Package 'GRAB'

July 25, 2025

Type Package

Title Genome-Wide Robust Analysis for Biobank Data (GRAB)

Version 0.2.1

Date 2025-08-01

Description Provides a comprehensive suite of genome-wide association study (GWAS) methods specifically designed for biobank-scale data. The package offers computationally efficient and robust association tests for time-to-event traits (e.g., Bi et al. (2020) <doi:10.1016/j.ajhg.2020.06.003>), ordinal categorical traits (e.g., Bi et al. (2021) <doi:10.1016/j.ajhg.2021.03.019>), and longitudinal traits (Xu et al. (2025) <doi:10.1038/s41467-025-56669-1>). Additionally, it includes functions for simulating genotype and phenotype data to support research and method development.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports dplyr, data.table, mvtnorm, Matrix, RSQLite, lme4, ordinal, survival, Rcpp, RcppParallel, igraph

Suggests dbplyr, optparse, tidyr, R.utils, SKAT

LinkingTo Rcpp, RcppArmadillo, RcppParallel, BH

RoxygenNote 7.3.2

NeedsCompilation yes

Author Wenjian Bi [aut],
Wei Zhou [aut],
Rounak Dey [aut],
Zhangchen Zhao [aut],
Seunggeun Lee [aut],
Woody Miao [cre]

Maintainer Woody Miao <miaolin@pku.edu.cn>

Repository CRAN

Date/Publication 2025-07-25 16:30:20 UTC

Contents

`Batcheffect.Test` *Test for batch effect*

Description

This function test for the allele frequency difference between the study cohort and the external datasets.

Usage

```
Batcheffect.Test(n0, n1, n.ext, maf0, maf1, maf.ext, pop.prev, var.ratio = 1)
```

Arguments

n0	A numeric. The sample size of cases in the study cohort
n1	A numeric. The sample size of controls in the study cohort
n.ext	A numeric. The sample size of external datasets
maf0	A numeric. The MAF of the cases.
maf1	A numeric. The MAF of the controls
maf.ext	A numeric. The MAF of the external datasets.
pop.prev	A numeric. The population prevalence of the disease.
var.ratio	A numeric. The variance ratio calculated by sparseGRM.

Value

A numeric of batch effect p-value

CCT *An analytical p-value combination method using the Cauchy distribution*

Description

The CCT function takes in a numeric vector of p-values, a numeric vector of non-negative weights, and return the aggregated p-value using Cauchy method.

Usage

```
CCT(pvals, weights = NULL)
```

Arguments

pvals	a numeric vector of p-values, where each of the element is between 0 to 1, to be combined. If a p-value equals to 1, we set it as 0.999. If a p-value equals to 0, an error action is executed.
weights	a numeric vector of non-negative weights. If NULL, the equal weights are assumed.

Value

the aggregated p-value combining p-values from the vector pvals.

References

Liu, Y., & Xie, J. (2020). Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures. *Journal of the American Statistical Association 115*(529), 393-402. (pub)

Examples

```
pvalues <- c(2e-02, 4e-04, 0.2, 0.1, 0.8)
CCT(pvals = pvalues)
```

`checkIfSampleIDsExist` *Check if sample identifiers are stored in a BGEN file*

Description

Check if sample identifiers are stored in a BGEN file, only support BGEN v1.2. Check link for more details.

Usage

```
checkIfSampleIDsExist(bgenFile)
```

Arguments

`bgenFile`	a character of BGEN file. Sometimes, BGEN file does not include sample IDs. This information can be extracted from BGEN file. Please refer to link for more details.

Value

A logical value indicating whether sample identifiers are stored in the BGEN file. Returns `TRUE` if sample IDs are present, `FALSE` otherwise.

Examples

```
BGENFile <- system.file("extdata", "simuBGEN.bgen", package = "GRAB")
checkIfSampleIDsExist(BGENFile)
```

`getDenseGRM` *Suppose that a dense GRM is Phi and input is bVec, return Phi * bVec (only for developers)*

Description

Suppose that a dense GRM is Phi and input is bVec, return Phi * bVec (only for developers), users can simply ignore this function

Usage

```
getDenseGRM(bVec)
```

Arguments

bVec	a numeric vector with the same length as in subjData (check the input of setDenseGRM)

Value

a numeric vector of Phi * bVec

Examples

```
# set up the dense GRM in C++
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
famData <- read.table(gsub("bed", "fam", GenoFile))
subjData <- famData$V2
genoList <- setDenseGRM(GenoFile, subjData = subjData)

set.seed(1)
bVec <- rnorm(1000)
KinbVec <- getDenseGRM(bVec)

# The following is based on the definition of GRM to validate the DenseGRM object
PlinkFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
IDsToIncludeFile <- system.file("extdata", "simuGENO.IDsToInclude", package = "GRAB")
GenoList <- GRAB.ReadGeno(PlinkFile, control = list(IDsToExcludeFile = IDsToIncludeFile))
GenoMat <- GenoList$GenoMat
markerInfo <- GenoList$markerInfo
pos <- which(markerInfo$CHROM != 1)
GenoMat <- GenoMat[, pos]
MAF <- apply(GenoMat, 2, mean) / 2
stdGenoMat <- (t(GenoMat) - 2 * MAF) / sqrt(2 * MAF * (1 - MAF)) / sqrt(ncol(GenoMat))
KinMat <- t(stdGenoMat) %*% stdGenoMat
KinbVec1 <- KinMat %*% bVec
# plot(KinbVec, KinbVec1)
head(cbind(KinbVec, KinbVec1))
```

---

getSampleIDsFromBGEN *Get sample identifiers from BGEN file*

Description

Extract sample identifiers from BGEN file (only support BGEN v1.2, check link)

Usage

```
getSampleIDsFromBGEN(bgenFile)
```

Arguments

bgenFile	a character of BGEN file.

Value

A character vector of sample identifiers extracted from the BGEN file.

Examples

```
BGENFile <- system.file("extdata", "simuBGEN.bgen", package = "GRAB")
getSampleIDsFromBGEN(BGENFile)
```

`getSparseGRM` *Make a* `SparseGRMFile` *for* `GRAB.NullModel`.

Description

If the sample size in analysis is greater than 100,000, we recommend using sparse GRM (instead of dense GRM) to adjust for sample relatedness. This function is to use `GCTA` (link) to make a `SparseGRMFile` to be passed to function `GRAB.NullModel`. This function can only support `Linux` and `PLINK` files as required by `GCTA` software. To make a `SparseGRMFile`, two steps are needed. Please check `Details` section for more details.

Usage

```
getSparseGRM(
  PlinkFile,
  nPartsGRM,
  SparseGRMFile,
  tempDir = NULL,
  relatednessCutoff = 0.05,
  minMafGRM = 0.01,
  maxMissingGRM = 0.1,
  rm.tempFiles = FALSE
)
```

Arguments

`PlinkFile`	a path to PLINK binary files (without file extension). Note that the current version (gcta_1.93.1beta) of `GCTA` software does not support different prefix names for BIM, BED, and FAM files.
`nPartsGRM`	a numeric value (e.g. 250): `GCTA` software can split subjects to multiple parts. For UK Biobank data analysis, it is recommended to set `nPartsGRM=250`.
`SparseGRMFile`	a path to file of output to be passed to `GRAB.NullModel`.
`tempDir`	a path to store temp files from `getTempFilesFullGRM`. This should be consistent to the input of `getTempFilesFullGRM`. Default is `system.file("SparseGRM", "temp", package = "GRAB")`.
`relatednessCutoff`	a cutoff for sparse GRM, only kinship coefficient greater than this cutoff will be retained in sparse GRM. *(default=0.05)*

minMafGRM	Minimal value of MAF cutoff to select markers (from PLINK files) to make sparse GRM. *(default=0.01)*
maxMissingGRM	Maximal value of missing rate to select markers (from PLINK files) to make sparse GRM. *(default=0.1)*
rm.tempFiles	a logical value indicating if the temp files generated in getTempFilesFullGRM will be deleted. *(default=FALSE)*

Details

- Step 1: Run getTempFilesFullGRM to save temporary files to tempDir.
- Step 2: Run getSparseGRM to combine the temporary files to make a SparseGRMFile to be passed to function GRAB.NullModel.

Users can customize parameters including (minMafGRM, maxMissingGRM, nPartsGRM), but functions getTempFilesFullGRM and getSparseGRM should use the same ones. Otherwise, package GRAB cannot accurately identify temporary files.

Value

A character string containing a message with the path to the output file where the sparse Genetic Relationship Matrix (SparseGRM) has been stored.

The following shows a typical workflow for creating a sparse GRM:

```
# Input data (We recommend setting nPartsGRM=250 for UKBB with N=500K):
GenoFile = system.file("extdata", "simuPLINK.bed", package = "GRAB")
PlinkFile = tools::file_path_sans_ext(GenoFile)
nPartsGRM = 2
```

Step 1: We strongly recommend parallel computing in high performance clusters (HPC).

```
# For Linux, get the file path of gcta64 by which command:
gcta64File <- system("which gcta64", intern = TRUE)
# For Windows, set the file path directly:
gcta64File <- "C:\\path\\to\\gcta64.exe"
# The temp outputs (may be large) will be in system.file("SparseGRM", "temp", package =
"GRAB") by default:
for(partParallel in 1:nPartsGRM) getTempFilesFullGRM(PlinkFile, nPartsGRM, partParallel,
gcta64File)
```

Step 2: Combine files in Step 1 to make a SparseGRMFile

```
tempDir = system.file("SparseGRM", "temp", package = "GRAB")
SparseGRMFile = gsub("temp", "SparseGRM.txt", tempDir)
getSparseGRM(PlinkFile, nPartsGRM, SparseGRMFile)
```

getTempFilesFullGRM *Make temporary files to be passed to function* getSparseGRM.

Description

Make temporary files to be passed to function getSparseGRM. We strongly suggest using parallel computing for different partParallel.

Usage

```
getTempFilesFullGRM(
  PlinkFile,
  nPartsGRM,
  partParallel,
  gcta64File,
  tempDir = NULL,
  subjData = NULL,
  minMafGRM = 0.01,
  maxMissingGRM = 0.1,
  threadNum = 8
)
```

Arguments

PlinkFile	a path to PLINK files (without file extensions of bed/bim/fam). Note that the current version (gcta_1.93.1beta) of gcta software does not support different prefix names for bim, bed, and fam files.
nPartsGRM	a numeric value (e.g. 250): GCTA software can split subjects to multiple parts. For UK Biobank data analysis, it is recommended to set nPartsGRM=250.
partParallel	a numeric value (from 1 to nPartsGRM) to split all jobs for parallel computation.
gcta64File	a path to GCTA program. GCTA can be downloaded from link.
tempDir	a path to store temp files to be passed to getSparseGRM. This should be consistent to the input of getSparseGRM. Default is system.file("SparseGRM", "temp", package = "GRAB").
subjData	a character vector to specify subject IDs to retain (i.e. IID). Default is NULL, i.e. all subjects are retained in sparse GRM. If the number of subjects is less than 1,000, the GRM estimation might not be accurate.
minMafGRM	Minimal value of MAF cutoff to select markers (from PLINK files) to make sparse GRM. *(default=0.01)*
maxMissingGRM	Maximal value of missing rate to select markers (from PLINK files) to make sparse GRM. *(default=0.1)*
threadNum	Number of threads (CPUs) to use.

Details

- Step 1: Run `getTempFilesFullGRM` to get temporary files.
- Step 2: Run `getSparseGRM` to combine the temporary files to make a `SparseGRMFile` to be passed to `GRAB.NullModel`.

Value

A character string message indicating the completion status and location of the temporary files.

Examples

```
## Please check help(getSparseGRM) for an example.
```

`getVersionFromBGEN` *Get version information from BGEN file*

Description

Get version information from BGEN file (check link)

Usage

```
getVersionFromBGEN(bgenFile)
```

Arguments

`bgenFile`	a character of BGEN file.

Value

A character string indicating the BGEN file version. Possible values include:

v1.1 BGEN format version 1.1

v1.2 BGEN format version 1.2

Version Layout = 0, which is not supported... Error message for unsupported version 0

Version Layout > 2, which is reserved for future use... Warning message for future versions

Examples

```
BGENFile <- system.file("extdata", "simuBGEN.bgen", package = "GRAB")
getVersionFromBGEN(BGENFile)
```

GRAB.getGenoInfo *Get allele frequency and missing rate information from genotype data*

Description

This function shares input as in function GRAB.ReadGeno, please check ?GRAB.ReadGeno for more details.

Usage

```
GRAB.getGenoInfo(
  GenoFile,
  GenoFileIndex = NULL,
  SampleIDs = NULL,
  control = NULL
)
```

Arguments

GenoFile	a character of genotype file. See Details section for more details.
GenoFileIndex	additional index file(s) corresponding to GenoFile. See Details section for more details.
SampleIDs	a character vector of sample IDs to extract. The default is NULL, that is, all samples in GenoFile will be extracted.
control	a list of parameters to decide which markers to extract. See Details section for more details.

Value

A data frame containing marker information with allele frequencies and missing rates. The data frame includes columns from marker information (CHROM, POS, ID, REF, ALT, etc.) plus additional columns:

altFreq Alternative allele frequency (before genotype imputation)

missingRate Missing rate for each marker

GRAB.makePlink *Make PLINK files using a numeric R matrix*

Description

Make PLINK files using a numeric matrix GenoMat (0,1,2,-9), rownames(GenoMat) are subject IDs and colnames(GenoMat) are marker IDs

Usage

```
GRAB.makePlink(
  GenoMat,
  OutputPrefix,
  A1 = "G",
  A2 = "A",
  CHR = NULL,
  BP = NULL,
  Pheno = NULL,
  Sex = NULL
)
```

Arguments

GenoMat	a numeric n*m genotype matrix (0,1,2,-9). Each row is for one subject and each column is for one marker. Row names of subject IDs and column names of marker IDs are required.
OutputPrefix	a character, prefix of the PLINK files to output (including path).
A1	a character to specify allele 1 (*default="G"*), usually minor (ALT).
A2	a character to specify allele 2 (*default="A"*), usually major (REF).
CHR	a character vector of the chromosome numbers for all markers. *Default=NULL*, that is, CHR=rep(1, m).
BP	a numeric vector of the base positions for all markers. *Default=NULL*, that is, BP=1:m).
Pheno	a character vector of the phenotypes for all subjects. *Default=NULL*, that is, Pheno=rep(-9, n).
Sex	a numeric vector of the sex for all subjects. *Default=NULL*, that is, Sex=rep(1, n)).

Details

Check link for detailed information of PLINK 2.00 alpha. Check link for detailed information of bgenix tool.

Convert PLINK text files to binary files:

Run plink --file simuPLINK --make-bed --out simuPLINK to convert PLINK text files (MAP and PED) to binary files (BED, BIM, and FAM).

Convert PLINK binary files to raw files:

Run plink --bfile simuPLINK --recode A --out simuRAW to convert PLINK binary files (BED, BIM, and FAM) to raw files (raw).

Convert PLINK binary files to bgen files:

RUN plink2 --bfile simuPLINK --export bgen-1.2 bits=8 ref-first --out simuBGEN to convert PLINK binary files (BED, BIM, and FAM) to BGEN binary files (BGEN).

Make bgi file using bgenix **tool:**

RUN bgenix -g simuBGEN.bgen --index

Value

PLINK text files (PED and MAP) are stored in 'OutputPrefix'. Suppose A1 is "G" and A2 is "A", then genotype of 0,1,2,-9 will be coded as "GG", "AG", "AA", "00". If PLINK binary files (BED, BIM, and FAM) are required, please download PLINK software and use option of "–make-bed". Please check Details section for the downstream process.

Examples

```
### Step 1: simulate a numeric genotype matrix
n <- 1000
m <- 20
MAF <- 0.3
set.seed(123)
GenoMat <- matrix(rbinom(n * m, 2, MAF), n, m)
rownames(GenoMat) <- paste0("Subj-", 1:n)
colnames(GenoMat) <- paste0("SNP-", 1:m)
outputDir <- system.file("results", package = "GRAB")
outputPrefix <- paste0(outputDir, "/simuPLINK")

### Step 2(a): make PLINK files without missing genotype
GRAB.makePlink(GenoMat, outputPrefix)

### Step 2(b): make PLINK files with genotype missing rate of 0.1
indexMissing <- sample(n * m, 0.1 * n * m)
GenoMat[indexMissing] <- -9
GRAB.makePlink(GenoMat, outputPrefix)

## The following are in shell environment
# plink --file simuPLINK --make-bed --out simuPLINK
# plink --bfile simuPLINK --recode A --out simuRAW
# plink2 --bfile simuPLINK --export bgen-1.2 bits=8 ref-first --out simuBGEN
# UK Biobank use 'ref-first'
# bgenix -g simuBGEN.bgen --index
```

GRAB.Marker *Conduct marker-level genetic association testing*

Description

Test for association between phenotype of interest and genetic marker.

Usage

```
GRAB.Marker(
  objNull,
  GenoFile,
  GenoFileIndex = NULL,
  OutputFile,
```

```
  OutputFileIndex = NULL,
  control = NULL
)
```

Arguments

objNull	the output object of function GRAB.NullModel.
GenoFile	a character of genotype file. Currently, two types of genotype formats are supported: PLINK and BGEN. Check GRAB.ReadGeno for more details.
GenoFileIndex	additional index files corresponding to the GenoFile. If NULL (default), the prefix is the same as GenoFile. Check GRAB.ReadGeno for more details.
OutputFile	a character of output file to save the analysis results.
OutputFileIndex	a character of output index file to record the end point. If the program ends unexpectedly, the end point can help GRAB package understand where to restart the analysis. If NULL (default), OutputFileIndex = paste0(OutputFile, ".index").
control	a list of parameters for controlling function GRAB.Marker, more details can be seen in Details section.

Details

GRAB package supports POLMM, SPACox, SPAGRM, SPAmix, and WtCoxG methods. Detailed information about the analysis methods is given in the Details section of GRAB.NullModel. Users do not need to specify them since functions GRAB.Marker and GRAB.Region will check the class(objNull).

The following details are about argument control:

The below is to let users customize markers to include in analysis. If these parameters are not specified, GRAB package will include all markers in analysis. For PLINK files, the default control$AlleleOrder = "alt-first"; for BGEN files, the default control$AlleleOrder = "ref-first".

- IDsToIncludeFile: please refer to the Details section of GRAB.ReadGeno.
- IDsToExcludeFile: please refer to the Details section of GRAB.ReadGeno.
- RangesToIncludeFile: please refer to the Details section of GRAB.ReadGeno.
- RangesToExcludeFile: please refer to the Details section of GRAB.ReadGeno.
- AlleleOrder: please refer to the Details section of GRAB.ReadGeno.

The below is to customize the quality-control (QC) process.

- omp_num_threads: (To be added later) a numeric value (default: value from data.table::getDTthreads()) to specify the number of threads in OpenMP for parallel computation.
- ImputeMethod: a character, "mean" (default), "bestguess", or "drop" (to be added later). Please refer to the Details section of GRAB.ReadGeno.
- MissingRateCutoff: a numeric value *(default=0.15)*. Markers with missing rate > this value will be excluded from analysis.
- MinMAFCutoff: a numeric value *(default=0.001)*. Markers with MAF < this value will be excluded from analysis.
- MinMACCutoff: a numeric value *(default=20)*. Markers with MAC < this value will be excluded from analysis.

- nMarkersEachChunk: number of markers *(default=10000)* in one chunk to output.

The below is to customize the columns in the OutputFile. Columns of Marker, Info, AltFreq, AltCounts, MissingRate, Pvalue are included for all methods.

- outputColumns: For example, for POLMM method, users can set control$outputColumns = c("beta", "seBeta", "AltFreqInGroup"):
  - POLMM: Default: beta, seBeta; Optional: zScore, AltFreqInGroup, nSamplesInGroup, AltCountsInGroup
  - SPACox: Optional: zScore

Value

The analysis results are written in a file of OutputFile, which includes the following columns.

Marker Marker IDs extracted from GenoFile and GenoFileIndex.

Info Marker Information of "CHR:POS:REF:ALT". The order of REF/ALT depends on control$AlleleOrder: "ref-first" or "alt-first".

AltFreq Alternative allele frequency (before genotype imputation, might be > 0.5). If the AltFreq of most markers are > 0.5, you should consider resetting control$AlleleOrder.

AltCounts Alternative allele counts (before genotype imputation).

MissingRate Missing rate for each marker

Pvalue Association test p-value

The following columns can be customized using control$outputColumns. Check makeGroup for details about phenotype grouping which are used for nSamplesInGroup, AltCountsInGroup, and AltFreqInGroup.

beta Estimated effect size of the ALT allele.

seBeta Estimated standard error (se) of the effect size.

zScore z score, standardized score statistics, usually follows a standard normal distribution.

nSamplesInGroup Number of samples in different phenotype groups. This can be slightly different from the original distribution due to the genotype missing.

AltCountsInGroup Alternative allele counts (before genotype imputation) in different phenotype groups.

AltFreqInGroup Alternative allele frequency (before genotype imputation) in different phenotype groups.

Examples

```
objNullFile <- system.file("results", "objPOLMMFile.RData", package = "GRAB")
load(objNullFile)
class(obj.POLMM) # "POLMM_NULL_Model", that indicates an object from POLMM method.

OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/simuOUTPUT.txt")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")

## make sure the output files does not exist at first
```

```
if (file.exists(OutputFile)) file.remove(OutputFile)
if (file.exists(paste0(OutputFile, ".index"))) file.remove(paste0(OutputFile, ".index"))

GRAB.Marker(obj.POLMM,
  GenoFile = GenoFile,
  OutputFile = OutputFile
)

data.table::fread(OutputFile)

## additional columns of "zScore", "nSamplesInGroup", "AltCountsInGroup", "AltFreqInGroup"
## We do not recommend adding too many columns for all markers

if (file.exists(OutputFile)) file.remove(OutputFile)
if (file.exists(paste0(OutputFile, ".index"))) file.remove(paste0(OutputFile, ".index"))
GRAB.Marker(obj.POLMM,
  GenoFile = GenoFile,
  OutputFile = OutputFile,
  control = list(outputColumns = c(
    "beta", "seBeta", "zScore",
    "nSamplesInGroup", "AltCountsInGroup",
    "AltFreqInGroup"
  ))
)
data.table::fread(OutputFile)
```

`GRAB.NullModel` *Fit a null model to estimate parameters and residuals*

Description

We fit a null model including response variable, covariates, and Genetic Relationship Matrix (GRM, if needed) to estimate parameters and residuals.

Usage

```
GRAB.NullModel(
  formula,
  data = NULL,
  subset = NULL,
  subjData,
  method = "SPACox",
  traitType = "time-to-event",
  GenoFile = NULL,
  GenoFileIndex = NULL,
  SparseGRMFile = NULL,
  control = NULL,
  ...
)
```

Arguments

formula	a formula object, with the response on the left of a ~ operator and the covariates on the right. Do not add a column of intercept (i.e. a vector of ones) on the right. Missing values should be denoted by NA and the corresponding samples will be removed from analysis. Other values (e.g. -9, -999) will be treated as ordinary numeric values in analysis.
data	a data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
subset	a specification of the rows to be used: defaults to all rows. This can be any valid indexing vector for the rows of data or if that is not supplied, a data frame made up of the variables used in formula.
subjData	a character vector of subject IDs. Its order should be the same as the subject order in the formula and data (before subset process).
method	a character: "SPACox" (check GRAB.SPACox), "POLMM" (check GRAB.POLMM), "SPAGE" (will be supported later), or "GATE" (will be supported later).
traitType	a character: "binary", "ordinal" (check GRAB.POLMM), "quantitative", or "time-to-event" (check GRAB.SPACox).
GenoFile	a character of genotype file. Currently, two types of genotype formats are supported: PLINK and BGEN. Check GRAB.ReadGeno for more details.
GenoFileIndex	additional index files corresponding to the GenoFile. If NULL (default), the same prefix as GenoFile is used. Check GRAB.ReadGeno for more details.
SparseGRMFile	a character of sparseGRM file. An example is system.file("SparseGRM","SparseGRM.txt",package=
control	a list of parameters for controlling the model fitting process. For more details, please check Details section.
...	other arguments passed to or from other methods.

Details

GRAB package uses score testing which consists of two steps. In Step 1, function GRAB.NullModel fits a null model including response variable, covariates, and Genetic Relationship Matrix (GRM) if needed. In Step 2, functions GRAB.Marker and GRAB.Region perform genome-wide marker-level analysis and region-level analysis, respectively. Step 1 fits a null model to get an R object, which is passed to Step 2 for association testing. Functions of save and load can save and load the object.

GRAB package includes multiple methods which support a wide variety of phenotypes as follows.

- POLMM: Support traitType = "ordinal". Check GRAB.POLMM for more details.
- SPACox: Support traitType = "time-to-event" or "Residual". Check GRAB.SPACox for more details.
- SPAmix: Support traitType = "time-to-event" or "Residual". Check GRAB.SPAmix for more details.
- SPAGRM: Support traitType = "time-to-event" or "Residual". Check GRAB.SPAGRM for more details.

GRAB package supports both Dense and Sparse GRM to adjust for sample relatedness. If Dense GRM is used, then GenoFile is required to construct GRM. If Sparse GRM is used, then SparseGRMFile is required, whose details can be seen in getTempFilesFullGRM and getSparseGRM.

The following details are about argument control:

Argument control includes a list of parameters for controlling the null model fitting process.

- maxiter: Maximum number of iterations used to fit the null model. *(default=100)*
- seed: An integer as a random seed. Used when random process is involved. *(default=12345678)*
- tolBeta: Positive tolerance: the iterations converge when |beta - beta_old| / (|beta| + |beta_old| + tolBeta) < tolBeta. *(default=0.001)*
- showInfo: Whether to show more detailed information for trouble shooting. *(default=FALSE)*

To adjust for sample relatedness, mixed effect model incorporates a random effect with a variance component. Argument control includes additional parameters to estimate the variance component.

- tau: Initial value of the variance component (tau). *(default=0.2)*.
- tolTau: Positive tolerance: the iterations converge when |tau - tau_old| / (|tau| + |tau_old| + tolTau) < tolTau. *(default=0.002)*

If dense GRM is used to adjust for sample relatedness, GenoFile should be PLINK files and argument control includes additional parameters as follows.

- maxiterPCG: Maximum number of iterations for PCG to converge. *(default=100)*
- tolEps: Positive tolerance for PCG to converge. *(default=1e-6)*
- minMafVarRatio: Minimal value of MAF cutoff to select markers (from PLINK files) to estimate variance ratio. *(default=0.1)*
- maxMissingVarRatio: Maximal value of missing rate cutoff to select markers (from PLINK files) to estimate variance ratio. *(default=0.1)*
- nSNPsVarRatio: Initial number of the selected markers to estimate variance ratio *(default=20)* the number will be automatically added by 10 until the coefficient of variantion (CV) of the variance ratio estimate is below CVcutoff.
- CVcutoff: Minimal cutoff of coefficient of variantion (CV) to estimate variance ratio *(default=0.0025)*
- LOCO: Whether to apply the leave-one-chromosome-out (LOCO) approach. *(default=TRUE)*
- stackSize: Stack size (in bytes) to use for worker threads. For more details, check setThreadOptions. *(default="auto")*
- grainSize: Grain size of a parallel algorithm sets a minimum chunk size for parallelization. In other words, at what point to stop processing input on separate threads. *(default=1)*
- minMafGRM: Minimal value of MAF cutoff to select markers (from PLINK files) to construct dense GRM. *(default=0.01)*
- memoryChunk: Size (Gb) for each memory chunk when reading in PLINK files. *(default=2)*
- tracenrun: Number of runs for trace estimator. *(default=30)*
- maxMissingGRM: Maximal value of missing rate to select markers (from PLINK files) to construct dense GRM. *(default=0.1)*
- onlyCheckTime: Not fit the null model, only check the computation time of reading PLINK files and running 30 KinbVec() functions. *(default=FALSE)*

Value

an R object with a class of "XXXXX_NULL_Model" in which XXXXX is the 'method' used in analysis. The following elements are required for all methods.

- N: Sample size in analysis
- yVec: Phenotype data
- beta: Coefficient parameters corresponding to covariates
- subjData: Subject IDs in analysis
- sessionInfo: Version information about R, the OS and attached or loaded packages.
- Call: A call in which all of the specified arguments are specified by their full names.
- time: The time when analysis is finished
- control: The R list of control in null model fitting

Examples

```
# For POLMM method (ordinal categorical data analysis while adjusting for sample relatedness)
# Step 1(a): fit a null model using a dense GRM (recommand using Linux OS)
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- read.table(PhenoFile, header = TRUE)
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")

# Limit threads for CRAN checks (optional for users).
Sys.setenv(RCPP_PARALLEL_NUM_THREADS = 2)

obj.POLMM <- GRAB.NullModel(
  factor(OrdinalPheno) ~ AGE + GENDER,
  data = PhenoData,
  subjData = IID,
  method = "POLMM",
  traitType = "ordinal",
  GenoFile = GenoFile,
  control = list(showInfo = FALSE, LOCO = FALSE, tolTau = 0.2, tolBeta = 0.1)
)

names(obj.POLMM)
obj.POLMM$tau

# Step 1(b): fit a null model using a sparse GRM (recommand using Linux OS)
# First use getSparseGRM() function to get a sparse GRM file
PhenoData <- read.table(PhenoFile, header = TRUE)
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")

obj.POLMM <- GRAB.NullModel(
  factor(OrdinalPheno) ~ AGE + GENDER,
  data = PhenoData,
  subjData = IID,
  method = "POLMM",
  traitType = "ordinal",
```

```
  GenoFile = GenoFile,
  SparseGRMFile = SparseGRMFile,
  control = list(showInfo = FALSE, LOCO = FALSE, tolTau = 0.2, tolBeta = 0.1)
)

names(obj.POLMM)
obj.POLMM$tau

# save(obj.POLMM, "obj.POLMM.RData")  # save the object for analysis in step 2

# For SPACox method, check ?GRAB.SPACox.
# For SPAmix method, check ?GRAB.SPAmix.
# For SPAGRM method, check ?GRAB.SPAGRM
# For WtCoxG method, check ?GRAB.WtCoxG
```

GRAB.POLMM *POLMM method in GRAB package*

Description

POLMM method is to analyze ordinal categorical data for related samples in a large-scale biobank.

Usage

```
GRAB.POLMM()
```

Details

Please check ?GRAB.control for the generic list of control in GRAB.NullModel() and GRAB.Marker().

Additional list of control in GRAB.NullModel() function Additional list of control in GRAB.Marker() function Additional list of control in GRAB.Region() function

Value

No return value, called for side effects (prints information about the POLMM method to the console).

Examples

```
### First, Read Data and Convert Phenotype to a Factor
library(dplyr)
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
PhenoData <- PhenoData %>% mutate(OrdinalPheno = factor(OrdinalPheno,
  levels = c(0, 1, 2)
))

### Step 1: Fit a null model
```

```
# If a sparse GRM is used in model fitting, SparseGRMFile is required.
# If SparseGRMFile isn't provided, GRAB.NullModel() will calculate dense GRM from GenoFile.

SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
obj.POLMM <- GRAB.NullModel(
  formula = OrdinalPheno ~ AGE + GENDER,
  data = PhenoData,
  subjData = PhenoData$IID,
  method = "POLMM",
  traitType = "ordinal",
  GenoFile = GenoFile,
  SparseGRMFile = SparseGRMFile,
  control = list(
    showInfo = FALSE,
    LOCO = FALSE,
    tolTau = 0.2,
    tolBeta = 0.1
  )
)

objPOLMMFile <- system.file("results", "objPOLMMFile.RData", package = "GRAB")
save(obj.POLMM, file = objPOLMMFile)


### Step 2(a): Single-variant tests using POLMM
objPOLMMFile <- system.file("results", "objPOLMMFile.RData", package = "GRAB")
load(objPOLMMFile) # read in an R object of "obj.POLMM"

GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/simuMarkerOutput.txt")
GRAB.Marker(obj.POLMM,
  GenoFile = GenoFile,
  OutputFile = OutputFile
)

results <- data.table::fread(OutputFile)
hist(results$Pvalue)


### Step 2(b): Set-based tests using POLMM-GENE
objPOLMMFile <- system.file("results", "objPOLMMFile.RData", package = "GRAB")
load(objPOLMMFile) # read in an R object of "obj.POLMM"

GenoFile <- system.file("extdata", "simuPLINK_RV.bed", package = "GRAB")
OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/simuRegionOutput.txt")
GroupFile <- system.file("extdata", "simuPLINK_RV.group", package = "GRAB")
SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")

## make sure the output files does not exist at first
file.remove(OutputFile)
```

```
file.remove(paste0(OutputFile, ".markerInfo"))
file.remove(paste0(OutputFile, ".index"))

GRAB.Region(
  objNull = obj.POLMM,
  GenoFile = GenoFile,
  GenoFileIndex = NULL,
  OutputFile = OutputFile,
  OutputFileIndex = NULL,
  GroupFile = GroupFile,
  SparseGRMFile = SparseGRMFile,
  MaxMAFVec = "0.01,0.005"
)

data.table::fread(OutputFile)
```

GRAB.ReadGeno *Read in genotype data*

Description

GRAB package provides functions to read in genotype data. Currently, we support genotype formats of PLINK and BGEN. Other formats such as VCF will be added later.

Usage

```
GRAB.ReadGeno(
  GenoFile,
  GenoFileIndex = NULL,
  SampleIDs = NULL,
  control = NULL,
  sparse = FALSE
)
```

Arguments

GenoFile	a character of genotype file. See Details section for more details.
GenoFileIndex	additional index file(s) corresponding to GenoFile. See Details section for more details.
SampleIDs	a character vector of sample IDs to extract. The default is NULL, that is, all samples in GenoFile will be extracted.
control	a list of parameters to decide which markers to extract. See Details section for more details.
sparse	a logical value *(default: FALSE)* to indicate if the output of genotype matrix is sparse.

Details

Details about `GenoFile` and `GenoFileIndex`:

Currently, we support two formats of genotype input including PLINK and BGEN. Other formats such as VCF will be added later. Users do not need to specify the genotype format, `GRAB` package will check the extension of the file name for that purpose. If `GenoFileIndex` is not specified, `GRAB` package assumes the prefix is the same as `GenoFile`.

PLINK format: Check link for more details about this format

- `GenoFile`: "prefix.bed". The full file name (including the extension ".bed") of the PLINK binary `bed` file.
- `GenoFileIndex`: c("prefix.bim", "prefix.fam"). If not specified, `GRAB` package assumes that `bim` and `fam` files have the same prefix as the `bed` file.

BGEN format: Check link for more details about this format. Currently, only version 1.2 with 8 bits suppression is supported

- `GenoFile`: "prefix.bgen". The full file name (including the extension ".bgen") of the BGEN binary `bgen` file.
- `GenoFileIndex`: "prefix.bgen.bgi" or c("prefix.bgen.bgi", "prefix.sample"). If not specified, `GRAB` package assumes that `bgi` and `sample` files have the same prefix as the `bgen` file. If only one element is given for `GenoFileIndex`, then it should be a `bgi` file. Check link for more details about `bgi` file.
- If the `bgen` file does not include sample identifiers, then `sample` file is required, whose detailed description can ben seen in link. If you are not sure if sample identifiers are in BGEN file, please refer to `checkIfSampleIDsExist`.

VCF format: will be supported later. `GenoFile`: "prefix.vcf"; `GenoFileIndex`: "prefix.vcf.tbi"

Details about argument `control`:

Argument `control` is used to include and exclude markers for function `GRAB.ReadGeno`. The function supports two include files of (`IDsToIncludeFile`, `RangesToIncludeFile`) and two exclude files of (`IDsToExcludeFile`, `RangesToExcludeFile`), but does not support both include and exclude files at the same time.

- `IDsToIncludeFile`: a file of marker IDs to include, one column (no header). Check `system.file("extdata", "IDsToInclude.txt", package = "GRAB")` for an example.
- `IDsToExcludeFile`: a file of marker IDs to exclude, one column (no header).
- `RangesToIncludeFile`: a file of ranges to include, three columns (no headers): chromosome, start position, end position. Check `system.file("extdata", "RangesToInclude.txt", package = "GRAB")` for an example.
- `RangesToExcludeFile`: a file of ranges to exclude, three columns (no headers): chromosome, start position, end position.
- `AlleleOrder`: a character, "ref-first" or "alt-first", to determine whether the REF/major allele should appear first or second. Default is "alt-first" for PLINK and "ref-first" for BGEN. If the ALT allele frequencies of most markers are > 0.5, you should consider resetting this option. NOTE, if you use plink2 to convert PLINK file to BGEN file, then 'ref-first' modifier is to reset the order.
- `AllMarkers`: a logical value (default: FALSE) to indicate if all markers are extracted. It might take too much memory to put genotype of all markers in R. This parameter is to remind users.

- ImputeMethod: a character, "none" (default), "bestguess", or "mean". By default, missing genotype is NA. Suppose alternative allele frequency is p, then missing genotype is imputed as 2p (ImputeMethod = "mean") or round(2p) (ImputeMethod = "bestguess").

Value

An R list including a genotype matrix and an information matrix.

- GenoMat: Genotype matrix, each row is for one sample and each column is for one marker.
- markerInfo: Information matrix including 5 columns of CHROM, POS, ID, REF, and ALT.

Examples

```
## Raw genotype data
RawFile <- system.file("extdata", "simuRAW.raw.gz", package = "GRAB")
GenoMat <- data.table::fread(RawFile)
GenoMat[1:10, 1:10]

## PLINK files
PLINKFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
# If include/exclude files are not specified, then control$AllMarker should be TRUE
GenoList <- GRAB.ReadGeno(PLINKFile, control = list(AllMarkers = TRUE))
GenoMat <- GenoList$GenoMat
markerInfo <- GenoList$markerInfo
head(GenoMat[, 1:6])
head(markerInfo)

## BGEN files (Note the different REF/ALT order for BGEN and PLINK formats)
BGENFile <- system.file("extdata", "simuBGEN.bgen", package = "GRAB")
GenoList <- GRAB.ReadGeno(BGENFile, control = list(AllMarkers = TRUE))
GenoMat <- GenoList$GenoMat
markerInfo <- GenoList$markerInfo
head(GenoMat[, 1:6])
head(markerInfo)

## The below is to demonstrate parameters in control
PLINKFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
IDsToIncludeFile <- system.file("extdata", "simuGENO.IDsToInclude", package = "GRAB")
RangesToIncludeFile <- system.file("extdata", "RangesToInclude.txt", package = "GRAB")
GenoList <- GRAB.ReadGeno(PLINKFile,
  control = list(
    IDsToIncludeFile = IDsToIncludeFile,
    RangesToIncludeFile = RangesToIncludeFile,
    AlleleOrder = "ref-first"
  )
)
GenoMat <- GenoList$GenoMat
head(GenoMat)
markerInfo <- GenoList$markerInfo
head(markerInfo)

## The below is for PLINK/BGEN files with missing data
```

```
PLINKFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
GenoList <- GRAB.ReadGeno(PLINKFile, control = list(AllMarkers = TRUE))
head(GenoList$GenoMat)

GenoList <- GRAB.ReadGeno(PLINKFile, control = list(AllMarkers = TRUE, ImputeMethod = "mean"))
head(GenoList$GenoMat)

BGENFile <- system.file("extdata", "simuBGEN.bgen", package = "GRAB")
GenoList <- GRAB.ReadGeno(BGENFile, control = list(AllMarkers = TRUE))
head(GenoList$GenoMat)
```

`GRAB.Region` *Conduct region-level genetic association testing*

Description

Test for association between phenotype of interest and regions including multiple genetic marker (mostly low-frequency or rare variants).

Usage

```
GRAB.Region(
  objNull,
  GenoFile,
  GenoFileIndex = NULL,
  OutputFile,
  OutputFileIndex = NULL,
  GroupFile,
  SparseGRMFile = NULL,
  SampleFile = NULL,
  MaxMAFVec = "0.01,0.001,0.0005",
  annoVec = "lof,lof:missense,lof:missense:synonymous",
  chrom = "LOCO=F",
  control = NULL
)
```

Arguments

`objNull`	the output object of function `GRAB.NullModel`.
`GenoFile`	a character of genotype file. Currently, two types of genotype formats are supported: PLINK and BGEN. Check `GRAB.ReadGeno` for more details.
`GenoFileIndex`	additional index files corresponding to the `GenoFile`. If `NULL` (default), the prefix is the same as GenoFile. Check `GRAB.ReadGeno` for more details.
`OutputFile`	a character of output file to save the analysis results.

`OutputFileIndex`
: a character of output index file to record the end point. If the program ends unexpectedly, the end point can help GRAB package understand where to restart the analysis. If NULL (default), `OutputFileIndex = paste0(OutputFile, ".index")`.

`GroupFile`
: a character of region file to specify region-marker mapping with annotation information. Each region includes two or three rows. Only alphabet, numbers, and `:,_+-` symbols are supported. Columns are separated by 'tab'.

`SparseGRMFile`
: a character of sparseGRM file. An example is `system.file("SparseGRM","SparseGRM.txt",package=`

`SampleFile`
: a character of file to include sample information with header.

`MaxMAFVec`
: a character of multiple max MAF cutoffs (comma separated) to include markers for region-level analysis. Default value is `"0.05,0.01,0.005"`.

`annoVec`
: a character of multiple annotation groups (comma separated) to include markers for region-level analysis. Default value is `"lof,lof:missense,lof:missense:synonymous"`.

`chrom`
: to be continued

`control`
: a list of parameters for controlling function `GRAB.Region`, more details can be seen in `Details` section.

Details

GRAB package supports POLMM, SPACox, SPAGRM, SPAmix, and WtCoxG methods. Detailed information about the analysis methods is given in the `Details` section of `GRAB.NullModel`. Users do not need to specify them since functions `GRAB.Marker` and `GRAB.Region` will check the `class(objNull)`.

The following details are about argument `control`:

For PLINK files, the default `control$AlleleOrder = "alt-first"`; for BGEN files, the default `control$AlleleOrder = "ref-first"`.

- `AlleleOrder`: please refer to the `Details` section of `GRAB.ReadGeno`.

The below is to customize the quality-control (QC) process.

- `omp_num_threads`: (To be added later) a numeric value (default: value from data.table::getDTthreads()) to specify the number of threads in OpenMP for parallel computation.
- `ImputeMethod`: a character, "mean", "bestguess" (default), or "drop" (to be added later). Please refer to the `Details` section of `GRAB.ReadGeno`.
- `MissingRateCutoff`: a numeric value *(default=0.15)*. Markers with missing rate > this value will be excluded from analysis.
- `MinMACCutoff`: a numeric value *(default=5)*. Markers with MAC < this value will be treated as Ultra-Rare Variants (URV) and collapsed as one value.
- `nRegionsEachChunk`: number of regions *(default=1)* in one chunk to output.

The below is for kernel-based approaches including SKAT and SKAT-O. For more details, please refer to the SKAT package.

- `kernel`: a type of kernel *(default="linear.weighted")*.
- `weights_beta`: a numeric vector of parameters for the beta weights for the weighted kernels *(default=c(1, 25))*. If you want to use your own weights, please use the `control$weights` parameter. It will be ignored if `control$weights` parameter is not NULL.
- `weights`: a numeric vector of weights for the weighted kernels. If it is NULL (default), the beta weight with the `control$weights.beta` parameter is used.

- `r.corr`: the rho parameter for the compound symmetric correlation structure kernels. If you give a vector value, SKAT will conduct the optimal test. It will be ignored if method="optimal" or method="optimal.adj" *(default=c(0, 0.1^2, 0.2^2, 0.3^2, 0.4^2, 0.5^2, 0.5, 1))*.

The below is to customize the columns in the `OutputMarkerFile`. Columns of `Marker`, `Info`, `AltFreq`, `AltCounts`, `MissingRate`, `Pvalue` are included for all methods.

- `outputColumns`: For example, for POLMM method, users can set `control$outputColumns = c("beta", "seBeta", "AltFreqInGroup")`:
  - `POLMM`: Default: `beta`, `seBeta`; Optional: `zScore`, `AltFreqInGroup`, `nSamplesInGroup`, `AltCountsInGroup`
  - `SPACox`: Optional: `zScore`

Value

Region-based analysis results are saved into two files: `OutputFile` and `OutputMarkerFile = paste0(OutputFile, ".markerInfo")`.

The file of `OutputMarkerFile` is the same as the results of `GRAB.Marker`. The file of `OutputFile` includes columns as below.

Region Region IDs from `RegionFile`

Anno.Type Annotation type from `RegionFile`

maxMAF the maximal cutoff of the MAF to select low-frequency/rare variants into analysis.

nSamples Number of samples in analysis.

nMarkers Number of markers whose MAF < `control$MaxMAFCutoff` and MAC > `control$MinMACCutoff`. Markers with annotation value <= 0 will be excluded from analysis.

nMarkersURV Number of Ultra-Rare Variants (URV) whose MAC < `control$MinMACCutoff`. Markers with annotation value <= 0 will be excluded from analysis.

pval.SKATO p-values based on SKAT-O method

pval.SKAT p-values based on SKAT method

pval.Burden p-values based on Burden test

Examples

```
objNullFile <- system.file("results", "objPOLMMFile.RData", package = "GRAB")
load(objNullFile)
class(obj.POLMM) # "POLMM_NULL_Model", that indicates an object from POLMM method.

OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/simuRegionOutput.txt")
GenoFile <- system.file("extdata", "simuPLINK_RV.bed", package = "GRAB")
GroupFile <- system.file("extdata", "simuPLINK_RV.group", package = "GRAB")
SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")

## make sure the output files does not exist at first
file.remove(OutputFile)
file.remove(paste0(OutputFile, ".markerInfo"))
file.remove(paste0(OutputFile, ".index"))
```

```
GRAB.Region(
  objNull = obj.POLMM,
  GenoFile = GenoFile,
  GenoFileIndex = NULL,
  OutputFile = OutputFile,
  OutputFileIndex = NULL,
  GroupFile = GroupFile,
  SparseGRMFile = SparseGRMFile,
  MaxMAFVec = "0.01,0.005"
)

data.table::fread(OutputFile)
data.table::fread(paste0(OutputFile, ".markerInfo"))
data.table::fread(paste0(OutputFile, ".otherMarkerInfo"))
data.table::fread(paste0(OutputFile, ".index"), sep = "\t", header = FALSE)

SampleFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
GRAB.Region(
  objNull = obj.POLMM,
  GenoFile = GenoFile,
  GenoFileIndex = NULL,
  OutputFile = OutputFile,
  OutputFileIndex = NULL,
  GroupFile = GroupFile,
  SparseGRMFile = SparseGRMFile,
  SampleFile = SampleFile,
  control = list(SampleLabelCol = "OrdinalPheno")
)

data.table::fread(OutputFile)
data.table::fread(paste0(OutputFile, ".markerInfo"))
data.table::fread(paste0(OutputFile, ".otherMarkerInfo"))
data.table::fread(paste0(OutputFile, ".index"), sep = "\t", header = FALSE)
```

GRAB.SAGELD *SAGELD method in GRAB package*

Description

SAGELD method is Scalable and Accurate algorithm for Gene-Environment interaction analysis using Longitudinal Data for related samples in a large-scale biobank. SAGELD extended SPAGRM to support gene-environment interaction analysis.

Usage

```
GRAB.SAGELD()
```

Details

Additional list of `control` in `SAGELD.NullModel()` function.

Additional list of `control` in `GRAB.Marker()` function.

Value

No return value, called for side effects (prints information about the SAGELD method to the console).

`GRAB.SimubVec` *GRAB: simulate random effect (i.e. bVec) based on family structure*

Description

Simulate random effect (i.e. bVec) based on family structure

Usage

```
GRAB.SimubVec(nSub, nFam, FamMode, tau)
```

Arguments

`nSub`	the number of unrelated subjects in simulations, if `nSub = 0`, then all subjects are related to at least one of the others.
`nFam`	the number of families in simulation, if `nFam = 0`, then all subjects are unrelated to each other.
`FamMode`	`"4-members"`, `"10-members"`, or `"20-members"`. Check `Details` section of function `help(GRAB.SimuGMat)` for more details.
`tau`	variance component

Value

a data frame including two columns: ID and random effect following a multivariate normal distribution

Examples

```
nSub <- 10
nFam <- 1
FamMode <- "10-members"
tau <- 2
bVec <- GRAB.SimubVec(nSub, nFam, FamMode, tau)
```

`GRAB.SimuGMat` *Simulate an R matrix of genotype data*

Description

`GRAB` package provides functions to simulate genotype data. We support simulations based on unrelated subjects and related subjects.

Usage

```
GRAB.SimuGMat(
  nSub,
  nFam,
  FamMode,
  nSNP,
  MaxMAF = 0.5,
  MinMAF = 0.05,
  MAF = NULL
)
```

Arguments

`nSub`	the number of unrelated subjects in simulations, if `nSub = 0`, then all subjects are related to at least one of the others.
`nFam`	the number of families in simulation, if `nFam = 0`, then all subjects are unrelated to each other.
`FamMode`	`"4-members"`, `"10-members"`, or `"20-members"`. Check `Details` section for more details.
`nSNP`	number of markers to simulate
`MaxMAF`	a numeric value *(default=0.5)*, haplotype is simulated with allele frequency <= this value.
`MinMAF`	a numeric value *(default=0.05)*, haplotype is simulated with allele frequency >= this value.
`MAF`	a numeric vector with a length of *nSNP*. If this argument is given, then arguments of *MaxMAF* and *MinMAF* would be ignored.

Details

Currently, function `GRAB.SimuGMat` supports both unrelated and related subjects. Genotype data is simulated following Hardy-Weinberg Equilibrium with allele frequency ~ `runif(MinMAF, MaxMAF)`.

If `FamMode = "4-members"`**:**

Total number of subjects is `nSub + 4 * nFam`. Each family includes 4 members with the family structure as below: 1+2->3+4.

If `FamMode = "10-members"`**:**

Total number of subjects is `nSub + 10 * nFam`. Each family includes 10 members with the family structure as below: 1+2->5+6, 3+5->7+8, 4+6->9+10.

If `FamMode = "20-members"`**:**

Total number of subjects is `nSub + 20 * nFam`. Each family includes 20 members with the family structure as below: 1+2->9+10, 3+9->11+12, 4+10->13+14, 5+11->15+16, 6+12->17, 7+13->18, 8+14->19+20.

Value

an R list including genotype matrix and marker information

- `GenoMat` a numeric matrix of genotype: each row is for one subject and each column is for one SNP
- `markerInfo` a data frame with the following 2 columns: SNP ID and minor allele frequency

See Also

`GRAB.makePlink` can make `PLINK` files using the genotype matrix.

Examples

```
nSub <- 100
nFam <- 10
FamMode <- "10-members"
nSNP <- 10000
OutList <- GRAB.SimuGMat(nSub, nFam, FamMode, nSNP)
GenoMat <- OutList$GenoMat
markerInfo <- OutList$markerInfo
GenoMat[1:10, 1:10]
head(markerInfo)

## The following is to calculate GRM
MAF <- apply(GenoMat, 2, mean) / 2
GenoMatSD <- t((t(GenoMat) - 2 * MAF) / sqrt(2 * MAF * (1 - MAF)))
GRM <- GenoMatSD %*% t(GenoMatSD) / ncol(GenoMat)
GRM1 <- GRM[1:10, 1:10]
GRM2 <- GRM[100 + 1:10, 100 + 1:10]
GRM1
GRM2
```

`GRAB.SimuGMatFromGenoFile`

GRAB: simulate genotype matrix based on family structure

Description

Simulate genotype matrix based on family structure using haplotype information from genotype files. This function is mainly to simulate genotype data for rare variants analysis. NOTE: if simulating related subjects, the genotype of two allele will be assigned to two haplotypes of one allele randomly.

Usage

```
GRAB.SimuGMatFromGenoFile(
  nFam,
  nSub,
  FamMode,
  GenoFile,
  GenoFileIndex = NULL,
  SampleIDs = NULL,
  control = NULL
)
```

Arguments

nFam	number of families in simulation
nSub	number of unrelated subjects in simulation
FamMode	"4-members", "10-members", or "20-members". Check Details section for more details.
GenoFile	this parameter is passed to GRAB.ReadGeno to read in genotype data.
GenoFileIndex	this parameter is passed to GRAB.ReadGeno to read in genotype data.
SampleIDs	this parameter is passed to GRAB.ReadGeno to read in genotype data.
control	this parameter is passed to GRAB.ReadGeno to read in genotype data.

Details

Currently, function GRAB.SimuGMatFromGenoFile supports both unrelated and related subjects. Genotype data of founders is from GenoFile and GenoFileIndex.

If FamMode = "4-members"**:**

Total number of subjects is nSub + 4 * nFam. Each family includes 4 members with the family structure as below: 1+2->3+4.

If FamMode = "10-members"**:**

Total number of subjects is nSub + 10 * nFam. Each family includes 10 members with the family structure as below: 1+2->5+6, 3+5->7+8, 4+6->9+10.

If FamMode = "20-members"**:**

Total number of subjects is nSub + 20 * nFam. Each family includes 20 members with the family structure as below: 1+2->9+10, 3+9->11+12, 4+10->13+14, 5+11->15+16, 6+12->17, 7+13->18, 8+14->19+20.

Value

a genotype matrix of genotype data

Examples

```
nFam <- 50
nSub <- 500
FamMode <- "10-members"

# PLINK data format. Currently, this function does not support BGEN data format.
PLINKFile <- system.file("extdata", "example_n1000_m236.bed", package = "GRAB")
IDsToIncludeFile <- system.file("extdata", "example_n1000_m236.IDsToInclude", package = "GRAB")

GenoList <- GRAB.SimuGMatFromGenoFile(nFam, nSub, FamMode, PLINKFile,
  control = list(IDsToIncludeFile = IDsToIncludeFile)
)
```

GRAB.SimuPheno *Simulate phenotype using linear predictor* eta

---

Description

GRAB package can help simulate a wide variaty of phenotypes

Usage

```
GRAB.SimuPheno(
  eta,
  traitType = "binary",
 control = list(pCase = 0.1, sdError = 1, pEachGroup = c(1, 1, 1), eventRate = 0.1),
  seed = NULL
)
```

Arguments

eta	linear predictors, usually covar x beta.covar + genotype x beta.genotype
traitType	"quantitative", "binary", "ordinal", or "time-to-event"
control	a list of parameters for controlling the simulation process
seed	a random number seed for reproducibility

Details

Check https://wenjianbi.github.io//grab.github.io/docs/simulation_phenotype.html for more details.

Value

a numeric vector of phenotype

GRAB.SPACox *SPACox method in GRAB package*

Description

SPACox method is an empirical approach to analyzing complex traits (including but not limited to time-to-event trait) for unrelated samples in a large-scale biobank.

Usage

```
GRAB.SPACox()
```

Details

Additional list of `control` in `GRAB.NullModel()` function.

Additional list of `control` in `GRAB.Marker()` function.

Value

No return value, called for side effects (prints information about the SPACox method to the console).

Examples

```
# Step 1: fit a null model
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
obj.SPACox <- GRAB.NullModel(survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  subjData = IID,
  method = "SPACox",
  traitType = "time-to-event"
)

# Using model residuals performs exactly the same as the above. Note that
# confounding factors are still required in the right of the formula.
obj.coxph <- survival::coxph(survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  x = TRUE
)
obj.SPACox <- GRAB.NullModel(obj.coxph$residuals ~ AGE + GENDER,
  data = PhenoData,
  subjData = IID,
  method = "SPACox",
  traitType = "Residual"
)

# Step 2: conduct score test
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputDir <- system.file("results", package = "GRAB")
```

```
OutputFile <- paste0(OutputDir, "/Results_SPACox.txt")
GRAB.Marker(obj.SPACox,
  GenoFile = GenoFile, OutputFile = OutputFile,
  control = list(outputColumns = "zScore")
)
data.table::fread(OutputFile)
```

GRAB.SPAGRM *SPAGRM method in GRAB package*

Description

SPAGRM method is an empirical approach to analyzing complex traits (including but not limited to longitudinal trait) for related samples in a large-scale biobank. SPAGRM extend SPACox to support an related populations.

Usage

```
GRAB.SPAGRM()
```

Details

Additional list of `control` in `SPAGRM.NullModel()` function.

Additional list of `control` in `GRAB.Marker()` function.

Value

No return value, called for side effects (prints information about the SPAGRM method to the console).

Examples

```
# Step 2a: process model residuals
ResidMatFile <- system.file("extdata", "ResidMat.txt", package = "GRAB")
SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")
PairwiseIBDFile <- system.file("PairwiseIBD", "PairwiseIBD.txt", package = "GRAB")
obj.SPAGRM <- SPAGRM.NullModel(
  ResidMatFile = ResidMatFile,
  SparseGRMFile = SparseGRMFile,
  PairwiseIBDFile = PairwiseIBDFile,
  control = list(ControlOutlier = FALSE)
)

# Step 2b: perform score test
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/SPAGRMMarkers.txt")
GRAB.Marker(
  objNull = obj.SPAGRM,
```

```
  GenoFile = GenoFile,
  OutputFile = OutputFile
)
head(read.table(OutputFile, header = TRUE))
```

GRAB.SPAmix *SPAmix method in GRAB package*

Description

SPAmix method is an empirical approach to analyzing complex traits (including but not limited to time-to-event trait) for unrelated samples in a large-scale biobank. SPAmix extend SPACox to support an admixture population or multiple populations.

Usage

```
GRAB.SPAmix()
```

Details

For `SPAmix`, the confounding factors of SNP-derived PCs are required and should be specified in `control`.

Value

No return value, called for side effects (prints information about the SPAmix method to the console).

Examples

```
# Step 1: fit a null model
library(dplyr)
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
N <- nrow(PhenoData)
PhenoData <- PhenoData %>% mutate(PC1 = rnorm(N), PC2 = rnorm(N))
# add two PCs, which are required for SPAmix

# Users can directly specify a time-to-event trait to analyze
obj.SPAmix <- GRAB.NullModel(survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData,
  subjData = IID,
  method = "SPAmix",
  traitType = "time-to-event",
  control = list(PC_columns = "PC1,PC2")
)

# Using model residuals performs exactly the same as the above. Note that
# confounding factors are still required in the right of the formula.
obj.coxph <- survival::coxph(survival::Surv(SurvTime, SurvEvent) ~
  AGE + GENDER + PC1 + PC2, data = PhenoData)
```

```
obj.SPAmix <- GRAB.NullModel(obj.coxph$residuals ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData,
  subjData = IID,
  method = "SPAmix",
  traitType = "Residual",
  control = list(PC_columns = "PC1,PC2")
)

# SPAmix also supports multiple residuals as below
obj.coxph <- survival::coxph(survival::Surv(SurvTime, SurvEvent) ~
  AGE + GENDER + PC1 + PC2, data = PhenoData)
obj.lm <- lm(QuantPheno ~ AGE + GENDER + PC1 + PC2, data = PhenoData)
obj.SPAmix <- GRAB.NullModel(obj.coxph$residuals + obj.lm$residuals ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData,
  subjData = IID,
  method = "SPAmix",
  traitType = "Residual",
  control = list(PC_columns = "PC1,PC2")
)

# Step 2: conduct score test
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputDir <- system.file("results", package = "GRAB")
OutputFile <- paste0(OutputDir, "/Results_SPAmix.txt")
GRAB.Marker(obj.SPAmix,
  GenoFile = GenoFile, OutputFile = OutputFile,
  control = list(outputColumns = "zScore")
)
data.table::fread(OutputFile)
```

`GRAB.WtCoxG` *WtCoxG method in GRAB package*

Description

WtCoxG is an accurate, powerful, and computationally efficient Cox-based approach to perform genome-wide time-to-event data analyses in study cohorts with case ascertainment.

Usage

```
GRAB.WtCoxG()
```

Details

Additional arguments in `GRAB.NullModel()`:

- `RefAfFile`: A character string specifying a reference allele frequency file, which is a csv file (with a header) and includes columns of `CHROM`, `POS`, `ID`, `REF`, `ALT`, `AF_ref`, and `AN_ref`.
- `OutputFile`: A character string specifying the output file name.

- SampleIDColumn: A character string specifying the column name in the input data that contains sample IDs.
- SurvTimeColumn: A character string specifying the column name in the input data that contains survival time information.
- IndicatorColumn: A character string specifying the column name in the input data that indicates case-control status (should be 0 for controls and 1 for cases).

Additional arguments in list control in GRAB.NullModel():

- RefPrevalence: A numeric value specifying the population-level disease prevalence used for weighting in the analysis.
- SNPnum: Minimum number of SNPs. Default is 1e4.

Additional arguments in list control in GRAB.Marker():

- cutoff: A numeric value specifying the batch effect p-value cutoff for method selection of an assocition test. Default is 0.1.

Value

No return value, called for side effects (prints information about the WtCoxG method to the console).

Examples

```
# Step0&1: fit a null model and estimate parameters according to batch effect p-values
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
SparseGRMFile <- system.file("SparseGRM", "SparseGRM.txt", package = "GRAB")

GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
RefAfFile <- system.file("extdata", "simuRefAf.txt", package = "GRAB")
RefPrevalence <- 0.1 # population-level disease prevalence

OutputDir <- system.file("results", package = "GRAB")
OutputStep1 <- paste0(OutputDir, "/WtCoxG_step1_out.txt")
OutputStep2 <- paste0(OutputDir, "/WtCoxG_step2_out.txt")

obj.WtCoxG <- GRAB.NullModel(
  formula = survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  subjData = PhenoData$IID,
  method = "WtCoxG",
  traitType = "time-to-event",
  GenoFile = GenoFile,
  SparseGRMFile = SparseGRMFile,
  control = list(
    AlleleOrder = "ref-first", AllMarkers = TRUE, RefPrevalence = RefPrevalence,
    SNPnum = 1000
  ), # minimum number of SNPs for to call TestforBatchEffect
  RefAfFile = RefAfFile,
  OutputFile = OutputStep1,
```

```
  SampleIDColumn = "IID",
  SurvTimeColumn = "SurvTime",
  IndicatorColumn = "SurvEvent"
)

resultStep1 <- data.table::fread(OutputStep1)
resultStep1[, c("CHROM", "POS", "pvalue_bat")]

# Step2: conduct association testing
GRAB.Marker(
  objNull = obj.WtCoxG,
  GenoFile = GenoFile,
  OutputFile = OutputStep2,
  control = list(
    AlleleOrder = "ref-first", AllMarkers = TRUE,
    cutoff = 0.1, nMarkersEachChunk = 5000
  )
)

resultStep2 <- data.table::fread(OutputStep2)
resultStep2[, c("CHROM", "POS", "WtCoxG.noext", "WtCoxG.ext")]
```

`handleFormula` *handle a formula (used in GRAB.NullModel function)*

Description

handle a formula (used in GRAB.NullModel function), this function can help users better understand the input of GRAB.NullModel() function

Usage

```
handleFormula(formula, data, subset, subjData)
```

Arguments

`formula`	a formula object, with the response on the left of a ~ operator and the covariates on the right. Do not add a column of intercept (e.g. a vector of ones) on the right. Missing values should be denoted by NA and the corresponding samples will be removed from analysis.
`data`	a data.frame in which to interpret the variables named in the formula, or in the subset argument. Check ?model.frame for more details.
`subset`	a specification of the rows to be used: defaults to all rows. This can be any valid indexing vector for the rows of data or if that is not supplied, a data frame made up of the variables used in formula. Check ?model.frame for more details.
`subjData`	a character vector of subject IDs. Its order should be the same as the subjects order in the formula and data.

Value

an R list with elements of 'response', 'designMat', and 'subjData'.

Examples

```
n <- 20
subjData <- paste0("ID-", 1:n)
pheno <- rbinom(n, 1, 0.5)
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rbinom(n, 2, 0.5)
objFormula <- handleFormula(pheno ~ x1 + x2 * x3, subset = x2 > 0, subjData = subjData)
objFormula
```

---

makeGroup *A lower function to make groups based on phenotype*

Description

In functions GRAB.Marker and GRAB.Region, users can get detailed information for each markers in different groups.

Usage

```
makeGroup(yVec)
```

Arguments

yVec	the phenotype recorded in objNull$yVec, the output object of function GRAB.NullModel.

Details

If yVec is categorical with groups <= 10, then Group is the same as yVec. Otherwise, Group is calcualted based on the rank of yVec.

Value

a numeric vector (Group, starting from 0) for group information.

`SAGELD.NullModel` *Fit a SAGELD Null Model*

Description

Fit a SAGELD Null Model

Usage

```
SAGELD.NullModel(
  NullModel,
  UsedMethod = "SAGELD",
  PlinkFile,
  SparseGRMFile,
  PairwiseIBDFile,
  PvalueCutoff = 0.001,
  control = list()
)
```

Arguments

`NullModel`	A fitted null model object from either `lme4::lmer()` or `glmmTMB::glmmTMB()`. This model should include the phenotype, environmental variable, covariates, and random effects structure.
`UsedMethod`	A character string specifying the method to use. Options are "SAGELD" (default) for gene-environment interaction analysis, or "GALLOP" for analysis using only unrelated samples.
`PlinkFile`	A character string specifying the path to PLINK files (without file extensions like ".bed", ".bim", or ".fam"). Used to read genotype data for calculating lambda values in gene-environment interaction models.
`SparseGRMFile`	A character string specifying the path to a sparse genetic relationship matrix (GRM) file. This file should be generated using the `getSparseGRM()` function and contain three columns: 'ID1', 'ID2', and 'Value'.
`PairwiseIBDFile`	A character string specifying the path to a pairwise identity-by-descent (IBD) file. This file should be generated using the `getPairwiseIBD()` function and contain five columns: 'ID1', 'ID2', 'pa', 'pb', and 'pc'.
`PvalueCutoff`	A numeric value (default: 0.001) specifying the p-value cutoff for marginal genetic effect on the environmental variable. Used to filter SNPs when calculating lambda values for gene-environment interaction models.
`control`	A list of control parameters for the null model fitting process. Available options include:

Value

A SAGELD null model object

setDenseGRM *Set up a dense GRM (only for developers)*

Description

Set up a dense GRM (only for developers), other users can ignore this function

Usage

```
setDenseGRM(GenoFile, GenoFileIndex = NULL, subjData = NULL)
```

Arguments

GenoFile	a character of genotype file. Three types of genotype files are supported: PLINK ("prefix.bed"), BGEN ("prefix.bgen"), and VCF ("prefix.vcf" or "prefix.vcf.gz").
GenoFileIndex	additional index files corresponding to the "GenoFile". If Null (default), the same prefix as GenoFile is used. PLINK: c("prefix.bim", "prefix.fam"), BGEN: c("prefix.bgi"), and VCF: c("prefix.vcf.tbi") or c("prefix.vcf.gz.tbi").
subjData	a character vector of subject IDs. Its order should be the same as the subjects order in the formula and data.

Value

no result is returned

Examples

```
# Check ?getDenseGRM() for an example.
```

SPAGRM.NullModel *Fit a SPAGRM Null Model*

Description

Fit a SPAGRM Null Model

Usage

```
SPAGRM.NullModel(
  ResidMatFile,
  SparseGRMFile,
  PairwiseIBDFile,
  control = list(MaxQuantile = 0.75, MinQuantile = 0.25, OutlierRatio = 1.5,
   ControlOutlier = TRUE, MaxNuminFam = 5, MAF_interval = c(1e-04, 5e-04, 0.001, 0.005,
    0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5))
)
```

Arguments

ResidMatFile	A file path (character) or data.frame containing residuals. If a file path, it should point to a tab-delimited file with two columns: 'SubjID' (subject IDs) and 'Resid' (residual values). If a data.frame, it should have the same structure with columns named 'SubjID' and 'Resid'.
SparseGRMFile	A file path (character) to a sparse genetic relationship matrix (GRM) file. This file should be generated using the `getSparseGRM()` function and contain three columns: 'ID1', 'ID2', and 'Value' representing the genetic relationships between pairs of individuals.
PairwiseIBDFile	A file path (character) to a pairwise identity-by-descent (IBD) file. This file should be generated using the `getPairwiseIBD()` function and contain five columns: 'ID1', 'ID2', 'pa', 'pb', and 'pc' representing IBD probabilities between pairs of individuals.
control	A list of control parameters for the null model fitting process. Available options include:

Value

A SPAGRM null model object

TestforBatchEffect *Quality control to check batch effect between study cohort and reference population.*

Description

This function performs quality control to test for the batch effect between a study cohort and a reference population. And fit a weighted null model.

Usage

```
TestforBatchEffect(
  objNull,
  data,
  GenoFile = NULL,
  GenoFileIndex = NULL,
  Geno.mtx = NULL,
  SparseGRMFile = NULL,
  RefAfFile,
  OutputFile,
  IndicatorColumn,
  SurvTimeColumn,
  SampleIDColumn
)
```

Arguments

objNull	a WtCoxG_NULL_Model object, which is the output of GRAB.NullModel.
data	a data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
GenoFile	A character string of the genotype file. See Details section for more details.
GenoFileIndex	Additional index file(s) corresponding to GenoFile. See Details section for more details.
Geno.mtx	A matrix of genotype data. If provided, it will be used instead of GenoFile. The matrix should have samples in rows and markers in columns.
SparseGRMFile	a path to file of output to be passed to GRAB.NullModel.
RefAfFile	A character string of the reference file. The reference file must be a txt file (header required) including at least 7 columns: CHROM, POS, ID, REF, ALT, AF_ref, AN_ref.
OutputFile	A character string of the output file name. The output file will be a txt file.
IndicatorColumn	A character string of the column name in data that indicates the case-control status. The value should be 0 for controls and 1 for cases.
SurvTimeColumn	A character string of the column name in data that indicates the survival time.
SampleIDColumn	A character string of the column name in data that indicates the sample ID.

Value

A dataframe of marker info and reference MAF.

Index