Package 'RAINBOWR'

June 29, 2022

Type Package

Title Genome-Wide Association Study with SNP-Set Methods

Version 0.1.31

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

Description By using 'RAINBOWR' (Reliable Association INference By Optimizing Weights with R), users can test multiple SNPs (Single Nucleotide Polymorphisms) simultaneously by kernel-based (SNP-set) methods. This package can also be applied to haplotype-based GWAS (Genome-Wide Association Study). Users can test not only additive effects but also dominance and epistatic effects. In detail, please check our paper on PLOS Computational Biology: Kosuke Hamazaki and Hiroyoshi Iwata (2020) <doi:10.1371/journal.pcbi.1007663>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports Rcpp, Matrix, cluster, MASS, pbmcapply, optimx, methods, ape, stringr, pegas, rrBLUP, expm, here, htmlwidgets, Rfast, gaston, MM4LMM

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.1.2

Suggests knitr, rmarkdown, plotly, haplotypes, adegenet, ggplot2, ggtree, scatterpie, phylobase, furrr, future, progressr, foreach, doParallel, data.table

VignetteBuilder knitr

Author Kosuke Hamazaki [aut, cre], Hiroyoshi Iwata [aut, ctb]

R topics documented:

`adjustGRM` *Function to adjust genomic relationship matrix (GRM) with subpopulations*

Description

Function to adjust genomic relationship matrix (GRM) with subpopulations

Usage

```
adjustGRM(
  y,
  X = NULL,
  ZETA,
  subpopInfo = NULL,
  nSubpop = 5,
  nPcsFindCluster = 10,
  include.epistasis = FALSE,
  package.MM = "gaston"
)
```

Arguments

`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X`	A $n \times p$ 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 only one kernel matrix for this function. For example, ZETA = list(A = list(Z = Z.A, K = K.A)) (A for additive) Please set names of lists "Z" and "K"!
`subpopInfo`	The information on group memberships (e.g., subgroups for the population) will be required. You can set a vector of group names (or clustering ids) for each genotype as this argument. This vector should be factor.
`nSubpop`	When ‘subpopInfo = NULL‘, ‘subpopInfo‘ will be automatically determined by using `find.clusters` function. You should specify the number of groups by this argument to decide ‘subpopInfo‘.
`nPcsFindCluster`	Number of principal components to be used for ‘adegenet::find.clusters‘. This argument is used inly when ‘subpopInfo‘ is ‘NULL‘.
`include.epistasis`	Whether or not including the genome-wide epistastic effects into the model to adjust ZETA.
`package.MM`	The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

Value

A List of

Adjusted ZETA including only one kernel.

$ZETAAd$subpopInfo A vector of ‘subpopInfo‘ used in this function.

$covariates A matrix of covariates used in the mixed effects model. #’

$nullModel Results of mixed-effects model for multiple kernels.

$nSubpop ‘nSubpop‘ used in this function.

$include.epistasis ‘include.epistasis‘ used in this function.

References

Rio S, Mary-Huard T, Moreau L, Bauland C, Palaffre C, et al. (2020) Disentangling group specific QTL allele effects from genetic background epistasis using admixed individuals in GWAS: An application to maize flowering. PLOS Genetics 16(3): e1008241.

---

calcGRM — *Function to calculate genomic relationship matrix (GRM)*

Description

Function to calculate genomic relationship matrix (GRM)

Usage

```
calcGRM(
  genoMat,
  methodGRM = "addNOIA",
  subpop = NULL,
  kernel.h = "tuned",
  returnWMat = FALSE,
  probaa = NULL,
  probAa = NULL
)
```

Arguments

`genoMat`	A $N \times M$ matrix of marker genotype
`methodGRM`	Method to calculate genomic relationship matrix (GRM). We offer the following methods; "addNOIA", "domNOIA", "A.mat", "linear", "gaussian", "exponential", "correlation". For NOIA methods, please refer to Vitezica et al. 2017.
`subpop`	Sub-population names corresponding to each individual. By utilizing ‘subpop‘ argument, you can consider the difference of allele frequencies between subpopulations when computing the genomic relationship matrix. This argument is only valid when NOIA methods are selected.
`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.

returnWMat
: If this argument is TRUE, we will return W matrix instead of GRM. Here, W satisfies $GRM = WW^T$. W corresponds to H matix in Vitezica et al. 2017.

probaa
: Probability of being homozygous for the reference allele for each marker. If NULL (default), it will be calculated from genoMat.

probAa
: Probability of being heterozygous for the reference and alternative alleles for each marker If NULL (default), it will be calculated from genoMat.

Value

genomic relationship matrix (GRM)

References

Vitezica, Z.G., Legarra, A., Toro, M.A. and Varona, L. (2017) Orthogonal Estimates of Variances for Additive, Dominance, and Epistatic Effects in Populations. Genetics. 206(3): 1297-1307.

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

`CalcThreshold` *Function to calculate threshold for GWAS*

Description

Calculate thresholds for the given GWAS (genome-wide association studies) 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.

`convertBlockList` *Function to convert haplotype block list from PLINK to RAINBOWR format*

Description

Function to convert haplotype block list from PLINK to RAINBOWR format

Usage

```
convertBlockList(
  fileNameBlocksDetPlink,
  map,
  blockNamesHead = "haploblock_",
  imputeOneSNP = FALSE,
  insertZeros = FALSE,
  n.core = 1,
  parallel.method = "mclapply",
  count = FALSE
)
```

Arguments

`fileNameBlocksDetPlink`
: File name of the haplotype block list generated by PLINK (See reference). The file names must contain ".blocks.det" in the tail.

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

`blockNamesHead`
: You can specify the header of block names for the returned data.frame.

`imputeOneSNP`
: As default, blocks including only one SNP will be discarded from the returned data. If you want to include them when creating haplotype-block list for RAINBOWR, please set ‘imputeOneSNP = TRUE‘.

`insertZeros`
: When naming blocks, whether or not inserting zeros to the name of blocks. For example, if there are 1,000 blocks in total, the function will name the block 1 as "block_1" when ‘insertZeros = FALSE‘ and "block_0001" when ‘insertZeros = TRUE‘.

`n.core`
: Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

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

Value

A data.frame object of

Block names for SNP-set methods in RAINBOWR

$block$marker Marker names in each block for SNP-set methods in RAINBOWR

Purcell, S. and Chang, C. (2018). PLINK 1.9, www.cog-genomics.org/plink/1.9/. Chang CC, Chow CC, Tellier LCAM, Vattikuti S, Purcell SM, Lee JJ (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience, 4. Gaunt T, Rodríguez S, Day I (2007) Cubic exact solutions for the estimation of pairwise haplotype frequencies: implications for linkage disequilibrium analyses and a web tool 'CubeX'. BMC Bioinformatics, 8. Taliun D, Gamper J, Pattaro C (2014) Efficient haplotype block recognition of very long and dense genetic sequences. BMC Bioinformatics, 15.

`cumsumPos` *Function to calculate cumulative position (beyond chromosome)*

Description

Function to calculate cumulative position (beyond chromosome)

Usage

```
cumsumPos(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,
  n.core = NA,
  optimizer = "nlminb",
  traceInside = 0,
  n.thres = 450,
  REML = TRUE,
```

```
  pred = TRUE,
  return.u.always = TRUE,
  return.u.each = TRUE,
  return.Hinv = TRUE
)
```

Arguments

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

X0 — A $n \times p$ 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 RAINBOWR. 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 RAINBOWR. 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.

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

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.

return.u.always — If TRUE, BLUP ('u'; u) will be returned.

return.u.each
: If TRUE, the function also computes each BLUP corresponding to different kernels (when solving multi-kernel mixed-effects model). It takes additional time compared to the one with 'return.u.each = FALSE'.

return.Hinv
: If TRUE, $H^{-1} = (Var[y] / \sum_{l=1}^{L} \sigma_l^2)^{-1}$ will be computed. It also returns $V^{-1} = (Var[y])^{-1}$.

Value

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

$Vu Estimator for σ_u^2, all of the genetic variance

$Ve Estimator for σ_e^2

$beta BLUE($\beta$)

$u BLUP(Sum of Zu)

$u.each BLUP(Each 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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM) & epistatic relationship matrix
K.A <- calcGRM(genoMat = x)
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, X0 = 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$beta)   ### Here, this is an intercept.
u.each <- EM3.res$u.each   ### estimated genotypic values (additive, additive x additive)
See(u.each)


### Perform genomic prediction with 10-fold cross validation (multi-kernel)
noNA <- !is.na(c(pheno.mat))   ### NA (missing) in the phenotype data

phenoNoNA <- pheno.mat[noNA, , drop = FALSE]   ### remove NA
ZETANoNA <- ZETA
ZETANoNA <- lapply(X = ZETANoNA, FUN = function (List) {
  List$Z <- List$Z[noNA, ]

  return(List)
})   ### remove NA


nFold <- 10    ### # of folds
nLine <- nrow(phenoNoNA)
idCV <- sample(1:nLine %% nFold)   ### assign random ids for cross-validation
idCV[idCV == 0] <- nFold

yPred <- rep(NA, nLine)

for (noCV in 1:nFold) {
  print(paste0("Fold: ", noCV))
  yTrain <- phenoNoNA
  yTrain[idCV == noCV, ] <- NA   ### prepare test data

  EM3.resCV <- EM3.cpp(y = yTrain, X0 = NULL, ZETA = ZETANoNA)   ### prediction
  yTest <-  EM3.resCV$y.pred     ### predicted values

  yPred[idCV == noCV] <- yTest[idCV == noCV]
```

```
}

### Plot the results
plotRange <- range(phenoNoNA, yPred)
plot(x = phenoNoNA, y = yPred,xlim = plotRange, ylim = plotRange,
     xlab = "Observed values", ylab = "Predicted values",
     main = "Results of Genomic Prediction (multi-kernel)",
     cex.lab = 1.5, cex.main = 1.5, cex.axis = 1.3)
abline(a = 0, b = 1, col = 2, lwd = 2, lty = 2)
R2 <- cor(x = phenoNoNA[, 1], y = yPred) ^ 2
text(x = plotRange[2] - 10,
     y = plotRange[1] + 10,
     paste0("R2 = ", round(R2, 3)),
     cex = 1.5)
```

EM3.general	*Equation of mixed model for multi-kernel including using other packages (with other packages, much faster than EM3.cpp)*

Description

This function solves the following multi-kernel linear mixed effects model using `MMEst` function in 'MM4LMM' package, `lmm.aireml` or `lmm.diago` functions in 'gaston' package, or `EM3.cpp` function in 'RAINBOWR' package.

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

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

Usage

```
EM3.general(
  y,
  X0 = NULL,
  ZETA,
  eigen.G = NULL,
  package = "gaston",
  tol = NULL,
  n.core = 1,
  optimizer = "nlminb",
  REML = TRUE,
  pred = TRUE,
  return.u.always = TRUE,
  return.u.each = TRUE,
  return.Hinv = TRUE,
  recheck.RAINBOWR = TRUE,
  var.ratio.range = c(1e-09, 1e+07)
)
```

Arguments

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

X0
: A $n \times p$ 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 RAINBOWR. 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.

package
: Package name to be used in this function. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘.

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.

n.core
: Setting n.core > 1 will enable parallel execution on a machine with multiple cores. (‘n.core‘ will be replaced by 1 for ‘package = ’gaston’‘)

optimizer
: The function used in the optimization process. We offer "optim", "optimx", and "nlminb" functions. This argument is only valid when ‘package = ’RAINBOWR’‘.

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.

return.u.always
: When using the "gaston" package with missing values or using the "MM4LMM" package (with/without missings), computing BLUP will take some time in addition to solving the mixed-effects model. You can choose whether BLUP (‘u‘; u) will be returned or not.

return.u.each
: If TRUE, the function also computes each BLUP corresponding to different kernels (when solving multi-kernel mixed-effects model). It takes additional time compared to the one with ‘return.u.each = FALSE‘ when using packages other than ‘RAINBOWR‘.

return.Hinv
: If TRUE, $H^{-1} = (Var[y]/\sum_{l=1}^{L}\sigma_l^2)^{-1}$ will be computed. It also returns $V^{-1} = (Var[y])^{-1}$. It will take some time in addition to solving the mixed-effects model when using packages other than ‘RAINBOWR‘.

recheck.RAINBOWR
: When you use the package other than ‘RAINBOWR‘ and the ratio of variance components is out of the range of ‘var.ratio.range‘, the function will solve the mixed-effects model again with ‘RAINBOWR‘ package, if ‘recheck.RAINBOWR = TRUE‘.

var.ratio.range
: The range of variance components to check that the results by the package other than RAINBOWR is correct or not when ‘recheck.RAINBOWR = TRUE‘.

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(Sum of Zu)

$u.each BLUP(Each 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.

Johnson, D. L., & Thompson, R. (1995). Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information. Journal of dairy science, 78(2), 449-456.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. The American Statistician, 58(1), 30-37.

Zhou, H., Hu, L., Zhou, J., & Lange, K. (2015). MM algorithms for variance components models. arXiv preprint arXiv:1509.07426.

Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995), Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models, Biometrics, 1440-1450.

See Also

MMEst, lmm.aireml, lmm.diago

Examples

```
### Import RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM) & epistatic relationship matrix
K.A <- calcGRM(genoMat = x)
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 using gaston package (2 random efects)
EM3.gaston.res <- EM3.general(y = pheno.mat, X0 = NULL, ZETA = ZETA,
                              package = "gaston", return.u.always = TRUE,
                              pred = TRUE, return.u.each = TRUE,
                              return.Hinv = TRUE)
(Vu <- EM3.gaston.res$Vu)   ### estimated genetic variance
(Ve <- EM3.gaston.res$Ve)   ### estimated residual variance
(weights <- EM3.gaston.res$weights)   ### estimated proportion of two genetic variances
(herit <- Vu * weights / (Vu + Ve))  ### genomic heritability (additive, additive x additive)

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


### Perform genomic prediction with 10-fold cross validation using gaston package (multi-kernel)
noNA <- !is.na(c(pheno.mat))   ### NA (missing) in the phenotype data

phenoNoNA <- pheno.mat[noNA, , drop = FALSE]   ### remove NA
ZETANoNA <- ZETA
ZETANoNA <- lapply(X = ZETANoNA, FUN = function (List) {
  List$Z <- List$Z[noNA, ]

  return(List)
})   ### remove NA


nFold <- 10    ### # of folds
nLine <- nrow(phenoNoNA)
idCV <- sample(1:nLine %% nFold)   ### assign random ids for cross-validation
idCV[idCV == 0] <- nFold
```

```
yPred <- rep(NA, nLine)

for (noCV in 1:nFold) {
  print(paste0("Fold: ", noCV))
  yTrain <- phenoNoNA
  yTrain[idCV == noCV, ] <- NA   ### prepare test data

  EM3.gaston.resCV <- EM3.general(y = yTrain, X0 = NULL, ZETA = ZETANoNA,
                                  package = "gaston", return.u.always = TRUE,
                                  pred = TRUE, return.u.each = TRUE,
                                  return.Hinv = TRUE)   ### prediction
  yTest <-  EM3.gaston.resCV$y.pred     ### predicted values

  yPred[idCV == noCV] <- yTest[idCV == noCV]
}

### Plot the results
plotRange <- range(phenoNoNA, yPred)
plot(x = phenoNoNA, y = yPred,xlim = plotRange, ylim = plotRange,
     xlab = "Observed values", ylab = "Predicted values",
     main = "Results of Genomic Prediction (multi-kernel)",
     cex.lab = 1.5, cex.main = 1.5, cex.axis = 1.3)
abline(a = 0, b = 1, col = 2, lwd = 2, lty = 2)
R2 <- cor(x = phenoNoNA[, 1], y = yPred) ^ 2
text(x = plotRange[2] - 10,
     y = plotRange[1] + 10,
     paste0("R2 = ", round(R2, 3)),
     cex = 1.5)
```

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,
```

```
  n.core = 1,
  tol = NULL,
  bounds = c(1e-06, 1e+06),
  optimizer = "nlminb",
  traceInside = 0,
  n.thres = 450,
  spectral.method = NULL,
  REML = TRUE,
  pred = TRUE,
  return.u.always = TRUE,
  return.u.each = TRUE,
  return.Hinv = TRUE
)
```

Arguments

- `y0` A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
- `X0` A $n \times p$ 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

  $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 RAINBOWR. 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 RAINBOWR. 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.
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.
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.
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.
return.u.always	If TRUE, BLUP ('u'; u) will be returned.
return.u.each	If TRUE, the function also computes each BLUP corresponding to different kernels (when solving multi-kernel mixed-effects model). It takes additional time compared to the one with 'return.u.each = FALSE'.
return.Hinv	If TRUE, $H^{-1} = (Var[y]/\sum_{l=1}^{L}\sigma_l^2)^{-1}$ will be computed. It also returns $V^{-1} = (Var[y])^{-1}$.

Value

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

$Vu Estimator for σ_u^2, all of the genetic variance

$Ve Estimator for σ_e^2

$beta BLUE($\beta$)

$u BLUP(Sum of Zu)

$u.each BLUP(Each 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.

Examples

```
### Import RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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))


### Including the additional linear kernel for chromosome 12
chrNo <- 12
W.A <- x[, map$chr == chrNo]    ### marker genotype data of chromosome 12

Zs0 <- list(A.part = Z)
Ws0 <- list(A.part = W.A)       ### This will be regarded as linear kernel
### for the variance-covariance matrix of another random effects.


### Solve multi-kernel linear mixed effects model (2 random efects)
EM3.linker.res <- EM3.linker.cpp(y0 = pheno.mat, X0 = NULL, ZETA = ZETA,
                                 Zs0 = Zs0, Ws0 = Ws0)
(Vu <- EM3.linker.res$Vu)   ### estimated genetic variance
(Ve <- EM3.linker.res$Ve)   ### estimated residual variance
(weights <- EM3.linker.res$weights)   ### estimated proportion of two genetic variances
(herit <- Vu * weights / (Vu + Ve))  ### genomic heritability (all chromosomes, chromosome 12)
```

```
(beta <- EM3.linker.res$beta)   ### Here, this is an intercept.
u.each <- EM3.linker.res$u.each  ### estimated genotypic values (all chromosomes, chromosome 12)
See(u.each)
```

EM3.op *Equation of mixed model for multi-kernel using other packages (much faster than EM3.cpp)*

Description

This function solves the following multi-kernel linear mixed effects model using `MMEst` function in 'MM4LMM' package, `lmm.aireml` or `lmm.diago` functions in 'gaston' package, or `EM3.cpp` function in 'RAINBOWR' package.

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

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

Usage

```
EM3.op(
  y,
  X0 = NULL,
  ZETA,
  eigen.G = NULL,
  package = "gaston",
  tol = NULL,
  n.core = 1,
  REML = TRUE,
  pred = TRUE,
  return.u.always = TRUE,
  return.u.each = TRUE,
  return.Hinv = TRUE
)
```

Arguments

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

`X0` A $n \times p$ 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 RAINBOWR. 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.

`package` Package name to be used in this function. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is 'gaston'.

`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.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores (only for 'MM4LMM').

`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.

`return.u.always` When using the "gaston" package with missing values or using the "MM4LMM" package (with/without missings), computing BLUP will take some time in addition to solving the mixed-effects model. You can choose whether BLUP ('u'; u) will be returned or not.

`return.u.each` If TRUE, the function also computes each BLUP corresponding to different kernels (when solving multi-kernel mixed-effects model). It takes additional time compared to the one with 'return.u.each = FALSE'.

`return.Hinv` If TRUE, $H^{-1} = (Var[y]/\sum_{l=1}^{L}\sigma_l^2)^{-1}$ will be computed. It also returns $V^{-1} = (Var[y])^{-1}$. It will take some time in addition to solving the mixed-effects model.

Value

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

$Vu Estimator for σ_u^2, all of the genetic variance

$Ve Estimator for σ_e^2

$beta BLUE($\beta$)

$u BLUP(Sum of Zu)

$u.each BLUP(Each 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.

Johnson, D. L., & Thompson, R. (1995). Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information. Journal of dairy science, 78(2), 449-456.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. The American Statistician, 58(1), 30-37.

Zhou, H., Hu, L., Zhou, J., & Lange, K. (2015). MM algorithms for variance components models. arXiv preprint arXiv:1509.07426.

Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995), Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models, Biometrics, 1440-1450.

See Also

`MMEst`, `lmm.aireml`, `lmm.diago`

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

Description

This function estimates maximum-likelihood (ML/REML; resticted maximum likelihood) 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 $Var[u] = K\sigma^2_u$. The residual variance is $Var[\epsilon] = I\sigma^2_e$.

Usage

```
EMM.cpp(
  y,
  X = NULL,
  ZETA,
  eigen.G = NULL,
  eigen.SGS = NULL,
  n.thres = 450,
  reestimation = FALSE,
  n.core = NA,
  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` A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

`X` A $n \times p$ 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 RAINBOWR. 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 RAINBOWR. 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.

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

`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 $\lambda = \sigma^2_e / \sigma^2_u$. 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 $\lambda = \sigma^2_e / \sigma^2_u$ (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

```
### Perform genomic prediction with 10-fold cross validation

  ### Import RAINBOWR
```

```
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)

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

### Estimate marker effects from estimated genotypic values
x.modi <- modify.res$geno.modi
WMat <- calcGRM(genoMat = x.modi, methodGRM = "addNOIA",
                returnWMat = TRUE)
K.A <- ZETA$A$K
if (min(eigen(K.A)$values) < 1e-08) {
  diag(K.A) <- diag(K.A) + 1e-06
}

mrkEffectsForW <- crossprod(x = WMat,
                            y = solve(K.A)) %*% as.matrix(u)
mrkEffects <- mrkEffectsForW / mean(scale(x.modi %*% mrkEffectsForW, scale = F) / u)
```

```
#### Cross-validation for genomic prediction
noNA <- !is.na(c(pheno.mat))   ### NA (missing) in the phenotype data

phenoNoNA <- pheno.mat[noNA, , drop = FALSE]   ### remove NA
ZETANoNA <- ZETA
ZETANoNA$A$Z <- ZETA$A$Z[noNA, ]   ### remove NA


nFold <- 10    ### # of folds
nLine <- nrow(phenoNoNA)
idCV <- sample(1:nLine %% nFold)   ### assign random ids for cross-validation
idCV[idCV == 0] <- nFold

yPred <- rep(NA, nLine)

for (noCV in 1:nFold) {
  yTrain <- phenoNoNA
  yTrain[idCV == noCV, ] <- NA   ### prepare test data

  EMM.resCV <- EMM.cpp(y = yTrain, X = NULL, ZETA = ZETANoNA)   ### prediction
  yTest <-  EMM.resCV$beta + EMM.resCV$u   ### predicted values

  yPred[idCV == noCV] <- (yTest[noNA])[idCV == noCV]
}

### Plot the results
plotRange <- range(phenoNoNA, yPred)
plot(x = phenoNoNA, y = yPred,xlim = plotRange, ylim = plotRange,
     xlab = "Observed values", ylab = "Predicted values",
     main = "Results of Genomic Prediction",
     cex.lab = 1.5, cex.main = 1.5, cex.axis = 1.3)
abline(a = 0, b = 1, col = 2, lwd = 2, lty = 2)
R2 <- cor(x = phenoNoNA[, 1], y = yPred) ^ 2
text(x = plotRange[2] - 10,
     y = plotRange[1] + 10,
     paste0("R2 = ", round(R2, 3)),
     cex = 1.5)
```

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

`X` A $n \times p$ 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 RAINBOWR. 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.

`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 $\lambda = \sigma_e^2/\sigma_u^2$. 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 $\lambda = \sigma_e^2/\sigma_u^2$

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

`X` A $n \times p$ 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 RAINBOWR. 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 RAINBOWR. 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

$beta BLUE($\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)

References

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

---

estNetwork — *Function to estimate & plot haplotype network*

Description

Function to estimate & plot haplotype network

Usage

```
estNetwork(
  blockInterest = NULL,
  gwasRes = NULL,
  nTopRes = 1,
  gene.set = NULL,
  indexRegion = 1:10,
```

```
  chrInterest = NULL,
  posRegion = NULL,
  blockName = NULL,
  pheno = NULL,
  geno = NULL,
  ZETA = NULL,
  chi2Test = TRUE,
  thresChi2Test = 0.05,
  plotNetwork = TRUE,
  distMat = NULL,
  distMethod = "manhattan",
  evolutionDist = FALSE,
  complementHaplo = "phylo",
  subpopInfo = NULL,
  groupingMethod = "kmedoids",
  nGrp = 3,
  nIterClustering = 100,
  iterRmst = 100,
  networkMethod = "rmst",
  autogamous = FALSE,
  probParsimony = 0.95,
  nMaxHaplo = 1000,
  kernelTypes = "addNOIA",
  nCores = parallel::detectCores() - 1,
  hOpt = "optimized",
  hOpt2 = "optimized",
  maxIter = 20,
  rangeHStart = 10^c(-1:1),
  saveName = NULL,
  saveStyle = "png",
  plotWhichMDS = 1:2,
  colConnection = c("grey40", "grey60"),
  ltyConnection = c("solid", "dashed"),
  lwdConnection = c(1.5, 0.8),
  pchBase = c(1, 16),
  colCompBase = c(2, 4),
  colHaploBase = c(3, 5, 6),
  cexMax = 2,
  cexMin = 0.7,
  ggPlotNetwork = FALSE,
  cexMaxForGG = 0.025,
  cexMinForGG = 0.008,
  alphaBase = c(0.9, 0.3),
  verbose = TRUE
)
```

Arguments

`blockInterest` A $n \times M$ matrix representing the marker genotype that belongs to the haplotype block of interest. If this argument is NULL, this argument will automatically be determined by ‘geno‘,

`gwasRes` You can use the results (data.frame) of haplotype-based (SNP-set) GWAS by

‘RGWAS.multisnp‘ function.

nTopRes — Haplotype blocks (or gene sets, SNP-sets) with top ‘nTopRes‘ p-values by ‘gwasRes‘ will be used.

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.

indexRegion — You can specify the haplotype block (or gene set, SNP-set) of interest by the marker index in ‘geno‘.

chrInterest — You can specify the haplotype block (or gene set, SNP-set) of interest by the marker position in ‘geno‘. Please assign the chromosome number to this argument.

posRegion — You can specify the haplotype block (or gene set, SNP-set) of interest by the marker position in ‘geno‘. Please assign the position in the chromosome to this argument.

blockName — You can specify the haplotype block (or gene set, SNP-set) of interest by the name of haplotype block in ‘geno‘.

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.

ZETA — A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

chi2Test — If TRUE, chi-square test for the relationship between haplotypes & subpopulations will be performed.

thresChi2Test — The threshold for the chi-square test.

plotNetwork — If TRUE, the function will return the plot of haplotype network.

distMat — You can assign the distance matrix of the block of interest. If NULL, the distance matrix will be computed in this function.

distMethod — You can choose the method to calculate distance between accessions. This argument corresponds to the ‘method‘ argument in the ‘dist‘ function.

evolutionDist — If TRUE, the evolution distance will be used instead of the pure distance. The ‘distMat‘ will be converted to the distance matrix by the evolution distance when you use ‘complementHaplo = "phylo"‘.

complementHaplo — how to complement unobserved haplotypes. When ‘complementHaplo = "all"‘, all possible haplotypes will be complemented from the observed haplotypes.

When ‘complementHaplo = "never"‘, unobserved haplotypes will not be complemented. When ‘complementHaplo = "phylo"‘, unobserved haplotypes will be complemented as nodes of phylogenetic tree. When ‘complementHaplo = "TCS"‘, unobserved haplotypes will be complemented by TCS methods (Clement et al., 2002).

`subpopInfo` The information of subpopulations. This argument should be a vector of factor.

`groupingMethod` If ‘subpopInfo‘ argument is NULL, this function estimates subpopulation information from marker genotype. You can choose the grouping method from ‘kmeans‘, ‘kmedoids‘, and ‘hclust‘.

`nGrp` The number of groups (or subpopulations) grouped by ‘groupingMethod‘. If this argument is 0, the subpopulation information will not be estimated.

`nIterClustering`

If ‘groupingMethod‘ = ‘kmeans‘, the clustering will be performed multiple times. This argument specifies the number of classification performed by the function.

`iterRmst` The number of iterations for RMST (randomized minimum spanning tree).

`networkMethod` Either one of ’mst’ (minimum spanning tree), ’msn’ (minimum spanning network), and ’rmst’ (randomized minimum spanning tree). ’rmst’ is recommended.

`autogamous` This argument will be valid only when you use ‘complementHaplo = "all"‘ or ‘complementHaplo = "TCS"‘. This argument specifies whether the plant is autogamous or not. If autogamous = TRUE, complemented haplotype will consist of only homozygous sites (-1, 1). If FALSE, complemented haplotype will consist of both homozygous & heterozygous sites (-1, 0, 1).

`probParsimony` Equal to the argument ‘prob‘ in ‘haplotypes::parsimnet‘ function:

A numeric vector of length one in the range [0.01, 0.99] giving the probability of parsimony as defined in Templeton et al. (1992). In order to set maximum connection steps to Inf (to connect all the haplotypes in a single network), set the probability to NULL.

`nMaxHaplo` The maximum number of haplotypes. If the number of total (complemented + original) haplotypes are larger than ‘nMaxHaplo‘, we will only show the results only for the original haplotypes to reduce the computational time.

`kernelTypes` In the function, similarlity matrix between accessions will be computed from marker genotype to estimate genotypic values. This argument specifies the method to compute similarity matrix: If this argument is ‘addNOIA‘ (or one of other options in ‘methodGRM‘ in ‘calcGRM‘), then the ‘addNOIA‘ (or corresponding) option in the ‘calcGRM‘ function will be used, and if this argument is ‘diffusion‘, the diffusion kernel based on Laplacian matrix will be computed from network. You can assign more than one kernelTypes for this argument; for example, kernelTypes = c("addNOIA", "diffusion").

`nCores` The number of cores used for optimization.

`hOpt` Optimized hyper parameter for constructing kernel when estimating haplotype effects. If hOpt = "optimized", hyper parameter will be optimized in the function. If hOpt is numeric, that value will be directly used in the function.

`hOpt2` Optimized hyper parameter for constructing kernel when estimating complemented haplotype effects. If hOpt2 = "optimized", hyper parameter will be optimized in the function. If hOpt2 is numeric, that value will be directly used in the function.

`maxIter` Max number of iterations for optimization algorithm.

`rangeHStart` The median of off-diagonal of distance matrix multiplied by rangeHStart will be used as the initial values for optimization of hyper parameters.

`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.
`saveStyle`	This argument specifies how to save the plot of phylogenetic tree. The function offers 'png', 'pdf', 'jpg', and 'tiff'.
`plotWhichMDS`	We will show the MDS (multi-dimensional scaling) plot, and this argument is a vector of two integers specifying that will define which MDS dimension will be plotted. The first and second integers correspond to the horizontal and vertical axes, respectively.
`colConnection`	A vector of two integers or characters specifying the colors of connection between nodes for the original and complemented haplotypes, respectively.
`ltyConnection`	A vector of two characters specifying the line types of connection between nodes for the original and complemented haplotypes, respectively.
`lwdConnection`	A vector of two integers specifying the line widths of connection between nodes for the original and complemented haplotypes, respectively.
`pchBase`	A vector of two integers specifying the plot types for the positive and negative genotypic values respectively.
`colCompBase`	A vector of two integers or characters specifying color of complemented haplotypes for the positive and negative genotypic values respectively.
`colHaploBase`	A vector of integers or characters specifying color of original haplotypes for the positive and negative genotypic values respectively. The length of the vector should equal to the number of subpopulations.
`cexMax`	A numeric specifying the maximum point size of the plot.
`cexMin`	A numeric specifying the minimum point size of the plot.
`ggPlotNetwork`	If TRUE, the function will return the ggplot version of haplotype network. It offers the precise information on subgroups for each haplotype.
`cexMaxForGG`	A numeric specifying the maximum point size of the plot for the ggplot version of haplotype network, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
`cexMinForGG`	A numeric specifying the minimum point size of the plot for the ggplot version of haplotype network, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
`alphaBase`	alpha (parameter that indicates the opacity of a geom) for original haplotype with positive / negative effects. alpha for complemented haplotype will be same as the alpha for original haplotype with negative effects.
`verbose`	If this argument is TRUE, messages for the current steps will be shown.

Value

A list / lists of

- **$haplotypeInfo** A list of haplotype information with
  - **$haploCluster** A vector indicating each individual belongs to which haplotypes.
  - **$haploBlock** Marker genotype of haplotype block of interest for the representing haplotypes.
- **$subpopInfo** The information of subpopulations.
- **$pValChi2Test** A p-value of the chi-square test for the dependency between haplotypes & subpopulations. If 'chi2Test = FALSE', 'NA' will be returned.

$mstResults A list of estimated results of MST / MSN / RMST:

Estimated results of MST / MSN / RMST for the data including original haplotypes.

$mstRes$mstResComp Estimated results of MST / MSN / RMST for the data including both original and complemented haplotype.

$distMats A list of distance matrix:

Distance matrix between haplotypes.

$distMat$distMatComp Distance matrix between haplotypes (including unobserved ones).

$laplacianMat Laplacian matrix between haplotypes (including unobserved ones).

$gvTotal Estimated genotypic values by kernel regression for each haplotype.

$gvTotalForLine Estimated genotypic values by kernel regression for each individual.

$minuslog10p $-log_{10}(p)$ for haplotype block of interest. p is the p-value for the siginifacance of the haplotype block effect.

$hOpts Optimized hyper parameters, hOpt1 & hOpt2.

$EMMResults A list of estimated results of kernel regression:

Estimated results of kernel regression for the estimation of haplotype effects. (1st step)

$EM3Res$EMMRes Estimated results of kernel regression for the estimation of haplotype effects of nodes. (2nd step)

$EMM0Res Estimated results of kernel regression for the null model.

$clusterNosForHaplotype A list of cluster Nos of individuals that belong to each haplotype.

`estPhylo` *Function to estimate & plot phylogenetic tree*

Description

Function to estimate & plot phylogenetic tree

Usage

```
estPhylo(
  blockInterest = NULL,
  gwasRes = NULL,
  nTopRes = 1,
  gene.set = NULL,
  indexRegion = 1:10,
  chrInterest = NULL,
  posRegion = NULL,
  blockName = NULL,
  pheno = NULL,
  geno = NULL,
  ZETA = NULL,
  chi2Test = TRUE,
  thresChi2Test = 0.05,
  plotTree = TRUE,
  distMat = NULL,
  distMethod = "manhattan",
  evolutionDist = FALSE,
```

```
  subpopInfo = NULL,
  groupingMethod = "kmedoids",
  nGrp = 3,
  nIterClustering = 100,
  kernelTypes = "addNOIA",
  nCores = parallel::detectCores() - 1,
  hOpt = "optimized",
  hOpt2 = "optimized",
  maxIter = 20,
  rangeHStart = 10^c(-1:1),
  saveName = NULL,
  saveStyle = "png",
  pchBase = c(1, 16),
  colNodeBase = c(2, 4),
  colTipBase = c(3, 5, 6),
  cexMax = 2,
  cexMin = 0.7,
  edgeColoring = TRUE,
  tipLabel = TRUE,
  ggPlotTree = FALSE,
  cexMaxForGG = 0.12,
  cexMinForGG = 0.06,
  alphaBase = c(0.9, 0.3),
  verbose = TRUE
)
```

Arguments

`blockInterest`	A $n \times M$ matrix representing the marker genotype that belongs to the haplotype block of interest. If this argument is NULL, this argument will automatically be determined by ‘geno‘,
`gwasRes`	You can use the results (data.frame) of haplotype-based (SNP-set) GWAS by ‘RGWAS.multisnp‘ function.
`nTopRes`	Haplotype blocks (or gene sets, SNP-sets) with top ‘nTopRes‘ p-values by ‘gwasRes‘ will be used.
`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.
`indexRegion`	You can specify the haplotype block (or gene set, SNP-set) of interest by the marker index in ‘geno‘.
`chrInterest`	You can specify the haplotype block (or gene set, SNP-set) of interest by the marker position in ‘geno‘. Please assign the chromosome number to this argument.
`posRegion`	You can specify the haplotype block (or gene set, SNP-set) of interest by the marker position in ‘geno‘. Please assign the position in the chromosome to this argument.
`blockName`	You can specify the haplotype block (or gene set, SNP-set) of interest by the name of haplotype block in ‘geno‘.

`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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`chi2Test` If TRUE, chi-square test for the relationship between haplotypes & subpopulations will be performed.

`thresChi2Test` The threshold for the chi-square test.

`plotTree` If TRUE, the function will return the plot of phylogenetic tree.

`distMat` You can assign the distance matrix of the block of interest. If NULL, the distance matrix will be computed in this function.

`distMethod` You can choose the method to calculate distance between accessions. This argument corresponds to the ‘method‘ argument in the ‘dist‘ function.

`evolutionDist` If TRUE, the evolution distance will be used instead of the pure distance. The ‘distMat‘ will be converted to the distance matrix by the evolution distance.

`subpopInfo` The information of subpopulations. This argument should be a vector of factor.

`groupingMethod` If ‘subpopInfo‘ argument is NULL, this function estimates subpopulation information from marker genotype. You can choose the grouping method from ‘kmeans‘, ‘kmedoids‘, and ‘hclust‘.

`nGrp` The number of groups (or subpopulations) grouped by ‘groupingMethod‘. If this argument is 0, the subpopulation information will not be estimated.

`nIterClustering`

If ‘groupingMethod‘ = ‘kmeans‘, the clustering will be performed multiple times. This argument specifies the number of classification performed by the function.

`kernelTypes` In the function, similarlity matrix between accessions will be computed from marker genotype to estimate genotypic values. This argument specifies the method to compute similarity matrix: If this argument is ‘addNOIA‘ (or one of other options in ‘methodGRM‘ in ‘calcGRM‘), then the ‘addNOIA‘ (or corresponding) option in the ‘calcGRM‘ function will be used, and if this argument is ‘phylo‘, the gaussian kernel based on phylogenetic distance will be computed from phylogenetic tree. You can assign more than one kernelTypes for this argument; for example, kernelTypes = c("addNOIA", "phylo").

`nCores` The number of cores used for optimization.

`hOpt` Optimized hyper parameter for constructing kernel when estimating haplotype effects. If hOpt = "optimized", hyper parameter will be optimized in the function. If hOpt = "tuned", hyper parameter will be replaced by the median of off-diagonal of distance matrix. If hOpt is numeric, that value will be directly used in the function.

`hOpt2`	Optimized hyper parameter for constructing kernel when estimating haplotype effects of nodes. If hOpt2 = "optimized", hyper parameter will be optimized in the function. If hOpt2 = "tuned", hyper parameter will be replaced by the median of off-diagonal of distance matrix. If hOpt2 is numeric, that value will be directly used in the function.
`maxIter`	Max number of iterations for optimization algorithm.
`rangeHStart`	The median of off-diagonal of distance matrix multiplied by rangeHStart will be used as the initial values for optimization of hyper parameters.
`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.
`saveStyle`	This argument specifies how to save the plot of phylogenetic tree. The function offers 'png', 'pdf', 'jpg', and 'tiff'.
`pchBase`	A vector of two integers specifying the plot types for the positive and negative genotypic values respectively.
`colNodeBase`	A vector of two integers or chracters specifying color of nodes for the positive and negative genotypic values respectively.
`colTipBase`	A vector of integers or chracters specifying color of tips for the positive and negative genotypic values respectively. The length of the vector should equal to the number of subpopulations.
`cexMax`	A numeric specifying the maximum point size of the plot.
`cexMin`	A numeric specifying the minimum point size of the plot.
`edgeColoring`	If TRUE, the edge branch of phylogenetic tree wiil be colored.
`tipLabel`	If TRUE, lavels for tips will be shown.
`ggPlotTree`	If TRUE, the function will return the ggplot version of phylogenetic tree. It offers the precise information on subgroups for each haplotype.
`cexMaxForGG`	A numeric specifying the maximum point size of the plot for ggtree, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
`cexMinForGG`	A numeric specifying the minimum point size of the plot for ggtree, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
`alphaBase`	alpha (parameter that indicates the opacity of a geom) for tip with positive / negative effects. alpha for node will be same as the alpha for tip with negative effects.
`verbose`	If this argument is TRUE, messages for the current step_s will be shown.

Value

A list / lists of

$haplotypeInfo A list of haplotype information with

- **$haploCluster** A vector indicating each individual belongs to which haplotypes.
- **$haploBlock** Marker genotype of haplotype block of interest for the representing haplotypes.

$subpopInfo The information of subpopulations.

$distMats A list of distance matrix:

- **$distMat** Distance matrix between haplotypes.
- **$distMatEvol** Evolutionary distance matrix between haplotypes.
- **$distMatNJ** Phylogenetic distance matrix between haplotypes including nodes.

$pValChi2Test A p-value of the chi-square test for the dependency between haplotypes & subpopulations. If 'chi2Test = FALSE', 'NA' will be returned.

$njRes The result of phylogenetic tree by neighborhood-joining method

$gvTotal Estimated genotypic values by kernel regression for each haplotype.

$gvTotalForLine Estimated genotypic values by kernel regression for each individual.

$minuslog10p $-log_{10}(p)$ for haplotype block of interest. p is the p-value for the siginifacance of the haplotype block effect.

$hOpts Optimized hyper parameters, hOpt1 & hOpt2.

$EMMResults A list of estimated results of kernel regression:

Estimated results of kernel regression for the estimation of haplotype effects. (1st step)

$EM3RES$EMMRes Estimated results of kernel regression for the estimation of haplotype effects of nodes. (2nd step)

$EMM0Res Estimated results of kernel regression for the null model.

$clusterNosForHaplotype A list of cluster Nos of individuals that belong to each haplotype.

`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	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
sample.sets	A 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	A 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

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

`num.qtn` The number of QTNs

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

`qtn.effect` Additive of dominance for each marker effect. This argument should be the same length as num.qtn.

`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.

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

`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.

Value

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 Genomic heritability for generated phenotypic values.

is.diag — *Function to judge the square matrix whether it is diagonal matrix or not*

Description

Function to judge the square matrix whether it is diagonal matrix or not

Usage

```
is.diag(x)
```

Arguments

`x`	Square matrix.

Value

If ‘x‘ is diagonal matrix, ‘TRUE‘. Otherwise the function returns ‘FALSE‘.

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`	A $n \times m$ 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	A $n \times p$ 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 = 1,
  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`	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.
`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,
  cex = 1,
  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.
`cex`	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.
`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",
  cex = 1,
  plot.col2 = 1,
  plot.type = "p",
  plot.pch = 16,
  cum.pos = NULL,
  lwd.thres = 1,
  cex.lab = 1,
  cex.axis = 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.
`cex`	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.
`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`	The font size of the axes.

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

Arguments

`input`	List of results of RGWAS.epistasis / RGWAS.twostep.epi. If the output of 'RGWAS.epistasis' is 'res', 'input' corresponds to 'res$scores'. If the output of 'RGWAS.twostep.epi.' is 'res', 'input' corresponds to 'res$epistasis$scores'. See: Value of RGWAS.epistasis
`map`	Data frame with the marker names in the first column. The second and third columns contain the chromosome and map position. This is map information for SNPs which are tested epistatic effects.
`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.

`parallel.compute` *Function to parallelize computation with various methods*

Description

Function to parallelize computation with various methods

Usage

```
parallel.compute(
  vec,
  func,
  n.core = 2,
  parallel.method = "mclapply",
  count = TRUE
)
```

Arguments

vec	Numeric vector including the values that are computed in parallel.
func	The function to be applied to each element of ‘vec‘ argument. This function must only have one argument.
n.core	Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when ‘parallel.method = "furrr"‘.
parallel.method	Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach". When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize future_map function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize progressor function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize foreach function in the ‘foreach‘ package with the utilization of makeCluster function in ‘parallel‘ package, and registerDoParallel function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize setTxtProgressBar and txtProgressBar functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

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

Value

List of the results for each element of ‘vec‘ argument.

`plotHaploNetwork` *Function to plot haplotype network from the estimated results*

Description

Function to plot haplotype network from the estimated results

Usage

```
plotHaploNetwork(
  estNetworkRes,
  traitName = NULL,
  blockName = NULL,
  plotNetwork = TRUE,
  subpopInfo = estNetworkRes$subpopInfo,
  saveName = NULL,
  saveStyle = "png",
  plotWhichMDS = 1:2,
  colConnection = c("grey40", "grey60"),
  ltyConnection = c("solid", "dashed"),
  lwdConnection = c(1.5, 0.8),
  pchBase = c(1, 16),
  colCompBase = c(2, 4),
  colHaploBase = c(3, 5, 6),
  cexMax = 2,
  cexMin = 0.7,
  ggPlotNetwork = FALSE,
  cexMaxForGG = 0.025,
```

```
  cexMinForGG = 0.008,
  alphaBase = c(0.9, 0.3)
)
```

Arguments

- `estNetworkRes` The estimated results of haplotype network by 'estNetwork' function for one
- `traitName` Name of trait of interest. This will be used in the title of the plots.
- `blockName` You can specify the haplotype block (or gene set, SNP-set) of interest by the name of haplotype block in 'geno'. This will be used in the title of the plots.
- `plotNetwork` If TRUE, the function will return the plot of haplotype network.
- `subpopInfo` The information of subpopulations.
- `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.
- `saveStyle` This argument specifies how to save the plot of phylogenetic tree. The function offers 'png', 'pdf', 'jpg', and 'tiff'.
- `plotWhichMDS` We will show the MDS (multi-dimensional scaling) plot, and this argument is a vector of two integers specifying that will define which MDS dimension will be plotted. The first and second integers correspond to the horizontal and vertical axes, respectively.
- `colConnection` A vector of two integers or characters specifying the colors of connection between nodes for the original and complemented haplotypes, respectively.
- `ltyConnection` A vector of two characters specifying the line types of connection between nodes for the original and complemented haplotypes, respectively.
- `lwdConnection` A vector of two integers specifying the line widths of connection between nodes for the original and complemented haplotypes, respectively.
- `pchBase` A vector of two integers specifying the plot types for the positive and negative genotypic values respectively.
- `colCompBase` A vector of two integers or characters specifying color of complemented haplotypes for the positive and negative genotypic values respectively.
- `colHaploBase` A vector of integers or characters specifying color of original haplotypes for the positive and negative genotypic values respectively. The length of the vector should equal to the number of subpopulations.
- `cexMax` A numeric specifying the maximum point size of the plot.
- `cexMin` A numeric specifying the minimum point size of the plot.
- `ggPlotNetwork` If TRUE, the function will return the ggplot version of haplotype network. It offers the precise information on subgroups for each haplotype.
- `cexMaxForGG` A numeric specifying the maximum point size of the plot for the ggplot version of haplotype network, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
- `cexMinForGG` A numeric specifying the minimum point size of the plot for the ggplot version of haplotype network, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).
- `alphaBase` alpha (parameter that indicates the opacity of a geom) for original haplotype with positive / negative effects. alpha for complemented haplotype will be same as the alpha for original haplotype with negative effects.

Value

Draw plot of haplotype network.

`plotPhyloTree` *Function to plot phylogenetic tree from the estimated results*

Description

Function to plot phylogenetic tree from the estimated results

Usage

```
plotPhyloTree(
  estPhyloRes,
  traitName = NULL,
  blockName = NULL,
  plotTree = TRUE,
  subpopInfo = estPhyloRes$subpopInfo,
  saveName = NULL,
  saveStyle = "png",
  pchBase = c(1, 16),
  colNodeBase = c(2, 4),
  colTipBase = c(3, 5, 6),
  cexMax = 2,
  cexMin = 0.7,
  edgeColoring = TRUE,
  tipLabel = TRUE,
  ggPlotTree = FALSE,
  cexMaxForGG = 0.12,
  cexMinForGG = 0.06,
  alphaBase = c(0.9, 0.3)
)
```

Arguments

`estPhyloRes`	The estimated results of phylogenetic analysis by 'estPhylo' function for one
`traitName`	Name of trait of interest. This will be used in the title of the plots.
`blockName`	You can specify the haplotype block (or gene set, SNP-set) of interest by the name of haplotype block in 'geno'. This will be used in the title of the plots.
`plotTree`	If TRUE, the function will return the plot of phylogenetic tree.
`subpopInfo`	The information of subpopulations.
`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.
`saveStyle`	This argument specifies how to save the plot of phylogenetic tree. The function offers 'png', 'pdf', 'jpg', and 'tiff'.
`pchBase`	A vector of two integers specifying the plot types for the positive and negative genotypic values respectively.

colNodeBase
: A vector of two integers or chracters specifying color of nodes for the positive and negative genotypic values respectively.

colTipBase
: A vector of integers or chracters specifying color of tips for the positive and negative genotypic values respectively. The length of the vector should equal to the number of subpopulations.

cexMax
: A numeric specifying the maximum point size of the plot.

cexMin
: A numeric specifying the minimum point size of the plot.

edgeColoring
: If TRUE, the edge branch of phylogenetic tree wiil be colored.

tipLabel
: If TRUE, lavels for tips will be shown.

ggPlotTree
: If TRUE, the function will return the ggplot version of phylogenetic tree. It offers the precise information on subgroups for each haplotype.

cexMaxForGG
: A numeric specifying the maximum point size of the plot for ggtree, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).

cexMinForGG
: A numeric specifying the minimum point size of the plot for ggtree, relative to the range of x and y-axes (0 < cexMaxForGG <= 1).

alphaBase
: alpha (parameter that indicates the opacity of a geom) for tip with positive / negative effects. alpha for node will be same as the alpha for tip with negative effects.

Value

Draw plots of phylogenetic tree.

`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

RAINBOWR — *RAINBOWR: Perform Genome-Wide Asscoiation Study (GWAS) By Kernel-Based Methods*

Description

By using 'RAINBOWR' (Reliable Association INference By Optimizing Weights with R), users can test multiple SNPs (Single Nucleotide Polymorphisms) simultaneously by kernel-based (SNP-set) methods. Users can test not only additive effects but also dominance and epistatic effects. In detail, please check our preprint on bioRxiv: Kosuke Hamazaki and Hiroyoshi Iwata (2019) <doi:10.1101/612028>.

RGWAS.epistasis — *Check epistatic effects by kernel-based GWAS (genome-wide association studies)*

Description

Check epistatic effects by kernel-based GWAS (genome-wide association studies)

Usage

```
RGWAS.epistasis(
  pheno,
  geno,
  ZETA = NULL,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  n.core = 1,
  parallel.method = "mclapply",
  test.method = "LR",
  dominance.eff = TRUE,
  skip.self.int = FALSE,
  haplotype = TRUE,
  num.hap = NULL,
  window.size.half = 5,
  window.slide = 1,
  chi0.mixture = 0.5,
  optimizer = "nlminb",
  gene.set = NULL,
  map.gene.set = NULL,
  plot.epi.3d = TRUE,
  plot.epi.2d = TRUE,
  main.epi.3d = NULL,
  main.epi.2d = NULL,
```

```
  saveName = NULL,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

ZETA
: A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

  Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

  K.A, K.D Different kernels which express some relationships between lines.

  For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

package.MM
: The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at EM3.general.

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. This argument is not valid when ‘parallel.method = "furrr"‘.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

  When ‘parallel.method = "mclapply"‘, we utilize pbmclapply function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and mclapply function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize future_map function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize progressor function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize foreach function in the ‘foreach‘ package with the utilization of makeCluster function in ‘parallel‘ package, and registerDoParallel function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize setTxtProgressBar and txtProgressBar functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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.

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set ‘skip.self.int = TRUE‘.

`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` RAINBOWR 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.

`map.gene.set`	Genotype map for ‘gene.set‘ (list of haplotype blocks). This is a data.frame with the haplotype block (SNP-set, or gene-set) names in the first column. The second and third columns contain the chromosome and map position for each block. The forth column contains the cumulative map position for each block, which can be computed by `cumsumPos` function. If this argument is NULL, the map will be constructed by `genesetmap` function after the SNP-set GWAS. It will take some time, so you can reduce the computational time by assigning this argument beforehand.
`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.
`skip.check`	As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting ‘skip.check = TRUE‘.
`verbose`	If this argument is TRUE, messages for the current steps will be shown.
`verbose2`	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, $y** The information of the positions of SNPs detected by regular GWAS. These vectors are used when drawing plots. Each output correspond to the replication of row and column of scores.
- **$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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno
Rice_haplo_block <- Rice_Zhao_etal$haploBlock

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

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
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 = "LR", gene.set = NULL,
                                 window.size.half = 5, window.slide = 11,
                                 package.MM = "gaston", parallel.method = "mclapply",
                                 skip.check = TRUE, n.core = 2)

See(epistasis.res$scores$scores)


### Check epistatic effects (by using the list of haplotype blocks estimated by PLINK)
### It will take almost 2 minutes...
epistasis_haplo_block.res <- RGWAS.epistasis(pheno = pheno.GWAS, geno = geno.GWAS,
                                             ZETA = ZETA, n.PC = 4,
                                       test.method = "LR", gene.set = Rice_haplo_block,
                                   package.MM = "gaston", parallel.method = "mclapply",
                                             skip.check = TRUE, n.core = 2)

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

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

Description

Print the R code which you should perform for RAINBOWR (Reliable Association INference By Optimizing Weights with R).

Usage

```
RGWAS.menu()
```

Value

The R code which you should perform for RAINBOWR GWAS

`RGWAS.multisnp` *Testing multiple SNPs simultaneously for GWAS*

Description

This function performs SNP-set GWAS (genome-wide association studies), which tests multiple SNPs (single nucleotide polymorphisms) simultaneously. 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,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  test.method = "LR",
  n.core = 1,
  parallel.method = "mclapply",
  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,
  map.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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`

Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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 NOIA methods for additive GRM.

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
: RAINBOWR 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.

map.gene.set
: Genotype map for ‘gene.set‘ (list of haplotype blocks). This is a data.frame with the haplotype block (SNP-set, or gene-set) names in the first column. The second and third columns contain the chromosome and map position for each block. The forth column contains the cumulative map position for each block, which can be computed by `cumsumPos` function. If this argument is NULL, the map will be constructed by `genesetmap` function after the SNP-set GWAS. It will take some time, so you can reduce the computational time by assigning this argument beforehand.

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 weighting.center = 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.
skip.check	As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting 'skip.check = TRUE'.
verbose	If this argument is TRUE, messages for the current steps will be shown.
verbose2	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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno
Rice_haplo_block <- Rice_Zhao_etal$haploBlock

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

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)
```

```
### 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 RAINBOWR
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Perform SNP-set GWAS (by regarding 21 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 = 10,
                              window.slide = 21, package.MM = "gaston",
                              parallel.method = "mclapply",
                              skip.check = TRUE, n.core = 2)
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 2 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, package.MM = "gaston",
                               parallel.method = "mclapply",
                               skip.check = TRUE, n.core = 2)
See(SNP_set.res2$D)  ### Column 4 contains -log10(p) values for markers


### Perform haplotype-block GWAS (by using the list of haplotype blocks estimated by PLINK)
haplo_block.res <- RGWAS.multisnp(pheno = pheno.GWAS, geno = geno.GWAS,
                                  ZETA = ZETA, n.PC = 4, test.method = "LR",
                                kernel.method = "linear", gene.set = Rice_haplo_block,
                                  test.effect = "additive", package.MM = "gaston",
                                  parallel.method = "mclapply",
                                  skip.check = TRUE, n.core = 2)
See(haplo_block.res$D)  ### Column 4 contains -log10(p) values for markers
```

`RGWAS.multisnp.interaction` *Testing multiple SNPs and their interaction with some kernel simultaneously for GWAS*

Description

This function performs SNP-set GWAS (genome-wide association studies), which tests multiple SNPs (single nucleotide polymorphisms) simultaneously. 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.interaction(
  pheno,
  geno,
  ZETA = NULL,
  interaction.kernel = NULL,
  include.interaction.kernel.null = FALSE,
  include.interaction.with.gb.null = FALSE,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  test.method = "LR",
  n.core = 1,
  parallel.method = "mclapply",
  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,
  map.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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`interaction.kernel` A $n \times n$ Gram (kernel) matrix which may indicate some interaction with SNP-sets to be tested.

`include.interaction.kernel.null` Whether or not including ‘iteraction.kernel‘ itself into the null and alternative models.

`include.interaction.with.gb.null` Whether or not including the interaction term between ‘iteraction.kernel‘ and the genetic background (= kinship matrix) into the null and alternative models. By setting this TRUE, you can avoid the false positives caused by epistastis between polygenes, especially you set kinship matrix as ‘interaction.kernel‘.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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 only one method to test effects of each SNP-set.

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

n.core
: Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when 'parallel.method = "furrr"'.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

  When 'parallel.method = "mclapply"', we utilize `pbmclapply` function in the 'pbmcapply' package with 'count = TRUE' and `mclapply` function in the 'parallel' package with 'count = FALSE'.

  When 'parallel.method = "furrr"', we utilize `future_map` function in the 'furrr' package. With 'count = TRUE', we also utilize `progressor` function in the 'progressr' package to show the progress bar, so please install the 'progressr' package from github (https://github.com/HenrikBengtsson/progressr). For 'parallel.method = "furrr"', you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to 'parallel.method = "mclapply"'.

  When 'parallel.method = "foreach"', we utilize `foreach` function in the 'foreach' package with the utilization of `makeCluster` function in 'parallel' package, and `registerDoParallel` function in 'doParallel' package. With 'count = TRUE', we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the 'utils' package to show the progress bar.

  We recommend that you use the option 'parallel.method = "mclapply"', but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option 'parallel.method = "foreach"'.

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 NOIA methods for additive GRM.

  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
: RAINBOWR 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.

map.gene.set
: Genotype map for ‘gene.set‘ (list of haplotype blocks). This is a data.frame with the haplotype block (SNP-set, or gene-set) names in the first column. The second and third columns contain the chromosome and map position for each block. The forth column contains the cumulative map position for each block, which can be computed by cumsumPos function. If this argument is NULL, the map will be constructed by genesetmap function after the SNP-set GWAS. It will take some time, so you can reduce the computational time by assigning this argument beforehand.

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 weighting.center = 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.
skip.check	As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting 'skip.check = TRUE'.
verbose	If this argument is TRUE, messages for the current steps will be shown.
verbose2	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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno
Rice_haplo_block <- Rice_Zhao_etal$haploBlock

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

```
See(Rice_pheno)
See(Rice_haplo_block)

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)


### Perform SNP-set GWAS with interaction
### by regarding 21 SNPs as one SNP-set
SNP_set.res.int <- RGWAS.multisnp.interaction(
  pheno = pheno.GWAS,
  geno = geno.GWAS,
  ZETA = ZETA,
  interaction.kernel = ZETA$A$K,
  include.interaction.kernel.null = FALSE,
  include.interaction.with.gb.null = TRUE,
  n.PC = 4,
  test.method = "LR",
  kernel.method = "linear",
  gene.set = NULL,
  test.effect = "additive",
  window.size.half = 10,
  window.slide = 21,
  package.MM = "gaston",
  parallel.method = "mclapply",
  skip.check = TRUE,
  n.core = 2
)
See(SNP_set.res.int$D)  ### Column 4 contains -log10(p) values for markers

### Perform SNP-set GWAS with interaction 2
### by regarding 11 SNPs as one SNP-set with sliding window
```

```
### It will take almost 2 minutes...
SNP_set.res.int2 <- RGWAS.multisnp.interaction(
  pheno = pheno.GWAS,
  geno = geno.GWAS,
  ZETA = ZETA,
  interaction.kernel = ZETA$A$K,
  include.interaction.kernel.null = FALSE,
  include.interaction.with.gb.null = TRUE,
  n.PC = 4,
  test.method = "LR",
  kernel.method = "linear",
  gene.set = NULL,
  test.effect = "additive",
  window.size.half = 5,
  window.slide = 1,
  package.MM = "gaston",
  parallel.method = "mclapply",
  skip.check = TRUE,
  n.core = 2
)
See(SNP_set.res.int2$D)  ### Column 4 contains -log10(p) values for markers


### Perform haplotype-block GWAS with interaction
### by using the list of haplotype blocks estimated by PLINK
haplo_block.res.int <- RGWAS.multisnp.interaction(
  pheno = pheno.GWAS,
  geno = geno.GWAS,
  ZETA = ZETA,
  interaction.kernel = ZETA$A$K,
  include.interaction.kernel.null = FALSE,
  include.interaction.with.gb.null = TRUE,
  n.PC = 4,
  test.method = "LR",
  kernel.method = "linear",
  gene.set = Rice_haplo_block,
  test.effect = "additive",
  package.MM = "gaston",
  parallel.method = "mclapply",
  skip.check = TRUE,
  n.core = 2
)
See(haplo_block.res.int$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 (genome-wide association studies). 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,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  P3D = TRUE,
  n.core = 1,
  parallel.method = "mclapply",
  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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

ZETA
: A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

package.MM
: The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

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. This argument is not valid when ‘parallel.method = "furrr"‘.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (`https://github.com/HenrikBengtsson/progressr`). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count =

TRUE‘, we also utilize setTxtProgressBar and txtProgressBar functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

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. This argument is only valid when ‘package.MM = 'RAINBOWR'‘.

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.

skip.check
: As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting ‘skip.check = TRUE‘.

verbose
: If this argument is TRUE, messages for the current steps will be shown.

verbose2
: 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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
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,
                           package.MM = "gaston", parallel.method = "mclapply",
                           skip.check = TRUE, n.core = 2)
See(normal.res$D)  ### Column 4 contains -log10(p) values for markers
```

`RGWAS.normal.interaction`

Perform normal GWAS including interaction (test each single SNP)

Description

This function performs single-SNP GWAS (genome-wide association studies), including the interaction between SNP and genetic background (or other environmental factors). The model of GWAS is quite similar to the one in the 'RGWAS.normal' function:

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

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 $S_i\alpha_i$, this term is only the difference compared to the model for normal single-SNP GWAS. Here, S_i includes the ith marker of genotype data and the interaction variables between the ith marker of genotype data and the matrix representing the genetic back ground (or some environmental factors). α_i is the cooresponding effects of that marker and the interaction term. 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.interaction(
  pheno,
  geno,
  ZETA = NULL,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  interaction.with.SNPs = NULL,
  interaction.mat.method = "PCA",
  n.interaction.element = 1,
  interaction.group = NULL,
  n.interaction.group = 3,
  interaction.group.method = "find.clusters",
  n.PC.dapc = 1,
  test.method.interaction = "simultaneous",
  n.PC = 0,
  min.MAF = 0.02,
  P3D = TRUE,
  n.core = 1,
  parallel.method = "mclapply",
  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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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.

`interaction.with.SNPs`
: A $m \times q$ matrix. Interaction between each SNP and this matrix will also be tested. For example, principal components of genomic relationship matrix can be used as this matrix to test the interaction between SNPs and the genetic background. Or you can test the interaction with some environmental factors by inputting some omics data that represent the environment. (Test inluding GxE effects.)

`interaction.mat.method`
: Method to compute 'interaction.with.SNPs' when 'interaction.with.SNPs' is NULL. We offer the following four different methods:

  "PCA": Principal component analysis for genomic relationship matrix ('K' in 'ZETA') using 'prcomp' function

  "LDA": Linear discriminant analysis with independent variables as genomic relationship matrix ('K' in 'ZETA') and dependent variables as some group information ('interaction.group') using 'lda' function

  "GROUP": Dummy variables for some group information ('interaction.group')

  "DAPC": Perform LDA to the principal components of PCAfor genomic relationship matrix ('K' in 'ZETA') using 'dapc' function in 'adgenet' package. See Jombart et al., 2010 and dapc for more details.

`n.interaction.element`
: Number of elements (variables) that are included in the model as interaction term for 'interaction.with.SNPs'. If 'interaction.with.SNPs = NULL' and 'n.interaction.element = 0', then the standard SNP-based GWAS will be performed by 'RGWAS.normal' function.

`interaction.group`
: When you use "LDA", "GROUP", or "DAPC", the information on groups (e.g., subgroups for the population) will be required. You can set a vector of group names (or clustering ids) for each genotype as this argument. This vector should be factor.

`n.interaction.group`
: When 'interaction.group = NULL', 'interaction.group' will be automatically determined by using k-medoids method ('pam' function in 'cluster' package). You should specify the number of groups by this argument to decide 'interaction.group'.

`interaction.group.method`
: The method to perform clustering when 'interaction.group = NULL'. We offer the following two methods "find.clusters" and "pam". "find.clusters" performs 'adegenet::find.clusters' functions to conduct successive K-means clustering, "pam" performs 'cluster::pam' functions to conduct k-medoids clustering. See find.clusters and pam for more details.

`n.PC.dapc`
: Number of principal components to be used for 'adegenet::find.clusters' or 'adegenet::dapc' functions.

`test.method.interaction`
: Method for how to test SNPs and the interactions between SNPs and the genetic background. We offer three methods as follows:

  "simultaneous": All effects (including SNP efects) are tested simultanously.

  "snpSeparate": SNP effects are tested as one effect, and the other interaction effects are simulateneously.

  "oneByOne": All efects are tested separately, one by one.

`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. This argument is not valid when 'parallel.method = "furrr"'.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

  When 'parallel.method = "mclapply"', we utilize `pbmclapply` function in the 'pbmcapply' package with 'count = TRUE' and `mclapply` function in the 'parallel' package with 'count = FALSE'.

  When 'parallel.method = "furrr"', we utilize `future_map` function in the 'furrr' package. With 'count = TRUE', we also utilize `progressor` function in the 'progressr' package to show the progress bar, so please install the 'progressr' package from github (https://github.com/HenrikBengtsson/progressr). For 'parallel.method = "furrr"', you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to 'parallel.method = "mclapply"'.

  When 'parallel.method = "foreach"', we utilize `foreach` function in the 'foreach' package with the utilization of `makeCluster` function in 'parallel' package, and `registerDoParallel` function in 'doParallel' package. With 'count = TRUE', we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the 'utils' package to show the progress bar.

  We recommend that you use the option 'parallel.method = "mclapply"', but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option 'parallel.method = "foreach"'.

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. This argument is only valid when ‘package.MM = ’RAINBOWR’‘.

`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.

`skip.check` As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting ‘skip.check = TRUE‘.

`verbose` If this argument is TRUE, messages for the current steps will be shown.

`verbose2` 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 List of data.frame which contains the information of the map you input and the results of RGWAS (-log10(p)) which correspond to the map for each tested effect.

$thres A matrix which contains the information of threshold determined by FDR = 0.05. (each trait x each tested effect)

$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.

Jombart, T., Devillard, S. and Balloux, F. (2010) Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. BMC Genet 11(1), 94.

Examples

```
### Import RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
See(pheno.GWAS)
See(geno.GWAS)
str(ZETA)



### Perform single-SNP GWAS with interaction
### by testing all effects (including SNP effects) simultaneously
normal.res.int <-
  RGWAS.normal.interaction(
    pheno = pheno.GWAS,
    geno = geno.GWAS,
    ZETA = ZETA,
    interaction.with.SNPs = NULL,
    interaction.mat.method = "PCA",
    n.interaction.element = 3,
    interaction.group = NULL,
    n.interaction.group = 3,
    interaction.group.method = "find.clusters",
    n.PC.dapc = 3,
    test.method.interaction = "simultaneous",
    n.PC = 3,
    P3D = TRUE,
    plot.qq = TRUE,
    plot.Manhattan = TRUE,
    verbose = TRUE,
    verbose2 = FALSE,
    count = TRUE,
    time = TRUE,
    package.MM = "gaston",
    parallel.method = "mclapply",
    skip.check = TRUE,
    n.core = 2
```

```
  )
See(normal.res.int$D[[1]])  ### Column 4 contains -log10(p) values
                            ### for all effects (including SNP effects)
```

RGWAS.twostep *Perform normal GWAS (genome-wide association studies) first, then perform SNP-set GWAS for relatively significant markers*

Description

Perform normal GWAS (genome-wide association studies) first, then perform SNP-set GWAS for relatively significant markers

Usage

```
RGWAS.twostep(
  pheno,
  geno,
  ZETA = NULL,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  n.core = 1,
  parallel.method = "mclapply",
  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,
  map.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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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. This argument is not valid when ‘parallel.method = "furrr"‘.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

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 NOIA methods for additive GRM.

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` RAINBOWR 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.

`map.gene.set` Genotype map for 'gene.set' (list of haplotype blocks). This is a data.frame with the haplotype block (SNP-set, or gene-set) names in the first column. The second and third columns contain the chromosome and map position for each block. The forth column contains the cumulative map position for each block, which can be computed by cumsumPos function. If this argument is NULL, the map will be constructed by genesetmap function after the SNP-set GWAS. It will take some time, so you can reduce the computational time by assigning this argument beforehand.

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 weighting.center = 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.col3
: Color of the points of manhattan plot which are added after the reestimation by SNP-set method. You should substitute this argument as color vector whose length is 2. 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".

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.

`skip.check` As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting 'skip.check = TRUE'.

`verbose` If this argument is TRUE, messages for the current steps will be shown.

`verbose2` 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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
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 = 3,
                                      window.slide = 2, package.MM = "gaston",
                                      parallel.method = "mclapply",
                                      skip.check = TRUE, n.core = 2)

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 (genome-wide association studies) first, then check epistatic effects for relatively significant markers*

Description

Perform normal GWAS (genome-wide association studies) first, then check epistatic effects for relatively significant markers

Usage

```
RGWAS.twostep.epi(
  pheno,
  geno,
  ZETA = NULL,
  package.MM = "gaston",
  covariate = NULL,
  covariate.factor = NULL,
  structure.matrix = NULL,
  n.PC = 0,
  min.MAF = 0.02,
  n.core = 1,
  parallel.method = "mclapply",
  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,
  skip.self.int = FALSE,
  haplotype = TRUE,
  num.hap = NULL,
  optimizer = "nlminb",
  window.size.half = 5,
  window.slide = 1,
  chi0.mixture = 0.5,
  gene.set = NULL,
```

```
  map.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,
  skip.check = FALSE,
  verbose = TRUE,
  verbose2 = 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.

`ZETA` A list of covariance (relationship) matrix (K: $m \times m$) and its design matrix (Z: $n \times m$) of random effects. Please set names of list "Z" and "K"! 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))

Z.A, Z.D Design matrix ($n \times m$) for the random effects. So, in many cases, you can use the identity matrix.

K.A, K.D Different kernels which express some relationships between lines.

For example, K.A is additive relationship matrix for the covariance between lines, and K.D is dominance relationship matrix.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is 'gaston'. See more details at `EM3.general`.

`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. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

  When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

  When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (`https://github.com/HenrikBengtsson/progressr`). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

  When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

  We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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.

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

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set 'skip.self.int = TRUE'.

`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` RAINBOWR 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.

`map.gene.set` Genotype map for 'gene.set' (list of haplotype blocks). This is a data.frame with the haplotype block (SNP-set, or gene-set) names in the first column. The second and third columns contain the chromosome and map position for each block. The forth column contains the cumulative map position for each block, which can be computed by `cumsumPos` function. If this argument is NULL, the map will be constructed by `genesetmap` function after the SNP-set GWAS. It will take some time, so you can reduce the computational time by assigning this argument beforehand.

`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.
`skip.check`	As default, RAINBOWR checks the type of input data and modifies it into the correct format. However, it will take some time, so if you prepare the correct format of input data, you can skip this procedure by setting 'skip.check = TRUE'.
`verbose`	If this argument is TRUE, messages for the current steps will be shown.
`verbose2`	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

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

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

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

- **$x, $y** The information of the positions of SNPs detected by regular GWAS. These vectors are used when drawing plots. Each output correspond to the replication of row and column of scores.
- **$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 RAINBOWR
require(RAINBOWR)

### Load example datasets
data("Rice_Zhao_etal")
Rice_geno_score <- Rice_Zhao_etal$genoScore
Rice_geno_map <- Rice_Zhao_etal$genoMap
Rice_pheno <- Rice_Zhao_etal$pheno

### 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 <- 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 genomic relationship matrix (GRM)
K.A <- calcGRM(genoMat = x)


### 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 RAINBOWR
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 = "LR", gene.set = NULL,
                                     window.size.half = 10, window.slide = 21,
                                  package.MM = "gaston", parallel.method = "mclapply",
                                     skip.check = TRUE, n.core = 2)

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

Rice_geno_map *Physical map of rice genome*

Description

A dataset containing the information of phycical map of rice genome (Zhao et al., 2010; PLoS One 5(5): e10780).

Format

A data frame with 1311 rows and 3 variables:

marker marker name for each marker, character

chr chromosome number for each marker, integer

pos physical position for each marker, integer, (b.p.)

Source

http://www.ricediversity.org/data/

References

Zhao K, Wright M, Kimball J, Eizenga G, McClung A, Kovach M, Tyagi W, Ali ML, Tung CW, Reynolds A, Bustamante CD, McCouch SR (2010). Genomic Diversity and Introgression in O. sativa Reveal the Impact of Domestication and Breeding on the Rice Genome. PLoS One. 2010; 5(5): e10780.

`Rice_geno_score` *Marker genotype of rice genome*

Description

A dataset containing the information of marker genotype (scored with -1, 0, 1) of rice genome (Zhao et al., 2010; PLoS One 5(5): e10780).

Format

A data frame with 1311 rows and 395 variables:

Each column shows the marker genotype of each accession. The column names are the names of accessions and the rownames are the names of markers.

Source

http://www.ricediversity.org/data/

References

Zhao K, Wright M, Kimball J, Eizenga G, McClung A, Kovach M, Tyagi W, Ali ML, Tung CW, Reynolds A, Bustamante CD, McCouch SR (2010). Genomic Diversity and Introgression in O. sativa Reveal the Impact of Domestication and Breeding on the Rice Genome. PLoS One. 2010; 5(5): e10780.

`Rice_haplo_block` *Physical map of rice genome*

Description

A dataset containing the information of haplotype block of rice genome (Zhao et al., 2010; PLoS One 5(5): e10780). The haplotype blocks were estimated using PLINK 1.9 (See reference).

Format

A data frame with 74 rows and 2 variables:

block names of haplotype blocks which consist of marker(s) in Rice_geno_score, character

marker marker names for each marker corresponding to those in Rice_geno_score, character

Source

http://www.ricediversity.org/data/

References

Zhao K, Wright M, Kimball J, Eizenga G, McClung A, Kovach M, Tyagi W, Ali ML, Tung CW, Reynolds A, Bustamante CD, McCouch SR (2010). Genomic Diversity and Introgression in O. sativa Reveal the Impact of Domestication and Breeding on the Rice Genome. PLoS One. 2010; 5(5): e10780. Purcell, S. and Chang, C. (2018). PLINK 1.9, www.cog-genomics.org/plink/1.9/. Chang CC, Chow CC, Tellier LCAM, Vattikuti S, Purcell SM, Lee JJ (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience, 4. Gaunt T, Rodríguez S, Day I (2007) Cubic exact solutions for the estimation of pairwise haplotype frequencies: implications for linkage disequilibrium analyses and a web tool 'CubeX'. BMC Bioinformatics, 8. Taliun D, Gamper J, Pattaro C (2014) Efficient haplotype block recognition of very long and dense genetic sequences. BMC Bioinformatics, 15.

`Rice_pheno` *Phenotype data of rice field trial*

Description

A dataset containing the information of phenotype data of rice field trial (Zhao et al., 2011; Nat Comm 2:467).

Format

A data frame with 413 rows and 36 variables:

Phenotypic data of 36 traits obtained by the field trial with 413 genotypes.

Source

http://www.ricediversity.org/data/

References

Zhao, K. et al. (2011) Genome-wide association mapping reveals a rich genetic architecture of complex traits in Oryza sativa. Nat Commun. 2: 467.

`Rice_Zhao_etal` *Rice_Zhao_etal:*

Description

A list containing the information of marker genotype of rice genome (Zhao et al., 2010; PLoS One 5(5): e10780) and phenotype data of rice field trial (Zhao et al., 2011; Nat Comm 2:467).

Usage

```
Rice_Zhao_etal
```

Format

A list of 4 data frames:

$genoScore marker genotyope, Rice_geno_score

$genoMap physical map, Rice_geno_map

$pheno phenotype, Rice_pheno

$haploBlock haplotype block, Rice_haplo_block

Details

Marker genotype and phenotype data of rice by Zhao et al., 2010.

Source

http://www.ricediversity.org/data/

References

Zhao K, Wright M, Kimball J, Eizenga G, McClung A, Kovach M, Tyagi W, Ali ML, Tung CW, Reynolds A, Bustamante CD, McCouch SR (2010). Genomic Diversity and Introgression in O. sativa Reveal the Impact of Domestication and Breeding on the Rice Genome. PLoS One. 2010; 5(5): e10780. Zhao, K. et al. (2011) Genome-wide association mapping reveals a rich genetic architecture of complex traits in Oryza sativa. Nat Commun. 2: 467. Purcell, S. and Chang, C. (2018). PLINK 1.9, www.cog-genomics.org/plink/1.9/. Chang CC, Chow CC, Tellier LCAM, Vattikuti S, Purcell SM, Lee JJ (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience, 4. Gaunt T, Rodríguez S, Day I (2007) Cubic exact solutions for the estimation of pairwise haplotype frequencies: implications for linkage disequilibrium analyses and a web tool 'CubeX'. BMC Bioinformatics, 8. Taliun D, Gamper J, Pattaro C (2014) Efficient haplotype block recognition of very long and dense genetic sequences. BMC Bioinformatics, 15.

See Also

Rice_geno_score, Rice_geno_map, Rice_pheno, Rice_haplo_block

---

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,
  package.MM = "gaston",
  Hinv,
  P3D = TRUE,
```

```
  eigen.G = NULL,
  optimizer = "nlminb",
  n.core = 1,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now`	A $n \times m$ genotype matrix where n is sample size and m 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`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ matrix. You should assign mean vector (rep(1, n)) and covariates. NA is not allowed.
`package.MM`	The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is 'gaston'. See more details at `EM3.general`.
`Hinv`	The inverse of $H = ZKZ' + \lambda I$ where $\lambda = \sigma^2_e / \sigma^2_u$.
`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.
`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 RAINBOWR. 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. This argument is only valid when 'package.MM = 'RAINBOWR''.
`n.core`	Setting n.core > 1 will enable parallel execution on a machine with multiple cores.
`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,
  package.MM = "gaston",
  eigen.SGS = NULL,
  eigen.G = NULL,
  n.core = 1,
  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,
  skip.self.int = FALSE,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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"!

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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 RAINBOWR. 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 RAINBOWR. 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.

`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` RAINBOWR 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.

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set 'skip.self.int = TRUE'.

`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.epistasis.LR.MC` *Calculate -log10(p) of epistatic effects by LR test (multi-cores)*

Description

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

Usage

```
score.calc.epistasis.LR.MC(
  M.now,
  y,
  X.now,
  ZETA.now,
  package.MM = "gaston",
  eigen.SGS = NULL,
  eigen.G = NULL,
  n.core = 2,
  parallel.method = "mclapply",
  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,
  skip.self.int = FALSE,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now` A $n \times m$ genotype matrix where n is sample size and m is the number of markers.

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

`X.now` A $n \times p$ 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"!

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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 RAINBOWR. 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 RAINBOWR. 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. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`

Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (`https://github.com/HenrikBengtsson/progressr`). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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` RAINBOWR 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.

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set ‘skip.self.int = TRUE‘.

`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.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,
  skip.self.int = FALSE,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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`	A $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

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

`P0` A $n \times n$ 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` RAINBOWR 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.

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set 'skip.self.int = TRUE'.

`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.epistasis.score.MC
Calculate -log10(p) of epistatic effects with score test (multi-cores)

Description

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

Usage

```
score.calc.epistasis.score.MC(
  M.now,
  y,
  X.now,
  ZETA.now,
  n.core = 2,
  parallel.method = "mclapply",
  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,
  skip.self.int = FALSE,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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"!
`n.core`	Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when ‘parallel.method = "furrr"‘.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

  When 'parallel.method = "mclapply"', we utilize pbmclapply function in the 'pbmcapply' package with 'count = TRUE' and mclapply function in the 'parallel' package with 'count = FALSE'.

  When 'parallel.method = "furrr"', we utilize future_map function in the 'furrr' package. With 'count = TRUE', we also utilize progressor function in the 'progressr' package to show the progress bar, so please install the 'progressr' package from github (https://github.com/HenrikBengtsson/progressr). For 'parallel.method = "furrr"', you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to 'parallel.method = "mclapply"'.

  When 'parallel.method = "foreach"', we utilize foreach function in the 'foreach' package with the utilization of makeCluster function in 'parallel' package, and registerDoParallel function in 'doParallel' package. With 'count = TRUE', we also utilize setTxtProgressBar and txtProgressBar functions in the 'utils' package to show the progress bar.

  We recommend that you use the option 'parallel.method = "mclapply"', but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option 'parallel.method = "foreach"'.

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

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

P0
: A $n \times n$ 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
: RAINBOWR 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.

`skip.self.int` As default, the function also tests the self-interactions among the same SNP-sets. If you want to avoid this, please set 'skip.self.int = TRUE'.

`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.int` *Calculate -log10(p) for single-SNP GWAS with interaction*

Description

Calculate -log10(p) of each SNP by the Wald test for the model inluding interaction term.

Usage

```
score.calc.int(
  M.now,
  ZETA.now,
  y,
  X.now,
  package.MM = "gaston",
  interaction.with.SNPs.now,
  test.method.interaction = "simultaneous",
  include.SNP.effect = TRUE,
  Hinv,
  P3D = TRUE,
  eigen.G = NULL,
  optimizer = "nlminb",
```

```
  n.core = 1,
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now` A $n \times m$ genotype matrix where n is sample size and m 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` A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

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

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`interaction.with.SNPs.now` A $m \times q$ matrix. Interaction between each SNP and this matrix will also be tested. For example, principal components of genomic relationship matrix can be used as this matrix to test the interaction between SNPs and the genetic background.

`test.method.interaction` Method for how to test SNPs and the interactions between SNPs and the genetic background. We offer three methods as follows:

"simultaneous": All effects (including SNP efects) are tested simultanously.

"snpSeparate": SNP effects are tested as one effect, and the other interaction effects are simulateneously.

"oneByOne": All efects are tested separately, one by one.

`include.SNP.effect` Whether or not including SNP effects into the tested effects.

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

`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.

`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 RAINBOWR. 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. This argument is only valid when ‘package.MM = ’RAINBOWR’‘.

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

`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.int.MC` *Calculate -log10(p) for single-SNP GWAS with interaction (multi-cores)*

Description

Calculate -log10(p) of each SNP by the Wald test for the model inluding interaction term.

Usage

```
score.calc.int.MC(
  M.now,
  ZETA.now,
  y,
  X.now,
  package.MM = "gaston",
  interaction.with.SNPs.now,
  test.method.interaction = "simultaneous",
  include.SNP.effect = TRUE,
  Hinv,
  n.core = 2,
  parallel.method = "mclapply",
  P3D = TRUE,
  eigen.G = NULL,
  optimizer = "nlminb",
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

`M.now` A $n \times m$ genotype matrix where n is sample size and m 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` A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

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

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is 'gaston'. See more details at `EM3.general`.

`interaction.with.SNPs.now`
A $m \times q$ matrix. Interaction between each SNP and this matrix will also be tested. For example, principal components of genomic relationship matrix can be used as this matrix to test the interaction between SNPs and the genetic background.

`test.method.interaction`
Method for how to test SNPs and the interactions between SNPs and the genetic background. We offer three methods as follows:

"simultaneous": All effects (including SNP efects) are tested simultanously.

"snpSeparate": SNP effects are tested as one effect, and the other interaction effects are simulateneously.

"oneByOne": All efects are tested separately, one by one.

`include.SNP.effect`
Whether or not including SNP effects into the tested effects.

`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. This argument is not valid when 'parallel.method = "furrr"'.

`parallel.method`
Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When 'parallel.method = "mclapply"', we utilize `pbmclapply` function in the 'pbmcapply' package with 'count = TRUE' and `mclapply` function in the 'parallel' package with 'count = FALSE'.

When 'parallel.method = "furrr"', we utilize `future_map` function in the 'furrr' package. With 'count = TRUE', we also utilize `progressor` function in the 'progressr' package to show the progress bar, so please install the 'progressr' package from github (https://github.com/HenrikBengtsson/progressr). For 'parallel.method = "furrr"', you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to 'parallel.method = "mclapply"'.

When 'parallel.method = "foreach"', we utilize `foreach` function in the 'foreach' package with the utilization of `makeCluster` function in 'parallel' package, and `registerDoParallel` function in 'doParallel' package. With 'count = TRUE', we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the 'utils' package to show the progress bar.

We recommend that you use the option 'parallel.method = "mclapply"', but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option 'parallel.method = "foreach"'.

`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.

`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 RAINBOWR. 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. This argument is only valid when ‘package.MM = ’RAINBOWR’‘.

`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.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 (likelihood-ratio) 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,
  package.MM = "gaston",
  LL0,
  eigen.SGS = NULL,
  eigen.G = NULL,
  n.core = 1,
  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`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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"!
`package.MM`	The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.
`LL0`	The log-likelihood for the null model.
`eigen.SGS`	A list with **$values** Eigen values **$vectors** Eeigen 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 RAINBOWR. 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 RAINBOWR. 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.

`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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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.int — *Calculate -log10(p) of each SNP-set and its interaction with kernels by the LR test*

Description

This function calculates -log10(p) of each SNP-set and its interaction with kernels by the LR (likelihood-ratio) 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.int(
  M.now,
  y,
  X.now,
  ZETA.now,
```

```
  interaction.kernel,
  package.MM = "gaston",
  LL0,
  eigen.SGS = NULL,
  eigen.G = NULL,
  n.core = 1,
  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` A $n \times m$ genotype matrix where n is sample size and m is the number of markers.

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

`X.now` A $n \times p$ 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"!

`interaction.kernel`
A $n \times n$ Gram (kernel) matrix which may indicate some interaction with SNP-sets to be tested.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

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

`eigen.SGS` A list with

$values Eigen values

$vectors Eeigen 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 RAINBOWR. 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 RAINBOWR. 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.

`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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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.int.MC *Calculate -log10(p) of each SNP-set and its interaction with kernels by the LR test (multi-cores)*

Description

This function calculates -log10(p) of each SNP-set and its interaction with kernels by the LR (likelihood-ratio) 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.int.MC(
  M.now,
  y,
  X.now,
  ZETA.now,
  interaction.kernel,
  package.MM = "gaston",
  LL0,
  eigen.SGS = NULL,
```

```
    eigen.G = NULL,
    n.core = 2,
    parallel.method = "mclapply",
    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` A $n \times m$ genotype matrix where n is sample size and m is the number of markers.

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

`X.now` A $n \times p$ 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"!

`interaction.kernel`
A $n \times n$ Gram (kernel) matrix which may indicate some interaction with SNP-sets to be tested.

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at EM3.general.

`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 RAINBOWR. 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 RAINBOWR. 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. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`
Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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 (likelihood-ratio) 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,
  package.MM = "gaston",
  LL0,
  eigen.SGS = NULL,
  eigen.G = NULL,
  n.core = 2,
  parallel.method = "mclapply",
  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`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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"!

`package.MM` The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at `EM3.general`.

`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 RAINBOWR. 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 RAINBOWR. 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. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`

Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (`https://github.com/HenrikBengtsson/progressr`). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize `setTxtProgressBar` and `txtProgressBar` functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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,
  package.MM = "gaston",
  Hinv,
  n.core = 2,
  parallel.method = "mclapply",
  P3D = TRUE,
  eigen.G = NULL,
  optimizer = "nlminb",
  min.MAF = 0.02,
  count = TRUE
)
```

Arguments

M.now A $n \times m$ genotype matrix where n is sample size and m 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 A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.

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

package.MM The package name to be used when solving mixed-effects model. We only offer the following three packages: "RAINBOWR", "MM4LMM" and "gaston". Default package is ‘gaston‘. See more details at EM3.general.

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

n.core
: Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when ‘parallel.method = "furrr"‘.

parallel.method
: Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize pbmclapply function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and mclapply function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize future_map function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize progressor function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize foreach function in the ‘foreach‘ package with the utilization of makeCluster function in ‘parallel‘ package, and registerDoParallel function in ‘doParallel‘ package. With ‘count = TRUE‘, we also utilize setTxtProgressBar and txtProgressBar functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

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.

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 RAINBOWR. 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. This argument is only valid when ‘package.MM = ’RAINBOWR’‘.

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`	A $n \times m$ genotype matrix where n is sample size and m is the number of markers.
`y`	A $n \times 1$ vector. A vector of phenotypic values should be used. NA is allowed.
`X.now`	A $n \times p$ 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` A $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.

`Ge` A $n \times n$ 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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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,
```

```
    parallel.method = "mclapply",
    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` A $n \times m$ genotype matrix where n is sample size and m is the number of markers.

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

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

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

`P0` A $n \times n$ 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.

`n.core` Setting n.core > 1 will enable parallel execution on a machine with multiple cores. This argument is not valid when ‘parallel.method = "furrr"‘.

`parallel.method`

Method for parallel computation. We offer three methods, "mclapply", "furrr", and "foreach".

When ‘parallel.method = "mclapply"‘, we utilize `pbmclapply` function in the ‘pbmcapply‘ package with ‘count = TRUE‘ and `mclapply` function in the ‘parallel‘ package with ‘count = FALSE‘.

When ‘parallel.method = "furrr"‘, we utilize `future_map` function in the ‘furrr‘ package. With ‘count = TRUE‘, we also utilize `progressor` function in the ‘progressr‘ package to show the progress bar, so please install the ‘progressr‘ package from github (https://github.com/HenrikBengtsson/progressr). For ‘parallel.method = "furrr"‘, you can perform multi-thread parallelization by sharing memories, which results in saving your memory, but quite slower compared to ‘parallel.method = "mclapply"‘.

When ‘parallel.method = "foreach"‘, we utilize `foreach` function in the ‘foreach‘ package with the utilization of `makeCluster` function in ‘parallel‘ package, and `registerDoParallel` function in ‘doParallel‘ package. With ‘count =

TRUE‘, we also utilize setTxtProgressBar and txtProgressBar functions in the ‘utils‘ package to show the progress bar.

We recommend that you use the option ‘parallel.method = "mclapply"‘, but for Windows users, this parallelization method is not supported. So, if you are Windows user, we recommend that you use the option ‘parallel.method = "foreach"‘.

`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 NOIA methods for additive GRM.

`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` RAINBOWR 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 weighting.center = 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`	A $n \times 1$ 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`	A $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.
`Ge`	A $n \times n$ matrix. You should assign identity matrix I (diag(n)).
`P0`	A $n \times n$ 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`	A $n \times 1$ 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`	A $n \times n$ matrix. You should assign ZKZ', where K is covariance (relationship) matrix and Z is its design matrix.
`Ge`	A $n \times n$ matrix. You should assign identity matrix I (diag(n)).
`P0`	A $n \times n$ 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,
  verbose = TRUE
)
```

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.
verbose	If TRUE, print the first part of data.

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"!

`ZWs` A list of additional linear kernels other than genomic relationship matrix (GRM). We utilize this argument in RGWAS.multisnp function, so you can ignore this.

`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.

$U Eigen vectors of G.

$delta Eigen 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 (genome-wide association studies) for simulation study*

Description

Calculate some summary statistics of GWAS (genome-wide association studies) 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` A 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.

inflator.plus — If 'the -log10(p) value for each marker' exceeds ('the inflation level' + 'inflator.plus'), that marker is regarded as significant.

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 Maximum 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