameter lambda. If the updated parameter goes out of this range, the parameter is reset to the value in this range.
`SE`	If TRUE, standard errors are calculated.
`return.Hinv`	If TRUE, the function returns the inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma_e^2/\sigma_u^2$. This is useful for GWAS.

Value

\$Vu estimator for σ_u^2

\$Ve estimator for σ_e^2

\$beta BLUE(β)

\$u BLUP(u)

\$LL maximized log-likelihood (full or restricted, depending on method)

\$beta.SE standard error for β (If SE = TRUE)

\$u.SE standard error for $u^* - u$ (If SE = TRUE)

\$Hinv the inverse of $H = ZKZ' + \lambda I$ (If return.Hinv = TRUE)

References

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

---

genesetmap *Function to generate map for gene set*

Description

Function to generate map for gene set

Usage

```
genesetmap(map, gene.set, cumulative = FALSE)
```

Arguments

`map`	Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position.
`gene.set`	Gene information with the format of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "map" argument.
`cumulative`	If this argument is TRUE, cumulative position will be returned.

Value

Map for gene set.

genetrait *Generate pseudo phenotypic values*

Description

This function generates pseudo phenotypic values according to the following formula.

$$y = X\beta + Zu + e$$

where effects of major genes are regarded as fixed effects β and polygenetic effects are regarded as random effects u. The variances of u and e are automatically determined by the heritability.

Usage

```
genetrait(x, sample.sets = NULL, candidate = NULL, pos = NULL,
  x.par = NULL, ZETA = NULL, x2 = NULL, num.qtn = 3,
  weight = c(2, 1, 1), qtn.effect = rep("A", num.qtn), prop = 1,
  polygene.weight = 1, polygene = TRUE, h2 = 0.6,
  h.correction = FALSE, seed = NULL, plot = TRUE, saveAt = NULL,
  subpop = NULL, return.all = FALSE, seed.env = TRUE)
```

Arguments

`x` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`sample.sets` n.sample x n.mark genotype matrix. Markers with fixed effects (QTNs) are chosen from sample.sets. If sample.sets = NULL, sample.sets = x.

`candidate` If you want to fix QTN postitions, please set the number where SNPs to be fixed are located in your data (so not position). If candidate = NULL, QTNs were randomly sampled from sample.sets or x.

`pos` n.mark x 1 vector. cumulative position (over chromosomes) of each marker.

`x.par` If you don't want to match the sampling population and the genotype data to QTN effects, then use this argument as the latter.

`x2` genotype matrix to calculate additive relationship matrix when Z.ETA = NULL. If Z.ETA = NULL & x2 = NULL, A.mat(x) will be calculated as kernel matrix.

`num.qtn` the number of QTNs

`weight` The weights for each QTN by their standard deviations. Minus value is also allowed.

`prop` The proportion of effects of QTNs to polygenetic effects.

`polygene.weight` If there are multiple kernels, this argument determines the weights of each kernel effect.

`polygene` If polygene = FALSE, pseudo phenotypes with only QTN effects will be generated.

`h2` The wide-sense heritability for generating phenotypes. 0 <= h2 < 1

`h.correction` If TRUE, this function will generate phenotypes to match the genomic heritability and "h2".

`seed` If seed is not NULL, some fixed phenotypic values will be generated according to set.seed(seed)

`plot` If TRUE, boxplot for generated phenotypic values will be drawn.

`subpop` If there is subpopulation structure, you can draw boxpots divide by subpopulations. n.sample x n.subpop matrix. Please indicate the subpopulation information by (0, 1) for each element. (0 means that line doen't belong to that subpopulation, and 1 means that line belongs to that subpopulation)

`return.all` If FALSE, only returns generated phenotypic values. If TRUE, this function will return other information such as positions of candidate QTNs.

`seed.env` If TRUE, this function will generate different environment effects every time.

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveAt = NULL, the plot is not saved.

Value

trait = trait, u = g.all2, e = e2, candidate = qtn.candidates, qtn.position = pos.qtns, heritability = true_h

trait generated phenotypic values

u generated genotyope values

e generated environmental effects

candidate the numbers where QTNs are located in your data (so not position).

qtn.position QTN positions

heritability var(u) / var(trait), genomic heritability for generated phenotypic values.

`MAF.cut` *Function to remove the minor alleles*

Description

Function to remove the minor alleles

Usage

```
MAF.cut(x.0, map.0 = NULL, min.MAF = 0.05, max.MS = 0.05,
  return.MAF = FALSE)
```

Arguments

`x.0` $n \times m$ original marker genotype matrix.

`map.0` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is removed from the original marker genotype data.

`max.MS` Specifies the maximum missing rate (MS). If a marker has a MS more than max.MS, it is removed from the original marker genotype data.

`return.MAF` If TRUE, MAF will be returned.

Value

$x The modified marker genotype data whose SNPs with MAF <= min.MAF were removed.

$map The modified map information whose SNPs with MAF <= min.MAF were removed.

$before Minor allele frequencies of the original marker genotype.

$after Minor allele frequencies of the modified marker genotype.

`make.full` *Change a matrix to full-rank matrix*

Description

Change a matrix to full-rank matrix

Usage

```
make.full(X)
```

Arguments

`X`	$n \times p$ matrix which you want to change into full-rank matrix.

Value

a full-rank matrix

`manhattan` *Draw manhattan plot*

Description

Draw manhattan plot

Usage

```
manhattan(input, sig.level = 0.05, method.thres = "BH", y.max = NULL,
  cex.lab = 1, lwd.thres = 1, plot.col1 = c("dark blue",
  "cornflowerblue"), cex.axis.x = 1, cex.axis.y = 1, plot.type = "p",
  plot.pch = 16)
```

Arguments

`input` Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`y.max` The maximum value for the vertical axis of manhattan plot. If NULL, automatically determined.

`cex.lab` The font size of the labels.

`lwd.thres` The line width for the threshold.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes.

`cex.axis.x` The font size of the x axis.

`cex.axis.y` The font size of the y axis.

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

Value

draw manhttan plot

manhattan.plus *Add points of -log10(p) corrected by kernel methods to manhattan plot*

Description

Add points of -log10(p) corrected by kernel methods to manhattan plot

Usage

```
manhattan.plus(input, checks, plot.col1 = c("dark blue",
  "cornflowerblue"), plot.col3 = c("red3", "orange3"), plot.type = "p",
  plot.pch = 16)
```

Arguments

`input` Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`checks` The marker numbers whose -log10(p)s are corrected by kernel methods.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as a color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes.

`plot.col3` Color of -log10(p) corrected by kernel methods. plot.col3[1] for odd chromosomes and plot.col3[2] for even chromosomes

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

Value

draw manhttan plot

manhattan2 *Draw manhattan plot (another method)*

Description

Draw manhattan plot (another method)

Usage

```
manhattan2(input, sig.level = 0.05, method.thres = "BH",
  plot.col2 = 1, plot.type = "p", plot.pch = 16, cum.pos = NULL,
  lwd.thres = 1, cex.lab = 1, cex.axis.x = 1, cex.axis.y = 1)
```

Arguments

`input` Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`sig.level` Siginifincance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`plot.col2` color of the manhattan plot. color changes with chromosome and it starts from plot.col2 + 1 (so plot.col2 = 1 means color starts from red.)

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

`cum.pos` cumulative position (over chromosomes) of each marker

`lwd.thres` The line width for the threshold.

`cex.lab` The font size of the labels.

`cex.axis.x` The font size of the x axis.

`cex.axis.y` The font size of the y axis.

Value

draw manhttan plot

`manhattan3` *Draw the effects of epistasis (3d plot and 2d plot)*

Description

Draw the effects of epistasis (3d plot and 2d plot)

Usage

```
manhattan3(input, cum.pos, plot.epi.3d = TRUE, plot.epi.2d = TRUE,
  main.epi.3d = NULL, main.epi.2d = NULL, saveName = NULL)
```

Arguments

`input`	Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.
`cum.pos`	cumulative position (over chromosomes) of each marker
`plot.epi.3d`	If TRUE, draw 3d plot
`plot.epi.2d`	If TRUE, draw 2d plot
`main.epi.3d`	The title of 3d plot. If this argument is NULL, trait name is set as the title.
`main.epi.2d`	The title of 2d plot. If this argument is NULL, trait name is set as the title.
`saveName`	When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveAt = NULL, the plot is not saved.

Value

draw 3d plot and 2d plot to show epistatic effects

`modify.data` *Function to modify genotype and phenotype data to match*

Description

Function to modify genotype and phenotype data to match

Usage

```
modify.data(pheno.mat, geno.mat, pheno.labels = NULL,
  geno.names = NULL, map = NULL, return.ZETA = TRUE,
  return.GWAS.format = FALSE)
```

Arguments

`pheno.mat` A $n_1 \times p$ matrix of phenotype data. rownames(pheno.mat) should be genotype (line; accesion; variety) names.

`geno.mat` A $n_2 \times m$ matrix of marker genotype data. rownames(geno.mat) should be genotype (line; accesion; variety) names.

`pheno.labels` A vector of genotype (line; accesion; variety) names which correpond to phenotypic values.

`geno.names` A vector of genotype (line; accesion; variety) names for marker genotype data (duplication is not recommended).

`map` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position.

`return.ZETA` If this argument is TRUE, the list for mixed model equation (ZETA) will be returned.

`return.GWAS.format`
If this argument is TRUE, phenotype and genotype data for GWAS will be returned.

Value

$geno.modi The modified marker genotype data.

$pheno.modi The modified phenotype data.

$ZETA The list for mixed model equation (ZETA).

$pheno.GWAS GWAS formatted phenotype data.

$geno.GWAS GWAS formatted marker genotype data.

`qq` *Draw qq plot*

Description

Draw qq plot

Usage

```
qq(scores)
```

Arguments

`scores` A vector of -log10(p) for each marker

Value

draw qq plot

RAINBOW *RAINBOW : Perform genome wide asscoiation study (GWAS) by kernel-based methods*

Description

Users can test multiple SNPs simultaneously by kernel-based methods. Users can test not only additive effects but also dominance and epistatic effects.

RGWAS.epistasis *Check epistatic effects by kernel-based GWAS*

Description

Check epistatic effects by kernel-based GWAS

Usage

```
RGWAS.epistasis(pheno, geno, ZETA = NULL, covariate = NULL,
  covariate.factor = NULL, structure.matrix = NULL, n.PC = 0,
  min.MAF = 0.02, n.core = 1, test.method = "LR",
  dominance.eff = TRUE, haplotype = TRUE, num.hap = NULL,
  window.size.half = 5, window.slide = 1, chi0.mixture = 0.5,
  optimizer = "nlminb", gene.set = NULL, plot.epi.3d = TRUE,
  plot.epi.2d = TRUE, main.epi.3d = NULL, main.epi.2d = NULL,
  saveName = NULL, verbose = FALSE, count = TRUE, time = TRUE)
```

Arguments

`pheno` Data frame where the first column is the line name (gid). The remaining columns should be a phenotype to test.

`geno` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. Columns 4 and higher contain the marker scores for each line, coded as -1, 0, 1 = aa, Aa, AA.

`covariate` A $n \times 1$ vector or a $n \times p_1$ matrix. You can insert continuous values, such as other traits or genotype score for special markers. This argument is regarded as one of the fixed effects.

`covariate.factor`
A $n \times p_2$ dataframe. You should assign a factor vector for each column. Then RGWAS changes this argument into model matrix, and this model matrix will be included in the model as fixed effects.

`structure.matrix`
You can use structure matrix calculated by structure analysis when there are population structure. You should not use this argument with n.PC > 0.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`test.method` RGWAS supports two methods to test effects of each SNP-set.

- **"LR"** Likelihood-ratio test, relatively slow, but accurate (default).
- **"score"** Score test, much faster than LR, but sometimes overestimate -log10(p).

`dominance.eff` If this argument is TRUE, dominance effect is included in the model, and additive x dominance and dominance x dominance are also tested as epistatic effects. When you use inbred lines, please set this argument FALSE.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`window.size.half` This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`gene.set` If you have information of gene (or haplotype block), you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`plot.epi.3d` If TRUE, draw 3d plot

`plot.epi.2d` If TRUE, draw 2d plot

`main.epi.3d` The title of 3d plot. If this argument is NULL, trait name is set as the title.

`main.epi.2d` The title of 2d plot. If this argument is NULL, trait name is set as the title.

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`verbose` If this argument is TRUE, welcome message will be shown.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

`time` When time is TRUE, you can know how much time it took to perform RGWAS.

Value

$map Map information for SNPs which are tested epistatic effects.

$scores **$scores** This is the matrix which contains -log10(p) calculated by the test about epistasis effects.

$x, $y The information of the positions of SNPs detected by regular GWAS. These vectors are used when drawing plots. Each output correspond to the repliction of row and column of scores.

$z This is a vector of $scores. This vector is also used when drawing plots.

References

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

Yu, J. et al. (2006) A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Nat Genet. 38(2): 203-208.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome J. 4(3): 250.

Endelman, J.B. and Jannink, J.L. (2012) Shrinkage Estimation of the Realized Relationship Matrix. G3 Genes, Genomes, Genet. 2(11): 1405-1413.

Su, G. et al. (2012) Estimating Additive and Non-Additive Genetic Variances and Predicting Genetic Merits Using Genome-Wide Dense Single Nucleotide Polymorphism Markers. PLoS One. 7(9): 1-7.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Jiang, Y. and Reif, J.C. (2015) Modeling epistasis in genomic selection. Genetics. 201(2): 759-768.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
```

```
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("RAINBOW")


### Modify data
modify.data.res <- modify.data(pheno.mat = y, geno.mat = x, map = map,
                               return.ZETA = TRUE, return.GWAS.format = TRUE)
pheno.GWAS <- modify.data.res$pheno.GWAS
geno.GWAS <- modify.data.res$geno.GWAS
ZETA <- modify.data.res$ZETA


### View each data for RAINBOW
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Check epistatic effects (by regarding 11 SNPs as one SNP-set)
epistasis.res <- RGWAS.epistasis(pheno = pheno.GWAS, geno = geno.GWAS, ZETA = ZETA,
                                 n.PC = 4, test.method = "score", gene.set = NULL,
                                 window.size.half = 40, window.slide = 81)

See(epistasis.res$scores$scores)
```

RGWAS.menu *Print the R code which you should perform for RAINBOW GWAS*

Description

Print the R code which you should perform for RAINBOW GWAS

Usage

```
RGWAS.menu()
```

Value

the R code which you should perform for RAINBOW GWAS

RGWAS.multisnp *Testing multiple SNPs simulataneously for GWAS*

Description

This function performs SNP-set GWAS, which tests multiple SNPs simulataneously. The model of SNP-set GWAS is

$$y = X\beta + Qv + Z_c u_c + Z_r u_r + \epsilon,$$

where y is the vector of phenotypic values, $X\beta$ and Qv are the terms of fixed effects, $Z_c u_c$ and $Z_c u_c$ are the term of random effects and e is the vector of residuals. $X\beta$ indicates all of the fixed effects other than population structure, and often this term also plays a role as an intercept. Qv is the term to correct the effect of population structure. $Z_c u_c$ is the term of polygenetic effects, and suppose that u_c follows the multivariate normal distribution whose variance-covariance matrix is the genetic covariance matrix. $u_c \sim MVN(0, K_c \sigma_c^2)$. $Z_r u_r$ is the term of effects for SNP-set of interest, and suppose that u_r follows the multivariate normal distribution whose variance-covariance matrix is the Gram matrix (linear, exponential, or gaussian kernel) calculated from marker genotype which belong to that SNP-set. Therefore, $u_r \sim MVN(0, K_r \sigma_r^2)$. Finally, the residual term is assumed to identically and independently follow a normal distribution as shown in the following equation. $e \sim MVN(0, I\sigma_e^2)$.

Usage

```
RGWAS.multisnp(pheno, geno, ZETA = NULL, covariate = NULL,
  covariate.factor = NULL, structure.matrix = NULL, n.PC = 0,
  min.MAF = 0.02, test.method = "LR", n.core = 1,
  kernel.method = "linear", kernel.h = "tuned", haplotype = TRUE,
  num.hap = NULL, test.effect = "additive", window.size.half = 5,
  window.slide = 1, chi0.mixture = 0.5, gene.set = NULL,
  weighting.center = TRUE, weighting.other = NULL, sig.level = 0.05,
  method.thres = "BH", plot.qq = TRUE, plot.Manhattan = TRUE,
  plot.method = 1, plot.col1 = c("dark blue", "cornflowerblue"),
  plot.col2 = 1, plot.type = "p", plot.pch = 16, saveName = NULL,
  main.qq = NULL, main.man = NULL, plot.add.last = FALSE,
  return.EMM.res = FALSE, optimizer = "nlminb", thres = TRUE,
  verbose = FALSE, count = TRUE, time = TRUE)
```

Arguments

`pheno` Data frame where the first column is the line name (gid). The remaining columns should be a phenotype to test.

`geno` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. Columns 4 and higher contain the marker scores for each line, coded as -1, 0, 1 = aa, Aa, AA.

`covariate` A $n \times 1$ vector or a $n \times p_1$ matrix. You can insert continuous values, such as other traits or genotype score for special markers. This argument is regarded as one of the fixed effects.

`covariate.factor`
A $n \times p_2$ dataframe. You should assign a factor vector for each column. Then RGWAS changes this argument into model matrix, and this model matrix will be included in the model as fixed effects.

`structure.matrix`
You can use structure matrix calculated by structure analysis when there are population structure. You should not use this argument with n.PC > 0.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`test.method` RGWAS supports two methods to test effects of each SNP-set.

"LR" Likelihood-ratio test, relatively slow, but accurate (default).

"score" Score test, much faster than LR, but sometimes overestimate -log10(p).

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`kernel.method` It determines how to calculate kernel. There are three methods.

"gaussian" It is the default method. Gaussian kernel is calculated by distance matrix.

"exponential" When this method is selected, exponential kernel is calculated by distance matrix.

"linear" When this method is selected, linear kernel is calculated by A.mat.

So local genomic relation matrix is regarded as kernel.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect` Effect of each marker to test. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`
This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`gene.set` If you have information of gene (or haplotype block), you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`weighting.center` In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other` You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`plot.qq` If TRUE, draw qq plot.

`plot.Manhattan` If TRUE, draw manhattan plot.

`plot.method` If this argument = 1, the default manhattan plot will be drawn. If this argument = 2, the manhattan plot with axis based on Position (bp) will be drawn. Also, this plot's color is changed by all chromosomes.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes

`plot.col2` color of the manhattan plot. color changes with chromosome and it starts from plot.col2 + 1 (so plot.col2 = 1 means color starts from red.)

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`main.qq` The title of qq plot. If this argument is NULL, trait name is set as the title.

`main.man` The title of manhattan plot. If this argument is NULL, trait name is set as the title.

`plot.add.last` If saveName is not NULL and this argument is TRUE, then you can add lines or dots to manhattan plots. However, you should also write "dev.off()" after adding something.

`return.EMM.res` When return.EMM.res = TRUE, the results of equation of mixed models are included in the result of RGWAS.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`thres` If thres = TRUE, the threshold of the manhattan plot is included in the result of RGWAS. When return.EMM.res or thres is TRUE, the results will be "list" class.

`verbose`	If this argument is TRUE, welcome message will be shown.
`count`	When count is TRUE, you can know how far RGWAS has ended with percent display.
`time`	When time is TRUE, you can know how much time it took to perform RGWAS.

Details

P-value for each SNP-set is calculated by performing the LR test or the score test (Lippert et al., 2014).

In the LR test, first, the function solves the multi-kernel mixed model and calaculates the maximum restricted log likelihood. Then it performs the LR test by using the fact that the deviance

$$D = 2 \times (LL_{alt} - LL_{null})$$

follows the chi-square distribution.

In the score test, the maximization of the likelihood is only performed for the null model. In other words, the function calculates the score statistic without solving the multi-kernel mixed model for each SNP-set. Then it performs the score test by using the fact that the score statistic follows the chi-square distribution.

Value

$D Dataframe which contains the information of the map you input and the results of RGWAS (-log10(p)) which correspond to the map. If there are more than one test.effects, then multiple lists for each test.effect are returned respectively.

$thres A vector which contains the information of threshold determined by FDR = 0.05.

$EMM.res This output is a list which contains the information about the results of "EMM" perfomed at first in regular GWAS. If you want to know details, see the description for the function "EMM1" or "EMM2".

References

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

Yu, J. et al. (2006) A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Nat Genet. 38(2): 203-208.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome J. 4(3): 250.

Endelman, J.B. and Jannink, J.L. (2012) Shrinkage Estimation of the Realized Relationship Matrix. G3 Genes, Genomes, Genet. 2(11): 1405-1413.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")


### Modify data
modify.data.res <- modify.data(pheno.mat = y, geno.mat = x, map = map,
                               return.ZETA = TRUE, return.GWAS.format = TRUE)
pheno.GWAS <- modify.data.res$pheno.GWAS
geno.GWAS <- modify.data.res$geno.GWAS
ZETA <- modify.data.res$ZETA


### View each data for RAINBOW
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Perform SNP-set GWAS (by regarding 11 SNPs as one SNP-set)
SNP_set.res <- RGWAS.multisnp(pheno = pheno.GWAS, geno = geno.GWAS, ZETA = ZETA,
n.PC = 4, test.method = "LR", kernel.method = "linear", gene.set = NULL,
test.effect = "additive", window.size.half = 5, window.slide = 11)
See(SNP_set.res$D)  ### Column 4 contains -log10(p) values for markers

### Perform SNP-set GWAS 2 (by regarding 11 SNPs as one SNP-set with sliding window)
### It will take almost 25 minutes...
SNP_set.res2 <- RGWAS.multisnp(pheno = pheno.GWAS, geno = geno.GWAS, ZETA = ZETA,
n.PC = 4, test.method = "LR", kernel.method = "linear", gene.set = NULL,
test.effect = "additive", window.size.half = 5, window.slide = 1)
See(SNP_set.res2$D)  ### Column 4 contains -log10(p) values for markers
```

RGWAS.normal *Perform normal GWAS (test each single SNP)*

Description

This function performs single-SNP GWAS. The model of GWAS is

$$y = X\beta + S_i\alpha_i + Qv + Zu + \epsilon,$$

where y is the vector of phenotypic values, $X\beta$, $S_i\alpha_i$, Qv are the terms of fixed effects, Zu is the term of random effects and e is the vector of residuals. $X\beta$ indicates all of the fixed effects other than the effect of SNPs to be tested and of population structure, and often this term also plays a role as an intercept. For $S_i\alpha_i$, S_i is the ith marker of genotype data and α_i is the effect of that marker. Qv is the term to correct the effect of population structure. Zu is the term of polygenetic effects, and suppose that u follows the multivariate normal distribution whose variance-covariance matrix is the genetic covariance matrix. $u \sim MVN(0, K\sigma_u^2)$. Finally, the residual term is assumed to identically and independently follow a normal distribution as shown in the following equation. $e \sim MVN(0, I\sigma_e^2)$.

Usage

```
RGWAS.normal(pheno, geno, ZETA = NULL, covariate = NULL,
  covariate.factor = NULL, structure.matrix = NULL, n.PC = 0,
  min.MAF = 0.02, P3D = TRUE, n.core = 1, sig.level = 0.05,
  method.thres = "BH", plot.qq = TRUE, plot.Manhattan = TRUE,
  plot.method = 1, plot.col1 = c("dark blue", "cornflowerblue"),
  plot.col2 = 1, plot.type = "p", plot.pch = 16, saveName = NULL,
  main.qq = NULL, main.man = NULL, plot.add.last = FALSE,
  return.EMM.res = FALSE, optimizer = "nlminb", thres = TRUE,
  verbose = FALSE, count = TRUE, time = TRUE)
```

Arguments

`pheno` Data frame where the first column is the line name (gid). The remaining columns should be a phenotype to test.

`geno` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. Columns 4 and higher contain the marker scores for each line, coded as -1, 0, 1 = aa, Aa, AA.

`covariate` A $n \times 1$ vector or a $n \times p_1$ matrix. You can insert continuous values, such as other traits or genotype score for special markers. This argument is regarded as one of the fixed effects.

`covariate.factor`
A $n \times p_2$ dataframe. You should assign a factor vector for each column. Then RGWAS changes this argument into model matrix, and this model matrix will be included in the model as fixed effects.

`structure.matrix`
You can use structure matrix calculated by structure analysis when there are population structure. You should not use this argument with n.PC > 0.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`P3D` When P3D = TRUE, variance components are estimated by REML only once, without any markers in the model. When P3D = FALSE, variance components are estimated by REML for each marker separately.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`plot.qq` If TRUE, draw qq plot.

`plot.Manhattan` If TRUE, draw manhattan plot.

`plot.method` If this argument = 1, the default manhattan plot will be drawn. If this argument = 2, the manhattan plot with axis based on Position (bp) will be drawn. Also, this plot's color is changed by all chromosomes.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes

`plot.col2` color of the manhattan plot. color changes with chromosome and it starts from plot.col2 + 1 (so plot.col2 = 1 means color starts from red.)

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`main.qq` The title of qq plot. If this argument is NULL, trait name is set as the title.

`main.man` The title of manhattan plot. If this argument is NULL, trait name is set as the title.

`plot.add.last` If saveName is not NULL and this argument is TRUE, then you can add lines or dots to manhattan plots. However, you should also write "dev.off()" after adding something.

`return.EMM.res` When return.EMM.res = TRUE, the results of equation of mixed models are included in the result of RGWAS.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`thres` If thres = TRUE, the threshold of the manhattan plot is included in the result of RGWAS. When return.EMM.res or thres is TRUE, the results will be "list" class.

`verbose` If this argument is TRUE, welcome message will be shown.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

`time` When time is TRUE, you can know how much time it took to perform RGWAS.

Details

P-value for each marker is calculated by performing F-test against the F-value as follows (Kennedy et al., 1992).

$$F = \frac{(L'\hat{b})'[L'(X'H^{-1}X)^{-1}L]^{-1}(L'\hat{b})}{f\hat{\sigma}_u^2},$$

where b is the vector of coefficients of the fixed effects, which combines β, α_i, v in the horizontal direction and L is a matrix to indicate which effects in b are tested. H is calculated by dividing the estimated variance-covariance matrix for the phenotypic values by σ_u^2, and is calculated by $H = ZKZ' + \hat{\lambda}I$. $\hat{\lambda}$ is the maximum likelihood estimator of the ratio between the residual variance and the additive genetic variance. $\hat{b}$ is the maximum likelihood estimator of b and is calculated by $\hat{b} = (X'H^{-1}X)^{-1}X'H^{-1}y$. f is the number of the fixed effects to be tested, and $\hat{\sigma}_u^2$ is estimated by the following formula.

$$\hat{\sigma}_u^2 = \frac{(y - X\hat{b})'H^{-1}(y - X\hat{b})}{n - p},$$

where n is the sample size and p is the number of the all fixed effects. We calculated each p-value using the fact that the above F-value follows the F distribution with the degree of freedom (f,$n-p$).

Value

$D Dataframe which contains the information of the map you input and the results of RGWAS (-log10(p)) which correspond to the map.

$thres A vector which contains the information of threshold determined by FDR = 0.05.

$EMM.res This output is a list which contains the information about the results of "EMM" perfomed at first in regular GWAS. If you want to know details, see the description for the function "EMM1" or "EMM2".

References

Kennedy, B.W., Quinton, M. and van Arendonk, J.A. (1992) Estimation of effects of single genes on quantitative traits. J Anim Sci. 70(7): 2000-2012.

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

Yu, J. et al. (2006) A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Nat Genet. 38(2): 203-208.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Kang, H.M. et al. (2010) Variance component model to account for sample structure in genome-wide association studies. Nat Genet. 42(4): 348-354.

Zhang, Z. et al. (2010) Mixed linear model approach adapted for genome-wide association studies. Nat Genet. 42(4): 355-360.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome J. 4(3): 250.

Endelman, J.B. and Jannink, J.L. (2012) Shrinkage Estimation of the Realized Relationship Matrix. G3 Genes, Genomes, Genet. 2(11): 1405-1413.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")


### Modify data
modify.data.res <- modify.data(pheno.mat = y, geno.mat = x, map = map,
                               return.ZETA = TRUE, return.GWAS.format = TRUE)
pheno.GWAS <- modify.data.res$pheno.GWAS
geno.GWAS <- modify.data.res$geno.GWAS
ZETA <- modify.data.res$ZETA


### View each data for RAINBOW
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)



### Perform single-SNP GWAS
normal.res <- RGWAS.normal(pheno = pheno.GWAS, geno = geno.GWAS,
                           ZETA = ZETA, n.PC = 4, P3D = TRUE)
See(normal.res$D)  ### Column 4 contains -log10(p) values for markers
```

RGWAS.twostep — *Perform normal GWAS first, then perform SNP-set GWAS for relatively significant markers*

Description

Perform normal GWAS first, then perform SNP-set GWAS for relatively significant markers

Usage

```
RGWAS.twostep(pheno, geno, ZETA = NULL, covariate = NULL,
  covariate.factor = NULL, structure.matrix = NULL, n.PC = 0,
  min.MAF = 0.02, n.core = 1, check.size = 40, check.gene.size = 4,
  kernel.percent = 0.1, GWAS.res.first = NULL, P3D = TRUE,
  test.method.1 = "normal", test.method.2 = "LR",
  kernel.method = "linear", kernel.h = "tuned", haplotype = TRUE,
  num.hap = NULL, test.effect.1 = "additive",
  test.effect.2 = "additive", window.size.half = 5, window.slide = 1,
  chi0.mixture = 0.5, optimizer = "nlminb", gene.set = NULL,
  weighting.center = TRUE, weighting.other = NULL, sig.level = 0.05,
  method.thres = "BH", plot.qq.1 = TRUE, plot.Manhattan.1 = TRUE,
  plot.qq.2 = TRUE, plot.Manhattan.2 = TRUE, plot.method = 1,
  plot.col1 = c("dark blue", "cornflowerblue"), plot.col2 = 1,
  plot.col3 = c("red3", "orange3"), plot.type = "p", plot.pch = 16,
  saveName = NULL, main.qq.1 = NULL, main.man.1 = NULL,
  main.qq.2 = NULL, main.man.2 = NULL, plot.add.last = FALSE,
  return.EMM.res = FALSE, thres = TRUE, verbose = FALSE,
  count = TRUE, time = TRUE)
```

Arguments

`pheno` Data frame where the first column is the line name (gid). The remaining columns should be a phenotype to test.

`geno` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. Columns 4 and higher contain the marker scores for each line, coded as -1, 0, 1 = aa, Aa, AA.

`covariate` A $n \times 1$ vector or a $n \times p_1$ matrix. You can insert continuous values, such as other traits or genotype score for special markers. This argument is regarded as one of the fixed effects.

`covariate.factor`
A $n \times p_2$ dataframe. You should assign a factor vector for each column. Then RGWAS changes this argument into model matrix, and this model matrix will be included in the model as fixed effects.

`structure.matrix`
You can use structure matrix calculated by structure analysis when there are population structure. You should not use this argument with n.PC > 0.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`check.size` This argument determines how many SNPs (around the SNP detected by normal GWAS) you will recalculate -log10(p).

`check.gene.size`
This argument determines how many genes (around the genes detected by normal GWAS) you will recalculate -log10(p). This argument is valid only when you assign "gene.set" argument.

`kernel.percent` This argument determines how many SNPs are detected by normal GWAS. For example, when kernel.percent = 0.1, SNPs whose value of -log10(p) is in the top 0.1 percent are chosen as candidate for recalculation by SNP-set GWAS.

`GWAS.res.first` If you have already performed normal GWAS and have the result, you can skip performing normal GWAS.

`P3D` When P3D = TRUE, variance components are estimated by REML only once, without any markers in the model. When P3D = FALSE, variance components are estimated by REML for each marker separately.

`test.method.1` RGWAS supports two methods to test effects of each SNP-set for 1st GWAS.

"normal" Normal GWAS (default).

"score" Score test, much faster than LR, but sometimes overestimate -log10(p).

`test.method.2` RGWAS supports two methods to test effects of each SNP-set for 2nd GWAS.

"LR" Likelihood-ratio test, relatively slow, but accurate (default).

"score" Score test, much faster than LR, but sometimes overestimate -log10(p).

`kernel.method` It determines how to calculate kernel. There are three methods.

"gaussian" It is the default method. Gaussian kernel is calculated by distance matrix.

"exponential" When this method is selected, exponential kernel is calculated by distance matrix.

"linear" When this method is selected, linear kernel is calculated by A.mat.

So local genomic relation matrix is regarded as kernel.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect.1` Effect of each marker to test for 1st GWAS. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". you can assign only one test effect for the 1st GWAS!

`test.effect.2` Effect of each marker to test for 2nd GWAS. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`gene.set` If you have information of gene (or haplotype block), you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`weighting.center`
In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other`
You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`plot.qq.1` If TRUE, draw qq plot for normal GWAS.

`plot.Manhattan.1`
If TRUE, draw manhattan plot for normal GWAS.

`plot.qq.2` If TRUE, draw qq plot for SNP-set GWAS.

`plot.Manhattan.2`
If TRUE, draw manhattan plot for SNP-set GWAS.

`plot.method` If this argument = 1, the default manhattan plot will be drawn. If this argument = 2, the manhattan plot with axis based on Position (bp) will be drawn. Also, this plot's color is changed by all chromosomes.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes

`plot.col2` color of the manhattan plot. color changes with chromosome and it starts from plot.col2 + 1 (so plot.col2 = 1 means color starts from red.)

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`main.qq.1` The title of qq plot for normal GWAS. If this argument is NULL, trait name is set as the title.

`main.man.1` The title of manhattan plot for normal GWAS. If this argument is NULL, trait name is set as the title.

`main.qq.2` The title of qq plot for SNP-set GWAS. If this argument is NULL, trait name is set as the title.

`main.man.2` The title of manhattan plot for SNP-set GWAS. If this argument is NULL, trait name is set as the title.

`plot.add.last` If saveName is not NULL and this argument is TRUE, then you can add lines or dots to manhattan plots. However, you should also write "dev.off()" after adding something.

`return.EMM.res` When return.EMM.res = TRUE, the results of equation of mixed models are included in the result of RGWAS.

`thres` If thres = TRUE, the threshold of the manhattan plot is included in the result of RGWAS. When return.EMM.res or thres is TRUE, the results will be "list" class.

`verbose` If this argument is TRUE, welcome message will be shown.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

`time` When time is TRUE, you can know how much time it took to perform RGWAS.

Value

$D Dataframe which contains the information of the map you input and the results of RGWAS (-log10(p)) which correspond to the map. -log10(p) by normal GWAS and recalculated -log10(p) by SNP-set GWAS will be obtained. If there are more than one test.effects, then multiple lists for each test.effect are returned respectively.

$thres A vector which contains the information of threshold determined by FDR = 0.05.

$EMM.res This output is a list which contains the information about the results of "EMM" perfomed at first in normal GWAS. If you want to know details, see the description for the function "EMM1" or "EMM2".

References

Kennedy, B.W., Quinton, M. and van Arendonk, J.A. (1992) Estimation of effects of single genes on quantitative traits. J Anim Sci. 70(7): 2000-2012.

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

Yu, J. et al. (2006) A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Nat Genet. 38(2): 203-208.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Kang, H.M. et al. (2010) Variance component model to account for sample structure in genome-wide association studies. Nat Genet. 42(4): 348-354.

Zhang, Z. et al. (2010) Mixed linear model approach adapted for genome-wide association studies. Nat Genet. 42(4): 355-360.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome J. 4(3): 250.

Endelman, J.B. and Jannink, J.L. (2012) Shrinkage Estimation of the Realized Relationship Matrix. G3 Genes, Genomes, Genet. 2(11): 1405-1413.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map


### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")


### Modify data
modify.data.res <- modify.data(pheno.mat = y, geno.mat = x, map = map,
                               return.ZETA = TRUE, return.GWAS.format = TRUE)
pheno.GWAS <- modify.data.res$pheno.GWAS
geno.GWAS <- modify.data.res$geno.GWAS
ZETA <- modify.data.res$ZETA


### View each data for RAINBOW
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Perform two step SNP-set GWAS (single-snp GWAS -> SNP-set GWAS for significant markers)
twostep.SNP_set.res <- RGWAS.twostep(pheno = pheno.GWAS, geno = geno.GWAS, ZETA = ZETA, kernel.percent = 0.2,
                          n.PC = 4, test.method.2 = "LR", kernel.method = "linear", gene.set = NULL,
                          test.effect.2 = "additive", window.size.half = 5, window.slide = 1)

See(twostep.SNP_set.res$D)
### Column 4 contains -log10(p) values for markers with the first method (single-SNP GWAS)
### Column 5 contains -log10(p) values for markers with the second method (SNP-set GWAS)
```

RGWAS.twostep.epi *Perform normal GWAS first, then check epistatic effects for relatively significant markers*

Description

Perform normal GWAS first, then check epistatic effects for relatively significant markers

Usage

```
RGWAS.twostep.epi(pheno, geno, ZETA = NULL, covariate = NULL,
  covariate.factor = NULL, structure.matrix = NULL, n.PC = 0,
  min.MAF = 0.02, n.core = 1, check.size.epi = 4,
  epistasis.percent = 0.05, check.epi.max = 200, your.check = NULL,
  GWAS.res.first = NULL, P3D = TRUE, test.method = "LR",
  dominance.eff = TRUE, haplotype = TRUE, num.hap = NULL,
  optimizer = "nlminb", window.size.half = 5, window.slide = 1,
  chi0.mixture = 0.5, gene.set = NULL, sig.level = 0.05,
  method.thres = "BH", plot.qq.1 = TRUE, plot.Manhattan.1 = TRUE,
  plot.epi.3d = TRUE, plot.epi.2d = TRUE, plot.method = 1,
  plot.col1 = c("dark blue", "cornflowerblue"), plot.col2 = 1,
  plot.type = "p", plot.pch = 16, saveName = NULL,
  main.qq.1 = NULL, main.man.1 = NULL, main.epi.3d = NULL,
  main.epi.2d = NULL, verbose = FALSE, count = TRUE, time = TRUE)
```

Arguments

`pheno` Data frame where the first column is the line name (gid). The remaining columns should be a phenotype to test.

`geno` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. Columns 4 and higher contain the marker scores for each line, coded as -1, 0, 1 = aa, Aa, AA.

`covariate` A $n \times 1$ vector or a $n \times p_1$ matrix. You can insert continuous values, such as other traits or genotype score for special markers. This argument is regarded as one of the fixed effects.

`covariate.factor`
A $n \times p_2$ dataframe. You should assign a factor vector for each column. Then RGWAS changes this argument into model matrix, and this model matrix will be included in the model as fixed effects.

`structure.matrix`
You can use structure matrix calculated by structure analysis when there are population structure. You should not use this argument with n.PC > 0.

`n.PC` Number of principal components to include as fixed effects. Default is 0 (equals K model).

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (use only at UNIX command line).

`check.size.epi` This argument determines how many SNPs (around the SNP detected by normal GWAS) you will check epistasis.

`epistasis.percent`

This argument determines how many SNPs are detected by normal GWAS. For example, when epistasis.percent = 0.1, SNPs whose value of -log10(p) is in the top 0.1 percent are chosen as candidate for checking epistasis.

`your.check` Because there are less SNPs that can be tested in epistasis than in kernel-based GWAS, you can select which SNPs you want to test. If you use this argument, please set the number where SNPs to be tested are located in your data (so not position). In the default setting, your_check = NULL and epistasis between SNPs detected by GWAS will be tested.

`GWAS.res.first` If you have already performed regular GWAS and have the result, you can skip performing normal GWAS.

`P3D` When P3D = TRUE, variance components are estimated by REML only once, without any markers in the model. When P3D = FALSE, variance components are estimated by REML for each marker separately.

`test.method` RGWAS supports two methods to test effects of each SNP-set.

"LR" Likelihood-ratio test, relatively slow, but accurate (default).

"score" Score test, much faster than LR, but sometimes overestimate -log10(p).

`dominance.eff` If this argument is TRUE, dominance effect is included in the model, and additive x dominance and dominance x dominance are also tested as epistatic effects. When you use inbred lines, please set this argument FALSE.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`gene.set` If you have information of gene (or haplotype block), you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`plot.qq.1` If TRUE, draw qq plot for normal GWAS.

`plot.Manhattan.1`
If TRUE, draw manhattan plot for normal GWAS.

`plot.epi.3d` If TRUE, draw 3d plot

`plot.epi.2d` If TRUE, draw 2d plot

`plot.method` If this argument = 1, the default manhattan plot will be drawn. If this argument = 2, the manhattan plot with axis based on Position (bp) will be drawn. Also, this plot's color is changed by all chromosomes.

`plot.col1` This argument determines the color of the manhattan plot. You should substitute this argument as color vector whose length is 2. plot.col1[1] for odd chromosomes and plot.col1[2] for even chromosomes

`plot.col2` color of the manhattan plot. color changes with chromosome and it starts from plot.col2 + 1 (so plot.col2 = 1 means color starts from red.)

`plot.type` This argument determines the type of the manhattan plot. See the help page of "plot".

`plot.pch` This argument determines the shape of the dot of the manhattan plot. See the help page of "plot".

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`main.qq.1` The title of qq plot for normal GWAS. If this argument is NULL, trait name is set as the title.

`main.man.1` The title of manhattan plot for normal GWAS. If this argument is NULL, trait name is set as the title.

`main.epi.3d` The title of 3d plot. If this argument is NULL, trait name is set as the title.

`main.epi.2d` The title of 2d plot. If this argument is NULL, trait name is set as the title.

`verbose` If this argument is TRUE, welcome message will be shown.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

`time` When time is TRUE, you can know how much time it took to perform RGWAS.

`check.size.epi.max`
It takes a lot of time to check epistasis, so you can decide the maximum number of SNPs to check epistasis.

Value

$first The results of first normal GWAS will be returned.

$epistasis **$map** Map information for SNPs which are tested epistatic effects.

$scores **$scores** This is the matrix which contains -log10(p) calculated by the test about epistasis effects.

$x, $y The information of the positions of SNPs detected by regular GWAS. These vectors are used when drawing plots. Each output correspond to the repliction of row and column of scores.

$z This is a vector of $scores. This vector is also used when drawing plots.

References

Kennedy, B.W., Quinton, M. and van Arendonk, J.A. (1992) Estimation of effects of single genes on quantitative traits. J Anim Sci. 70(7): 2000-2012.

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proc Natl Acad Sci. 100(16): 9440-9445.

Yu, J. et al. (2006) A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Nat Genet. 38(2): 203-208.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Kang, H.M. et al. (2010) Variance component model to account for sample structure in genome-wide association studies. Nat Genet. 42(4): 348-354.

Zhang, Z. et al. (2010) Mixed linear model approach adapted for genome-wide association studies. Nat Genet. 42(4): 355-360.

Endelman, J.B. (2011) Ridge Regression and Other Kernels for Genomic Selection with R Package rrBLUP. Plant Genome J. 4(3): 250.

Endelman, J.B. and Jannink, J.L. (2012) Shrinkage Estimation of the Realized Relationship Matrix. G3 Genes, Genomes, Genet. 2(11): 1405-1413.

Su, G. et al. (2012) Estimating Additive and Non-Additive Genetic Variances and Predicting Genetic Merits Using Genome-Wide Dense Single Nucleotide Polymorphism Markers. PLoS One. 7(9): 1-7.

Zhou, X. and Stephens, M. (2012) Genome-wide efficient mixed-model analysis for association studies. Nat Genet. 44(7): 821-824.

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Jiang, Y. and Reif, J.C. (2015) Modeling epistasis in genomic selection. Genetics. 201(2): 759-768.

Examples

```
### Import RAINBOW
require(RAINBOW)

### Load example datasets
data("Rice_Zhao_etal")

### View each dataset
See(Rice_geno_score)
See(Rice_geno_map)
See(Rice_pheno)

### Select one trait for example
trait.name <- "Flowering.time.at.Arkansas"
y <- as.matrix(Rice_pheno[, trait.name, drop = FALSE])

### Remove SNPs whose MAF <= 0.05
x.0 <- t(Rice_geno_score)
MAF.cut.res <- MAF.cut(x.0 = x.0, map.0 = Rice_geno_map)
x <- MAF.cut.res$x
map <- MAF.cut.res$map
```

```
### Estimate genetic relationship matrix
K.A <- rrBLUP::A.mat(x) ### rrBLUP package can be installed by install.packages("rrBLUP")


### Modify data
modify.data.res <- modify.data(pheno.mat = y, geno.mat = x, map = map,
                               return.ZETA = TRUE, return.GWAS.format = TRUE)
pheno.GWAS <- modify.data.res$pheno.GWAS
geno.GWAS <- modify.data.res$geno.GWAS
ZETA <- modify.data.res$ZETA


### View each data for RAINBOW
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Perform two step epistasis GWAS (single-snp GWAS -> Check epistasis for significant markers)
twostep.epi.res <- RGWAS.twostep.epi(pheno = pheno.GWAS, geno = geno.GWAS, ZETA = ZETA,
                                     n.PC = 4, test.method = "score", gene.set = NULL,
                                     window.size.half = 5, window.slide = 11)

See(twostep.epi.res$epistasis$scores)
```

score.calc *Calculate -log10(p) for single-SNP GWAS*

Description

Calculate -log10(p) of each SNP by the Wald test.

Usage

```
score.calc(M.now, ZETA.now, y, X.now, Hinv, P3D = TRUE,
  optimizer = "nlminb", eigen.G = NULL, min.MAF = 0.02,
  count = TRUE)
```

Arguments

`M.now` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`ZETA.now` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X.now` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`Hinv` the inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma_e^2/\sigma_u^2$.

`P3D` When P3D = TRUE, variance components are estimated by REML only once, without any markers in the model. When P3D = FALSE, variance components are estimated by REML for each marker separately.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each marker

References

Kennedy, B.W., Quinton, M. and van Arendonk, J.A. (1992) Estimation of effects of single genes on quantitative traits. J Anim Sci. 70(7): 2000-2012.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Kang, H.M. et al. (2010) Variance component model to account for sample structure in genome-wide association studies. Nat Genet. 42(4): 348-354.

Zhang, Z. et al. (2010) Mixed linear model approach adapted for genome-wide association studies. Nat Genet. 42(4): 355-360.

score.calc.epistasis.LR

Calculate -log10(p) of epistatic effects by LR test

Description

Calculate -log10(p) of epistatic effects by LR test

Usage

```
score.calc.epistasis.LR(M.now, y, X.now, ZETA.now, eigen.SGS = NULL,
  eigen.G = NULL, optimizer = "nlminb", map, haplotype = TRUE,
  num.hap = NULL, window.size.half = 5, window.slide = 1,
  chi0.mixture = 0.5, gene.set = NULL, dominance.eff = TRUE,
  min.MAF = 0.02, count = TRUE)
```

Arguments

`M.now` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X.now` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA.now` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`map` Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the tdeviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`gene.set` If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`dominance.eff` If this argument is TRUE, dominance effect is included in the model, and additive x dominance and dominance x dominance are also tested as epistatic effects. When you use inbred lines, please set this argument FALSE.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

`LL0` The log-likelihood for the null model.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

Value

-log10(p) of epistatic effects for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Jiang, Y. and Reif, J.C. (2015) Modeling epistasis in genomic selection. Genetics. 201(2): 759-768.

score.calc.epistasis.score

Calculate -log10(p) of epistatic effects with score test

Description

Calculate -log10(p) of epistatic effects with score test

Usage

```
score.calc.epistasis.score(M.now, y, X.now, ZETA.now, Gu, Ge, P0, map,
  haplotype = TRUE, num.hap = NULL, window.size.half = 5,
  window.slide = 1, chi0.mixture = 0.5, gene.set = NULL,
  dominance.eff = TRUE, min.MAF = 0.02, count = TRUE)
```

Arguments

M.now
: n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

y
: $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

X.now
: $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

ZETA.now
: A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

Gu
: $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

Ge
: $n \times n$ matrix. You should assign identity matrix I (diag(n)).

P0
: $n \times n$ matrix. The Moore-Penrose generalized inverse of $SV0S$, where $S = X(X'X)^{-1}X'$ and $V0 = \sigma_u^2 Gu + \sigma_e^2 Ge$. σ_u^2 and σ_e^2 are estimators of the null model.

map
: Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

haplotype
: If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

num.hap
: When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

window.size.half
: This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

window.slide
: This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

chi0.mixture
: RAINBOW assumes the test statistic $l1'Fl1$ is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where l1 is the first derivative of the log-likelihood and F is the Fisher information. And r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

gene.set
: If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

dominance.eff
: If this argument is TRUE, dominance effect is included in the model, and additive x dominance and dominance x dominance are also tested as epistatic effects. When you use inbred lines, please set this argument FALSE.

min.MAF
: Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

count
: When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) of epistatic effects for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

Jiang, Y. and Reif, J.C. (2015) Modeling epistasis in genomic selection. Genetics. 201(2): 759-768.

`score.calc.LR` *Calculate -log10(p) of each SNP-set by the LR test*

Description

This function calculates -log10(p) of each SNP-set by the LR test. First, the function solves the multi-kernel mixed model and calaculates the maximum restricted log likelihood. Then it performs the LR test by using the fact that the deviance

$$D = 2 \times (LL_{alt} - LL_{null})$$

follows the chi-square distribution.

Usage

```
score.calc.LR(M.now, y, X.now, ZETA.now, LL0, eigen.SGS = NULL,
  eigen.G = NULL, optimizer = "nlminb", map,
  kernel.method = "linear", kernel.h = "tuned", haplotype = TRUE,
  num.hap = NULL, test.effect = "additive", window.size.half = 5,
  window.slide = 1, chi0.mixture = 0.5, weighting.center = TRUE,
  weighting.other = NULL, gene.set = NULL, min.MAF = 0.02,
  count = TRUE)
```

Arguments

M.now
: n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

y
: $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

X.now
: $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

ZETA.now
: A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

LL0
: The log-likelihood for the null model.

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`map` Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`kernel.method` It determines how to calculate kernel. There are three methods.

"gaussian" It is the default method. Gaussian kernel is calculated by distance matrix.

"exponential" When this method is selected, exponential kernel is calculated by distance matrix.

"linear" When this method is selected, linear kernel is calculated by A.mat.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect` Effect of each marker to test. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`weighting.center` In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other` You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`gene.set` If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

`score.calc.LR.MC` *Calculate -log10(p) of each SNP-set by the LR test (multi-cores)*

Description

This function calculates -log10(p) of each SNP-set by the LR test. First, the function solves the multi-kernel mixed model and calaculates the maximum restricted log likelihood. Then it performs the LR test by using the fact that the deviance

$$D = 2 \times (LL_{alt} - LL_{null})$$

follows the chi-square distribution.

Usage

```
score.calc.LR.MC(M.now, y, X.now, ZETA.now, LL0, eigen.SGS = NULL,
  eigen.G = NULL, n.core = 2, map, kernel.method = "linear",
  kernel.h = "tuned", haplotype = TRUE, num.hap = NULL,
  test.effect = "additive", window.size.half = 5, window.slide = 1,
  optimizer = "nlminb", chi0.mixture = 0.5, weighting.center = TRUE,
  weighting.other = NULL, gene.set = NULL, min.MAF = 0.02,
  count = TRUE)
```

Arguments

`M.now` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X.now` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA.now` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`LL0` The log-likelihood for the null model.

`eigen.SGS` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of SGS, where $S = I - X(X'X)^{-1}X'$, $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores.

`map` Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`kernel.method` It determines how to calculate kernel. There are three methods.

"gaussian" It is the default method. Gaussian kernel is calculated by distance matrix.

"exponential" When this method is selected, exponential kernel is calculated by distance matrix.

"linear" When this method is selected, linear kernel is calculated by A.mat.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect` Effect of each marker to test. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`chi0.mixture` RAINBOW assumes the deviance is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`weighting.center`

In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other`

You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`gene.set` If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

score.calc.MC *Calculate -log10(p) for single-SNP GWAS (multi-cores)*

Description

Calculate -log10(p) of each SNP by the Wald test.

Usage

```
score.calc.MC(M.now, ZETA.now, y, X.now, Hinv, n.core = 2, P3D = TRUE,
  optimizer = "nlminb", eigen.G = NULL, min.MAF = 0.02,
  count = TRUE)
```

Arguments

`M.now` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`ZETA.now` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X.now` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`Hinv` the inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma^2_e / \sigma^2_u$.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores.

`P3D` When P3D = TRUE, variance components are estimated by REML only once, without any markers in the model. When P3D = FALSE, variance components are estimated by REML for each marker separately.

`optimizer` The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions.

`eigen.G` A list with

$values eigen values

$vectors eigen vectors

The result of the eigen decompsition of $G = ZKZ'$. You can use "spectralG.cpp" function in RAINBOW. If this argument is NULL, the eigen decomposition will be performed in this function. We recommend you assign the result of the eigen decomposition beforehand for time saving.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each marker

References

Kennedy, B.W., Quinton, M. and van Arendonk, J.A. (1992) Estimation of effects of single genes on quantitative traits. J Anim Sci. 70(7): 2000-2012.

Kang, H.M. et al. (2008) Efficient Control of Population Structure in Model Organism Association Mapping. Genetics. 178(3): 1709-1723.

Kang, H.M. et al. (2010) Variance component model to account for sample structure in genome-wide association studies. Nat Genet. 42(4): 348-354.

Zhang, Z. et al. (2010) Mixed linear model approach adapted for genome-wide association studies. Nat Genet. 42(4): 355-360.

`score.calc.score` *Calculate -log10(p) of each SNP-set by the score test*

Description

This function calculates -log10(p) of each SNP-set by the score test. First, the function calculates the score statistic without solving the multi-kernel mixed model for each SNP-set. Then it performs the score test by using the fact that the score statistic follows the chi-square distribution.

Usage

```
score.calc.score(M.now, y, X.now, ZETA.now, LL0, Gu, Ge, P0, map,
  kernel.method = "linear", kernel.h = "tuned", haplotype = TRUE,
  num.hap = NULL, test.effect = "additive", window.size.half = 5,
  window.slide = 1, chi0.mixture = 0.5, weighting.center = TRUE,
  weighting.other = NULL, gene.set = NULL, min.MAF = 0.02,
  count = TRUE)
```

Arguments

`M.now` n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X.now` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`ZETA.now` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`LL0` The log-likelihood for the null model.

`Gu` $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

`Ge` $n \times n$ matrix. You should assign identity matrix I (diag(n)).

`P0` $n \times n$ matrix. The Moore-Penrose generalized inverse of $SV0S$, where $S = X(X'X)^{-1}X'$ and $V0 = \sigma_u^2 Gu + \sigma_e^2 Ge$. σ_u^2 and σ_e^2 are estimators of the null model.

`map` Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`kernel.method` It determines how to calculate kernel. There are three methods.

- **"gaussian"** It is the default method. Gaussian kernel is calculated by distance matrix.
- **"exponential"** When this method is selected, exponential kernel is calculated by distance matrix.
- **"linear"** When this method is selected, linear kernel is calculated by A.mat.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect` Effect of each marker to test. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the test statistic $l1'Fl1$ is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where l1 is the first derivative of the log-likelihood and F is the Fisher information. And r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`weighting.center`

In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other`

You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`gene.set` If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should

	assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.
`min.MAF`	Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.
`count`	When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

`score.calc.score.MC` *Calculate -log10(p) of each SNP-set by the score test (multi-cores)*

Description

This function calculates -log10(p) of each SNP-set by the score test. First, the function calculates the score statistic without solving the multi-kernel mixed model for each SNP-set. Then it performs the score test by using the fact that the score statistic follows the chi-square distribution.

Usage

```
score.calc.score.MC(M.now, y, X.now, ZETA.now, LL0, Gu, Ge, P0,
  n.core = 2, map, kernel.method = "linear", kernel.h = "tuned",
  haplotype = TRUE, num.hap = NULL, test.effect = "additive",
  window.size.half = 5, window.slide = 1, chi0.mixture = 0.5,
  weighting.center = TRUE, weighting.other = NULL, gene.set = NULL,
  min.MAF = 0.02, count = TRUE)
```

Arguments

`M.now`	n.sample x n.mark genotype matrix where n.sample is sample size and n.mark is the number of markers.
`y`	$n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	$n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.
`ZETA.now`	A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!
`LL0`	The log-likelihood for the null model.
`Gu`	$n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

`Ge` $n \times n$ matrix. You should assign identity matrix I (diag(n)).

`P0` $n \times n$ matrix. The Moore-Penrose generalized inverse of $SV0S$, where $S = X(X'X)^{-1}X'$ and $V0 = \sigma_u^2 Gu + \sigma_e^2 Ge$. σ_u^2 and σ_e^2 are estimators of the null model.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores.

`map` Data frame of map information where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`kernel.method` It determines how to calculate kernel. There are three methods.

- **"gaussian"** It is the default method. Gaussian kernel is calculated by distance matrix.
- **"exponential"** When this method is selected, exponential kernel is calculated by distance matrix.
- **"linear"** When this method is selected, linear kernel is calculated by A.mat.

`kernel.h` The hyper parameter for gaussian or exponential kernel. If kernel.h = "tuned", this hyper parameter is calculated as the median of off-diagonals of distance matrix of genotype data.

`haplotype` If the number of lines of your data is large (maybe > 100), you should set haplotype = TRUE. When haplotype = TRUE, haplotype-based kernel will be used for calculating -log10(p). (So the dimension of this gram matrix will be smaller.) The result won't be changed, but the time for the calculation will be shorter.

`num.hap` When haplotype = TRUE, you can set the number of haplotypes which you expect. Then similar arrays are considered as the same haplotype, and then make kernel(K.SNP) whose dimension is num.hap x num.hap. When num.hap = NULL (default), num.hap will be set as the maximum number which reflects the difference between lines.

`test.effect` Effect of each marker to test. You can choose "test.effect" from "additive", "dominance" and "additive+dominance". You also can choose more than one effect, for example, test.effect = c("additive", "aditive+dominance")

`window.size.half`

This argument decides how many SNPs (around the SNP you want to test) are used to calculated K.SNP. More precisely, the number of SNPs will be 2 * window.size.half + 1.

`window.slide` This argument determines how often you test markers. If window.slide = 1, every marker will be tested. If you want to perform SNP set by bins, please set window.slide = 2 * window.size.half + 1.

`chi0.mixture` RAINBOW assumes the test statistic $l1'Fl1$ is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where l1 is the first derivative of the log-likelihood and F is the Fisher information. And r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

`weighting.center`

In kernel-based GWAS, weights according to the Gaussian distribution (centered on the tested SNP) are taken into account when calculating the kernel if Rainbow = TRUE. If Rainbow = FALSE, weights are not taken into account.

`weighting.other`

You can set other weights in addition to weighting.center. The length of this argument should be equal to the number of SNPs. For example, you can assign SNP effects from the information of gene annotation.

`gene.set` If you have information of gene, you can use it to perform kernel-based GWAS. You should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "geno" argument.

`min.MAF` Specifies the minimum minor allele frequency (MAF). If a marker has a MAF less than min.MAF, it is assigned a zero score.

`count` When count is TRUE, you can know how far RGWAS has ended with percent display.

Value

-log10(p) for each SNP-set

References

Listgarten, J. et al. (2013) A powerful and efficient set test for genetic markers that handles confounders. Bioinformatics. 29(12): 1526-1533.

Lippert, C. et al. (2014) Greater power and computational efficiency for kernel-based association testing of sets of genetic variants. Bioinformatics. 30(22): 3206-3214.

`score.cpp` *Calculte -log10(p) by score test (slow, for general cases)*

Description

Calculte -log10(p) by score test (slow, for general cases)

Usage

```
score.cpp(y, Gs, Gu, Ge, P0, chi0.mixture = 0.5)
```

Arguments

`y` $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`Gs` A list of kernel matrices you want to test. For example, Gs = list(A.part = K.A.part, D.part = K.D.part)

`Gu` $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

`Ge` $n \times n$ matrix. You should assign identity matrix I (diag(n)).

`P0` $n \times n$ matrix. The Moore-Penrose generalized inverse of $SV0S$, where $S = X(X'X)^{-1}X'$ and $V0 = \sigma^2_u Gu + \sigma^2_e Ge$. σ^2_u and σ^2_e are estimators of the null model.

`chi0.mixture` RAINBOW assumes the test statistic $l1'Fl1$ is considered to follow a x chisq(df = 0) + (1 - a) x chisq(df = r). where l1 is the first derivative of the log-likelihood and F is the Fisher information. And r is the degree of freedom. The argument chi0.mixture is a (0 <= a < 1), and default is 0.5.

Value

-log10(p) calculated by score test

`score.linker.cpp` *Calculte -log10(p) by score test (fast, for limited cases)*

Description

Calculte -log10(p) by score test (fast, for limited cases)

Usage

```
score.linker.cpp(y, Ws, Gammas, gammas.diag = TRUE, Gu, Ge, P0,
  chi0.mixture = 0.5)
```

Arguments

`y`	$n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`Ws`	A list of low rank matrices (ZW; $n \times k$ matrix). This forms linear kernel $ZKZ' = ZW\Gamma(ZW)'$. For example, Ws = list(A.part = ZW.A, D.part = ZW.D)
`Gammas`	A list of matrices for weighting SNPs (Gamma; $k \times k$ matrix). This forms linear kernel $ZKZ' = ZW\Gamma(ZW)'$. For example, if there is no weighting, Gammas = lapply(Ws, function(x) diag(ncol(x)))
`gammas.diag`	If each Gamma is the diagonal matrix, please set this argument TRUE. The calculation time can be saved.
`Gu`	$n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.
`Ge`	$n \times n$ matrix. You should assign identity matrix I (diag(n)).
`P0`	$n \times n$ matrix. The Moore-Penrose generalized inverse of $SV0S$, where $S = X(X'X)^{-1}X'$ and $V0 = \sigma_u^2 Gu + \sigma_e^2 Ge$. σ_u^2 and σ_e^2 are estimators of the null model.
`chi0.mixture`	RAINBOW assumes the statistic $l1'Fl1$ follows the mixture of χ_0^2 and χ_r^2, where l1 is the first derivative of the log-likelihood and F is the Fisher information. And r is the degree of freedom. chi0.mixture determins the proportion of χ_0^2

Value

-log10(p) calculated by score test

`See` *Function to view the first part of data (like head(), tail())*

Description

Function to view the first part of data (like head(), tail())

Usage

```
See(data, fh = TRUE, fl = TRUE, rown = 6, coln = 6, rowst = 1,
  colst = 1, narray = 2, drop = FALSE, save.variable = FALSE)
```

Arguments

`data` Your data. 'vector', 'matrix', 'array' (whose dimensions <= 4), 'data.frame' are supported format. If other formatted data is assigned, str(data) will be returned.

`fh` From head. If this argument is TRUE, first part (row) of data will be shown (like head() function). If FALSE, last part (row) of your data will be shown (like tail() function).

`fl` From left. If this argument is TRUE, first part (column) of data will be shown (like head() function). If FALSE, last part (column) of your data will be shown (like tail() function).

`rown` The number of rows shown in console.

`coln` The number of columns shown in console.

`rowst` The start point for the direction of row.

`colst` The start point for the direction of column.

`narray` The number of dimensions other than row and column shown in console. This argument is effective only your data is array (whose dimensions >= 3).

`drop` When rown = 1 or coln = 1, the dimension will be reduced if this argument is TRUE.

`save.variable` If you want to assign the result to a variable, please set this agument TRUE.

Value

If save.variable is FALSE, NULL. If TRUE, the first part of your data will be returned.

`spectralG.cpp` *Perform spectral decomposition (inplemented by Rcpp)*

Description

Perform spectral decomposition for $G = ZKZ'$ or SGS where $S = I - X(X'X)^{-1}X$.

Usage

```
spectralG.cpp(ZETA, ZWs = NULL, X = NULL, weights = 1,
  return.G = TRUE, return.SGS = FALSE, spectral.method = NULL,
  tol = NULL, df.H = NULL)
```

Arguments

`ZETA` A list of variance (relationship) matrix (K; $m \times m$) and its design matrix (Z; $n \times m$) of random effects. You can use only one kernel matrix. For example, ZETA = list(A = list(Z = Z, K = K)) Please set names of list "Z" and "K"!

`X` $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.

`weights` If the length of ZETA >= 2, you should assign the ratio of variance components to this argument.

`return.G` If thie argument is TRUE, spectral decomposition results of G will be returned. ($G = ZKZ'$)

`return.SGS` If this argument is TRUE, spectral decomposition results of SGS will be returned. $(S = I - X(X'X)^{-1}X, G = ZKZ')$

`spectral.method` The method of spectral decomposition. In this function, "eigen" : eigen decomposition and "cholesky" : cholesky and singular value decomposition are offered. If this argument is NULL, either method will be chosen accorsing to the dimension of Z and X.

`tol` The tolerance for detecting linear dependencies in the columns of G = ZKZ'. Eigen vectors whose eigen values are less than "tol" argument will be omitted from results. If tol is NULL, top 'n' eigen values will be effective.

`df.H` The degree of freedom of K matrix. If this argument is NULL, min(n, sum(nrow(K1), nrow(K2), ...)) will be assigned.

Value

**$spectral.G** The spectral decomposition results of G. \item$Ueigen vectors of G. \item$deltaeigen values of G.

$spectral.SGS estimator for σ_e^2

$Q eigen vectors of SGS.

$theta eigen values of sGS.

SS_gwas *Calculate some summary statistics of GWAS (for simulation study)*

Description

Calculate some summary statistics of GWAS (for simulation study)

Usage

```
SS_gwas(res, x, map.x, qtn.candidate, gene.set = NULL,
  n.top.false.block = 10, sig.level = c(0.05, 0.01),
  method.thres = "BH", inflator.plus = 2, LD_length = 150000,
  cor.thres = 0.35, window.size = 0, saveName = NULL,
  plot.ROC = TRUE)
```

Arguments

`res` Data frame of GWAS results where the first column is the marker names, the second and third column is the chromosome amd map position, and the forth column is -log10(p) for each marker.

`x` N (lines) x M (markers) marker genotype data (matrix), coded as -1, 0, 1 = aa, Aa, AA.

`map.x` Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position.

`qtn.candidate` A vector of causal markers. You should assign where those causal markers are positioned in our marker genotype, rather than physical position of those causal markers.

`gene.set` If you have information of gene (or haplotype block), and if you used it to perform kernel-based GWAS, you should assign your gene information to gene.set in the form of a "data.frame" (whose dimension is (the number of gene) x 2). In the first column, you should assign the gene name. And in the second column, you should assign the names of each marker, which correspond to the marker names of "x" argument.

`n.top.false.block` We will calculate the mean of -log10(p) values of top 'n.top.false.block' blocks to evaluate the inflation level of results. The default is 10.

`sig.level` Significance level for the threshold. The default is 0.05.

`method.thres` Method for detemining threshold of significance. "BH" and "Bonferroni are offered.

`LD_length` SNPs within the extent of LD are regareded as one set. This LD_length determines the size of LD block, and 2 x LD_length (b.p.) will be the size of LD block.

`cor.thres` SNPs within the extent of LD are regareded as one set. This cor.thres also determines the size of LD block, and the region with square of correlation coefficients >= cor.thres is regareded as one LD block. More precisely, the regions which satisfies both LD_length and cor.thres condition is rearded as one LD block.

`window.size` If you peform SNP-set analysis with slinding window, we can consider the effect of window size by this argument.

`saveName` When drawing any plot, you can save plots in png format. In saveName, you should substitute the name you want to save. When saveName = NULL, the plot is not saved.

`plot.ROC` If this argunent is TRUE, ROC (Reciever Operating Characteristic) curve will be drawn with AUC (Area Under the Curve).

Value

$log.p -log10(p)) values of the causals.

$qtn.logp.order The rank of -log10(p) of causals.

$thres A vector which contains the information of threshold.

$overthres The number of markers which exceed the threshold.

$AUC Area under the curve.

$AUC.relax Area under the curve calculated with LD block units.

$FDR False discovery rate. 1 - Precision.

$FPR False positive rate.

$FNR False negative rate. 1 - Recall.

$Recall The proportion of the number of causals dected by GWAS to the number of causals you set.

$Precision The proportion of the number of causals dected by GWAS to the number of markers detected by GWAS.

$Accuracy The accuracy of GWAS results.

$Hm Harmonic mean of Recall and Precision.

$haplo.name The haplotype block name which correspond to causals.

$mean.false The mean of -log10(p) values of top 'n.top.false.block' blocks.

$max.trues Max of the -log10(p) values of the region near causals.

`welcome_to_RGWAS` *Function to greet to users*

Description

Function to greet to users

Usage

```
welcome_to_RGWAS()
```

Value

show welcome messages

Index