Package: GRAB (via r-universe)

October 29, 2026

Type Package

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

Version 0.2.5

Date 2026-07-03

Description Provides a comprehensive suite of genome-wide association study (GWAS) methods specifically designed for biobank-scale data, including but not limited to, robust approaches for time-to-event traits (Li et al., 2025 <doi:10.1038/s43588-025-00864-z>) and ordinal categorical traits (Bi et al., 2021 <doi:10.1016/j.ajhg.2021.03.019>). The package also offers general frameworks for GWAS of any trait type (Bi et al., 2020 <doi:10.1016/j.ajhg.2020.06.003>), while accounting for sample relatedness (Xu et al., 2025 <doi:10.1038/s41467-025-56669-1>) or population structure (Ma et al., 2025 <doi:10.1186/s13059-025-03827-9>). By accurately approximating score statistic distributions using saddlepoint approximation (SPA), these methods can effectively control type I error rates for rare variants and in the presence of unbalanced phenotype distributions. Additionally, the package 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, rlang

Suggests SKAT, dbplyr, tidyr, R.utils

LinkingTo Rcpp, RcppArmadillo, RcppParallel, BH

RoxygenNote 8.0.0

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 https://cran.r-universe.dev

Date/Publication 2026-07-03 08:30:39 UTC

RemoteUrl https://github.com/cran/GRAB

RemoteRef HEAD

RemoteSha 537b5e0ab534bb3effc8f9a7d9355bd6dca2ca18

Contents

CCT . . . 2
getPairwiseIBD . . . 3
getSparseGRM . . . 4
getTempFilesFullGRM . . . 6
GRAB.getGenoInfo . . . 8
GRAB.makePlink . . . 9
GRAB.Marker . . . 10
GRAB.NullModel . . . 12
GRAB.POLMM . . . 14
GRAB.POLMM.Region . . . 17
GRAB.ReadGeno . . . 19
GRAB.Region . . . 22
GRAB.SAGELD . . . 23
GRAB.SimubVec . . . 24
GRAB.SimuGMat . . . 25
GRAB.SimuGMatFromGenoFile . . . 27
GRAB.SimuPheno . . . 28
GRAB.SPACox . . . 29
GRAB.SPAGRM . . . 31
GRAB.SPAmix . . . 32
GRAB.WtCoxG . . . 34
SAGELD.NullModel . . . 36
SPAGRM.NullModel . . . 38

Index **40**

`CCT` *Cauchy Combination Test for p-value aggregation*

Description

Combines multiple p-values using the Cauchy distribution method, which provides analytical p-value calculation under arbitrary dependency structures.

Usage

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

Arguments

pvals	Numeric vector of p-values to combine (each between 0 and 1). P-values equal to 1 are automatically adjusted to 0.999. P-values equal to 0 will cause an error.
weights	Numeric vector of non-negative weights for each p-value. If NULL, equal weights are used. Must have same length as pvals.

Details

The Cauchy Combination Test (CCT) transforms p-values using the inverse Cauchy distribution and combines them with specified weights. This method is particularly powerful because it:

- Works under arbitrary dependency structures
- Provides exact analytical p-values (no simulation needed)
- Maintains good power properties across different scenarios

Special Cases: - If any p-value equals 0, returns 0 immediately - P-values equal to 1 are adjusted to 0.999 with a warning - Very small p-values (< 1e-16) receive special numerical treatment

Value

Single aggregated p-value combining all input p-values.

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. doi:10.1080/01621459.2018.1554485

Examples

```
# Basic usage with equal weights
pvalues <- c(0.02, 0.0004, 0.2, 0.1, 0.8)
CCT(pvals = pvalues)

# Usage with custom weights
weights <- c(2, 3, 1, 1, 1)
CCT(pvals = pvalues, weights = weights)
```

---

getPairwiseIBD *Calculate Pairwise IBD (Identity By Descent)*

Description

This function calculates pairwise IBD probabilities for related samples using PLINK genotype data and sparse GRM. It follows the getSparseGRM() function workflow.

Usage

```
getPairwiseIBD(
  PlinkPrefix,
  SparseGRMFile,
  PairwiseIBDOutput,
  frqFile = NULL,
  tempDir = NULL,
  maxSampleNums = 2500,
  minMafIBD = 0.01,
  rm.tempFile = FALSE
)
```

Arguments

PlinkPrefix	Character. Path to PLINK file (without file extensions .bed/.bim/.fam).
SparseGRMFile	Character. Path to sparse GRM file from getSparseGRM() function.
PairwiseIBDOutput	Character. Output path to save pairwise IBD results.
frqFile	Character. Path to frequency file corresponding to PLINK file. If NULL (default), uses PlinkPrefix.frq.
tempDir	Character. Directory to save temporary files. If NULL (default), uses tempdir().
maxSampleNums	Integer. Maximum number of subjects' genotypes to read for analysis (default: 2500).
minMafIBD	Numeric. Minimum MAF cutoff to select markers (default: 0.01).
rm.tempFile	Logical. Whether to delete temporary files (default: FALSE).

Value

Character. Message indicating where the pairwise IBD results have been stored.

Examples

```
PlinkPrefix <- file.path(system.file(package = "GRAB"), "extdata", "simuPLINK")
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
PairwiseIBDOutput <- file.path(tempdir(), "PairwiseIBD.txt")
getPairwiseIBD(PlinkPrefix, SparseGRMFile, PairwiseIBDOutput)
```

---

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](https://cnsgenomics.com/software/gcta/#Ov 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(
  PlinkPrefix,
  nPartsGRM,
  SparseGRMFile,
  tempDir = NULL,
  relatednessCutoff = 0.05,
  minMafGRM = 0.01,
  maxMissingGRM = 0.1,
  rm.tempFiles = FALSE
)
```

Arguments

PlinkPrefix	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 tempdir().
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.

```
# 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")

PlinkPrefix = 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 tempdir() by default:

for(partParallel in 1:nPartsGRM) getTempFilesFullGRM(PlinkPrefix, nPartsGRM, partParallel,
gcta64File)

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

tempDir = tempdir()

SparseGRMFile = file.path(tempDir, "SparseGRM.txt")

getSparseGRM(PlinkPrefix, nPartsGRM, SparseGRMFile)
```

Value

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

`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(
  PlinkPrefix,
  nPartsGRM,
  partParallel,
  gcta64File,
  tempDir = NULL,
```

```
  subjData = NULL,
  minMafGRM = 0.01,
  maxMissingGRM = 0.1,
  threadNum = 8
)
```

Arguments

PlinkPrefix	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](https://yanglab.westlake.edu.cn/software/g
tempDir	a path to store temp files to be passed to getSparseGRM. This should be consistent to the input of getSparseGRM. Default is tempdir().
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.
```

`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`	List of control parameters with the following options: • `AlleleOrder`: Allele order in genotype file. Options: "ref-first", "alt-first", or NULL (default: "alt-first" for BGEN, "ref-first" for PLINK). • **Marker Selection:** – `AllMarkers`: Set to TRUE (default) to analyze all markers. Automatically set to FALSE if any include/exclude files are provided. – `IDsToIncludeFile`: Path to file with marker IDs to include. – `RangesToIncludeFile`: Path to file with genomic ranges to include. Can be used with IDsToIncludeFile (union will be used). – `IDsToExcludeFile`: Path to file with marker IDs to exclude. – `RangesToExcludeFile`: Path to file with genomic ranges to exclude. Can be used with IDsToExcludeFile (union will be excluded). – Note: Cannot use both include and exclude files simultaneously.

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 *Convert genotype matrix to PLINK format files*

Description

Converts a numeric genotype matrix to PLINK text files (PED and MAP format) for use with PLINK software and other genetic analysis tools.

Usage

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

Arguments

GenoMat	Numeric genotype matrix (n×m) with values 0, 1, 2, or -9. Rows = subjects, columns = markers. Row and column names are required.
OutputPrefix	Output file prefix including path (without extension).
A1	Allele 1 character, usually minor/ALT allele (default: "G").
A2	Allele 2 character, usually major/REF allele (default: "A").
CHR	Chromosome numbers for markers (default: all chromosome 1).
BP	Base positions for markers (default: 1:m).
Pheno	Phenotype values for subjects (default: all missing as -9).
Sex	Sex codes for subjects (default: all coded as 1).

Details

Genotype Encoding: - 0, 1, 2 → copies of minor allele - -9 → missing genotype (coded as "00" in PED) - A1="G", A2="A": 0→"GG", 1→"AG", 2→"AA", -9→"00"

Output Files: - '.ped': Pedigree file with genotype data - '.map': Marker map file with positions

Downstream Processing: "'bash # Convert to binary format plink –file prefix –make-bed –out prefix

Convert to raw format plink –bfile prefix –recode A –out prefix_raw

Convert to BGEN format plink2 –bfile prefix –export bgen-1.2 bits=8 ref-first –out prefix_bgen

Create BGEN index bgenix -g prefix_bgen.bgen –index "'

Value

Character message confirming file creation location.

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 <- tempdir()
outputPrefix <- file.path(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 *Perform single-marker association tests using a fitted null model*

Description

Conducts single-marker association tests between genetic variants and phenotypes using various statistical methods supported by GRAB.

Usage

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

```
  control = NULL
)
```

Arguments

objNull	(S3 object) Null model object from GRAB.NullModel, SPAGRM.NullModel or SAGELD.NullModel. Supported classes: • POLMM_NULL_Model: See ?GRAB.POLMM. • SPACox_NULL_Model: See ?GRAB.SPACox. • SPAmix_NULL_Model: See ?GRAB.SPAmix. • WtCoxG_NULL_Model: See ?GRAB.WtCoxG. • SPAGRM_NULL_Model: See ?GRAB.SPAGRM. • SAGELD_NULL_Model: See ?GRAB.SAGELD.
GenoFile	Path to genotype file. Supported formats determined by extension: • PLINK: "prefix.bed" • BGEN: "prefix.bgen" (version 1.2 with 8-bit compression)
OutputFile	(character) Path for saving association test results.
GenoFileIndex	(character vector or NULL) Associated files for the genotype file (auto-detected if NULL): • PLINK: c("prefix.bim", "prefix.fam") • BGEN: c("prefix.bgen.bgi", "prefix.sample") or c("prefix.bgen.bgi")
OutputFileIndex	(character or NULL) #' Path to the progress tracking file from a previous unfinished run. Enables analysis to restart if interrupted. If NULL (default), uses paste0(OutputFile, ".index").
control	(list or NULL) List of control parameters with the following elements: • AlleleOrder (character or NULL): Allele order in genotype file. Options: "ref-first", "alt-first", or NULL (default: "alt-first" for BGEN, "ref-first" for PLINK). • **Marker Selection:** – AllMarkers (logical): Set to TRUE (default) to analyze all markers. Automatically set to FALSE if any include/exclude files are provided. – IDsToIncludeFile (character or NULL): Path to file with marker IDs to include. – RangesToIncludeFile (character or NULL): Path to file with genomic ranges to include. Can be used with IDsToIncludeFile (union will be used). – IDsToExcludeFile (character or NULL): Path to file with marker IDs to exclude. – RangesToExcludeFile (character or NULL): Path to file with genomic ranges to exclude. Can be used with IDsToExcludeFile (union will be excluded). – Note: Cannot use both include and exclude files simultaneously.

- `impute_method` (character): Imputation method for handling missing genotypes during analysis in C++ backend. Applies to all genotype formats. Options: "mean" (default), "minor", "drop".
- `missing_cutoff` (numeric): Exclude markers with missing rate above this threshold. Range: 0 to 0.5. Default: 0.15.
- `min_maf_marker` (numeric): Exclude markers with MAF below this threshold. Range: 0 to 0.1. Default: 0.001.
- `min_mac_marker` (numeric): Exclude markers with MAC below this threshold. Range: 0 to 100. Default: 20.
- `nMarkersEachChunk` (integer): Number of markers processed per chunk. Range: 1000 to 100000. Default: 10000.
- `SPA_Cutoff` (numeric): Z-score cutoff for saddlepoint approximation. When the absolute value of the test statistic exceeds this cutoff, SPA is used to calculate more accurate p-values. Default: 2.

Value

The function returns `NULL` invisibly. Results are written to `OutputFile`. For method-specific examples and output columns and format, see:

- POLMM method: `GRAB.POLMM`
- SPACox method: `GRAB.SPACox`
- SPAmix method: `GRAB.SPAmix`
- WtCoxG method: `GRAB.WtCoxG`
- SPAGRM method: `GRAB.SPAGRM`
- SAGELD method: `GRAB.SAGELD`

`GRAB.NullModel` *Top-level API for generating a null model object used by GRAB.Marker and GRAB.Region*

Description

GRAB performs two-step genetic association testing. This function implements the first step: fitting a null model and preparing the dataset required for downstream marker-level (`GRAB.Marker`) and region-level (`GRAB.Region`) analyses.

Usage

```
GRAB.NullModel(
  formula,
  data,
  subjIDcol = NULL,
  subjData = NULL,
  method,
```

```
  traitType,
  GenoFile = NULL,
  GenoFileIndex = NULL,
  SparseGRMFile = NULL,
  control = NULL,
  ...
)
```

Arguments

formula	(formula) Formula with response variable(s) on the left and covariates on the right. Do not include an intercept (added automatically). For SPAmix with traitType "Residual", multiple response variables (separated by "+") are supported.
data	(data.frame) Data frame containing response variables and covariates in the formula. Parameter "subset" is deprecated. All subjects with phenotype data will be used. Missing values should be coded as NA. Other values (e.g., -9, -999) are treated as numeric.
subjIDcol	(character or NULL) Column name in data containing subject IDs.
subjData	(character vector or NULL) Subject IDs aligned with rows of data. Exactly one of subjIDcol or subjData must be provided.
method	(character) Supported methods: • "POLMM": Ordinal traits. See ?GRAB.POLMM. • "SPACox": Time-to-event or Residual. See ?GRAB.SPACox. • "SPAmix": Time-to-event or Residual. See ?GRAB.SPAmix. • "WtCoxG": Time-to-event traits. See ?GRAB.WtCoxG.
traitType	(character) Supported: "ordinal", "time-to-event", and "Residual".
GenoFile	(character or NULL) Path to genotype file. Supported formats determined by extension: • PLINK: "prefix.bed" • BGEN: "prefix.bgen" (version 1.2 with 8-bit compression)
GenoFileIndex	(character vector or NULL) Associated files for the genotype file (auto-detected if NULL): • PLINK: c("prefix.bim", "prefix.fam") • BGEN: c("prefix.bgen.bgi", "prefix.sample") or c("prefix.bgen.bgi")
SparseGRMFile	(character or NULL) Path to a sparse GRM file. The file must be whitespace-delimited with three columns in the order: • Column 1: Subject ID 1 • Column 2: Subject ID 2 • Column 3: Genetic correlation between the two subjects See system.file("extdata", "SparseGRM.txt", package = "GRAB") for an example. See ?getSparseGRM for details on generating a sparse GRM.
control	(list or NULL) List of additional, less commonly used parameters. See the corresponding method documentation for available options and defaults.
...	Additional method-specific parameters.

Value

An S3 object with class "{method}_NULL_Model". All returned objects contain the following elements:

N Sample size (integer).

subjData Character vector of subject IDs included in analysis.

Call Original function call.

sessionInfo R session and package information.

time Analysis completion timestamp (character).

control List of control parameters used in fitting.

This object serves as input for `GRAB.Marker` or `GRAB.Region`.

See method-specific documentation for additional elements included in the returned object.

`GRAB.POLMM` *Instruction of POLMM method*

Description

POLMM inplements single-variant association tests for ordinal categorical phenotypes, which accounts for sample relatedness. It can control type I error rates at a stringent significance level regardless of the phenotypic distribution, and is more powerful than alternative methods. This instruction covers null model fitting and marker-level analysis using POLMM. For region-based analysis with POLMM-GENE, see `GRAB.POLMM.Region`.

Usage

```
GRAB.POLMM()
```

Details

Genotype file: `GenoFile` is mandatory for `GRAB.NullModel()` when using POLMM. It is required for estimating the variance ratio parameter, which is essential for calibrating the test statistics in subsequent association tests.

Genetic Relationship Matrix (GRM) Options: POLMM supports both sparse and dense GRM for modeling genetic relatedness:

- If `SparseGRMFile` is provided to `GRAB.NullModel()`, the sparse GRM will be used in model fitting.
- If `SparseGRMFile` is not provided, `GRAB.NullModel()` will calculate a dense GRM from `GenoFile`.

Additional Control Parameters for GRAB.NullModel():

- `memoryChunk` (numeric, default: 2): Memory chunk size for computation.
- `seed` (integer, default: -1): Random seed (-1 means no seed is set).

- `tracenrun` (integer, default: 30): Number of runs for trace calculation.
- `maxiter` (integer, default: 100): Maximum number of iterations for model fitting.
- `tolBeta` (numeric, default: 0.001): Convergence tolerance for beta estimates.
- `tolTau` (numeric, default: 0.002): Convergence tolerance for tau estimates.
- `tau` (numeric, default: 0.2): Initial variance component value.
- `maxiterPCG` (integer, default: 100): Maximum iterations for preconditioned conjugate gradient.
- `tolPCG` (numeric, default: 1e-6): Tolerance for preconditioned conjugate gradient.
- `showInfo` (logical, default: FALSE): Whether to print PCG iteration information for debugging.
- `maxiterEps` (integer, default: 100): Maximum iterations for epsilon estimation.
- `tolEps` (numeric, default: 1e-10): Tolerance for epsilon estimation.
- `minMafVarRatio` (numeric, default: 0.1): Minimum MAF for variance ratio estimation.
- `maxMissingVarRatio` (numeric, default: 0.1): Maximum missing rate for variance ratio estimation.
- `nSNPsVarRatio` (integer, default: 20): Number of SNPs used for variance ratio estimation.
- `CVcutoff` (numeric, default: 0.0025): Coefficient of variation cutoff.
- `grainSize` (integer, default: 1): Grain size for parallel processing.
- `minMafGRM` (numeric, default: 0.01): Minimum MAF for GRM construction.
- `maxMissingGRM` (numeric, default: 0.1): Maximum missing rate for GRM construction.

Method-specific elements in the `POLMM_NULL_Model` **object returned by** `GRAB.NullModel()`**:**:

- `M`: Number of ordinal categories (integer).
- `iter`: Number of iterations to convergence (numeric).
- `eta`: Linear predictor (matrix).
- `yVec`: Phenotype matrix (matrix).
- `Cova`: Design matrix of covariates (matrix).
- `muMat`: Fitted probabilities for each category (matrix).
- `YMat`: Indicator matrix for ordinal categories (matrix).
- `beta`: Estimated covariate coefficients (matrix).
- `bVec`: Random effect estimates (matrix).
- `tau`: Variance component estimate (numeric).
- `eps`: Cutpoints for ordinal categories (matrix).

Additional Control Parameters for GRAB.Marker():

- `ifOutGroup` (logical, default: FALSE): Whether to output group-specific statistics (alternative allele frequency, counts, and sample size for each ordinal category). When TRUE, adds columns AltFreqInGroup.*, AltCountsInGroup.*, and nSamplesInGroup.* to the output file.

Marker-level results (`OutputFile`) columns:

Marker Marker identifier (rsID or CHR:POS:REF:ALT).

Info Marker information in format CHR:POS:REF:ALT.

AltFreq Alternative allele frequency in the overall sample.

AltCounts Total count of alternative alleles.

MissingRate Proportion of missing genotypes.

Pvalue P-value from the score test.

beta Effect size estimate (log-odds scale).

seBeta Standard error of beta.

zScore Z-score from the score test.

AltFreqInGroup.1, AltFreqInGroup.2, ... (Only if `ifOutGroup = TRUE`) Alternative allele frequency in each ordinal category.

AltCountsInGroup.1, AltCountsInGroup.2, ... (Only if `ifOutGroup = TRUE`) Alternative allele counts in each ordinal category.

nSamplesInGroup.1, nSamplesInGroup.2, ... (Only if `ifOutGroup = TRUE`) Sample size in each ordinal category.

References

Bi et al. (2021). Efficient mixed model approach for large-scale genome-wide association studies of ordinal categorical phenotypes. doi:10.1016/j.ajhg.2021.03.019

Examples

```
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultPOLMMmarker.txt")

PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
PhenoData$OrdinalPheno <- factor(PhenoData$OrdinalPheno, levels = c(0, 1, 2))
# Step 1
obj.POLMM <- GRAB.NullModel(
 OrdinalPheno ~ AGE + GENDER,
 data = PhenoData,
 subjIDcol = "IID",
 method = "POLMM",
 traitType = "ordinal",
 GenoFile = GenoFile,
 SparseGRMFile = SparseGRMFile
)

# Step 2
GRAB.Marker(obj.POLMM, GenoFile, OutputFile,
  control = list(ifOutGroup = TRUE))

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

GRAB.POLMM.Region *Instruction of POLMM-GENE method*

Description

POLMM-GENE implements region-based association tests for ordinal categorical phenotypes, adjusting for sample relatedness. It is well-suited for analyzing rare variants in large-scale biobank data, and effectively controls type I error rates while maintaining statistical power.

Usage

```
GRAB.POLMM.Region()
```

Details

For single-variant tests, see GRAB.POLMM.

See GRAB.POLMM for details on step 1.

Additional Control Parameters for GRAB.Region() with POLMM:

- showInfo (logical, default: FALSE): Whether to print PCG iteration information for debugging.
- tolPCG (numeric, default: 0.001): Tolerance for PCG in region testing.
- maxiterPCG (integer, default: 100): Maximum PCG iterations in region testing.

Results are saved to four files:

1. OutputFile: Region-based test results (SKAT-O, SKAT, Burden p-values).
2. paste0(OutputFile, ".markerInfo"): Marker-level results for rare variants (MAC >= min_mac_region) included in region tests.
3. paste0(OutputFile, ".otherMarkerInfo"): Information for excluded markers (ultra-rare variants or failed QC).
4. paste0(OutputFile, ".infoBurdenNoWeight"): Summary statistics for burden tests without weights.

Region-level results (OutputFile) columns:

Region Region identifier from GroupFile.

nMarkers Number of rare variants with MAF < cutoff and MAC >= min_mac_region.

nMarkersURV Number of ultra-rare variants with MAC < min_mac_region.

Anno.Type Annotation type from GroupFile.

MaxMAF.Cutoff Maximum MAF cutoff used for variant selection.

pval.SKATO SKAT-O test p-value.

pval.SKAT SKAT test p-value.

pval.Burden Burden test p-value.

Marker-level results (`paste0(OutputFile, ".markerInfo")`) columns:

Region Region identifier.

ID Marker identifier.

Info Marker information in format CHR:POS:REF:ALT.

Anno Annotation from `GroupFile`.

AltFreq Alternative allele frequency.

MAC Minor allele count.

MAF Minor allele frequency.

MissingRate Proportion of missing genotypes.

IndicatorVec Marker status indicator (1 = rare variant included, 3 = ultra-rare variant included).

StatVec Score test statistic.

altBetaVec Effect size estimate.

seBetaVec Standard error of effect size estimate.

pval0Vec Unadjusted p-value.

pval1Vec SPA-adjusted p-value.

posRow Position row index.

Other marker info (`paste0(OutputFile, ".otherMarkerInfo")`) columns:

ID Marker identifier.

Annos Annotation from `GroupFile`.

Region Region identifier.

Info Marker information in format CHR:POS:REF:ALT.

Anno Annotation category.

AltFreq Alternative allele frequency.

MAC Minor allele count.

MAF Minor allele frequency.

MissingRate Proportion of missing genotypes.

IndicatorVec Status indicator (0 or 2 for excluded markers).

Burden test summary (`paste0(OutputFile, ".infoBurdenNoWeight")`) columns:

region Region identifier.

anno Annotation type.

max_maf Maximum MAF cutoff.

sum Sum of genotypes.

Stat Score test statistic.

beta Effect size estimate.

se.beta Standard error of effect size estimate.

pvalue P-value for burden test.

References

Bi et al. (2023). Scalable mixed model methods for set-based association studies on large-scale categorical data analysis and its application to exome-sequencing data in UK Biobank. doi:10.1016/j.ajhg.2023.03.010

Examples

```
GenoFileStep1 <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
GenoFileStep2 <- system.file("extdata", "simuPLINK_RV.bed", package = "GRAB")
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
GroupFile <- system.file("extdata", "simuPLINK_RV.group", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultPOLMMregion.txt")

PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
PhenoData$OrdinalPheno <- factor(PhenoData$OrdinalPheno, levels = c(0, 1, 2))
# Step 1
obj.POLMM <- GRAB.NullModel(
 OrdinalPheno ~ AGE + GENDER,
 data = PhenoData,
 subjIDcol = "IID",
 method = "POLMM",
 traitType = "ordinal",
 GenoFile = GenoFileStep1,
 SparseGRMFile = SparseGRMFile,
 control = list(tolTau = 0.2, tolBeta = 0.1)
)

# Step 2
GRAB.Region(obj.POLMM, GenoFileStep2, OutputFile,
  GroupFile = GroupFile,
  SparseGRMFile = SparseGRMFile,
  MaxMAFVec = "0.01,0.005"
)

head(data.table::fread(OutputFile))
head(data.table::fread(paste0(OutputFile, ".markerInfo")))
head(data.table::fread(paste0(OutputFile, ".otherMarkerInfo")))
head(data.table::fread(paste0(OutputFile, ".infoBurdenNoWeight")))
```

GRAB.ReadGeno *Read genotype data from multiple file formats*

Description

Reads genotype data from PLINK or BGEN format files with flexible filtering and processing options. Supports efficient memory usage and various imputation methods for missing genotypes.

Usage

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

Arguments

`GenoFile`	Path to genotype file. Supported formats determined by extension: • PLINK: "prefix.bed" (binary format) • BGEN: "prefix.bgen" (version 1.2 with 8-bit compression)
`GenoFileIndex`	Associated index files for the genotype file: • PLINK: c("prefix.bim", "prefix.fam") (auto-detected if NULL) • BGEN: "prefix.bgen.bgi" or c("prefix.bgen.bgi", "prefix.sample")
`SampleIDs`	Character vector of sample IDs to extract. If NULL, extracts all samples.
`control`	List of control parameters with the following options: • `imputeMethod`: Imputation method for genotype data. Options: "none" (default), "mean" (2 times allele frequency). "bestguess" (round mean to the nearest integer, 0, 1, or 2). • `AlleleOrder`: Allele order in genotype file. Options: "ref-first", "alt-first", or NULL (default: "alt-first" for BGEN, "ref-first" for PLINK). • **Marker Selection:** – `AllMarkers`: Set to TRUE (default) to analyze all markers. Automatically set to FALSE if any include/exclude files are provided. – `IDsToIncludeFile`: Path to file with marker IDs to include. – `RangesToIncludeFile`: Path to file with genomic ranges to include. Can be used with IDsToIncludeFile (union will be used). – `IDsToExcludeFile`: Path to file with marker IDs to exclude. – `RangesToExcludeFile`: Path to file with genomic ranges to exclude. Can be used with IDsToExcludeFile (union will be excluded). – Note: Cannot use both include and exclude files simultaneously.
`sparse`	Logical indicating whether to return sparse genotype matrix (default: FALSE).

Details

File Format Support:

PLINK Format: Binary BED/BIM/FAM files. See https://www.cog-genomics.org/plink/2.0/ for specifications.

BGEN Format: Version 1.2 with 8-bit compression. See https://www.well.ox.ac.uk/~gav/bgen_format/spec/v1.2.html for details. Requires BGI index file created with bgenix tool.

Value

List containing:

GenoMat Genotype matrix (samples × markers) with values 0, 1, 2, or NA.

markerInfo Data frame with columns CHROM, POS, ID, REF, 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 *Perform region-based association tests*

Description

Tests for association between phenotypes and genomic regions containing multiple genetic variants, primarily low-frequency and rare variants.

Usage

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

Arguments

objNull	(S3 object) Null model object from GRAB.NullModel. Currently supports POLMM_NULL_Model.
GenoFile	(character) Path to genotype file (PLINK or BGEN format). See GRAB.ReadGeno for details.
OutputFile	(character) Path for saving region-based association results.
GenoFileIndex	(character or NULL) Index files for the genotype file. If NULL (default), uses same prefix as GenoFile. See GRAB.ReadGeno for details.
OutputFileIndex	(character or NULL) Path for progress tracking file. If NULL (default), uses paste0(OutputFile, ".index").
GroupFile	(character) Path to region definition file specifying region-marker mappings and annotation information. Tab-separated format with 2-3 columns per region.
SparseGRMFile	(character or NULL) Path to sparse GRM file (optional).
MaxMAFVec	(character) Comma-separated MAF cutoffs for including variants in analysis (default: "0.01,0.001,0.0005").

`annoVec` (character) Comma-separated annotation groups for analysis (default: "lof,lof:missense,lof:missense:syno

`control` (list or NULL) List of the following parameters:

- `impute_method` (character): Method for imputing missing genotypes: "mean", "minor", or "drop". Default: "minor".
- `missing_cutoff` (numeric): Exclude markers with missing rate > this value. Range: 0 to 0.5. Default: 0.15.
- `min_mac_region` (numeric): Minimum MAC threshold; markers with MAC < this value are treated as ultra-rare variants. Default: 5.
- `max_markers_region` (integer): Maximum number of markers allowed per region. Default: 100.
- `r.corr` (numeric vector): Rho parameters for SKAT-O test. Range: 0 to 1. Default: c(0, 0.1^2, 0.2^2, 0.3^2, 0.4^2, 0.5^2, 0.5, 1).
- `weights.beta` (numeric vector): Beta distribution parameters for variant weights (length 2). Default: c(1, 25).
- `omp_num_threads` (integer): Number of OpenMP threads for parallel computation. Default: data.table::getDTthreads().
- `min_nMarker` (integer): Minimum number of markers required for region analysis. Default: 3.
- `SPA_Cutoff` (numeric): Z-score cutoff for saddlepoint approximation. When the absolute value of the test statistic exceeds this cutoff, SPA is used to calculate more accurate p-values. Default: 2.

Value

The function returns `NULL` invisibly. Results are saved to four files:

1. `OutputFile`: Region-based test results (SKAT-O, SKAT, Burden p-values).
2. `paste0(OutputFile, ".markerInfo")`: Marker-level results for rare variants (MAC >= `min_mac_region`) included in region tests.
3. `paste0(OutputFile, ".otherMarkerInfo")`: Information for excluded markers (ultra-rare variants or failed QC).
4. `paste0(OutputFile, ".infoBurdenNoWeight")`: Summary statistics for burden tests without weights.

For method-specific examples and output columns and format, see:

- POLMM method: `GRAB.POLMM.Region`

`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 SPA_{GRM} 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).

Examples

```
LongDataFile <- system.file("extdata", "simuLongPHENO.txt", package = "GRAB")
ResidMatFile <- system.file("extdata", "ResidMat.txt", package = "GRAB")
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
PairwiseIBDFile <- system.file("extdata", "PairwiseIBD.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultSAGELD.txt")
LongPheno <- data.table::fread(LongDataFile)

# Step 1: fit null model using lme4
nullmodel <- lme4::lmer(LongPheno ~ AGE + GENDER + (AGE|IID), data = LongPheno)

# Step 2a: pre-calculate genotype distributions
obj.SAGELD <- SAGELD.NullModel(
  NullModel = nullmodel,
  UsedMethod = "SAGELD",
  PlinkFile = GenoFile,
  SparseGRMFile = SparseGRMFile,
  PairwiseIBDFile = PairwiseIBDFile,
)

# Step 2b: perform association tests
GRAB.Marker(obj.SAGELD, GenoFile, OutputFile)

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

GRAB.SimubVec *Simulate random effects based on family structure*

Description

Generates random effect vectors (bVec) that account for family relationships through kinship-based correlation structures.

Usage

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

Arguments

nSub	Number of unrelated subjects. If 0, all subjects are related.
nFam	Number of families. If 0, all subjects are unrelated.
FamMode	Family structure: "4-members", "10-members", or "20-members". See GRAB.SimuGMat for pedigree details.
tau	Variance component for random effects.

Details

For related subjects, random effects follow a multivariate normal distribution with covariance proportional to kinship coefficients. For unrelated subjects, effects are independent normal random variables.

Value

Data frame with columns:

IID Subject identifiers.

bVec Random effect values following appropriate correlation structure.

GRAB.SimuGMat *Simulate genotype data matrix for related and unrelated subjects*

Description

Generates genotype data for association studies, supporting both unrelated subjects and family-based designs with various pedigree structures.

Usage

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

Arguments

nSub	Number of unrelated subjects. If 0, all subjects are related.
nFam	Number of families. If 0, all subjects are unrelated.
FamMode	Family structure: "4-members", "10-members", or "20-members". See Details for pedigree structures.
nSNP	Number of genetic markers to simulate.
MaxMAF	Maximum minor allele frequency for simulation (default: 0.5).
MinMAF	Minimum minor allele frequency for simulation (default: 0.05).
MAF	Optional vector of specific MAF values for each marker. If provided, MaxMAF and MinMAF are ignored.

Details

Genotypes are simulated under Hardy-Weinberg equilibrium with MAF ~ Uniform(MinMAF, MaxMAF).

Family Structures:

- **4-members**: 1+2→3+4 (parents 1,2 → offspring 3,4)
- **10-members**: 1+2→5+6, 3+5→7+8, 4+6→9+10
- **20-members**: Complex multi-generational pedigree with 20 members

Total subjects: nSub + nFam × family_size

Value

List containing:

GenoMat Numeric genotype matrix (subjects × markers) with values 0, 1, 2.

markerInfo Data frame with marker IDs and MAF values.

See Also

GRAB.makePlink for converting to PLINK format.

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` *Simulate genotype matrix from external genotype file*

Description

Generates genotype matrices for families and unrelated subjects using haplotype data from existing genotype files. Primarily designed for rare variant analysis simulations.

Usage

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

Arguments

`nFam`	Number of families to simulate.
`nSub`	Number of unrelated subjects to simulate.
`FamMode`	Family structure: "4-members", "10-members", or "20-members". See Details for pedigree structures.
`GenoFile`	Path to genotype file (passed to `GRAB.ReadGeno`).
`GenoFileIndex`	Index file for genotype data (optional, for BGEN files).
`SampleIDs`	Vector of sample IDs to include (optional).
`control`	List of control parameters passed to `GRAB.ReadGeno`.

Details

This function supports both unrelated and related subjects. Founder genotypes are sampled from the input genotype file, and offspring genotypes are generated through Mendelian inheritance.

Note: When simulating related subjects, alleles are randomly assigned to haplotypes during the phasing process.

Family Structures: - **4-members**: Total subjects = nSub + 4×nFam. Structure: 1+2→3+4 - **10-members**: Total subjects = nSub + 10×nFam. Structure: 1+2→5+6, 3+5→7+8, 4+6→9+10 - **20-members**: Total subjects = nSub + 20×nFam. Structure: 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

List containing:

GenoMat Simulated genotype matrix (subjects × variants).

SubjIDs Subject identifiers for simulated samples.

markerInfo Variant information from input file.

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 phenotypes from linear predictors*

Description

Generates various types of phenotypes (quantitative, binary, ordinal, time-to-event) from linear predictors using appropriate link functions.

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`	Vector of linear predictors (typically covariates×beta + genotypes×beta).
`traitType`	Type of phenotype: "quantitative", "binary", "ordinal", or "time-to-event".
`control`	List of simulation parameters specific to each trait type: **pCase** Proportion of cases (binary traits). **sdError** Error term standard deviation (quantitative traits). **pEachGroup** Group proportions (ordinal traits). **eventRate** Event rate (time-to-event traits).
`seed`	Random seed for reproducibility (optional).

Details

Trait Type Details: - **Quantitative**: Y = eta + error, where error ~ N(0, sdError²) - **Binary**: Logistic model with specified case proportion - **Ordinal**: Proportional odds model with specified group proportions - **Time-to-event**: Weibull survival model with specified event rate

For more details, see: https://wenjianbi.github.io/grab.github.io/docs/simulation_phenotype.html

Value

Simulated phenotype vector or data frame:

quantitative Numeric vector of continuous values.

binary Numeric vector of 0/1 values.

ordinal Numeric vector of categorical levels.

time-to-event Data frame with SurvTime and SurvEvent columns.

`GRAB.SPACox` *Instruction of SPACox method*

Description

SPACox is primarily intended for time-to-event traits in unrelated samples from large-scale biobanks. It uses the empirical cumulant generating function (CGF) to perform SPA-based single-variant association tests, enabling analysis with residuals from any null model.

Usage

```
GRAB.SPACox()
```

Details

Additional Control Parameters for GRAB.NullModel():

- `range` (numeric vector, default: c(-100, 100)): Range for saddlepoint approximation grid. Must be symmetric (range\[2\] = -range\[1\]).
- `length.out` (integer, default: 10000): Number of grid points for saddlepoint approximation.

Method-specific elements in the `SPACox_NULL_Model` object returned by `GRAB.NullModel()`::

- `mresid`: Martingale residuals (numeric or "Residual" class).
- `cumul`: Cumulative sums for variance estimation (matrix).
- `tX`: Transpose of design matrix (matrix).
- `yVec`: Event indicator (numeric or "Residual" class).
- `X.invXX`: Matrix for variance calculations (matrix).

Additional Control Parameters for GRAB.Marker():

- `pVal_covaAdj_Cutoff` (numeric, default: 5e-05): P-value cutoff for covariate adjustment.

Output file columns:

Marker Marker identifier (rsID or CHR:POS:REF:ALT).

Info Marker information in format CHR:POS:REF:ALT.

AltFreq Alternative allele frequency in the sample.

AltCounts Total count of alternative alleles.

MissingRate Proportion of missing genotypes.

Pvalue P-value from the score test.

zScore Z-score from the score test.

References

Bi et al. (2020). Fast and accurate method for genome-wide time-to-event data analysis and its application to UK Biobank. doi:10.1016/j.ajhg.2020.06.003

Examples

```
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultSPACox.txt")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)

# Step 1 option 1
obj.SPACox <- GRAB.NullModel(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  subjIDcol = "IID",
  method = "SPACox",
  traitType = "time-to-event"
)
```

```
# Step 1 option 2
residuals <- survival::coxph(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  x = TRUE
)$residuals

obj.SPACox <- GRAB.NullModel(
  residuals ~ AGE + GENDER,
  data = PhenoData,
  subjIDcol = "IID",
  method = "SPACox",
  traitType = "Residual"
)

# Step 2
GRAB.Marker(obj.SPACox, GenoFile, OutputFile)

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

GRAB.SPAGRM *Instruction of SPAGRM method*

Description

SPAGRM is a scalable and accurate framework for retrospective association tests. It treats genetic loci as random vectors and uses a precise approximation of their joint distribution. This approach enables SPAGRM to handle any type of complex trait, including longitudinal and unbalanced phenotypes. SPAGRM extends SPACox to support sample relatedness.

Usage

```
GRAB.SPAGRM()
```

Details

See SPAGRM.NullModel for detailed instructions on preparing a SPAGRM_NULL_Model object required for GRAB.Marker().

Additional Control Parameters for GRAB.Marker():

- `zeta` (numeric, default: 0): SPA moment approximation parameter.
- `tol` (numeric, default: 1e-5): Numerical tolerance for SPA convergence.

Marker-level results (`OutputFile`) columns:

Marker Marker identifier (rsID or CHR:POS:REF:ALT).

Info Marker information in format CHR:POS:REF:ALT.

AltFreq Alternative allele frequency in the sample.

AltCounts Total count of alternative alleles.

MissingRate Proportion of missing genotypes.

zScore Z-score from the score test.

Pvalue P-value from the score test.

hwepval Hardy-Weinberg equilibrium p-value.

References

Xu et al. (2025). SPAGRM: effectively controlling for sample relatedness in large-scale genome-wide association studies of longitudinal traits. doi:10.1038/s41467025566691

Examples

```
ResidMatFile <- system.file("extdata", "ResidMat.txt", package = "GRAB")
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
PairwiseIBDFile <- system.file("extdata", "PairwiseIBD.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultSPAGRM.txt")

# Step 2a: pre-calculate genotype distributions
obj.SPAGRM <- SPAGRM.NullModel(
  ResidMatFile = ResidMatFile,
  SparseGRMFile = SparseGRMFile,
  PairwiseIBDFile = PairwiseIBDFile,
  control = list(ControlOutlier = FALSE)
)

# Step 2b: perform association tests
GRAB.Marker(obj.SPAGRM, GenoFile, OutputFile)

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

`GRAB.SPAmix` *Instruction of SPAmix method*

Description

SPAmix performs retrospective single-variant association tests using genotypes and residuals from null models of any complex trait in large-scale biobanks. It extends SPACox to support complex population structures, such as admixed ancestry and multiple populations, but does not account for sample relatedness.

Usage

```
GRAB.SPAmix()
```

Details

Additional Control Parameters for GRAB.NullModel():

- `PC_columns` (character, required): Comma-separated column names of principal components (e.g., "PC1,PC2").
- `OutlierRatio` (numeric, default: 1.5): IQR multiplier for outlier detection. Outliers are defined as values outside \[Q1 - r*IQR, Q3 + r*IQR\].

Method-specific elements in the `SPAmix_NULL_Model` object returned by `GRAB.NullModel()`:

- `resid`: Residuals from mixed model (matrix or "Residual" class).
- `yVec`: Phenotype vector (numeric or "Residual" class).
- `PCs`: Principal components for dimension reduction (matrix).
- `nPheno`: Number of phenotypes analyzed (integer).
- `outLierList`: List identifying outlier subjects for SPA adjustment.

Additional Control Parameters for GRAB.Marker():

- `dosage_option` (character, default: "rounding_first"): Dosage handling option. Must be either "rounding_first" or "rounding_last".

Output file columns:

Pheno Phenotype identifier (for multi-trait analysis).

Marker Marker identifier (rsID or CHR:POS:REF:ALT).

Info Marker information in format CHR:POS:REF:ALT.

AltFreq Alternative allele frequency in the sample.

AltCounts Total count of alternative alleles.

MissingRate Proportion of missing genotypes.

Pvalue P-value from the score test.

zScore Z-score from the score test.

References

Ma et al. (2025). SPAmix: a scalable, accurate, and universal analysis framework for large-scale genetic association studies in admixed populations. doi:10.1186/s13059025038279

Examples

```
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultSPAmix.txt")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)

# Step 1 option 1
obj.SPAmix <- GRAB.NullModel(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData,
```

```
  subjIDcol = "IID",
  method = "SPAmix",
  traitType = "time-to-event",
  control = list(PC_columns = "PC1,PC2")
)

# Step 1 option 2
residuals <- survival::coxph(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData
)$residuals

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

# Step 1 option 2: analyze multiple traits at once
res_cox <- survival::coxph(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData
)$residuals

res_lm <- lm(QuantPheno ~ AGE + GENDER + PC1 + PC2, data = PhenoData)$residuals

obj.SPAmix <- GRAB.NullModel(
  res_cox + res_lm ~ AGE + GENDER + PC1 + PC2,
  data = PhenoData,
  subjIDcol = "IID",
  method = "SPAmix",
  traitType = "Residual",
  control = list(PC_columns = "PC1,PC2")
)

# Step 2
GRAB.Marker(obj.SPAmix, GenoFile, OutputFile)

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

`GRAB.WtCoxG` *Instruction of WtCoxG method*

Description

WtCoxG is a Cox-based association test method for time-to-event traits. It effectively addresses case ascertainment and rare variant analysis. By leveraging external minor allele frequencies from

public resources, WtCoxG can further boost statistical power.

Usage

```
GRAB.WtCoxG()
```

Details

Additional Parameters for `GRAB.NullModel()`:

- `RefAfFile` (character, required): Reference allele frequency file path. File must contain columns: CHROM, POS, ID, REF, ALT, AF_ref, AN_ref
- `RefPrevalence` (numeric, required): Population-level disease prevalence for weighting. Must be in range (0, 0.5)

Additional Control Parameters for GRAB.NullModel():

- `OutlierRatio` (numeric, default: 1.5): IQR multiplier for outlier detection

Method-specific elements in the `WtCoxG_NULL_Model` object returned by `GRAB.NullModel()`::

- `mresid`: Martingale residuals from weighted Cox model (numeric).
- `Cova`: Design matrix of covariates (matrix).
- `yVec`: Event indicator (numeric).
- `weight`: Observation weights based on reference prevalence (numeric).
- `RefPrevalence`: Reference population prevalence used for weighting (numeric).
- `outLierList`: List identifying outlier subjects for SPA adjustment.
- `mergeGenoInfo`: Data frame with batch effect QC results and external reference data.

Additional Control Parameters for GRAB.Marker():

- `cutoff` (numeric, default: 0.1): Cutoff of batch effect test p-value for association testing. Variants with batch effect p-value below this cutoff will be excluded from association testing.

Output file columns:

Pheno Phenotype identifier (for multi-trait analysis).

Marker Marker identifier (rsID or CHR:POS:REF:ALT).

Info Marker information in format CHR:POS:REF:ALT.

AltFreq Alternative allele frequency in the sample.

AltCounts Total count of alternative alleles.

MissingRate Proportion of missing genotypes.

Pvalue P-value from the score test.

zScore Z-score from the score test.

References

Li et al. (2025). Applying weighted Cox regression to genome-wide association studies of time-to-event phenotypes. doi:10.1038/s4358802500864z

Examples

```
# Step 1: fit null model and test batch effect
PhenoFile <- system.file("extdata", "simuPHENO.txt", package = "GRAB")
PhenoData <- data.table::fread(PhenoFile, header = TRUE)
SparseGRMFile <- system.file("extdata", "SparseGRM.txt", package = "GRAB")
GenoFile <- system.file("extdata", "simuPLINK.bed", package = "GRAB")
RefAfFile <- system.file("extdata", "simuRefAf.txt", package = "GRAB")
OutputFile <- file.path(tempdir(), "resultWtCoxG.txt")

obj.WtCoxG <- GRAB.NullModel(
  survival::Surv(SurvTime, SurvEvent) ~ AGE + GENDER,
  data = PhenoData,
  subjIDcol = "IID",
  method = "WtCoxG",
  traitType = "time-to-event",
  GenoFile = GenoFile,
  SparseGRMFile = SparseGRMFile,
  RefAfFile = RefAfFile,
  RefPrevalence = 0.1
)

# Step2
GRAB.Marker(obj.WtCoxG, GenoFile, OutputFile)

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

`SAGELD.NullModel` *Construct SAGELD/GALLOP null model from a mixed-effects fit*

Description

Builds the SAGELD (or GALLOP) null model from a fitted mixed-effects model and relatedness inputs. Extracts variance components, forms the penalization matrix, derives residual summaries, and (for SAGELD) integrates sparse GRM and pairwise IBD to prepare graph-based components for marker testing.

Usage

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

Arguments

`NullModel`	A fitted model from **lme4** (class `merMod`) or **glmmTMB** with a subject-specific random intercept (e.g., `(1|ID)`).
`UsedMethod`	Character; either `"SAGELD"` (default) or `"GALLOP"`.
`PlinkFile`	Character. Path to PLINK .bed file.
`SparseGRMFile`	Character. Path to sparse GRM file produced by `getSparseGRM()`.
`PairwiseIBDFile`	Character. Path to pairwise IBD file produced by `getPairwiseIBD()`.
`PvalueCutoff`	Numeric p-value threshold for screening gene–environment association when estimating λ.
`control`	List of options (forwarded to internal checks; see `checkControl.SAGELD.NullModel`).

Value

A list of class `"SAGELD_NULL_Model"` with elements:

subjData Character vector of subject IDs.

N Number of subjects.

Method Method label: `"SAGELD"` or `"GALLOP"`.

XTs Per-subject sums for crossprod(X, G) terms.

SS Per-subject Rot %*% Si matrices for random effects.

AtS Per-subject cross-products used in variance assembly.

Q Fixed-effect precision matrix (p x p).

A21 Block matrix linking random and fixed effects.

TTs Per-subject sums for crossprod(G).

Tys Per-subject sums for crossprod(G, y).

sol Fixed-effects solution vector.

blups Random-effects BLUPs per subject.

sig Scale parameter extracted from VarCorr.

Resid Residuals used in SAGELD testing.

Resid_G Genetic component residuals.

Resid_GxE GxE component residuals.

Resid_E Environmental component residuals.

Resid.unrelated.outliers Residuals for unrelated outlier subjects.

Resid.unrelated.outliers_G G residuals for unrelated outliers.

Resid.unrelated.outliers_GxE GxE residuals for unrelated outliers.

R_GRM_R Quadratic form Resid' * GRM * Resid (all subjects).

R_GRM_R_G Quadratic form for G residuals.

R_GRM_R_GxE Quadratic form for GxE residuals.

R_GRM_R_G_GxE Cross-term quadratic form between G and GxE.

R_GRM_R_E Quadratic form for E residuals.

R_GRM_R_TwoSubjOutlier Contribution from two-subject outlier families.

R_GRM_R_TwoSubjOutlier_G Two-subject outlier contribution (G).

R_GRM_R_TwoSubjOutlier_GxE Two-subject outlier contribution (GxE).

R_GRM_R_TwoSubjOutlier_G_GxE Two-subject outlier cross-term (G,GxE).

sum_R_nonOutlier Sum of residuals for non-outlier unrelated subjects.

sum_R_nonOutlier_G Sum of G residuals for non-outlier unrelated subjects.

sum_R_nonOutlier_GxE Sum of GxE residuals for non-outlier unrelated subjects.

R_GRM_R_nonOutlier Quadratic form for non-outlier unrelated subjects.

R_GRM_R_nonOutlier_G Quadratic form for G (non-outlier unrelated).

R_GRM_R_nonOutlier_GxE Quadratic form for GxE (non-outlier unrelated).

R_GRM_R_nonOutlier_G_GxE Cross-term for G/GxE (non-outlier unrelated).

TwoSubj_list Per-family lists for N=2 outlier families.

ThreeSubj_list CLT and standardized scores for larger families.

MAF_interval MAF breakpoints used in CLT construction.

zScoreE_cutoff Z-score threshold for E used in screening.

`SPAGRM.NullModel` *Fit SPAGRM null model from residuals and relatedness inputs*

Description

Builds the SPAGRM null model object using subject residuals, sparse GRM, and pairwise IBD estimates, detecting residual outliers and constructing family-level graph structures for downstream saddlepoint marker tests.

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`	Data frame or file path with columns `SubjID`, `Resid`.
`SparseGRMFile`	File path to sparse GRM (tab-delimited: ID1, ID2, Value).
`PairwiseIBDFile`	File path to pairwise IBD table (ID1, ID2, pa, pb, pc).
`control`	List of options controlling outlier handling and family decomposition (see `checkControl.SPAGRM.NullM`

Value

A list of class `"SPAGRM_NULL_Model"` with elements:

Resid Numeric vector of residuals used in analysis.

subjData Character vector of subject IDs (length = N).

N Number of subjects.

Resid.unrelated.outliers Residuals of unrelated outlier subjects.

R_GRM_R Sum of quadratic form Resid' * GRM * Resid for all subjects.

R_GRM_R_TwoSubjOutlier Aggregate contribution from two-subject outlier families.

sum_R_nonOutlier Sum of residuals for non-outlier unrelated subjects.

R_GRM_R_nonOutlier Quadratic form contribution for non-outlier unrelated subjects.

TwoSubj_list List with per two-member family residual/Rho info.

ThreeSubj_list List with Chow–Liu tree structures and standardized scores for larger families.

MAF_interval Vector of MAF breakpoints used in tree construction.

Index

CCT, 2

getPairwiseIBD, 3
getSparseGRM, 4, *6, 7*, *13*
getTempFilesFullGRM, *5*, 6, *6*
GRAB.getGenoInfo, 8
GRAB.makePlink, 9, *26*
GRAB.Marker, 10, *14*
GRAB.NullModel, *4, 5*, *7*, *11*, 12, *22*
GRAB.POLMM, *11–13*, 14, *17*
GRAB.POLMM.Region, *14*, 17, *23*
GRAB.ReadGeno, 19, *22*, *27*
GRAB.Region, *14*, 22
GRAB.SAGELD, *11, 12*, 23
GRAB.SimubVec, 24
GRAB.SimuGMat, 25, *25*
GRAB.SimuGMatFromGenoFile, 27
GRAB.SimuPheno, 28
GRAB.SPACox, *11–13*, 29
GRAB.SPAGRM, *11, 12*, 31
GRAB.SPAmix, *11–13*, 32
GRAB.WtCoxG, *11–13*, 34

SAGELD.NullModel, *11*, 36
SPAGRM.NullModel, *11*, *31*, 38