GEM

Generated by Doxygen 1.9.8

Chapter 1

GEM

GEM (Gene-Environment interaction analysis for Millions of samples) is a software program for large-scale gene-environment interaction testing in cross-sectional and longitudinal data, from unrelated and related individuals. It enables genome-wide association studies in up to millions of samples while allowing for multiple exposures, and control for genotype-covariate interactions.

Current version: 2.2

Additional documentation:
https://large-scale-gxe-methods.github.io/GEMShowcaseWorkspace

1.1 Contents

1.2 Quick Installation

Option 1: Use the binary executable file for Linux

- Download the MKL binary file from: https://github.com/large-scale-gxe-methods/GEM/releases/download/v2.2/GEM_2.2_MKL
- Download the OpenBLAS binary file from: https://github.com/large-scale-gxe-methods/GEM/releases/download/v2.2/GEM_2.2_OpenBLAS

After downloading, make the file executable:

```
chmod +x GEM_2.2_*
```

Option 2: Build GEM Library Dependencies

- C++17 compiler or later
- BLAS/LAPACK. For Intel processors, we recommend that GEM be compiled with an optimized math routine library such as the Intel oneAPI Math Kernal Library to replace BLAS/LAPACK for optimal performance.
- Boost C++ libraries. GEM links to the following Boost libraries: `boost_program_options, boost_thread, boost_system, and boost_filesystem`
- SuiteSparse Library, download and compile it using CMake.

To install GEM, run the following lines of code:

```
git clone https://github.com/large-scale-gxe-methods/GEM
cd GEM/
cmake -B build
cmake --build build
```

1.3 Dependencies

C/C++ Compiler

- A compiler with C++17 (or later) support is required.

LAPACK and BLAS

- The LAPACK (Linear Algebra PACKage) and BLAS (Basic Linear Algebra Subprograms) libraries are used for matrix operations in GEM.

Intel processors:

- We recommend linking GEM to the Intel oneAPI Math Kernal Library (oneMKL), instead of classical BLAS/↩ LAPACK, for a greater performance boost. This can be done by replacing -llapack and -lblas in the makefile with -lmkl_gf_lp64 -lmkl_sequential -lmkl_core before compiling.
  - It is important to compile with -lmkl_sequential since GEM already does multi-threading across SNPs.

AMD processors:

- For AMD processors, OpenBLAS (-lopenblas) may be a better alternative.

Boost C++ Libraries

- The Boost C++ libraries are used for command-line, file management and multi-threading purposes.
- The following Boost libraries are required :
  1. libboost_system
  2. libboost_program_options
  3. libboost_filesystem
  4. libboost_thread

Eigen Library

- The Eigen library is used for linear algebra of dense and sparse matrices.

SuiteSparse Library

- The SuiteSparse library is used for linear algebra operations on dense and sparse matrices. It is particularly utilized in in the GLMM model to fit the null model during association test.

Armadillo Library

- The Armadillo library is used for linear algebra of dense and sparse matrices. It is particularly utilized in G × E (Gene-Environment) interaction test.

1.4 Usage

Running GEM

1. Command Line Options
2. Input Files
3. Output File Format
4. Examples

1.4.1 Command Line Options

Once GEM is installed, the executable `./GEM_2.2` can be used to run the program.
For a list of options, use `./GEM_2.2 --help`.

List of Options

```
**General Options**

--help
  Prints the available options of GEM and exits.

--help
    Prints the available options of GEM and exits.

--version
    Prints the version of GEM and exits.



Input/Output File Options:

--pheno-file
     Path to the phenotype file.
--kin-file
     Path to the kinship file.
--bgen
     Path to the BGEN file.

--sample
     Path to the sample file.
     Required when the BGEN file does not contain sample identifiers.

--pfile
     Path and prefix to the .pgen, .pvar, and .psam files.
     If this flag is used, then --pgen/--pvar/--psam don't need to be specified.

--pgen
     Path to the pgen file.

--pvar
     Path to the pvar file.

--psam
     Path to the psam file.

--bfile
     Path and prefix to the .bed, .bim and .fam files.
     If this flag is used, then --bed/--bim/--fam don't need to be specified.

--bed
     Path to the bed file.

--bim
     Path to the bim file.

--fam
     Path to the fam file.

--out
     Full path and extension to where GEM output results.
     Default: gem.out

--output-style
     Modifies the output of GEM. Must be one of the following:
     minimum: Output the summary statistics for only the GxE and marginal G terms.
     meta: 'minimum' output plus additional fields for the main G and any GxCovariate terms. For a robust
analysis, additional columns for the model-based summary statistics will be included.
     full: 'meta' output plus additional fields needed for re-analyses of a subset of interactions.
          Default: meta



Phenotype File Options:

--sampleid-name
     Column name in the phenotype file that contains sample identifiers.

--pheno-name
     Column name in the phenotype file that contains the phenotype of interest.
     If the number of levels (unique observations) is 2, the phenotype is treated as binary; otherwise it
is assumed to be continuous.

--exposure-names
```

```
     One or more column names in the phenotype file naming the exposure(s) to be included in interaction
tests.
     If no exposures are included, GEM will only perform the marginal test.

--int-covar-names
     Any column names in the phenotype file naming the covariate(s) for which interactions should be
included for adjustment (do not include with --exposure-names).

--covar-names
     Any column names in the phenotype file naming the covariates for which only main effects should be
included for adjustment (do not include with --exposure-names or --int-covar-names).

--random-slope-name
     Column name in the phenotype file that contains random slope.

--group-name
     Column name in the phenotype file that contains the group.

--robust
     0 for model-based standard errors and 1 for robust standard errors.
        Default: 0

--tol
     Convergence tolerance for logistic regression.
        Default: 0.0000001

--delim
     Delimiter separating values in the phenotype file. Tab delimiter should be represented as \t and
space delimiter as \0.
        Default: , (comma-separated)

--missing-value
     Indicates how missing values in the phenotype file are stored.
        Default: NA

--center
     0 for no centering to be done, 1 to center ALL exposures and covariates, and 2 to center all the
interaction covariates only.
          Default: 2

--scale
     0 for no scaling to be done and 1 to scale ALL exposures and covariates by the standard deviation.
        Default: 0

--categorical-names
     Names of the exposure or interaction covariate that should be treated as categorical.
        Default: None

--cat-threshold
     A cut-off to determine which exposure or interaction covariate not specified using
--categorical-names
     should be automatically treated as categorical based on the number of levels (unique observations).
        Default: 2



Kinship File Options:
--kin-delim
     Delimiter separating values in the kinship file. Tab delimiter should be represented as \t and space
delimiter as \0.
        Default: , (comma-separated)

--kin-diag
     Diagonal value of kinship matrix that not accounting for inbreeding.
          Default: 1.0 (for 2x the kinship matrix).



Filtering Options:

--maf
     Variants with a minor allele frequency less than this threshold value (range: [0, 0.5]) will be
excluded.
        Default: 0.001

--miss-geno-cutoff
     Variants with a missing genotype rate greater than this threshold value (range: [0, 1.0]) will be
excluded.
        Default: 0.05

--include-snp-file
     Path to file containing a subset of variants in the specified genotype file to be used for analysis.
     The first line in this file is the header that specifies which variant identifier in the genotype
file
     is used for ID matching. This must be 'snpid' (PLINK or BGEN) or 'rsid' (BGEN only). There should be
one variant
     identifier per line after the header.
```

```
Performance Options:

--threads
     Set number of compute threads.
       Default: ceiling(detected threads / 2)

--stream-snps
     Number of SNPs to analyze in a batch. Memory consumption will increase for larger values of
stream-snps.
        Default: 1
```

1.4.2 Input Files

1.4.2.1 Phenotype File

A file which should contain a sample identifier column and columns for the phenotypes, exposures, and covariates. The ordering of the columns does not matter. All inputs should be coded numerically (e.g., males/females as 0/1)

1.4.2.2 Kinship File

A file contains nonzero values in a three-column format, where the first two columns represent sample identifiers, and the third column indicates the kinship coefficient or genetic relatedness between individuals, or the inbreeding coefficient for the same individual. The scale of the kinship matrix does not matter (e.g., you can use two times the kinship matrix, or identity-by-descent information), as long as the values are on the same scale of the diagonal elements to be added in –kin-diag. For genetic relationship matrices computed from genotypes that have different values on the diagonals, we recommend specifying all diagonal values explicitly in the kinship file, along with –kin-diag 0.

Example 1 of Kinship Matrix (with an inbreeding coefficient of 0.05 for the fifth individual):

```
 0.5    0    0.25  0.25   0
  0    0.5   0.25  0.25   0
 0.25  0.25  0.5   0.25   0
 0.25  0.25  0.25  0.5    0
  0     0     0     0    0.55
```

The Kinship File Corresponding to Example 1 (along with –kin-diag 0.5):

```
| ID1 | ID2 | Kinship|
|-----|-----|--------|
|  1  |  3  |  0.25  |
|  1  |  4  |  0.25  |
|  2  |  3  |  0.25  |
|  2  |  4  |  0.25  |
|  3  |  4  |  0.25  |
|  5  |  5  |  0.05  |
```

Example 2 of Kinship Matrix (two times the kinship matrix in Example 1):

```
  1    0    0.5   0.5   0
  0    1    0.5   0.5   0
 0.5  0.5    1    0.5   0
 0.5  0.5   0.5    1    0
  0    0     0     0   1.1
```

The Kinship File Corresponding to Example 2 (along with –kin-diag 1):

ID1	ID2	Kinship
1	3	0.5
1	4	0.5
2	3	0.5
2	4	0.5
3	4	0.5
5	5	0.1

1.4.2.3 Genotype Files

BGEN

Variants that are non-biallelic should be filtered from the BGEN file. Note that since there are no indication of a REF/ALT allele in the BGEN file, the second allele is the effect allele counted in association testing.

A `.sample file` is required as input when the .bgen file does not contain a sample identifier block.

Plink BED

`**.fam**` - The .fam file can be space or tab-delimited and must contain at least 2 columns where the first column is the family ID (FID) and the second column is the individual ID (IID). GEM will use the IID column for sample identifier matching with the phenotype file.

`**.bim**` - The .bim file can also be space or tab-delimited and should be in the following order: the chromosome, variant id, cM (optional), base-pair coordinate, ALT allele, and REF allele.

`**.bed**` - A bed file must be stored in variant-major form. The ALT allele specified in the .bim file is the effect allele counted in association testing.

Plink 2.0 PGEN

`**.psam**` - The .psam file is a tab-delimited text file containing the sample information. If header lines are present, the last header line should contain a column with the name #IID (if the first column is not #FID) or IID (if the first column is #FID) that holds the individual ID for sample identifier matching with the phenotype file. All previous header lines will be ignored. If no header line beginning with #IID or #FID is present, then the columns are assumed to be in .fam file order.

`**.pvar**` - The .pvar file is a tab-delimited text file containing the variant information. If header lines are present, the last header line should start with #CHROM. If #CHROM is present, then the columns POS, ID, REF, and ALT must also be present. All previous header lines will be ignored. If the .pvar file contain no header lines beginning with #CHROM, it is assumed that the columns are in .bim file order.

`**.pgen**` - The .pgen file should be filtered for non-biallelic variants. The ALT allele specified in the .pvar file is the effect allele counted in association testing.

1.4.3 Output File Format

GEM will write results to the output file specified with the –out parameter (or 'gem.out' if no output file is specified). Below are details of the possible column headers in the output file.

```
SNPID              - The SNP identifier as retrieved from the genotype file.
RSID               - The reference SNP ID number. (BGEN only)
CHR                - The chromosome of the SNP.
POS                - The physical position of the SNP.
Non_Effect_Allele  - The allele not counted in association testing.
Effect_Allele      - The allele that is counted in association testing.
N_Samples          - The number of samples without missing genotypes.
AF                 - The allele frequency of the effect allele.
GV                 - The variance of the effected allele.
N_catE_*           - The number of non-missing samples in each combination of strata for all of the
      categorical exposures and interaction covariates.
AF_catE_*          - The allele frequency of the effect allele for each combination of strata for all of the
      catgorical exposure or interaction covariate.
GV_catE_*          - The variance of the effect allele for each combination of strata for all of the
      catgorical exposure or interaction covariate.

Beta_Marginal           - The coefficient estimate for the marginal genetic effect (i.e., from a model with
      no interaction terms).
SE_Beta_Marginal        - The model-based SE associated with the marginal genetic effect estimate.
robust_SE_Beta_Marginal - The robust SE associated with the marginal genetic effect estimate.

Beta_G             - The coefficient estimate for the genetic main effect (G) (i.e., from a model with
      interaction terms).
Beta_G-*           - The coefficient estimate for the interaction or interaction covariate terms.
SE_Beta_G          - Model-based SE associated with the the genetic main effect (G).
SE_Beta_G-*        - Model-based SE associated with any GxE or interaction covariate terms.
robust_SE_Beta_G   - Robust SE associated with the the genetic main effect (G).
robust_SE_Beta_G-* - Robust SE associated with any GxE or interaction covariate terms.
Cov_Beta_G_G-*          - Model-based covariance between the genetic main effect (G) and any GxE or
      interaction covariate terms.
Cov_Beta_G-*_G-*        - Model-based covariance between any GxE or interaction covariate terms.
robust_Cov_Beta_G_G-*   - Robust covariance between the genetic main effect (G) and any GxE or interaction
      covariate terms.
robust_Cov_Beta_G-*_G-* - Robust covariance between any GxE or interaction covariate terms.

P_Value_Marginal           - Marginal genetic effect p-value from model-based SE.
P_Value_Interaction        - Interaction effect p-value (K degrees of freedom test of interaction effect)
      from model-based SE. (K is number of major exposures)
P_Value_Joint              - Joint test p-value (K+1 degrees of freedom test of genetic and interaction
      effect) from model-based SE.
robust_P_Value_Marginal    - Marginal genetic effect p-value from robust SE.
robust_P_Value_Interaction - Interaction effect p-value from robust SE.
robust_P_Value_Joint       - Joint test p-value (K+1 degrees of freedom test of genetic and interaction
      effect) from robust SE.
```

The –output-style flag can be used to specify which columns should be included in the output file:

1.4.3.1 minimum:

Includes the variant information, Beta_Marginal, SE_Beta_Marginal, coefficient estimates for only the GxE terms, and depending on the –robust option, SE and covariance for only the GxE terms.

1.4.3.2 meta:

Includes each of the possible outputs listed above when applicable. For a model-based analysis (–robust 0), the columns containing the "robust" prefix (robust_$*$) are excluded in the output file.

1.4.3.3 full:

Includes, in addition to "meta", an initial header line with the residual variance estimate necessary for re-analysis of a subset of interactions using only summary statistics (for example, switching an exposure and interaction covariate).

1.4.4 Examples

To run GEM using the example data, execute GEM with the following code.

Example for cross-sectional data without kinship:

```
./GEM --bgen example.bgen --sample example.sample --pheno-file example.pheno --sampleid-name sampleid
      --pheno-name pheno2 --covar-names cov3 --exposure-names cov1 --out cross_sectional_without_kinship.out
```

The results should look like the following output file `cross_sectional_without_kinship.out`.

Example for cross-sectional data with kinship:

```
./GEM --kin-file example.kinship --kin-diag 0.5 --pheno-file example.pheno --pheno-name pheno2
      --sampleid-name sampleid --exposure-names cov1  --covar-names cov3 --random-slope-name cov3 --bgen
      example.bgen --sample example.sample --output-style meta --out cross_sectional_with_kinship.out
```

The results should look like the following output file `cross_sectional_with_kinship.out`.

Example for longitudinal data without kinship:

```
./GEM --pheno-file example.pheno2 --pheno-name pheno2 --sampleid-name sampleid --exposure-names cov1
      --covar-names cov3 --random-slope-name cov3 --bgen example.bgen --sample example.sample --output-style
      meta --out longitudinal_without_kinship.out
```

The results should look like the following output file `longitudinal_without_kinship.out`.

Example for longitudinal data with kinship:

```
./GEM --kin-file example.kinship --kin-diag 0.5 --pheno-file example.pheno2 --pheno-name pheno2
      --sampleid-name sampleid --exposure-names cov1  --covar-names cov3 --random-slope-name cov3 --bgen
      example.bgen --sample example.sample --output-style  meta --out longitudinal_with_kinship.out
```

The results should look like the following output file `longitudinal_with_kinship.out`.

1.5 Recent Updates

`Version 2.2` - March 4, 2026:

- Added convergence check for logistic regression. The program now reports detailed coefficient estimates when the model fails to converge and exits safely after 500 iterations.
- Added support for `group` variable in null model fitting to allow heteroscedastic residual variance across sample groups.
- Improved memory handling and internal limits to support larger numbers of observations (larger sample sizes)

`Version 2.1.3` - June 14, 2025:

- Added log file support
- Updated headers in the output file

Version 2.1.2 - May 27, 2025:

- Added output columns GV and GV_catE_$*$

Version 2.1.1 - April 18, 2025:

- Logged the SNP ID when a singular matrix is detected

Version 2.1 - March 10, 2025:

- Added support for GLMM on PGEN and BED files
- Added support for GEI test on PGEN and BED files

Version 2.0 - February 14, 2025:

- Added generalized linear mixed model (GLMM)
- Added gen-environment interaction (GEI) test

Version 1.5.3 - May 20, 2024:

- Included stratified values for binary outcomes

Version 1.5.2 - August 16, 2023:

- Fixed the output when there is no exposure

Version 1.5.1 - April 20, 2023:

- Treated empty strings as missing values
- Fixed a bug for empty strings at the end of each line
- Minor changes to messages printed to stdout
- Error out if the sample size is not greater than the number of predictors (intercept, exposures, interaction covariates, and covariates) in the null model fitting

Version 1.5 - March 9, 2023:

- Changed the default of the –center flag to 2 to center all the interaction covariates only

Version 1.4.5 - November 11, 2022:

- Added collinearity check of the covariates before fitting the null model

Version 1.4.4 - October 5, 2022:

- Fixed the bugs of include-snp-file
- Removed the default value of flag "--center"

`Version 1.4.3` - March 23, 2022:

- Sorted the output headers of categorical variables

`Version 1.4.2` - November 22, 2021:

- Add math.h library to install GEM through Docker desktop
- Added a binary executable file

`Version 1.4.1` - September 14, 2021:

- Added to read phenotype files created from the Windows system

`Version 1.4` - July 2, 2021:

- Remove –pheno-type flag. If the number of levels (unique observations) is 2, the phenotype is treated as binary; otherwise it is assumed to be continuous
- Check for categorical exposures and interaction covariates
- Output number of non-missing samples (N) and allele frequency (AF) for effect allele for each combination of strata for all exposures and interaction covariates
- Add two additional flags –categorical-names and –cat-threshold for user definition of categorical variables
- Output the SE instead of variance for the coefficient estimates
- Output only the lower triangle of the covariance matrix instead of the full matrix
- For robust analysis and "meta"/"full" output style, include model-based summary statistics in the output file
- Column names for the robust summary statistics will include the prefix "robust_"
- For "full" output style, an initial header line with the dispersion is included in the output file
- The V matrix no longer included in the output file for "full" output style

`Version 1.3` - April 7, 2021:

- Add a new flag (–output-style) to modify which summary statistics should be included in the the output file. Column names now include the exposure and interaction covariate names instead of numbers.
- The –exposure-names flag is now optional. If no exposures are specified, GEM will run a G-only model. Covariates (not of interest) can still be adjusted for using –covar-names flag.

`Version 1.2` - January 22, 2021:

- Fix issue to allow for space and tab delimited phenotype files.

- Allow for centering and scaling of exposures and covariates.
- Update calculations for model-based joint test
- Update calculations for robust joint test
- Output covariance, coefficients and standard errors to the log.
- Change Allele1 and Allele2 in outfile file to Non_Effect_Allele and Effect_Allele.
- Fix bug when phenotype is binary and there are missing genotypes.
- Support PGEN/BED files.

Version 1.1 - July 21, 2020:

- Allow GEM to subset the BGEN file based on a list of variants to include for analysis. –include-snp-file
- Use matrix operation to adjust for covariates instead of for-loop. Use the libdeflate package for faster zlib decompression of BGEN genotype blocks. Compile GEM with -O2 (optimizer flag). Prioritize BGEN sample file over the BGEN sample identifier block. Error if phenotype (–pheno-name) is also included as an exposoure or covariate
- Support BGEN v1.1, v1.2 and v1.3 uncompressed genotype blocks.
- Fix major printing bug.
- Handle missing genotypes in BGEN files.

1.6 Contact

For comments, suggestions, bug reports and questions, please contact Han Chen (hanchenphd@gmail.com), Alisa Manning (AKMANNING@mgh.harvard.edu), Kenny Westerman (KEWESTERMAN@mgh.harvard.edu) or Samaneh Salehi Nasab (Samaneh.SalehiNasab@uth.tmc.edu). For bug reports, please include an example to reproduce the problem without having to access your confidential data.

1.7 References

If you use GEM in your analysis, please cite

- Westerman KE, Pham DT, Hong L, Chen Y, Sevilla-González M, Sung YJ, Sun YV, Morrison AC, Chen H, Manning AK. (2021) GEM: scalable and flexible gene-environment interaction analysis in millions of samples. Bioinformatics 37(20):3514-3520. PubMed PMID: **34695175**. PMCID: **PMC8545347**. DOI: **10.1093/bioinformatics/btab223**.

1.8 License

```
GEM : Gene-Environment interaction analysis for Millions of samples
Copyright (C) 2018-2026  Liang Hong, Han Chen, Duy Pham, Cong Pan, Samaneh Salehi Nasab

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

The GEM package is distributed under GPL (>= 3). It includes source code from open source third-party software:

- libdeflate: MIT
- Plink: LGPLv3+
- Zstandard (zstd): BSD_3_clause | GPL-2

  The binary release of GEM also links to third-party libraries:
- Boost: Boost Software License, Version 1.0
- Eigen: Mozilla Public License, Version 2.0
- Intel oneAPI Math Kernel Library (oneMKL): Intel Simplified Software License (Version October 2022 or later)
- Armadillo: Apache License 2.0
- SuiteSparse:
  - libcholmod: GNU Lesser General Public License (LGPL), version 2.1 or later
  - libcxsparse: GNU Lesser General Public License (LGPL), version 2.1 or later
  - libspqr:GNU General Public License (GPL), version 2 or later
  - libumfpack: GNU General Public License (GPL), version 2 or later
  - libcamd: BSD 3-Clause License
  - libccolamd: BSD 3-Clause License
  - libcolamd: BSD 3-Clause License
  - libamd: BSD 3-Clause License
  - libsuitesparseconfig: BSD-3-clause
- fmt: MIT License

  Full copies of license agreements for GEM, third-party source code, linked libraries can be found here.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 details Namespace Reference

Functions

- SpaMat diag (DensVec vec)
- SpaMat diag (std::ext::V_double vec)
- void add_dquot (std::ext::V_string &v_strs)

6.1.1 Function Documentation

6.1.1.1 add_dquot()

```
void details::add_dquot (
            std::ext::V_string & v_strs )
```

6.1.1.2 diag() [1/2]

```
SpaMat details::diag (
            DensVec vec )
```

6.1.1.3 diag() [2/2]

```
SpaMat details::diag (
            std::ext::V_double vec )
```

6.2 std Namespace Reference

Namespaces

- namespace ext

6.3 std::ext Namespace Reference

Typedefs

- using FitNull_f = std::function< void(int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret, std::vector< double > &beta_ret, std::vector< double > &Xbeta_ret)>

  Function signature for the GEM null model fitting routine.
- using Matrix_variant = std::variant< DensMat, SpaMat >

  Variant type for storing dense or sparse matrices.
- using V_string = std::vector< std::string >
- using VV_string = std::vector< V_string >
- using Data = unordered_map< string, V_string >
- using V_double = std::vector< double >
- using V_float = std::vector< float >
- using V_int = std::vector< int >
- using V_size_t = std::vector< std::size_t >
- using VV_int = std::vector< V_int >
- using V_bool = std::vector< bool >
- using V_lluint = std::vector< long long unsigned int >
- using map_str_int = std::map< std::string, int >
- using Umap_str_int = std::unordered_map< std::string, int >
- using V_opt_string = std::vector< std::optional< std::string > >
- using Var_bool_int = std::variant< V_bool, V_int >
- using Map_str_Vint = std::map< string, V_int >
- using IndexMapUnq = unordered_map< string, int >
- using IndexMap = unordered_map< string, V_int >
- using IndexMapRev = unordered_map< int, string >
- using Triplet_d = Eigen::Triplet< double >
- using Triplet_i = Eigen::Triplet< int >
- using VecTuples4spmat = vector< Triplet_d >
- using VecTriple_i = vector< Triplet_i >

Functions

- template<typename Tuple , size_t... Indices>
  void tuplePrint (const Tuple &t, std::index_sequence< Indices... >)
- template<typename... Args>
  void tuplePrint (const std::tuple< Args... > &t)

6.3.1 Typedef Documentation

6.3.1.1 Data

```
using std::ext::Data = typedef unordered_map<string, V_string>
```

6.3.1.2 FitNull_f

```
using std::ext::FitNull_f = typedef std::function<void (int samSize, int numSelCol, int pheno↩
Type, double epsilon, int robust, std::vector<string> covSelHeadersName, std::vector<double>
phenodata, std::vector<double> covdata, std::vector<double>* XinvXTX_ret, vector<double>*
miu_ret, vector<double>* resid_ret, double* sigma2_ret, std::vector<double>& beta_ret, std↩
::vector<double>& Xbeta_ret)>
```

Function signature for the GEM null model fitting routine.

This alias defines the callable interface used to invoke the null model fitting implementation (e.g., `fitNull↩Model2`). It returns results such as fitted values, residuals, regression coefficients, and variance estimates.

6.3.1.3 IndexMap

```
using std::ext::IndexMap = typedef unordered_map<string, V_int>
```

6.3.1.4 IndexMapRev

```
using std::ext::IndexMapRev = typedef unordered_map<int, string>
```

6.3.1.5 IndexMapUnq

```
using std::ext::IndexMapUnq = typedef unordered_map<string, int>
```

6.3.1.6 map_str_int

```
using std::ext::map_str_int = typedef std::map<std::string, int>
```

6.3.1.7 Map_str_Vint

```
using std::ext::Map_str_Vint = typedef std::map<string, V_int>
```

6.3.1.8 Matrix_variant

```
using std::ext::Matrix_variant = typedef std::variant<DensMat, SpaMat>
```

Variant type for storing dense or sparse matrices.

This type is used to represent matrices that may be stored either in dense format (`DensMat`) or sparse format (`SpaMat`).

6.3.1.9 Triplet_d

```
using std::ext::Triplet_d = typedef Eigen::Triplet<double>
```

6.3.1.10 Triplet_i

```
using std::ext::Triplet_i = typedef Eigen::Triplet<int>
```

6.3.1.11 Umap_str_int

```
using std::ext::Umap_str_int = typedef std::unordered_map<std::string, int>
```

6.3.1.12 V_bool

```
using std::ext::V_bool = typedef std::vector<bool>
```

6.3.1.13 V_double

```
using std::ext::V_double = typedef std::vector<double>
```

6.3.1.14 V_float

```
using std::ext::V_float = typedef std::vector<float>
```

6.3.1.15 V_int

```
using std::ext::V_int = typedef std::vector<int>
```

6.3.1.16 V_lluint

```
using std::ext::V_lluint = typedef std::vector<long long unsigned int>
```

6.3.1.17 V_opt_string

```
using std::ext::V_opt_string = typedef std::vector<std::optional<std::string> >
```

6.3.1.18 V_size_t

```
using std::ext::V_size_t = typedef std::vector<std::size_t>
```

6.3.1.19 V_string

```
using std::ext::V_string = typedef std::vector<std::string>
```

6.3.1.20 Var_bool_int

```
using std::ext::Var_bool_int = typedef std::variant<V_bool, V_int>
```

6.3.1.21 VecTriple_i

```
using std::ext::VecTriple_i = typedef vector<Triplet_i>
```

6.3.1.22 VecTuples4spmat

```
using std::ext::VecTuples4spmat = typedef vector<Triplet_d>
```

6.3.1.23 VV_int

```
using std::ext::VV_int = typedef std::vector<V_int>
```

6.3.1.24 VV_string

```
using std::ext::VV_string = typedef std::vector<V_string>
```

6.3.2 Function Documentation

6.3.2.1 tuplePrint() [1/2]

```
template<typename... Args>
void std::ext::tuplePrint (
            const std::tuple< Args... > & t )
```

6.3.2.2 tuplePrint() [2/2]

```
template<typename Tuple , size_t... Indices>
void std::ext::tuplePrint (
            const Tuple & t,
            std::index_sequence< Indices... > )
```

Chapter 7

Class Documentation

7.1 Bed Class Reference

`#include <ReadBed.h>`

Public Member Functions

- void processBed (string bedFile, string bimFile, string famFile)
- void processFam (Bed bed, string famFile, unordered_map< string, vector< vector< string > > > phenomap, string phenoMissingKey, int numSelCol, int samSize)
- void getBedVariantPos (Bed bed, CommandLine cmd)

Public Attributes

- uint32_t n_samples = 0
- uint32_t n_variants = 0
- char famDelim
- char bimDelim
- int bimLast
- int new_samSize
- std::vector< string > sampleID
- std::vector< string > sampleID_all
- std::vector< double > new_covdata
- std::vector< double > new_phenodata
- std::vector< long int > include_idx
- int numIntSelCol_new
- int numExpSelCol_new
- int numSelCol_new
- std::vector< int > excludeCol
- int phenoType
- vector< uint32_t > begin
- vector< uint32_t > end
- uint32_t threads
- bool filterVariants = false
- std::vector< long long unsigned int > bedVariantPos

7.1.1 Member Function Documentation

7.1.1.1 getBedVariantPos()

```
void Bed::getBedVariantPos (
            Bed bed,
            CommandLine cmd )
```

7.1.1.2 processBed()

```
void Bed::processBed (
            string bedFile,
            string bimFile,
            string famFile )
```

7.1.1.3 processFam()

```
void Bed::processFam (
            Bed bed,
            string famFile,
            unordered_map< string, vector< vector< string > > > phenomap,
            string phenoMissingKey,
            int numSelCol,
            int samSize )
```

7.1.2 Member Data Documentation

7.1.2.1 bedVariantPos

```
std::vector<long long unsigned int> Bed::bedVariantPos
```

7.1.2.2 begin

```
vector<uint32_t> Bed::begin
```

7.1.2.3 bimDelim

```
char Bed::bimDelim
```

7.1.2.4 bimLast

```
int Bed::bimLast
```

7.1.2.5 end

```
vector<uint32_t> Bed::end
```

7.1.2.6 excludeCol

```
std::vector<int> Bed::excludeCol
```

7.1.2.7 famDelim

```
char Bed::famDelim
```

7.1.2.8 filterVariants

```
bool Bed::filterVariants = false
```

7.1.2.9 include_idx

```
std::vector<long int> Bed::include_idx
```

7.1.2.10 n_samples

```
uint32_t Bed::n_samples = 0
```

7.1.2.11 n_variants

```
uint32_t Bed::n_variants = 0
```

7.1.2.12 new_covdata

```
std::vector<double> Bed::new_covdata
```

7.1.2.13 new_phenodata

```
std::vector<double> Bed::new_phenodata
```

7.1.2.14 new_samSize

```
int Bed::new_samSize
```

7.1.2.15 numExpSelCol_new

```
int Bed::numExpSelCol_new
```

7.1.2.16 numIntSelCol_new

```
int Bed::numIntSelCol_new
```

7.1.2.17 numSelCol_new

```
int Bed::numSelCol_new
```

7.1.2.18 phenoType

```
int Bed::phenoType
```

7.1.2.19 sampleID

```
std::vector<string> Bed::sampleID
```

7.1.2.20 sampleID_all

```
std::vector<string> Bed::sampleID_all
```

7.1.2.21 threads

```
uint32_t Bed::threads
```

The documentation for this class was generated from the following files:

- include/ReadBed.h
- src/ReadBed.cpp

7.2 Bgen Class Reference

Reader and processor for BGEN genotype files (v1.1–v1.3).

```
#include <ReadBGEN.h>
```

Public Member Functions

- void processBgenHeaderBlock (string bgenfile)
- void processBgenSampleBlock (Bgen bgen, char samplefile[300], bool useSample, unordered_map< string, vector< vector< string > > > phenomap, string phenoMissingKey, int numSelCol, int samSize)
- void getPositionOfBgenVariant (Bgen bgen, CommandLine cmd)

Public Attributes

- FILE * fin
- uint offset
- uint Mbgen
- uint Nbgen
- uint CompressedSNPBlocks
- uint Layout
- uint SampleIdentifiers
- int new_samSize
- std::vector< string > sampleID
- std::vector< string > sampleID_all
- std::vector< double > new_covdata
- std::vector< double > new_phenodata
- std::vector< long int > include_idx
- vector< long int > variant_pos
- std::vector< unsigned int > includeVariantIndex
- int numIntSelCol_new
- int numExpSelCol_new
- int numSelCol_new
- std::vector< int > excludeCol
- int phenoType
- uint threads
- bool filterVariants
- std::vector< uint > Mbgen_begin
- std::vector< uint > Mbgen_end
- std::vector< long long unsigned int > bgenVariantPos
- std::vector< vector< uint > > keepVariants

7.2.1 Detailed Description

Reader and processor for BGEN genotype files (v1.1–v1.3).

This class provides utilities for:

- Reading the BGEN header block (variant/sample metadata)
- Loading sample identifiers (from file or embedded block)
- Matching genotype samples with phenotype/covariate data
- Locating variant byte positions for multithreaded processing

7.2.2 Member Function Documentation

7.2.2.1 getPositionOfBgenVariant()

```
void Bgen::getPositionOfBgenVariant (
            Bgen bgen,
            CommandLine cmd )
```

7.2.2.2 processBgenHeaderBlock()

```
void Bgen::processBgenHeaderBlock (
            string bgenfile )
```

7.2.2.3 processBgenSampleBlock()

```
void Bgen::processBgenSampleBlock (
            Bgen bgen,
            char samplefile[300],
            bool useSample,
            unordered_map< string, vector< vector< string > > > phenomap,
            string phenoMissingKey,
            int numSelCol,
            int samSize )
```

7.2.3 Member Data Documentation

7.2.3.1 bgenVariantPos

```
std::vector<long long unsigned int> Bgen::bgenVariantPos
```

7.2.3.2 CompressedSNPBlocks

```
uint Bgen::CompressedSNPBlocks
```

7.2.3.3 excludeCol

```
std::vector<int> Bgen::excludeCol
```

7.2.3.4 filterVariants

```
bool Bgen::filterVariants
```

7.2.3.5 fin

```
FILE* Bgen::fin
```

7.2.3.6 include_idx

```
std::vector<long int> Bgen::include_idx
```

7.2.3.7 includeVariantIndex

```
std::vector<unsigned int> Bgen::includeVariantIndex
```

7.2.3.8 keepVariants

```
std::vector<vector<uint> > Bgen::keepVariants
```

7.2.3.9 Layout

```
uint Bgen::Layout
```

7.2.3.10 Mbgen

```
uint Bgen::Mbgen
```

7.2.3.11 Mbgen_begin

```
std::vector<uint> Bgen::Mbgen_begin
```

7.2.3.12 Mbgen_end

```
std::vector<uint> Bgen::Mbgen_end
```

7.2.3.13 Nbgen

```
uint Bgen::Nbgen
```

7.2.3.14 new_covdata

```
std::vector<double> Bgen::new_covdata
```

7.2.3.15 new_phenodata

```
std::vector<double> Bgen::new_phenodata
```

7.2.3.16 new_samSize

```
int Bgen::new_samSize
```

7.2.3.17 numExpSelCol_new

```
int Bgen::numExpSelCol_new
```

7.2.3.18 numIntSelCol_new

```
int Bgen::numIntSelCol_new
```

7.2.3.19 numSelCol_new

```
int Bgen::numSelCol_new
```

7.2.3.20 offset

```
uint Bgen::offset
```

7.2.3.21 phenoType

```
int Bgen::phenoType
```

7.2.3.22 sampleID

```
std::vector<string> Bgen::sampleID
```

7.2.3.23 sampleID_all

```
std::vector<string> Bgen::sampleID_all
```

7.2.3.24 SampleIdentifiers

```
uint Bgen::SampleIdentifiers
```

7.2.3.25 threads

```
uint Bgen::threads
```

7.2.3.26 variant_pos

```
vector<long int> Bgen::variant_pos
```

The documentation for this class was generated from the following files:

- include/ReadBGEN.h
- src/ReadBGEN.cpp

7.3 BinE Class Reference

```
#include <BinaryEUtils.h>
```

Public Member Functions

- void checkBinaryCovariates (BinE binE, CommandLine cmd, unordered_map< string, vector< vector< string > > > phenoMap, vector< string > sampleID, vector< long int > include_idx, int samSize, int phenoType, string phenoHeaderName, std::vector< string > covSelHeadersName, std::vector< string > covSelHeadersName_new, int Sq_new)

Public Attributes

- int nBinE = 0
- int strataLen = 0
- std::vector< int > stratum_idx
- std::vector< string > bin_headers

7.3.1 Member Function Documentation

7.3.1.1 checkBinaryCovariates()

```
void BinE::checkBinaryCovariates (
            BinE binE,
            CommandLine cmd,
            unordered_map< string, vector< vector< string > > > phenoMap,
            vector< string > sampleID,
            vector< long int > include_idx,
            int samSize,
            int phenoType,
            string phenoHeaderName,
            std::vector< string > covSelHeadersName,
            std::vector< string > covSelHeadersName_new,
            int Sq_new )
```

7.3.2 Member Data Documentation

7.3.2.1 bin_headers

```
std::vector<string> BinE::bin_headers
```

7.3.2.2 nBinE

```
int BinE::nBinE = 0
```

7.3.2.3 strataLen

```
int BinE::strataLen = 0
```

7.3.2.4 stratum_idx

```
std::vector<int> BinE::stratum_idx
```

The documentation for this class was generated from the following files:

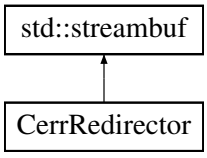
- include/BinaryEUtils.h
- src/BinaryEUtils.cpp

7.4 CerrRedirector Class Reference

Redirects standard error output (std::cerr) into the spdlog logging system.

```
#include <Logger.h>
```

Inheritance diagram for CerrRedirector:



Protected Member Functions

- int overflow (int c) override
- int sync () override

7.4.1 Detailed Description

Redirects standard error output (std::cerr) into the spdlog logging system.

This class captures all output written to std::cerr and forwards it to spdlog using the error-level logger.

Characters are buffered until a newline or flush occurs.

7.4.2 Member Function Documentation

7.4.2.1 overflow()

```
int CerrRedirector::overflow (
            int c )  [override], [protected]
```

7.4.2.2 sync()

```
int CerrRedirector::sync ( )  [override], [protected]
```

The documentation for this class was generated from the following files:

- include/Logger.h
- src/Logger.cpp

7.5 CommandLine Class Reference

```
#include <ReadParameters.h>
```

Public Member Functions

- void processCommandLine (int argc, char *argv[ ])

Public Attributes

- char samplefile [300]
- std::string bgenFile
- std::string sampleFile
- std::string pgenFile
- std::string pvarFile
- std::string psamFile
- std::string bedFile
- std::string bimFile
- std::string famFile
- bool useSampleFile = false
- bool useBgenFile = false
- bool usePgenFile = false
- bool useBedFile = false
- std::string pheno_file
- std::string delim
- std::string missing
- char pheno_delim
- std::string kin_file
- std::string delim_k
- bool kin_flag = false
- double kin_diag
- char kin_delim

- int cat_threshold
- std::vector< std::string > cat_names
- std::string outFile
- std::string outStyle
- std::string phenoName
- std::string randomSlope
- std::string sampleID
- int numSelCol = 0
- int numExpSelCol = 0
- int numIntSelCol = 0
- std::vector< std::string > cov
- std::vector< std::string > icov
- std::vector< std::string > exp
- std::string group
- std::unordered_map< std::string, int > covHM
- std::unordered_map< std::string, int > intHM
- std::unordered_map< std::string, int > expHM
- double MAF
- double missGenoRate
- std::string includeVariantFile
- bool doFilters
- int center
- int scale
- double tol
- int robust
- int threads
- int stream_snps

7.5.1 Member Function Documentation

7.5.1.1 processCommandLine()

```
void CommandLine::processCommandLine (
            int argc,
            char * argv[] )
```

7.5.2 Member Data Documentation

7.5.2.1 bedFile

```
std::string CommandLine::bedFile
```

7.5.2.2 bgenFile

```
std::string CommandLine::bgenFile
```

7.5.2.3 bimFile

```
std::string CommandLine::bimFile
```

7.5.2.4 cat_names

```
std::vector<std::string> CommandLine::cat_names
```

7.5.2.5 cat_threshold

```
int CommandLine::cat_threshold
```

7.5.2.6 center

```
int CommandLine::center
```

7.5.2.7 cov

```
std::vector<std::string> CommandLine::cov
```

7.5.2.8 covHM

```
std::unordered_map<std::string, int> CommandLine::covHM
```

7.5.2.9 delim

```
std::string CommandLine::delim
```

7.5.2.10 delim_k

```
std::string CommandLine::delim_k
```

7.5.2.11 doFilters

```
bool CommandLine::doFilters
```

7.5.2.12 exp

```
std::vector<std::string> CommandLine::exp
```

7.5.2.13 expHM

```
std::unordered_map<std::string, int> CommandLine::expHM
```

7.5.2.14 famFile

```
std::string CommandLine::famFile
```

7.5.2.15 group

```
std::string CommandLine::group
```

7.5.2.16 icov

```
std::vector<std::string> CommandLine::icov
```

7.5.2.17 includeVariantFile

```
std::string CommandLine::includeVariantFile
```

7.5.2.18 intHM

```
std::unordered_map<std::string, int> CommandLine::intHM
```

7.5.2.19 kin_delim

```
char CommandLine::kin_delim
```

7.5.2.20 kin_diag

```
double CommandLine::kin_diag
```

7.5.2.21 kin_file

```
std::string CommandLine::kin_file
```

7.5.2.22 kin_flag

```
bool CommandLine::kin_flag = false
```

7.5.2.23 MAF

```
double CommandLine::MAF
```

7.5.2.24 missGenoRate

```
double CommandLine::missGenoRate
```

7.5.2.25 missing

```
std::string CommandLine::missing
```

7.5.2.26 numExpSelCol

```
int CommandLine::numExpSelCol = 0
```

7.5.2.27 numIntSelCol

```
int CommandLine::numIntSelCol = 0
```

7.5.2.28 numSelCol

```
int CommandLine::numSelCol = 0
```

7.5.2.29 outFile

```
std::string CommandLine::outFile
```

7.5.2.30 outStyle

```
std::string CommandLine::outStyle
```

7.5.2.31 pgenFile

```
std::string CommandLine::pgenFile
```

7.5.2.32 pheno_delim

```
char CommandLine::pheno_delim
```

7.5.2.33 pheno_file

```
std::string CommandLine::pheno_file
```

7.5.2.34 phenoName

```
std::string CommandLine::phenoName
```

7.5.2.35 psamFile

```
std::string CommandLine::psamFile
```

7.5.2.36 pvarFile

```
std::string CommandLine::pvarFile
```

7.5.2.37 randomSlope

```
std::string CommandLine::randomSlope
```

7.5.2.38 robust

```
int CommandLine::robust
```

7.5.2.39 samplefile

```
char CommandLine::samplefile[300]
```

7.5.2.40 sampleFile

```
std::string CommandLine::sampleFile
```

7.5.2.41 sampleID

```
std::string CommandLine::sampleID
```

7.5.2.42 scale

```
int CommandLine::scale
```

7.5.2.43 stream_snps

```
int CommandLine::stream_snps
```

7.5.2.44 threads

```
int CommandLine::threads
```

7.5.2.45 tol

```
double CommandLine::tol
```

7.5.2.46 useBedFile

```
bool CommandLine::useBedFile = false
```

7.5.2.47 useBgenFile

```
bool CommandLine::useBgenFile = false
```

7.5.2.48 usePgenFile

```
bool CommandLine::usePgenFile = false
```

7.5.2.49 useSampleFile

```
bool CommandLine::useSampleFile = false
```

The documentation for this class was generated from the following files:

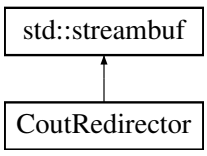
- include/ReadParameters.h
- src/ReadParameters.cpp

7.6 CoutRedirector Class Reference

Redirects standard output (std::cout) into the spdlog logging system.

```
#include <Logger.h>
```

Inheritance diagram for CoutRedirector:



Protected Member Functions

- int overflow (int c) override
- int sync () override

7.6.1 Detailed Description

Redirects standard output (std::cout) into the spdlog logging system.

This class inherits from std::streambuf and overrides the virtual functions overflow() and sync() in order to intercept characters written to std::cout.

Typical use:

- Installed via std::cout.rdbuf(&redirector)
- Enables capturing of all std::cout output into log files and console sinks.

7.6.2 Member Function Documentation

7.6.2.1 overflow()

```
int CoutRedirector::overflow (
            int c )  [override], [protected]
```

7.6.2.2 sync()

```
int CoutRedirector::sync ( )  [override], [protected]
```

The documentation for this class was generated from the following files:

- include/Logger.h
- src/Logger.cpp

7.7 DataFrame Class Reference

A class for storing infromation in the tabular files.

```
#include <ReadFiles.h>
```

Public Member Functions

- bool isDataFrameCreated () const
- int n_rows () const
- int n_cols () const
- void head (int n=5)

  Print the n rows of the dataframe.
- DataFrame copy_by_hdrs (std::ext::V_string const &v_hdrs)

  Make a copy of data from based on a given headers.
- void read_file (std::string_view path, char delim=',')

  Read input file and store its data.
- void read_file (std::string_view path, std::ext::V_string const &keep_headers, char delim=',')

  Read input file and store its data.
- std::ext::V_string get_header (std::string const &hdr) const

  Return columns of a given header.
- void remove_duplicates (std::ext::V_string const &v_hdrs)

  Remove duplicate entries based on the values in the given header fields.
- size_t size_wo_duplicates (std::string const &hdr)
- DataFrame remove_duplicates (std::string const &hdr)
- bool any_duplicated (std::string const &hdr)
- std::set< std::string > list_duplicates (std::string const &hdr)
- void match_genoids (std::string hdr_id, std::ext::V_string const &v_hdrs)
- std::ext::V_string list_unique (std::string const &hdr)

Public Attributes

- std::ext::Data m_data
- int m_nrows = 0
- int m_ncols = 0
- std::ext::V_string m_headers
- std::string m_missing_key
- std::ext::V_string m_geno_ids

7.7.1 Detailed Description

A class for storing infromation in the tabular files.

7.7.2 Member Function Documentation

7.7.2.1 any_duplicated()

```
bool DataFrame::any_duplicated (
            std::string const & hdr )
```

7.7.2.2 copy_by_hdrs()

```
DataFrame DataFrame::copy_by_hdrs (
            std::ext::V_string const & v_hdrs )
```

Make a copy of data from based on a given headers.

Parameters

v_hdrs	: vector of requested headers to copy

Returns

DataFrame

7.7.2.3 get_header()

```
std::ext::V_string DataFrame::get_header (
            std::string const & hdr ) const
```

Return columns of a given header.

Parameters

hdr	: a given header to return its values

Returns

std::ext::V_string

7.7.2.4 head()

```
void DataFrame::head (
            int n = 5 )
```

Print the n rows of the dataframe.

Parameters

n	: number of rows to show (by default 5)

7.7.2.5 isDataFrameCreated()

```
bool DataFrame::isDataFrameCreated ( ) const
```

7.7.2.6 list_duplicates()

```
std::set< std::string > DataFrame::list_duplicates (
            std::string const & hdr )
```

7.7.2.7 list_unique()

```
std::ext::V_string DataFrame::list_unique (
            std::string const & hdr )
```

7.7.2.8 match_genoids()

```
void DataFrame::match_genoids (
            std::string hdr_id,
            std::ext::V_string const & v_hdrs )
```

7.7.2.9 n_cols()

```
int DataFrame::n_cols ( ) const
```

7.7.2.10 n_rows()

```
int DataFrame::n_rows ( ) const
```

7.7.2.11 read_file() [1/2]

```
void DataFrame::read_file (
            std::string_view path,
            char delim = ',' )
```

Read input file and store its data.

Parameters

path	: path to the input file
delim	: delimiter to separate columns

7.7.2.12 read_file() [2/2]

```
void DataFrame::read_file (
            std::string_view path,
            std::ext::V_string const & keep_headers,
            char delim = ',' )
```

Read input file and store its data.

Parameters

path	
keep_headers	
delim	

7.7.2.13 remove_duplicates() [1/2]

```
void DataFrame::remove_duplicates (
            std::ext::V_string const & v_hdrs )
```

Remove duplicate entries based on the values in the given header fields.

Parameters

v_hdrs	

7.7.2.14 remove_duplicates() [2/2]

```
DataFrame DataFrame::remove_duplicates (
            std::string const & hdr )
```

7.7.2.15 size_wo_duplicates()

```
size_t DataFrame::size_wo_duplicates (
            std::string const & hdr )
```

7.7.3 Member Data Documentation

7.7.3.1 m_data

```
std::ext::Data DataFrame::m_data
```

7.7.3.2 m_geno_ids

```
std::ext::V_string DataFrame::m_geno_ids
```

7.7.3.3 m_headers

```
std::ext::V_string DataFrame::m_headers
```

7.7.3.4 m_missing_key

```
std::string DataFrame::m_missing_key
```

7.7.3.5 m_ncols

```
int DataFrame::m_ncols = 0
```

7.7.3.6 m_nrows

```
int DataFrame::m_nrows = 0
```

The documentation for this class was generated from the following files:

- include/ReadFiles.h
- src/ReadFiles.cpp

7.8 Fit Struct Reference

A structure that contains all required parameters and functions related to the fitting function and return parameters of the method fitglmm_ai in GMMAT class.

```
#include <GMMAT.h>
```

Public Member Functions

- void calc_dmu_deta (std::string const &family_t, int size)

  A function to calculate derivative of mu in respect of eta based on the family type.
- DensVec calc_sqrtW ()

  A function to calculate the square root of W.

Public Attributes

- DensVec eta
- DensVec mu
- DensVec dmu_deta
- DensMat cov
- DensVec alpha
- SpaMat sigma_i
- DensVec diag_sigma_i
- SpaMat diag_sigma_i_ZPchol
- DensMat sigma_ix
- std::optional< DensVec > dtau
- DensVec W

7.8.1 Detailed Description

A structure that contains all required parameters and functions related to the fitting function and return parameters of the method fitglmm_ai in GMMAT class.

7.8.2 Member Function Documentation

7.8.2.1 calc_dmu_deta()

```
void Fit::calc_dmu_deta (
            std::string const & family_t,
            int size )
```

A function to calculate derivative of mu in respect of eta based on the family type.

Parameters

family_t	
size	

7.8.2.2 calc_sqrtW()

```
DensVec Fit::calc_sqrtW ( )
```

A function to calculate the square root of W.

Returns

DensVec

7.8.3 Member Data Documentation

7.8.3.1 alpha

```
DensVec Fit::alpha
```

7.8.3.2 cov

```
DensMat Fit::cov
```

7.8.3.3 diag_sigma_i

```
DensVec Fit::diag_sigma_i
```

7.8.3.4 diag_sigma_i_ZPchol

```
SpaMat Fit::diag_sigma_i_ZPchol
```

7.8.3.5 dmu_deta

```
DensVec Fit::dmu_deta
```

7.8.3.6 dtau

```
std::optional<DensVec> Fit::dtau
```

7.8.3.7 eta

```
DensVec Fit::eta
```

7.8.3.8 mu

```
DensVec Fit::mu
```

7.8.3.9 sigma_i

```
SpaMat Fit::sigma_i
```

7.8.3.10 sigma_ix

```
DensMat Fit::sigma_ix
```

7.8.3.11 W

```
DensVec Fit::W
```

The documentation for this struct was generated from the following files:

- include/GMMAT.h
- src/GMMAT.cpp

7.9 GEMFit Struct Reference

A structure to interconnect GEM with GMMAT, especially parameters in Fit structure for the null model.

```
#include <GMMAT.h>
```

Public Member Functions

- Fit convert_2_fit ()

  A function to convert GEMFit to Fit structure data type.

Public Attributes

- std::ext::V_double XinvXTX
- std::ext::V_double mu
- std::ext::V_double resid
- double sigma2
- std::ext::V_double alpha
- std::ext::V_double eta

7.9.1 Detailed Description

A structure to interconnect GEM with GMMAT, especially parameters in Fit structure for the null model.

7.9.2 Member Function Documentation

7.9.2.1 convert_2_fit()

```
Fit GEMFit::convert_2_fit ( )
```

A function to convert GEMFit to Fit structure data type.

Returns

Fit

7.9.3 Member Data Documentation

7.9.3.1 alpha

```
std::ext::V_double GEMFit::alpha
```

7.9.3.2 eta

```
std::ext::V_double GEMFit::eta
```

7.9.3.3 mu

```
std::ext::V_double GEMFit::mu
```

7.9.3.4 resid

```
std::ext::V_double GEMFit::resid
```

7.9.3.5 sigma2

```
double GEMFit::sigma2
```

7.9.3.6 XinvXTX

```
std::ext::V_double GEMFit::XinvXTX
```

The documentation for this struct was generated from the following files:

- include/GMMAT.h
- src/GMMAT.cpp

7.10 Glmmkin Struct Reference

A structure containing Fit and other variables to pass between GMMAT methods, also return GMMAT object type.

`#include <GMMAT.h>`

Public Attributes

- DensVec residuals
- DensVec scaled_residuals
- std::ext::V_string id_include
- bool converged
- Fit fit
- double sigma2
- bool run_wb = false

7.10.1 Detailed Description

A structure containing Fit and other variables to pass between GMMAT methods, also return GMMAT object type.

7.10.2 Member Data Documentation

7.10.2.1 converged

```
bool Glmmkin::converged
```

7.10.2.2 fit

```
Fit Glmmkin::fit
```

7.10.2.3 id_include

```
std::ext::V_string Glmmkin::id_include
```

7.10.2.4 residuals

```
DensVec Glmmkin::residuals
```

7.10.2.5 run_wb

```
bool Glmmkin::run_wb = false
```

7.10.2.6 scaled_residuals

```
DensVec Glmmkin::scaled_residuals
```

7.10.2.7 sigma2

```
double Glmmkin::sigma2
```

The documentation for this struct was generated from the following file:

- include/GMMAT.h

7.11 GMMAT Class Reference

A class to run association test.

```
#include <GMMAT.h>
```

Public Member Functions

- void build_Psi (int const ng, bool calc_diag_kin=false)

  Construct the covariance matrix for the random effects.
- void build_Z ()

  Construct the random-effects design matrix.
- Fit fitglmm_ai (DensVec const &W)

  Fits the null GLMM model using Average Information (AI) algorithm.
- Glmmkin glmmkin_ai (Fit fit_null, int maxiter=500, double tol=1e-5)

  Fits the GLMM using AI iterations starting from a null model.
- Glmmkin glmmkin_fit (Fit fit_null, std::ext::V_int group_id, std::string const method="REML", std::string method_optim="AI", int maxiter=500, double tol=1e-5, double tau_min=1e-5, double tau_max=1e+5, int tau↩_region=10)

  Fits the GLMM using the selected optimization method (currently AI-REML).
- Glmmkin glmmkin_init (std::ext::FitNull_f fit0, Pheno pheno, std::ext::V_string cov_selected_hdrs, std::string phenoname, std::string const &id, std::ext::V_string int_cov_hdrs_name_new, std::string randomSlope↩Name, std::string const &groups, int center, int scale, std::string const method="REML", std::string method↩_optim="AI", int maxiter=500, double tol=1e-5, double tau_min=1e-5, double tau_max=1e+5, int tau_↩region=10)

  Initializes and fits a GLMM starting from phenotype and covariate data.

Public Attributes

- int m_groups = 0
- int m_n_sel_col
- std::vector< SparseInverse > m_vkins_sp
- std::ext::VV_int m_covariance_idx
- DensVec m_sqrtW
- DensVec m_Y
- std::ext::V_int m_group_id
- std::unordered_map< int, std::ext::V_int > m_group_idx
- SpaMat m_J
- SpaMat m_Psi
- SpaMat m_Z
- SpaMat m_Zt_Z
- size_t m_N
- size_t m_Nobs
- bool m_dup
- int m_kins_size
- DensVec m_tau
- std::ext::V_int m_fixtau
- std::ext::V_int m_fixrho
- std::ext::V_int m_fixrho_idx
- std::string m_family_t = "binomial"
- std::string m_link = "logit"
- DensVec m_offset
- DensVec m_y
- DensMat m_X
- DensVec m_rand_slope
- std::ext::map_str_int m_hdrsMap
- int m_robust = 0
- std::ext::V_int m_idxtau
- std::ext::V_int m_idxtau2
- ModelType m_modeltype
- SpaMat m_diag_sigma_im_Z

7.11.1 Detailed Description

A class to run association test.

7.11.2 Member Function Documentation

7.11.2.1 build_Psi()

```
void GMMAT::build_Psi (
            int const ng,
            bool calc_diag_kin = false )
```

Construct the covariance matrix for the random effects.

This function builds the random-effect covariance matrix

7.11.2.2 used in Woodbury longitudinal generalized linear mixed models (GLMMs).

1. Random Intercept (RI) model

- Φ_i are sparse kinship matrices
- θ_i are variance component parameters stored in `m_tau`
- I is the identity matrix

The resulting matrix has dimension $N \times N$.

2. Random Intercept + Random Slope (RS) model

where:

- Φ_i are sparse kinship matrices
- θ_i are variance component parameters stored in `m_tau`
- I is the identity matrix

The resulting matrix has dimension $2N \times 2N$.

Implementation Notes

- The matrix is assembled efficiently using sparse triplets.

Parameters

ng	Offset index in `m_tau` where variance components begin.
calc_diag_kin	To calculate the mean of the diagonal of kinsip

Note

The resulting matrix is stored in the member variable `m_Psi`.

7.11.2.3 build_Z()

```
void GMMAT::build_Z ( )
```

Construct the random-effects design matrix.

This function builds the sparse design matrix that links random effects to observations in Woodbury longitudinal GLMM models.

The structure depends on the model type:

7.11.2.4 1. Random Intercept (RI) model

where:

- J is the subject-indicator matrix of size $N_{obs} \times N$
- each row assigns an observation to its corresponding subject

---

7.11.2.5 2. Random Intercept + Random Slope (RS) model

For `LONGITUDINAL_RS`, the design matrix includes both intercept and slope random effects:
where:

- the first block represents J
- the second block represents random slope effects scaled by J

The resulting matrix has dimension:
$N_{obs} \times 2N$

7.11.2.6 Implementation Notes

- The matrix is assembled using sparse triplets for efficiency.
- The subject-indicator matrix `m_J` must already be initialized.

Note

The resulting matrix is stored in the member variable `m_Z`.

7.11.2.7 fitglmm_ai()

```
Fit GMMAT::fitglmm_ai (
            DensVec const & W )
```

Fits the null GLMM model using Average Information (AI) algorithm.

Parameters

W	Working weight vector

Returns

Fit Structure

7.11.2.8 glmmkin_ai()

```
Glmmkin GMMAT::glmmkin_ai (
            Fit fit_null,
            int maxiter = 500,
            double tol = 1e-5 )
```

Fits the GLMM using AI iterations starting from a null model.
Iteratively updates:

- fixed-effect coefficients (alpha)
- variance components (tau)
- working response, weights, and residuals until convergence.

Supports both Woodbury-based and direct sparse implementations.

Parameters

fit_null	Initial null model fit.
maxiter	Maximum number of iterations.
tol	Convergence tolerance.

Returns

Glmmkin object.

7.11.2.9 glmmkin_fit()

```
Glmmkin GMMAT::glmmkin_fit (
            Fit fit_null,
            std::ext::V_int group_id,
            std::string const method = "REML",
            std::string method_optim = "AI",
            int maxiter = 500,
            double tol = 1e-5,
            double tau_min = 1e-5,
            double tau_max = 1e+5,
            int tau_region = 10 )
```

Fits the GLMM using the selected optimization method (currently AI-REML).
Prepares group structure and variance components, runs AI-based fitting, and refits automatically if variance estimates hit parameter boundaries.

Parameters

fit_null	Initial null model fit.
group_id	Group membership for heteroscedastic modeling.
method	Estimation method ("REML" or "ML").
method_optim	Optimization method ("AI").
maxiter	Maximum number of iterations.
tol	Convergence tolerance.
tau_min	Lower bound for variance components.
tau_max	Upper bound for variance components.
tau_region	

Returns

Fitted GLMM model object.

7.11.2.10 glmmkin_init()

```
Glmmkin GMMAT::glmmkin_init (
            std::ext::FitNull_f fit0,
            Pheno pheno,
            std::ext::V_string cov_selected_hdrs,
            std::string phenoname,
            std::string const & id,
            std::ext::V_string int_cov_hdrs_name_new,
            std::string randomSlopeName,
            std::string const & groups,
            int center,
            int scale,
```

```
        std::string const method = "REML",
        std::string method_optim = "AI",
        int maxiter = 500,
        double tol = 1e-5,
        double tau_min = 1e-5,
        double tau_max = 1e+5,
        int tau_region = 10 )
```

Initializes and fits a GLMM starting from phenotype and covariate data.

Parameters

fit0	Function to compute the initial null model.
pheno	Phenotype data container.
cov_selected_hdrs	Selected covariate column names.
phenoname	Phenotype column name.
id	Sample ID column.
randomSlopeName	Optional random slope variable.
group	Grouping variable for heteroscedastic models.
method	Estimation method ("REML" or "ML").
method_optim	Optimization method ("AI").
maxiter	Maximum number of iterations.
tol	Convergence tolerance.
tau_min	Minimum variance component value.
tau_max	Maximum variance component value.
tau_region	

Returns

Fitted GLMM object.

7.11.3 Member Data Documentation

7.11.3.1 m_covariance_idx

```
std::ext::VV_int GMMAT::m_covariance_idx
```

7.11.3.2 m_diag_sigma_im_Z

```
SpaMat GMMAT::m_diag_sigma_im_Z
```

7.11.3.3 m_dup

```
bool GMMAT::m_dup
```

7.11.3.4 m_family_t

```
std::string GMMAT::m_family_t = "binomial"
```

7.11.3.5 m_fixrho

```
std::ext::V_int GMMAT::m_fixrho
```

7.11.3.6 m_fixrho_idx

```
std::ext::V_int GMMAT::m_fixrho_idx
```

7.11.3.7 m_fixtau

std::ext::V_int GMMAT::m_fixtau

7.11.3.8 m_group_id

std::ext::V_int GMMAT::m_group_id

7.11.3.9 m_group_idx

std::unordered_map<int, std::ext::V_int> GMMAT::m_group_idx

7.11.3.10 m_groups

int GMMAT::m_groups = 0

7.11.3.11 m_hdrsMap

std::ext::map_str_int GMMAT::m_hdrsMap

7.11.3.12 m_idxtau

std::ext::V_int GMMAT::m_idxtau

7.11.3.13 m_idxtau2

std::ext::V_int GMMAT::m_idxtau2

7.11.3.14 m_J

SpaMat GMMAT::m_J

7.11.3.15 m_kins_size

int GMMAT::m_kins_size

7.11.3.16 m_link

std::string GMMAT::m_link = "logit"

7.11.3.17 m_modeltype

ModelType GMMAT::m_modeltype

7.11.3.18 m_N

size_t GMMAT::m_N

7.11.3.19 m_n_sel_col

int GMMAT::m_n_sel_col

7.11.3.20 m_Nobs

size_t GMMAT::m_Nobs

7.11.3.21 m_offset

DensVec GMMAT::m_offset

7.11.3.22 m_Psi

```
SpaMat GMMAT::m_Psi
```

7.11.3.23 m_rand_slope

```
DensVec GMMAT::m_rand_slope
```

7.11.3.24 m_robust

```
int GMMAT::m_robust = 0
```

7.11.3.25 m_sqrtW

```
DensVec GMMAT::m_sqrtW
```

7.11.3.26 m_tau

```
DensVec GMMAT::m_tau
```

7.11.3.27 m_vkins_sp

```
std::vector<SparseInverse> GMMAT::m_vkins_sp
```

7.11.3.28 m_X

```
DensMat GMMAT::m_X
```

7.11.3.29 m_Y

```
DensVec GMMAT::m_Y
```

7.11.3.30 m_y

```
DensVec GMMAT::m_y
```

7.11.3.31 m_Z

```
SpaMat GMMAT::m_Z
```

7.11.3.32 m_Zt_Z

```
SpaMat GMMAT::m_Zt_Z
```

The documentation for this class was generated from the following files:

- include/GMMAT.h
- src/GMMAT.cpp

7.12 Kinship Class Reference

```
#include <Kinship.h>
```

Public Member Functions

- void set_path (std::string const &path)

  Set the kinship path.
- void read_file (std::string_view, char delim=',')

  Read the kinship path.
- unsigned int size ()

  A function to get the size of kinship.

Public Attributes

- DataFrame m_data_frame
- std::string m_path
- bool m_null_kin = false
- double m_diag

7.12.1 Member Function Documentation

7.12.1.1 read_file()

```
void Kinship::read_file (
            std::string_view path,
            char delim = ',' )
```

Read the kinship path.

Parameters

delim	

7.12.1.2 set_path()

```
void Kinship::set_path (
            std::string const & path )
```

Set the kinship path.

Parameters

path	

7.12.1.3 size()

```
unsigned int Kinship::size ( )
```

A function to get the size of kinship.

Returns

unsigned int

7.12.2 Member Data Documentation

7.12.2.1 m_data_frame

```
DataFrame Kinship::m_data_frame
```

7.12.2.2 m_diag

```
double Kinship::m_diag
```

7.12.2.3 m_null_kin

```
bool Kinship::m_null_kin = false
```

7.12.2.4 m_path

```
std::string Kinship::m_path
```

The documentation for this class was generated from the following files:

- include/Kinship.h
- src/Kinship.cpp

7.13 LoggerSetup Class Reference

Initializes a global spdlog logger and redirects std::cout/std::cerr.
`#include <Logger.h>`

Static Public Member Functions

- static void init (const std::string &filename="log.txt")

7.13.1 Detailed Description

Initializes a global spdlog logger and redirects std::cout/std::cerr.
This utility class configures spdlog with:

- Console logging (colored output)
- File logging (persistent log file)
- Custom log formatting
- Automatic flushing behavior

After initialization, all output written to std::cout and std::cerr is captured and forwarded into spdlog, allowing unified logging of both C++ and library output.

7.13.2 Member Function Documentation

7.13.2.1 init()

```
void LoggerSetup::init (
            const std::string & filename = "log.txt" )  [static]
```

The documentation for this class was generated from the following files:

- include/Logger.h
- src/Logger.cpp

7.14 MAGEE Class Reference

A class to run GEI test.
`#include <MAGEE.h>`

Public Types

- enum class GENOTYPE { Bgen , Pgen , Bed }

Public Member Functions

- MAGEE ()=default
- MAGEE (GMMAT &gmmat, Glmmkin &glmmkin, CommandLine &cmd, Bgen bgen, std::ext::V_string interaction_exp, std::ext::V_string interaction_cov)
- MAGEE (GMMAT &gmmat, Glmmkin &glmmkin, CommandLine &cmd, Pgen pgen, std::ext::V_string interaction_exp, std::ext::V_string interaction_cov)
- MAGEE (GMMAT &gmmat, Glmmkin &glmmkin, CommandLine &cmd, Bed bed, std::ext::V_string interaction_exp, std::ext::V_string interaction_cov)
- void fitglmm ()

  Run GEI based on glmm.

7.14.1 Detailed Description

A class to run GEI test.

7.14.2 Member Enumeration Documentation

7.14.2.1 GENOTYPE

```
enum class MAGEE::GENOTYPE  [strong]
```

Enumerator

Bgen	
Pgen	
Bed	

7.14.3 Constructor & Destructor Documentation

7.14.3.1 MAGEE() [1/4]

```
MAGEE::MAGEE ( )  [default]
```

7.14.3.2 MAGEE() [2/4]

```
MAGEE::MAGEE (
            GMMAT & gmmat,
            Glmmkin & glmmkin,
            CommandLine & cmd,
            Bgen bgen,
            std::ext::V_string interaction_exp,
            std::ext::V_string interaction_cov )
```

7.14.3.3 MAGEE() [3/4]

```
MAGEE::MAGEE (
            GMMAT & gmmat,
            Glmmkin & glmmkin,
            CommandLine & cmd,
            Pgen pgen,
            std::ext::V_string interaction_exp,
            std::ext::V_string interaction_cov )
```

7.14.3.4 MAGEE() [4/4]

```
MAGEE::MAGEE (
            GMMAT & gmmat,
            Glmmkin & glmmkin,
            CommandLine & cmd,
            Bed bed,
            std::ext::V_string interaction_exp,
            std::ext::V_string interaction_cov )
```

7.14.4 Member Function Documentation

7.14.4.1 fitglmm()

```
void MAGEE::fitglmm ( )
```

Run GEI based on glmm.

The documentation for this class was generated from the following files:

- include/MAGEE.h
- src/MAGEE.cpp

7.15 Magee_Arma Struct Reference

Struct to create an object with armadillo type.
`#include <MAGEE.h>`

Public Attributes

- arma::mat EC
- std::ext::V_int select
- bool dupflag = false
- arma::sp_mat cov
- size_t n_obs
- size_t n
- arma::mat Jres
- arma::mat JEresblock
- arma::sp_mat Psi
- arma::sp_mat Xi
- arma::sp_mat sigma_iJJ
- arma::sp_mat sigma_ixJ
- arma::vec diag_sigma_i
- arma::sp_mat diag_sigma_i_ZPchol

7.15.1 Detailed Description

Struct to create an object with armadillo type.

7.15.2 Member Data Documentation

7.15.2.1 cov

```
arma::sp_mat Magee_Arma::cov
```

7.15.2.2 diag_sigma_i

```
arma::vec Magee_Arma::diag_sigma_i
```

7.15.2.3 diag_sigma_i_ZPchol

```
arma::sp_mat Magee_Arma::diag_sigma_i_ZPchol
```

7.15.2.4 dupflag

```
bool Magee_Arma::dupflag = false
```

7.15.2.5 EC

```
arma::mat Magee_Arma::EC
```

7.15.2.6 JEresblock

```
arma::mat Magee_Arma::JEresblock
```

7.15.2.7 Jres

```
arma::mat Magee_Arma::Jres
```

7.15.2.8 n

```
size_t Magee_Arma::n
```

7.15.2.9 n_obs

```
size_t Magee_Arma::n_obs
```

7.15.2.10 Psi

```
arma::sp_mat Magee_Arma::Psi
```

7.15.2.11 select

```
std::ext::V_int Magee_Arma::select
```

7.15.2.12 sigma_iJJ

```
arma::sp_mat Magee_Arma::sigma_iJJ
```

7.15.2.13 sigma_ixJ

```
arma::sp_mat Magee_Arma::sigma_ixJ
```

7.15.2.14 Xi

```
arma::sp_mat Magee_Arma::Xi
```

The documentation for this struct was generated from the following file:

- include/MAGEE.h

7.16 Magee_Glmmkin Struct Reference

Structure for storing GEI-related data in Eigen matrices.

`#include <MAGEE.h>`

Public Attributes

- DensMat E
- DensMat EC
- SpaMat J
- std::ext::V_int select
- bool dupflag = false
- SpaMat sigma_i
- DensVec diag_sigma_i
- SpaMat diag_sigma_i_ZPchol
- SpaMat sigma_ix
- SpaMat cov
- DensVec residuals
- DensMat JEres
- SpaMat JPJ
- SpaMat JEPJ
- SpaMat JEPEJ

7.16.1 Detailed Description

Structure for storing GEI-related data in Eigen matrices.
This structure holds intermediate matrices and vectors used in the GMMAT + MAGEE workflow, including design matrices, covariance components, residuals, and Woodbury-related terms.

7.16.2 Member Data Documentation

7.16.2.1 cov

```
SpaMat Magee_Glmmkin::cov
```

7.16.2.2 diag_sigma_i

```
DensVec Magee_Glmmkin::diag_sigma_i
```

7.16.2.3 diag_sigma_i_ZPchol

```
SpaMat Magee_Glmmkin::diag_sigma_i_ZPchol
```

7.16.2.4 dupflag

```
bool Magee_Glmmkin::dupflag = false
```

7.16.2.5 E

```
DensMat Magee_Glmmkin::E
```

7.16.2.6 EC

```
DensMat Magee_Glmmkin::EC
```

7.16.2.7 J

```
SpaMat Magee_Glmmkin::J
```

7.16.2.8 JEPEJ

```
SpaMat Magee_Glmmkin::JEPEJ
```

7.16.2.9 JEPJ

```
SpaMat Magee_Glmmkin::JEPJ
```

7.16.2.10 JEres

```
DensMat Magee_Glmmkin::JEres
```

7.16.2.11 JPJ

```
SpaMat Magee_Glmmkin::JPJ
```

7.16.2.12 residuals

```
DensVec Magee_Glmmkin::residuals
```

7.16.2.13 select

```
std::ext::V_int Magee_Glmmkin::select
```

7.16.2.14 sigma_i

```
SpaMat Magee_Glmmkin::sigma_i
```

7.16.2.15 sigma_ix

```
SpaMat Magee_Glmmkin::sigma_ix
```

The documentation for this struct was generated from the following file:

- include/MAGEE.h

7.17 pair_hash Struct Reference

```
#include <SparseInverse.h>
```

Public Member Functions

- template<class T1 , class T2 >
  std::size_t operator() (std::pair< T1, T2 > const &pair) const

7.17.1 Member Function Documentation

7.17.1.1 operator()()

```
template<class T1 , class T2 >
std::size_t pair_hash::operator() (
            std::pair< T1, T2 > const & pair ) const  [inline]
```

The documentation for this struct was generated from the following file:

- include/SparseInverse.h

7.18 Pgen Class Reference

```
#include <ReadPGEN.h>
```

Public Member Functions

- void processPgenHeader (string pgenFile)
- void processPvar (Pgen pgen, string pvarFile)
- void processPsam (Pgen pgen, string psamFile, unordered_map< string, vector< vector< string > > > phenomap, string phenoMissingKey, int numSelCol, int samSize)
- void getPgenVariantPos (Pgen pgen, CommandLine cmd)

Public Attributes

- uint32_t raw_variant_ct
- uint32_t raw_sample_ct
- int new_samSize
- std::vector< string > sampleID
- std::vector< string > sampleID_all
- std::vector< double > new_covdata
- std::vector< double > new_phenodata
- std::vector< long int > include_idx
- int NumExcludeCol
- int phenoType
- vector< uint32_t > begin
- vector< uint32_t > end
- uint32_t threads
- bool filterVariants = false
- std::vector< long long unsigned int > pgenVariantPos
- vector< int > pvarIndex

- int pvarLast
- vector< int > binE_idx
- int numIntSelCol_new
- int numExpSelCol_new
- int numSelCol_new
- std::vector< int > excludeCol

7.18.1 Member Function Documentation

7.18.1.1 getPgenVariantPos()

```
void Pgen::getPgenVariantPos (
            Pgen pgen,
            CommandLine cmd )
```

7.18.1.2 processPgenHeader()

```
void Pgen::processPgenHeader (
            string pgenFile )
```

7.18.1.3 processPsam()

```
void Pgen::processPsam (
            Pgen pgen,
            string psamFile,
            unordered_map< string, vector< vector< string > > > phenomap,
            string phenoMissingKey,
            int numSelCol,
            int samSize )
```

7.18.1.4 processPvar()

```
void Pgen::processPvar (
            Pgen pgen,
            string pvarFile )
```

7.18.2 Member Data Documentation

7.18.2.1 begin

```
vector<uint32_t> Pgen::begin
```

7.18.2.2 binE_idx

```
vector<int> Pgen::binE_idx
```

7.18.2.3 end

```
vector<uint32_t> Pgen::end
```

7.18.2.4 excludeCol

```
std::vector<int> Pgen::excludeCol
```

7.18.2.5 filterVariants

```
bool Pgen::filterVariants = false
```

7.18.2.6 include_idx

```
std::vector<long int> Pgen::include_idx
```

7.18.2.7 new_covdata

```
std::vector<double> Pgen::new_covdata
```

7.18.2.8 new_phenodata

```
std::vector<double> Pgen::new_phenodata
```

7.18.2.9 new_samSize

```
int Pgen::new_samSize
```

7.18.2.10 NumExcludeCol

```
int Pgen::NumExcludeCol
```

7.18.2.11 numExpSelCol_new

```
int Pgen::numExpSelCol_new
```

7.18.2.12 numIntSelCol_new

```
int Pgen::numIntSelCol_new
```

7.18.2.13 numSelCol_new

```
int Pgen::numSelCol_new
```

7.18.2.14 pgenVariantPos

```
std::vector<long long unsigned int> Pgen::pgenVariantPos
```

7.18.2.15 phenoType

```
int Pgen::phenoType
```

7.18.2.16 pvarIndex

```
vector<int> Pgen::pvarIndex
```

7.18.2.17 pvarLast

```
int Pgen::pvarLast
```

7.18.2.18 raw_sample_ct

```
uint32_t Pgen::raw_sample_ct
```

7.18.2.19 raw_variant_ct

```
uint32_t Pgen::raw_variant_ct
```

7.18.2.20 sampleID

```
std::vector<string> Pgen::sampleID
```

7.18.2.21 sampleID_all

```
std::vector<string> Pgen::sampleID_all
```

7.18.2.22 threads

```
uint32_t Pgen::threads
```

The documentation for this class was generated from the following files:

- include/ReadPGEN.h
- src/ReadPGEN.cpp

7.19 Pheno Class Reference

```
#include <Pheno.h>
```

Public Member Functions

- unsigned int size ()

  a function to get the size of pheno data
- void set_path (std::string_view path)

  Set the path.
- std::string get_path () const

  Get pheno path.
- void read_file (std::string_view, char delim=',')

  Read pheno path.
- std::pair< std::string, std::string > check_binary (std::string pheno_name)

Public Attributes

- DataFrame m_data_frame
- std::string m_sam_id
- std::ext::V_string m_v_hdrs

7.19.1 Member Function Documentation

7.19.1.1 check_binary()

```
std::pair< std::string, std::string > Pheno::check_binary (
            std::string pheno_name )
```

7.19.1.2 get_path()

```
std::string Pheno::get_path ( ) const
```

Get pheno path.

Returns

std::string

7.19.1.3 read_file()

```
void Pheno::read_file (
            std::string_view path,
            char delim = ',' )
```

Read pheno path.

Parameters

delim	

7.19.1.4 set_path()

```
void Pheno::set_path (
            std::string_view path )
```

Set the path.

Parameters

path	

7.19.1.5 size()

```
unsigned int Pheno::size ( )
```

a function to get the size of pheno data

Returns

unsigned int

7.19.2 Member Data Documentation

7.19.2.1 m_data_frame

```
DataFrame Pheno::m_data_frame
```

7.19.2.2 m_sam_id

```
std::string Pheno::m_sam_id
```

7.19.2.3 m_v_hdrs

```
std::ext::V_string Pheno::m_v_hdrs
```

The documentation for this class was generated from the following files:

- include/Pheno.h
- src/Pheno.cpp

7.20 SparseInverse Class Reference

```
#include <SparseInverse.h>
```

Public Member Functions

- SparseInverse ()=default
- SparseInverse (std::string kin_add, std::string pheno_add, char kin_delim, double kin_diag, char pheno_delim, std::string &m_sam_id, std::ext::V_string &m_v_hdrs, std::ext::V_string &bgen_sample_id, std::string phenoMissingKey)
- void set_idx_mp_unq (std::ext::V_string const &vec)
- void set_idx_mp (std::ext::V_string v_strs)
- std::ext::IndexMap get_idx_mp ()
- std::ext::IndexMapUnq get_idx_mp_unq ()
- void set_kin_Nsize (std::ext::VecTuples4spmat &vt4spmat)
- void set_kin_Nobssize (std::ext::VecTuples4spmat &vt4spmat)

- void set_spmat ()
- void set_spmat (SpaMat sm)
- SpaMat & get_spmat ()
- SpaMat inv_spamat ()

Static Public Member Functions

- static SpaMat solve_chol (const SpaMat &sm)
- static SpaMat inv_spamat (SpaMat const &sm)
- static SpaMat inv_spamat_chol (SpaMat const &sm, bool return_full_inverse=false)
- static DensMat inv (DensMat const &dm)

Public Attributes

- Pheno pheno
- Kinship kin
- char pheno_delim
- char kin_delim
- int m_ratio_Nob2N
- int m_thr = 2

7.20.1 Constructor & Destructor Documentation

7.20.1.1 SparseInverse() [1/2]

```
SparseInverse::SparseInverse ( )  [default]
```

7.20.1.2 SparseInverse() [2/2]

```
SparseInverse::SparseInverse (
            std::string kin_add,
            std::string pheno_add,
            char kin_delim,
            double kin_diag,
            char pheno_delim,
            std::string & m_sam_id,
            std::ext::V_string & m_v_hdrs,
            std::ext::V_string & bgen_sample_id,
            std::string phenoMissingKey )
```

7.20.2 Member Function Documentation

7.20.2.1 get_idx_mp()

```
std::ext::IndexMap SparseInverse::get_idx_mp ( )
```

7.20.2.2 get_idx_mp_unq()

```
std::ext::IndexMapUnq SparseInverse::get_idx_mp_unq ( )
```

7.20.2.3 get_spmat()

```
SpaMat & SparseInverse::get_spmat ( )
```

7.20.2.4 inv()

```
DensMat SparseInverse::inv (
            DensMat const & dm )  [static]
```

7.20.2.5 inv_spamat() [1/2]

```
SpaMat SparseInverse::inv_spamat ( )
```

7.20.2.6 inv_spamat() [2/2]

```
SpaMat SparseInverse::inv_spamat (
            SpaMat const & sm )  [static]
```

7.20.2.7 inv_spamat_chol()

```
SpaMat SparseInverse::inv_spamat_chol (
            SpaMat const & sm,
            bool return_full_inverse = false )  [static]
```

7.20.2.8 set_idx_mp()

```
void SparseInverse::set_idx_mp (
            std::ext::V_string v_strs )
```

7.20.2.9 set_idx_mp_unq()

```
void SparseInverse::set_idx_mp_unq (
            std::ext::V_string const & vec )
```

7.20.2.10 set_kin_Nobssize()

```
void SparseInverse::set_kin_Nobssize (
            std::ext::VecTuples4spmat & vt4spmat )
```

7.20.2.11 set_kin_Nsize()

```
void SparseInverse::set_kin_Nsize (
            std::ext::VecTuples4spmat & vt4spmat )
```

7.20.2.12 set_spmat() [1/2]

```
void SparseInverse::set_spmat ( )
```

7.20.2.13 set_spmat() [2/2]

```
void SparseInverse::set_spmat (
            SpaMat sm )
```

7.20.2.14 solve_chol()

```
SpaMat SparseInverse::solve_chol (
            const SpaMat & sm )  [static]
```

7.20.3 Member Data Documentation

7.20.3.1 kin

```
Kinship SparseInverse::kin
```

7.20.3.2 kin_delim

```
char SparseInverse::kin_delim
```

7.20.3.3 m_ratio_Nob2N

```
int SparseInverse::m_ratio_Nob2N
```

7.20.3.4 m_thr

```
int SparseInverse::m_thr = 2
```

7.20.3.5 pheno

```
Pheno SparseInverse::pheno
```

7.20.3.6 pheno_delim

```
char SparseInverse::pheno_delim
```

The documentation for this class was generated from the following files:

- include/SparseInverse.h
- src/SparseInverse.cpp

7.21 Time Class Reference

```
#include <TimeUtils.h>
```

The documentation for this class was generated from the following file:

- include/TimeUtils.h

Chapter 8

File Documentation

8.1 include/BinaryEUtils.h File Reference

```
#include <vector>
#include <string>
#include "ReadParameters.h"
```

Classes

- class BinE

Macros

- #define BinaryEUtils_H

Functions

- std::unordered_map< int, vector< int > > GetPowerSet (std::string v)
- std::vector< std::string > vector_binstrings (int width)
- std::unordered_map< std::string, unsigned int > cartesian_map (vector< vector< string > > &v)
- std::vector< std::string > cartesian_vec_sep (vector< vector< string > > &v)

8.1.1 Macro Definition Documentation

8.1.1.1 BinaryEUtils_H

```
#define BinaryEUtils_H
```

8.1.2 Function Documentation

8.1.2.1 cartesian_map()

```
std::unordered_map< std::string, unsigned int > cartesian_map (
            vector< vector< string > > & v )
```

8.1.2.2 cartesian_vec_sep()

```
std::vector< std::string > cartesian_vec_sep (
            vector< vector< string > > & v )
```

8.1.2.3 GetPowerSet()

```
std::unordered_map< int, vector< int > > GetPowerSet (
            std::string v )
```

8.1.2.4 vector_binstrings()

```
std::vector< std::string > vector_binstrings (
            int width )
```

8.2 BinaryEUtils.h

Go to the documentation of this file.

```
00001
00002 #include <vector>
00003 #include <string>
00004 #include "ReadParameters.h"
00005
00006 #ifndef BinaryEUtils_H
00007 #define BinaryEUtils_H
00008
00009 class BinE {
00010
00011     public:
00012         int nBinE = 0;
00013         int strataLen = 0;
00014
00015         std::vector<int> stratum_idx;
00016
00017         std::vector<string> bin_headers;
00018
00019        void checkBinaryCovariates(BinE binE, CommandLine cmd, unordered_map<string,
      vector<vector<string>> phenoMap, vector<string> sampleID, vector<long int> include_idx, int samSize,
      int phenoType, string phenoHeaderName, std::vector<string> covSelHeadersName, std::vector<string>
      covSelHeadersName_new, int Sq_new);
00020 };
00021
00022 std::unordered_map <int, vector<int>> GetPowerSet(std::string v);
00023 std::vector<std::string> vector_binstrings(int width);
00024 std::unordered_map<std::string, unsigned int> cartesian_map( vector<vector<string> >& v);
00025 std::vector<std::string> cartesian_vec_sep( vector<vector<string> >& v);
00026
00027 #endif
```

8.3 include/declars.h File Reference

```
#include <vector>
#include <cstring>
#include <string>
#include <set>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <numeric>
#include <utility>
#include <cctype>
#include <typeinfo>
#include <sstream>
#include <cstdlib>
#include <cassert>
#include <ctime>
#include <chrono>
#include <omp.h>
#include <cmath>
#include <unordered_map>
#include <unordered_set>
#include <exception>
#include <limits>
#include <boost/thread.hpp>
#include <boost/thread/tss.hpp>
#include <boost/asio.hpp>
#include <boost/make_shared.hpp>
```

```
#include <boost/algorithm/string/replace.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/iostreams/filtering_stream.hpp>
#include <boost/iostreams/filter/gzip.hpp>
#include <boost/iostreams/copy.hpp>
#include <boost/math/distributions/chi_squared.hpp>
#include <boost/program_options.hpp>
#include <boost/filesystem.hpp>
#include <boost/format.hpp>
#include "zlib.h"
#include <Eigen/Sparse>
#include <Eigen/SparseLU>
#include <Eigen/SparseCholesky>
#include <Eigen/SparseCore>
#include <Eigen/IterativeLinearSolvers>
#include <Eigen/SparseQR>
#include <Eigen/Dense>
#include <Eigen/Core>
#include "MatrixUtils.h"
#include "ReadParameters.h"
#include "ReadBGEN.h"
#include "ReadPGEN.h"
#include "ReadBed.h"
#include "BinaryEUtils.h"
```

Typedefs

- typedef unsigned char uchar
- typedef unsigned int uint
- typedef unsigned long long uint64
- typedef unsigned short ushort

8.3.1 Typedef Documentation

8.3.1.1 uchar

```
typedef unsigned char uchar
```

8.3.1.2 uint

```
typedef unsigned int uint
```

8.3.1.3 uint64

```
typedef unsigned long long uint64
```

8.3.1.4 ushort

```
typedef unsigned short ushort
```

8.4 declars.h

Go to the documentation of this file.

```
00001 #include <vector>
00002 #include <cstring>
00003 #include <string>
00004 #include <set>
00005 #include <iostream>
00006 #include <fstream>
00007 #include <algorithm>
```

```
00008 #include <numeric>
00009 #include <utility>
00010 #include <cctype>
00011 #include <typeinfo>
00012 #include <sstream>
00013 #include <cstdlib>
00014 #include <cassert>
00015 #include <ctime>
00016 #include <chrono>
00017 #include <omp.h>
00018 #include <cmath>
00019 #include <unordered_map>
00020 #include <unordered_set>
00021 #include <exception>
00022 #include <limits>
00023
00024
00025 #include <boost/thread.hpp>
00026 #include <boost/thread/tss.hpp>
00027 #include <boost/asio.hpp>
00028 #include <boost/make_shared.hpp>
00029 #include <boost/algorithm/string/replace.hpp>
00030 #include <boost/algorithm/string.hpp>
00031 #include <boost/iostreams/filtering_stream.hpp>
00032 #include <boost/iostreams/filter/gzip.hpp>
00033 #include <boost/iostreams/copy.hpp>
00034 #include <boost/math/distributions/chi_squared.hpp>
00035 #include <boost/program_options.hpp>
00036 #include <boost/make_shared.hpp>
00037 #include <boost/thread.hpp>
00038 #include <boost/thread/tss.hpp>
00039 #include <boost/filesystem.hpp>
00040 #include <boost/format.hpp>
00041 #include "zlib.h"
00042
00043 #include <Eigen/Sparse>
00044 #include <Eigen/SparseLU>
00045 #include <Eigen/SparseCholesky>
00046 #include <Eigen/SparseCore>
00047 #include <Eigen/IterativeLinearSolvers>
00048 #include <Eigen/SparseQR>
00049 #include <Eigen/Dense>
00050 #include <Eigen/Core>
00051
00052 using std::string;
00053 using std::vector;
00054 using std::cin;
00055 using std::cout;
00056 using std::endl;
00057 using std::cerr;
00058 using std::getline;
00059 using std::find;
00060 using std::unordered_map;
00061 using std::unordered_set;
00062 using Eigen::MatrixXd;
00063 using Eigen::MatrixXi;
00064 using Eigen::VectorXd;
00065 namespace po = boost::program_options;
00066
00067
00068 typedef unsigned char uchar;
00069 typedef unsigned int uint;
00070 typedef unsigned long long uint64;
00071 typedef unsigned short ushort;
00072
00073
00074 #include "MatrixUtils.h"
00075 #include "ReadParameters.h"
00076 #include "ReadBGEN.h"
00077 #include "ReadPGEN.h"
00078 #include "ReadBed.h"
00079 #include "BinaryEUtils.h"
00080
00081
```

8.5 include/GEM.h File Reference

```
#include "GMMAT.h"
#include "MAGEE.h"
#include "declars.h"
#include "Logger.h"
```

Functions

- std::string get_log_name (int argc, char *argv[])

  Extract the output/log file name from command-line arguments.
- int checkBinary (unordered_map< string, vector< vector< string > > > phenoMap, vector< string > sampleID, double epsilon)

  Determine whether the phenotype is binary or continuous.
- void center (int center, int scale, int samSize, int numSelCol, vector< double > covdata, vector< double > *covdata_ret)
- void fitNullModel2 (int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret, std::vector< double > &beta_ret, std::vector< double > &Xbeta_ret)

  Fit null regression model and return additional intermediate outputs.
- void fitNullModel (int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret)

  Fit the null regression model (linear or logistic).
- void printCovVarMat (int numCovs, vector< string > covNames, double *covVarMat, double *beta, int phenoType, int samSize)

  Print regression coefficients and covariance matrix of estimates.
- void printOutputHeader (bool useBgen, int numExpSelCol_new, int Sq1, vector< string > covNames, string output, string outStyle, int robust, double sigma2, BinE binE)

  Write the header line for the GWAS output results file.

8.5.1 Function Documentation

8.5.1.1 center()

```
void center (
            int center,
            int scale,
            int samSize,
            int numSelCol,
            vector< double > covdata,
            vector< double > * covdata_ret )
```

8.5.1.2 checkBinary()

```
int checkBinary (
            unordered_map< string, vector< vector< string > > > phenoMap,
            vector< string > sampleID,
            double epsilon )
```

Determine whether the phenotype is binary or continuous.
This function inspects the phenotype values across all provided sample IDs. It counts the number of unique phenotype categories:

- If only 2 unique values exist → phenotype is treated as binary.
- If more than 2 unique values exist → phenotype is treated as continuous.

If all phenotype values are identical, an error is printed.
When binary, the logistic regression convergence threshold is also reported.

Parameters

phenoMap	Map from sample ID to phenotype vector.
sampleID	Vector of sample IDs in analysis order.
epsilon	Convergence tolerance used in logistic regression.

Returns

int Returns 1 if phenotype is binary, 0 if continuous.

8.5.1.3 fitNullModel()

```
void fitNullModel (
            int samSize,
            int numSelCol,
            int phenoType,
            double epsilon,
            int robust,
            std::vector< string > covSelHeadersName,
            std::vector< double > phenodata,
            std::vector< double > covdata,
            std::vector< double > * XinvXTX_ret,
            vector< double > * miu_ret,
            vector< double > * resid_ret,
            double * sigma2_ret )
```

Fit the null regression model (linear or logistic).
This function fits a regression model without genotype effects:

- Linear regression for continuous traits
- Logistic regression for binary traits

Outputs include:

- Residual vector
- Dispersion estimate (sigma^2)
- Mean response (mu)
- Precomputed matrix X If logistic regression fails to converge after MAX_ITER, the program terminates with an error.

Parameters

samSize	
numSelCol	
phenoType	
epsilon	
robust	
covSelHeadersName	
phenodata	
covdata	
XinvXTX_ret	
miu_ret	
resid_ret	
sigma2_ret	

8.5.1.4 fitNullModel2()

```
void fitNullModel2 (
            int samSize,
            int numSelCol,
```

```
            int phenoType,
            double epsilon,
            int robust,
            std::vector< string > covSelHeadersName,
            std::vector< double > phenodata,
            std::vector< double > covdata,
            std::vector< double > * XinvXTX_ret,
            vector< double > * miu_ret,
            vector< double > * resid_ret,
            double * sigma2_ret,
            std::vector< double > & beta_ret,
            std::vector< double > & Xbeta_ret )
```

Fit null regression model and return additional intermediate outputs.
This is an extended version of fitNullModel().
In addition to standard null model outputs, it also returns:

- beta estimates (alpha)
- linear predictor X∗beta (eta)

These intermediate quantities are required for downstream GMMAT-style score statistic calculations.
Supports both:

- Linear regression
- Logistic regression

Parameters

samSize	
numSelCol	
phenoType	
epsilon	
robust	
covSelHeadersName	
phenodata	
covdata	
XinvXTX_ret	
miu_ret	
resid_ret	
sigma2_ret	
beta_ret	
Xbeta_ret	

8.5.1.5 get_log_name()

```
std::string get_log_name (
            int argc,
            char * argv[] )
```

Extract the output/log file name from command-line arguments.
This function scans the command-line inputs and looks for the option `--out`. If found, the following argument is returned as the log/output file name.
If the option is not provided, an empty string (fallback) is returned.

Parameters

argc	Number of command-line arguments.

Parameters

argv	Array of command-line argument strings.

Returns

std::string The file name specified after `--out`, or an empty string.

8.5.1.6 printCovVarMat()

```
void printCovVarMat (
            int numCovs,
            vector< string > covNames,
            double * covVarMat,
            double * beta,
            int phenoType,
            int samSize )
```

Print regression coefficients and covariance matrix of estimates.
Function outputs:

- Estimated regression coefficients (beta)
- Standard errors
- Z-values
- Chi-square p-values

It also prints the full variance-covariance matrix.
Used after fitting the null model in linear or logistic regression.

Parameters

numCovs	Number of covariates.
covNames	Names of covariates.
covVarMat	Variance-covariance matrix.
beta	Estimated regression coefficients.
phenoType	Phenotype type (1=binary, 0=continuous).
samSize	Number of samples.

8.5.1.7 printOutputHeader()

```
void printOutputHeader (
            bool useBgen,
            int numExpSelCol_new,
            int Sq1,
            vector< string > covNames,
            string output,
            string outStyle,
            int robust,
            double sigma2,
            BinE binE )
```

Write the header line for the GWAS output results file.
This function generates the correct column header depending on:
The header includes:

- Variant information (CHR, POS, alleles, AF)

- Marginal effects
- Interaction effects
- Standard errors
- Covariance terms
- P-values

Parameters

useBgen	
numExpSelCol_new	
Sq1	
covNames	
output	
outStyle	
robust	
sigma2	
binE	

8.6 GEM.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "GMMAT.h"
00003 #include "MAGEE.h"
00004 #include "declars.h"
00005 #include "Logger.h"
00006
00007
00022 std::string get_log_name(int argc, char* argv[]);
00023
00043 int  checkBinary(unordered_map<string, vector<vector<string>> phenoMap, vector<string> sampleID,
      double epsilon);
00044
00045
00046 void center(int center, int scale, int samSize, int numSelCol, vector<double> covdata, vector<double>*
      covdata_ret);
00047
00080 void fitNullModel2(int samSize, int numSelCol, int phenoType, double epsilon,
00081                    int robust, std::vector<string> covSelHeadersName, std::vector<double> phenodata,
00082                    std::vector<double> covdata, std::vector<double>* XinvXTX_ret, vector<double>*
      miu_ret,
00083                    vector<double>* resid_ret, double* sigma2_ret, std::vector<double>& beta_ret,
00084                    std::vector<double>& Xbeta_ret);
00085
00086
00087
00117 void fitNullModel(int samSize, int numSelCol, int phenoType, double epsilon,
00118                    int robust, std::vector<string> covSelHeadersName, std::vector<double> phenodata,
00119                    std::vector<double> covdata, std::vector<double>* XinvXTX_ret, vector<double>*
      miu_ret,
00120                    vector<double>* resid_ret, double* sigma2_ret);
00121
00143 void printCovVarMat(int numCovs, vector<string> covNames, double* covVarMat, double* beta, int
      phenoType, int samSize);
00144
00145
00169 void printOutputHeader(bool useBgen, int numExpSelCol_new, int Sq1, vector<string> covNames, string
      output, string outStyle,
00170                        int robust, double sigma2, BinE binE);
00171
00172
00173
```

8.7 include/GMMAT.h File Reference

```
#include "SparseInverse.h"
#include "Kinship.h"
#include <optional>
#include <unordered_set>
#include <iterator>
#include <functional>
#include <variant>
```

Classes

- struct Fit

  A structure that contains all required parameters and functions related to the fitting function and return parameters of the method fitglmm_ai in GMMAT class.
- struct Glmmkin

  A structure containing Fit and other variables to pass between GMMAT methods, also return GMMAT object type.
- struct GEMFit

  A structure to interconnect GEM with GMMAT, especially parameters in Fit structure for the null model.
- class GMMAT

  A class to run association test.

Namespaces

- namespace std
- namespace std::ext

Typedefs

- using std::ext::FitNull_f = std::function< void(int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret, std::vector< double > &beta_ret, std::vector< double > &Xbeta_ret)>

  Function signature for the GEM null model fitting routine.
- using std::ext::Matrix_variant = std::variant< DensMat, SpaMat >

  Variant type for storing dense or sparse matrices.

Enumerations

- enum ModelType { LONGITUDINAL_RI , LONGITUDINAL_RS }

Functions

- SpaMat slice_mat (SpaMat const &spm, std::ext::V_int const &indices)

  Slice sparse matrix based on rows index.
- DensMat slice_mat (DensMat const &dm, std::ext::V_int const &indices)
- DensMat slice_mat (DensMat const &dm, std::ext::V_int ind1, std::ext::V_int ind2)

  Slice dense matrix based on cols and rows index.
- DensMat slice_mat (DensMat const &dm, std::ext::V_bool const &ind1, std::ext::V_bool const &ind2)

  Slice dense matrix based on column and row logical vectors.
- DensMat slice_mat (DensMat const &dm, std::ext::Var_bool_int const &ind1, std::ext::Var_bool_int const &ind2)

  Slice matrix based on columns and rows variant vectors(either int or bool)
- DensMat slice_mat (const DensMat &dm, const std::ext::V_bool &indices)

  Slice dense matrix based on rows indexes.

- DensMat slice_mat (const DensMat &dm, const std::ext::Var_bool_int &indices)

  Slice dense matrix based on variant row vector(either int or bool)
- DensMat slice_mat_cols (DensMat const &dm, std::ext::V_int const &ind2)

  Slice dense matrix based on column index vector.
- DensVec slice_vec (DensVec const &dv, std::ext::V_int const &indices)

  Slice dense vector based on indexes.
- DensVec slice_vec (DensVec const &dv, std::ext::V_bool const &indices)

  Slice dense vector based on logical vector.
- DensVec slice_vec (DensVec const &dv, std::ext::Var_bool_int const &indices)

  Slice dense vector based on variant vector(either int or bool)
- SpaMat slice_mat (SpaMat const &spm, std::ext::V_int ind1, std::ext::V_int ind2, bool check_size=true)

  Slice sparse matrix based on cols and rows index.
- SpaMat slice_mat (SpaMat const &spm, std::ext::V_bool const &ind1, std::ext::V_bool const &ind2, bool check_size=true)

  Slice sparse matrix based on column and row logical vectors.
- SpaMat slice_mat (SpaMat const &spm, std::ext::Var_bool_int const &ind1, std::ext::Var_bool_int const &ind2, bool check_size=true)

  Slice sparse matrix based on cols and rows variant(either int or bool)
- template<typename T , typename Pred >
  std::ext::V_int which (std::vector< T > const &vec, Pred pred)

  Returns a vector of indices for elements that satisfy the predicate.
- template<typename Pred >
  std::ext::V_int which (DensVecInt const &vec, Pred pred)

  Returns a vector of indices for elements that satisfy the predicate.
- template<typename First , typename Second >
  auto crossprod (First const &first, Second const &second)

  Analogy of crossprod function in R language. it gets two matrix from eigen library of any type (dens or sparse) and calculates the transpose of the first one and mulitplies it by the second one.
- template<typename First , typename Second >
  auto tcrossprod (First const &first, Second const &second)

  Analogy of tcrossprod function in R language.
- template<typename T >
  std::unordered_set< T > unique (std::vector< T > const &vec)

  Removes duplicate values from a vector.
- template<typename T >
  std::vector< T > unique_id (std::vector< T > const &vec)

  Returns a vector of unique elements from the input vector.
- std::ext::V_int match_indices (std::ext::V_string const &original, std::ext::V_string const &filtered)

  Matches indices from the original vector to the filtered vector.

Variables

- const int MAX_N_ITER = 500

8.7.1 Enumeration Type Documentation

8.7.1.1 ModelType

```
enum ModelType
```

Enumerator

LONGITUDINAL_RI	
LONGITUDINAL_RS	

8.7.2 Function Documentation

8.7.2.1 crossprod()

```
template<typename First , typename Second >
auto crossprod (
            First const & first,
            Second const & second )
```

Analogy of crossprod function in R language. it gets two matrix from eigen library of any type (dens or sparse) and calculates the transpose of the first one and mulitplies it by the second one.

Template Parameters

First	matrix from eigen library
Second	matrix from eigen libray

Parameters

first	first matrix from eigen library, it can be either dense or sparse.
second	second matrix from eigen library, it can be either dense or sparse.

Returns

auto

8.7.2.2 match_indices()

```
std::ext::V_int match_indices (
            std::ext::V_string const & original,
            std::ext::V_string const & filtered )  [inline]
```

Matches indices from the original vector to the filtered vector.
Similar to the above `match_indices`, but this version accepts non-optional strings in both the original and filtered vectors. If an element in the original vector does not have a match, the corresponding index is set to -1.

Parameters

original	The original vector of strings.
filtered	The filtered vector of strings.

Returns

A vector of matched indices or -1 for non-matches.

8.7.2.3 slice_mat() [1/10]

```
DensMat slice_mat (
            const DensMat & dm,
            const std::ext::V_bool & indices )
```

Slice dense matrix based on rows indexes.

Parameters

Dense	matrix
std::vector<bool>	

Returns

Dense matrix

8.7.2.4 slice_mat() [2/10]

```
DensMat slice_mat (
            const DensMat & dm,
            const std::ext::Var_bool_int & indices )
```

Slice dense matrix based on variant row vector(either int or bool)

Parameters

Dense	matrix
std::variant<int,bool>	

Returns

Dense matrix

8.7.2.5 slice_mat() [3/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_bool const & ind1,
            std::ext::V_bool const & ind2 )
```

Slice dense matrix based on column and row logical vectors.

Parameters

Dense	matrix
std::vector<bool>	
std::vector<bool>	

Returns

Dense matrix

8.7.2.6 slice_mat() [4/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_int const & indices )
```

8.7.2.7 slice_mat() [5/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_int ind1,
            std::ext::V_int ind2 )
```

Slice dense matrix based on cols and rows index.

Parameters

Dense	matrix
std::vector<int>	
std::vector<int>	

Returns

Dens matrix

8.7.2.8 slice_mat() [6/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::Var_bool_int const & ind1,
            std::ext::Var_bool_int const & ind2 )
```

Slice matrix based on columns and rows variant vectors(either int or bool)

Parameters

Dense	matrix
std::variant<int,bool>	
std::variant<int,bool>	

Returns

Dense matrix

8.7.2.9 slice_mat() [7/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_bool const & ind1,
            std::ext::V_bool const & ind2,
            bool check_size = true )
```

Slice sparse matrix based on column and row logical vectors.

Parameters

Sparse	matrix
std::vector<bool>	
std::vector<bool>	

Returns

sparse matrix

8.7.2.10 slice_mat() [8/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_int const & indices )
```

Slice sparse matrix based on rows index.

Parameters

spm	
indices	

Returns

SpaMat

8.7.2.11 slice_mat() [9/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_int ind1,
            std::ext::V_int ind2,
            bool check_size = true )
```

Slice sparse matrix based on cols and rows index.

Parameters

Sparse	matrix
std::vector<int>	
std::vector<int>	

Returns

sparse matrix

8.7.2.12 slice_mat() [10/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::Var_bool_int const & ind1,
            std::ext::Var_bool_int const & ind2,
            bool check_size = true )
```

Slice sparse matrix based on cols and rows variant(either int or bool)

Parameters

Sparse	matrix
std::variant<int,bool>	
std::variant<int,bool>	

Returns

sparse matrix

8.7.2.13 slice_mat_cols()

```
DensMat slice_mat_cols (
            DensMat const & dm,
            std::ext::V_int const & ind2 )
```

Slice dense matrix based on column index vector.

Parameters

Dense	matrix
std::vector<int>	

Returns

Dense matrix

8.7.2.14 slice_vec() [1/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::V_bool const & indices )
```

Slice dense vector based on logical vector.

Parameters

Dense	vector
std::vector<bool>	

Returns

Dense vector

8.7.2.15 slice_vec() [2/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::V_int const & indices )
```

Slice dense vector based on indexes.

Parameters

Dense	vector
std::vector<int>	

Returns

Dense vector

8.7.2.16 slice_vec() [3/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::Var_bool_int const & indices )
```

Slice dense vector based on variant vector(either int or bool)

Parameters

Dense	vector
std::variant<int,bool>	

Returns

Dense vector

8.7.2.17 tcrossprod()

```
template<typename First , typename Second >
auto tcrossprod (
```

```
            First const & first,
            Second const & second )
```

Analogy of tcrossprod function in R language.
it gets two matrix from eigen library of any type (dens or sparse) and mulitplies the first one by the transpose of the second one.

Template Parameters

First	matrix from eigen library
Second	matrix from eigen libray

Parameters

first	first matrix from eigen library, it can be either dense or sparse.
second	second matrix from eigen library, it can be either dense or sparse.

Returns

auto

8.7.2.18 unique()

```
template<typename T >
std::unordered_set< T > unique (
            std::vector< T > const & vec )
```

Removes duplicate values from a vector.
Constructs an unordered set from the input vector, keeping only unique elements.

Template Parameters

T	Element type of the input vector.

Parameters

vec	Input vector.

Returns

std::unordered_set<T> Set containing the unique elements.

8.7.2.19 unique_id()

```
template<typename T >
std::vector< T > unique_id (
            std::vector< T > const & vec )
```

Returns a vector of unique elements from the input vector.

Template Parameters

T	The type of elements in the vector.

Parameters

vec	The input vector.

Returns

std::vector<T> A vector of unique elements.

8.7.2.20 which() [1/2]

```
template<typename Pred >
std::ext::V_int which (
            DensVecInt const & vec,
            Pred pred )
```

Returns a vector of indices for elements that satisfy the predicate.

Template Parameters

T	type of the vector
Pred	predicate

Parameters

DensVecInt	
pred	

Returns

std::ext::V_int

8.7.2.21 which() [2/2]

```
template<typename T , typename Pred >
std::ext::V_int which (
            std::vector< T > const & vec,
            Pred pred )
```

Returns a vector of indices for elements that satisfy the predicate.

Template Parameters

T	type of the vector.
Pred	predicate.

Parameters

std::vector<T>	input vector.
pred	prdicate to fill a vector on integer values to return.

Returns

std::ext::V_int, Vector of indices where the predicate is true.

8.7.3 Variable Documentation

8.7.3.1 MAX_N_ITER

```
const int MAX_N_ITER = 500
```

8.8 GMMAT.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "SparseInverse.h"
00004 #include "Kinship.h"
00005 #include <optional>
00006 #include <unordered_set>
00007 #include <iterator>
00008 #include <functional>
00009 #include <variant>
00010
00011 const int MAX_N_ITER = 500;
00012
00013
00014 namespace std
00015 {
00016     namespace ext
00017     {
00026         using FitNull_f = std::function<void (int samSize, int numSelCol, int phenoType, double
      epsilon,
00027                     int robust, std::vector<string> covSelHeadersName, std::vector<double> phenodata,
00028                     std::vector<double> covdata, std::vector<double>* XinvXTX_ret, vector<double>*
      miu_ret,
00029                     vector<double>* resid_ret, double* sigma2_ret, std::vector<double>& beta_ret,
00030                     std::vector<double>& Xbeta_ret)>;
00037         using Matrix_variant = std::variant<DensMat, SpaMat>;
00038     }
00039 }
00040
00041
00047 struct Fit
00048 {
00049     DensVec eta;
00050     DensVec mu;
00051     DensVec dmu_deta;
00052     DensMat cov;
00053     DensVec alpha;
00054     SpaMat sigma_i;
00055     DensVec diag_sigma_i;
00056     SpaMat  diag_sigma_i_ZPchol;
00057     DensMat sigma_ix;
00058     std::optional<DensVec> dtau;
00059     DensVec W;
00066     void calc_dmu_deta(std::string const& family_t, int size);
00072     DensVec calc_sqrtW();
00073 };
00074
00079 struct Glmmkin
00080 {
00081     DensVec residuals; //
00082     DensVec scaled_residuals;
00083     std::ext::V_string id_include;
00084     bool converged;
00085     Fit fit;
00086     double sigma2;
00087     bool run_wb = false;
00088 };
00089
00090
00095 struct  GEMFit
00096 {
00097     std::ext::V_double XinvXTX;
00098     std::ext::V_double mu;
00099     std::ext::V_double resid;
00100     double sigma2; // Keep gf.sigma2 from fitNullModel2
00101     std::ext::V_double alpha;
00102     std::ext::V_double eta;
00108     Fit convert_2_fit();
00109 };
00110
```

```
00111 enum ModelType
00112 {
00113     LONGITUDINAL_RI,      // Random intercept only
00114     LONGITUDINAL_RS       // Random slope + intercept
00115 };
00116
00117
00122 class GMMAT
00123 {
00124     public:
00125         int m_groups = 0;
00126         int m_n_sel_col;
00127         std::vector<SparseInverse> m_vkins_sp;
00128         std::ext::VV_int m_covariance_idx ;
00129         DensVec m_sqrtW; //sqrtW is the weight matrix
00130         DensVec m_Y; // working vector
00131         std::ext::V_int m_group_id;
00132         std::unordered_map<int, std::ext::V_int> m_group_idx;
00133         SpaMat m_J;
00134         SpaMat m_Psi;
00135         SpaMat m_Z;
00136         SpaMat m_Zt_Z;
00137         size_t m_N;
00138         size_t m_Nobs;
00139         bool m_dup;
00140         int m_kins_size;
00141         DensVec m_tau;
00142         std::ext::V_int m_fixtau;
00143         std::ext::V_int m_fixrho;
00144         std::ext::V_int m_fixrho_idx;
00145         std::string m_family_t = "binomial";
00146         std::string m_link = "logit";
00147         DensVec m_offset;
00148         DensVec m_y;
00149         DensMat m_X;
00150         DensVec m_rand_slope;
00151         std::ext::map_str_int m_hdrsMap;
00152         int m_robust = 0;
00153         std::ext::V_int m_idxtau;
00154         std::ext::V_int m_idxtau2;
00155         ModelType m_modeltype;
00156         SpaMat m_diag_sigma_im_Z;
00157
00192         void build_Psi(int const ng, bool calc_diag_kin = false);
00193
00229         void build_Z();
00230
00237         Fit fitglmm_ai(DensVec const& W);
00255         Glmmkin glmmkin_ai(Fit fit_null, int maxiter = 500, double tol = 1e-5);
00274         Glmmkin glmmkin_fit(Fit fit_null, std::ext::V_int group_id,
00275                             std::string const method = "REML",
00276                             std::string method_optim = "AI",
00277                             int maxiter = 500,
00278                             double tol = 1e-5, double tau_min = 1e-5,
00279                             double tau_max = 1e+5, int tau_region = 10);
00300         [[nodiscard]] Glmmkin glmmkin_init(std::ext::FitNull_f fit0, Pheno pheno,
00301                             std::ext::V_string cov_selected_hdrs,
00302                             std::string phenoname,
00303                             std::string const& id,
00304                             std::ext::V_string int_cov_hdrs_name_new,
00305                             std::string randomSlopeName,
00306                             std::string const& groups,
00307                             int center, int scale,
00308                             std::string const method = "REML",
00309                             std::string method_optim = "AI",
00310                             int maxiter = 500,
00311                             double tol = 1e-5, double tau_min = 1e-5,
00312                             double tau_max = 1e+5, int tau_region = 10);
00313
00314     private:
00315         void fill_mat(const int numRows, std::ext::V_int& indixes_col, int value);
00316         void fill_J(std::ext::V_string const& sample_ids);
00317         void set_ai_low_ng(int i, DensVec& score, DensMat& ai, DensVec const& wpy, Fit const& fit,
00318                            DensVec const& py, DensVec diagp, DensMat sigma_ixcov);
00319         void set_ai_high_ng(int i, DensVec& score, DensMat& ai, DensVec const& wpy, Fit const& fit,
00320                             DensVec const& py, DensMat const& sigma_ixcov,
00321                             int ng);
00322         void set_ai(DensVec& score, DensMat& ai, DensVec const& wpy, Fit const& fit, DensVec const&
      py,
00323                     DensVec diagp, DensMat sigma_ixcov, int ng, int q2);
00324         bool any_negative();
00325         bool any_negative(std::ext::V_int vec);
00326         bool any_nonzero();
00327         void update_tau(DensVec const& tau0, DensVec const& dtau);
00328         void update_tau_below_tol(DensVec const& tau0, double tol);
00329         void update_tau_below_tol2(DensVec const& tau0, double tol);
00330         void extract_group_idx(std::unordered_set<int> const& group_unique, std::ext::V_int group_id);
```

```
00331         int check_pheno_type();
00338         void calc_covariance(int &kins_size, int ng);
00339         void set_dspy(DensMat& dspy, DensVec const& wpy, int ng);
00340         void set_VZpy(DensMat& VZpy, DensVec const& Zpy, int nk);
00341         void set_mtau(DensVec V_tr_corr, DensVec tau0, SpaMat& Ztsigma_iZ,
00342             DensMat const& Ztsigma_ix, DensMat const& Ztsigma_ixcov,
00343             DensVec diagp, int nk, int dimZ, int ng, bool has_random_slope);
00344         void calc_tr_corr(int i, DensVec &score, DensMat const& Ztsigma_ix,
00345             DensMat const& Ztsigma_ixcov, SpaMat& Ztsigma_iZ,
00346             int nk, int dimZ, int const& ng, bool has_random_slope);
00347         void set_score(DensVec &score, DensMat const& Ztsigma_ix,
00348             DensMat const& Ztsigma_ixcov, SpaMat& Ztsigma_iZ,
00349             DensVec const& diagp, int nk,
00350             int dimZ, int const& ng, bool has_random_slope);
00351
00352         void fill_fixrho_idx(double tol);
00353         void update_fixtau_fixrho(std::ext::V_int &fixtau_new,std::ext::V_int &fixrho_new, double
      tol);
00354         void fill_fixrho_idx0(std::ext::V_int &fixrho_idx0, DensVec const& tau0, double tol);
00355         void update_mtau_with_m_covariance_idx_tau0(DensVec const& tau0, double tol);
00356         void update_mtau_with_m_covariance_idx_mfixrho(std::ext::V_int const& idxrho);
00357         void update_mtau_with_m_covariance_mfixrho_idx_fixrho_idx0(std::ext::V_int const&
      fixrho_idx0);
00358         void update_mtau_with_m_covariance_mfixrho_idx();
00359         void update_mtau_with_m_covariance_idx(double tol);
00360         std::ext::V_int exclude_idx();
00361         bool covariate_larger_slope_intercept(double tol);
00362     };
00363
00372 SpaMat slice_mat(SpaMat const& spm, std::ext::V_int const& indices);
00373
00374 DensMat slice_mat(DensMat const& dm, std::ext::V_int const& indices);
00375
00385 DensMat slice_mat(DensMat const& dm, std::ext::V_int ind1 , std::ext::V_int ind2);
00386
00395 DensMat slice_mat(DensMat const& dm, std::ext::V_bool const& ind1, std::ext::V_bool const& ind2);
00396
00405 DensMat slice_mat(DensMat const& dm, std::ext::Var_bool_int const& ind1, std::ext::Var_bool_int const&
      ind2);
00406
00414 DensMat slice_mat(const DensMat& dm, const std::ext::V_bool& indices);
00422 DensMat slice_mat(const DensMat& dm, const std::ext::Var_bool_int& indices);
00423
00431 DensMat slice_mat_cols(DensMat const& dm, std::ext::V_int const& ind2);
00432
00433
00442 DensVec slice_vec(DensVec const& dv, std::ext::V_int const& indices);
00443
00451 DensVec slice_vec(DensVec const& dv, std::ext::V_bool const& indices);
00452
00460 DensVec slice_vec(DensVec const& dv, std::ext::Var_bool_int const& indices);
00461
00470 SpaMat slice_mat(SpaMat const& spm, std::ext::V_int ind1, std::ext::V_int ind2, bool check_size =
      true);
00471
00480 SpaMat slice_mat(SpaMat const& spm, std::ext::V_bool const& ind1, std::ext::V_bool const& ind2, bool
      check_size = true);
00481
00490 SpaMat slice_mat(SpaMat const& spm, std::ext::Var_bool_int const& ind1, std::ext::Var_bool_int const&
      ind2, bool check_size = true);
00491
00492
00502 template <typename T, typename Pred>
00503 std::ext::V_int which(std::vector<T> const& vec, Pred pred)
00504 {
00505     std::vector<int> v_idx;
00506     for(size_t i{0}; i < vec.size(); ++i)
00507     {
00508         if(pred(vec[i]))
00509         {
00510             v_idx.push_back(i);
00511         }
00512     }
00513     return v_idx;
00514 }
00515
00525 template<typename Pred>
00526 std::ext::V_int which(DensVecInt const& vec, Pred pred)
00527 {
00528     std::vector<int> v_idx;
00529     for(int i{0}; i < vec.size(); ++i)
00530     {
00531         if(pred(vec(i)))
00532         {
00533             v_idx.push_back(i);
00534         }
00535     }
```

```
00536     return v_idx;
00537 }
00538
00550 template<typename First, typename Second>
00551 auto crossprod(First const& first, Second const& second) {
00552     return first.transpose() * second;
00553 }
00554
00567 template<typename First, typename Second>
00568 auto tcrossprod(First const& first, Second const& second) {
00569     return first * second.transpose();
00570 }
00571
00581 template<typename T>
00582 std::unordered_set<T> unique(std::vector<T> const& vec)
00583 {
00584     std::unordered_set<T> u_set(vec.begin(), vec.end());
00585     return u_set;
00586 }
00587
00595 template <typename T>
00596 std::vector<T> unique_id(std::vector<T> const& vec)
00597 {
00598     std::unordered_set<T> seen;
00599     std::vector<T> result;
00600
00601     for (auto const& val : vec)
00602     {
00603         if (seen.find(val) == seen.end())
00604         {
00605             result.push_back(val);
00606             seen.insert(val);
00607         }
00608     }
00609
00610     return result;
00611 }
00612
00625 inline std::ext::V_int match_indices(std::ext::V_string const& original, std::ext::V_string const&
      filtered) {
00626     std::ext::map_str_int filtered_map;
00627
00628     for (size_t i = 0; i < filtered.size(); ++i)
00629     {
00630         filtered_map[filtered[i]] = i;
00631     }
00632
00633     std::ext::V_int indices;
00634     for (const auto& id : original)
00635     {
00636         auto it = filtered_map.find(id);
00637
00638         if (it != filtered_map.end())
00639         {
00640             indices.push_back(it->second);
00641         }
00642         else
00643         {
00644             indices.push_back(-1); // Use -1 to indicate not found
00645         }
00646     }
00647     return indices;
00648 }
```

8.9 include/Kinship.h File Reference

```
#include "ReadFiles.h"
```

Classes

- class Kinship

8.10 Kinship.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003
```

```
00004 #include "ReadFiles.h"
00005
00006 class Kinship
00007 {
00008     public:
00009         DataFrame m_data_frame;
00010         std::string m_path;
00011         bool m_null_kin = false;
00012         double m_diag;
00018         void set_path(std::string const& path);
00024         void read_file(std::string_view, char delim = ',');
00030         unsigned int size();
00031     private:
00032         void add_dquot();
00033 };
00034
```

8.11 include/Logger.h File Reference

```
#include <iostream>
#include <streambuf>
#include <memory>
#include <string>
#include <spdlog/spdlog.h>
#include <spdlog/sinks/stdout_color_sinks.h>
#include <spdlog/sinks/basic_file_sink.h>
#include <vector>
```

Classes

- class CoutRedirector

  Redirects standard output (std::cout) into the spdlog logging system.
- class CerrRedirector

  Redirects standard error output (std::cerr) into the spdlog logging system.
- class LoggerSetup

  Initializes a global spdlog logger and redirects std::cout/std::cerr.

8.12 Logger.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <streambuf>
00005 #include <memory>
00006 #include <string>
00007 #include <spdlog/spdlog.h>
00008 #include <spdlog/sinks/stdout_color_sinks.h>
00009 #include <spdlog/sinks/basic_file_sink.h>
00010 #include <vector>
00011
00023 class CoutRedirector : public std::streambuf
00024 {
00025     protected:
00026         int overflow(int c) override;
00027         int sync() override;
00028     private:
00029         std::string buffer_;
00030 };
00031
00041 class CerrRedirector : public std::streambuf
00042 {
00043     protected:
00044         int overflow(int c) override;
00045         int sync() override;
00046     private:
00047         std::string buffer_;
00048 };
00049
00063 class LoggerSetup
```

```
00064 {
00065     public:
00066         static void init(const std::string& filename = "log.txt");
00067 };
```

8.13 include/MAGEE.h File Reference

```
#include "GMMAT.h"
#include "declars.h"
#include "ReadBGEN.h"
#include <numeric>
#include <thread>
```

Classes

- struct Magee_Glmmkin

  Structure for storing GEI-related data in Eigen matrices.
- struct Magee_Arma

  Struct to create an object with armadillo type.
- class MAGEE

  A class to run GEI test.

Functions

- void conver_eigen_to_arma (SpaMat const &eigenMat, arma::sp_mat &armaMat)

  Convert Eigen matrix to armadillo.
- std::ext::V_int match_indices (std::ext::V_string const &original, std::ext::V_opt_string const &filtered)

  Matches indices from the original vector to the filtered vector.
- template<typename T >
  std::ext::V_opt_string slice (std::vector< T > const &vec, std::ext::V_int const &indices)

  Extracts elements from a vector based on specified indices.
- bool any_isna (std::ext::V_int vec)

  Checks if any element in the vector is NA (represented as -1).
- std::ext::V_bool in_op (std::ext::V_string const &vec1, std::ext::V_string const &vec2)

  Checks if elements of vec1 are in vec2.
- std::ext::V_string filtered_ids (std::ext::V_string const &vec1, std::ext::V_bool const &vec2)

  Filters a vector of strings based on a boolean vector.
- void check_not_empty (std::ext::V_string const &sample_id)

  Checks if the input vector of strings is not empty.
- bool any_duplicated (std::ext::V_string const &vec)

  Checks if there are any duplicated elements in the vector.
- std::ext::V_bool list_duplicates_bool (std::ext::V_string const &vec)

  Identifies duplicates in a vector of strings.
- DensMat filter_unique_rows (DensMat const &mat, std::ext::V_string &id_include)

  Filters out duplicate rows from a matrix.
- std::ext::V_bool apply_on_columns (DensMat const &mat, const std::function< bool(VectorXd const &)> &func, int threshold)

  Applies a function to each column of a matrix and returns a boolean vector.
- bool unique_less_equal (DensVec const &vec, int threshold)

  Checks if the number of unique elements in a vector is less than or equal to a threshold.
- std::ext::Map_str_Vint generate_strata_list (std::ext::V_string const &vec)

  Generates a strata list mapping observations to their corresponding strata.
- void scale (DensMat &mat, bool center=true, bool scale=true)

Scales a matrix by centering and/or scaling its columns.

- DensMat cbind (DensMat const &A, DensMat const &B)

  Combines two matrices column-wise.
- std::ext::V_string match_id_include (std::ext::V_string const &id_include, std::ext::V_string const &sample_id)

  Matches included IDs with sample IDs.
- std::ext::V_int remove_minus_one (std::ext::V_int vec)

  Removes elements equal to -1 from the vector.
- std::ext::V_string create_strata (std::ext::VV_string const &Ecat)

  Creates strata by concatenating elements in each row of the binary columns of matrix E.

8.13.1 Function Documentation

8.13.1.1 any_duplicated()

```
bool any_duplicated (
            std::ext::V_string const & vec )
```

Checks if there are any duplicated elements in the vector.

Parameters

vec	The vector to check.

Returns

True if there are duplicates, otherwise false.

8.13.1.2 any_isna()

```
bool any_isna (
            std::ext::V_int vec )
```

Checks if any element in the vector is NA (represented as -1).

Parameters

vec	The vector to check.

Returns

True if any element is -1, otherwise false.

8.13.1.3 apply_on_columns()

```
std::ext::V_bool apply_on_columns (
            DensMat const & mat,
            const std::function< bool(VectorXd const &)> & func,
            int threshold )
```

Applies a function to each column of a matrix and returns a boolean vector.

Parameters

mat	The matrix to process.
func	The function to apply to each column.
threshold	The threshold value used in the function.

Returns

A boolean vector indicating which columns meet the criteria.

8.13.1.4 cbind()

```
DensMat cbind (
            DensMat const & A,
            DensMat const & B )
```

Combines two matrices column-wise.
This function concatenates two matrices `A` and `B` horizontally to create a new matrix.

Parameters

A	The first matrix.
B	The second matrix.

Returns

The combined matrix.

8.13.1.5 check_not_empty()

```
void check_not_empty (
            std::ext::V_string const & sample_id )
```

Checks if the input vector of strings is not empty.

Parameters

sample↩ _id	The vector of sample IDs to check.

Exceptions

std::runtime_error	if the vector is empty.

8.13.1.6 conver_eigen_to_arma()

```
void conver_eigen_to_arma (
            SpaMat const & eigenMat,
            arma::sp_mat & armaMat )
```

Convert Eigen matrix to armadillo.

Parameters

eigenMat	
armaMat	

8.13.1.7 create_strata()

```
std::ext::V_string create_strata (
            std::ext::VV_string const & Ecat )
```

Creates strata by concatenating elements in each row of the binary columns of matrix E.

Parameters

Ecat	A vector of vector of strings representing categorical data.

Returns

A vector of concatenated strings representing strata.

8.13.1.8 filter_unique_rows()

```
DensMat filter_unique_rows (
            DensMat const & mat,
            std::ext::V_string & id_include )
```

Filters out duplicate rows from a matrix.
This function removes duplicate rows from the input matrix and returns a matrix with only unique rows, as well as updating the `id_include` vector accordingly.

Parameters

mat	The input matrix.
id_include	The vector of IDs to filter.

Returns

A matrix with unique rows.

8.13.1.9 filtered_ids()

```
std::ext::V_string filtered_ids (
            std::ext::V_string const & vec1,
            std::ext::V_bool const & vec2 )
```

Filters a vector of strings based on a boolean vector.

Parameters

vec1	The vector to filter.
vec2	The boolean vector indicating which elements to keep.

Returns

A filtered vector of strings.

8.13.1.10 generate_strata_list()

```
std::ext::Map_str_Vint generate_strata_list (
            std::ext::V_string const & vec )
```

Generates a strata list mapping observations to their corresponding strata.

Parameters

vec	The vector of strata identifiers.

Returns

A map from strata identifiers to lists of observation indices.

8.13.1.11 in_op()

```
std::ext::V_bool in_op (
            const std::ext::V_string & vec1,
            const std::ext::V_string & vec2 )
```

Checks if elements of vec1 are in vec2.

Parameters

vec1	The first vector of strings.
vec2	The second vector of strings.

Returns

A boolean vector indicating whether elements of vec1 are found in vec2.

Checks if elements of vec1 are in vec2.

Parameters

vec1	
vec2	

Returns

std::ext::V_bool

8.13.1.12 list_duplicates_bool()

```
std::ext::V_bool list_duplicates_bool (
            std::ext::V_string const & vec )
```

Identifies duplicates in a vector of strings.

Parameters

vec	The vector to check for duplicates.

Returns

A boolean vector indicating the presence of duplicates.

8.13.1.13 match_id_include()

```
std::ext::V_string match_id_include (
            std::ext::V_string const & id_include,
            std::ext::V_string const & sample_id )
```

Matches included IDs with sample IDs.
This function returns the subset of IDs in `id_include` if they are present in `sample_id`.

Parameters

id_include	A vector of IDs to include.
sample_id	A vector of sample IDs.

Returns

A vector of matched IDs.

8.13.1.14 match_indices()

```
std::ext::V_int match_indices (
            std::ext::V_string const & original,
            std::ext::V_opt_string const & filtered )
```

Matches indices from the original vector to the filtered vector.
This function returns a vector of indices that match the elements in the original vector with those in the filtered vector. If an element in the original vector does not have a match, the corresponding index is set to -1.

Parameters

original	The original vector of strings.
filtered	The filtered vector of optional strings.

Returns

A vector of matched indices or -1 for non-matches.

8.13.1.15 remove_minus_one()

```
std::ext::V_int remove_minus_one (
            std::ext::V_int vec )
```

Removes elements equal to -1 from the vector.

Parameters

vec	The vector to filter.

Returns

A vector with all elements that are not equal to -1.

8.13.1.16 scale()

```
void scale (
            DensMat & mat,
            bool center = true,
            bool scale = true )
```

Scales a matrix by centering and/or scaling its columns.

Parameters

mat	The matrix to scale.
center	Whether to center the matrix by subtracting the mean of each column.
scale	Whether to scale the matrix by dividing by the standard deviation of each column.

8.13.1.17 slice()

```
template<typename T >
std::ext::V_opt_string slice (
            std::vector< T > const & vec,
```

```
            std::ext::V_int const & indices )
```

Extracts elements from a vector based on specified indices.

Template Parameters

T	The type of elements in the vector.

Parameters

vec	The original vector.
indices	The indices of the elements to extract.

Returns

A vector of optional strings, with nullopt representing NA.

8.13.1.18 unique_less_equal()

```
bool unique_less_equal (
            DensVec const & vec,
            int threshold )
```

Checks if the number of unique elements in a vector is less than or equal to a threshold.

Parameters

vec	The vector to check.
threshold	The threshold value.

Returns

True if the number of unique elements is less than or equal to the threshold, otherwise false.

8.14 MAGEE.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "GMMAT.h"
00004 #include "declars.h"
00005 #include "ReadBGEN.h"
00006 #include <numeric>
00007 #include <thread>
00008
00016 struct Magee_Glmmkin
00017 {
00018     DensMat E;
00019     DensMat EC;
00020     SpaMat J;
00021     std::ext::V_int select;
00022     bool dupflag = false;
00023     SpaMat sigma_i;
00024     DensVec diag_sigma_i;           //  Woodbury
00025     SpaMat  diag_sigma_i_ZPchol; //  Woodbury
00026     SpaMat sigma_ix;           //new sigma_ix which is sparse
00027     SpaMat cov;
00028     DensVec residuals;
00029     DensMat JEres;
00030     SpaMat JPJ;
00031     SpaMat JEPJ;
00032     SpaMat JEPEJ;
00033 };
00034
00039 struct Magee_Arma {
00040     arma::mat EC;
00041     std::ext::V_int select;
```

```
00042     bool dupflag = false;
00043     arma::sp_mat cov;
00044     size_t n_obs;
00045     size_t n;
00046     arma::mat Jres;
00047     arma::mat JEresblock;
00048     arma::sp_mat Psi;
00049     arma::sp_mat Xi;
00050     arma::sp_mat sigma_iJJ;
00051     arma::sp_mat sigma_ixJ;
00052     arma::vec diag_sigma_i;
00053     arma::sp_mat diag_sigma_i_ZPchol;
00054 };
00055
00060 class MAGEE
00061 {
00062     public:
00063         enum class GENOTYPE {Bgen, Pgen, Bed};
00064         MAGEE() = default;
00065         MAGEE(GMMAT &gmmat, Glmmkin &glmmkin,  CommandLine &cmd, Bgen bgen,
00066                 std::ext::V_string interaction_exp, std::ext::V_string interaction_cov);
00067         MAGEE(GMMAT &gmmat, Glmmkin &glmmkin,  CommandLine &cmd, Pgen pgen,
00068                 std::ext::V_string interaction_exp, std::ext::V_string interaction_cov);
00069         MAGEE(GMMAT &gmmat, Glmmkin &glmmkin,  CommandLine &cmd, Bed bed,
00070                 std::ext::V_string interaction_exp, std::ext::V_string interaction_cov);
00075         void fitglmm();
00076     private:
00077         GMMAT *m_gmmat;
00078         Magee_Glmmkin m_magee_glmmkin;
00079         Glmmkin &m_glmmkin_fitnull;
00080         CommandLine &m_cmd;
00081         // Bgen &m_bgen;
00082         std::variant<Bgen, Pgen, Bed> m_genotype;
00083         GENOTYPE m_active_genotype;
00084         std::ext::V_string m_interaction;
00085         std::ext::V_string m_interaction_exp;
00086         std::ext::V_string m_interaction_new;
00087         std::ext::V_string m_interaction_cov;
00088         // int m_numSelCol;
00089         std::ext::Map_str_Vint m_strata_list;
00090         std::ext::V_string m_bin_headers;
00095         void clear_m_magee_glmmkin();
00100         void create_E();
00101
00111         void fill_sel(std::ext::V_string& sample_id);
00112
00122         void fill_mat(const int numRows, const int numCols, std::ext::V_int& indixes_col, int value);
00123
00132         void fill_J(std::ext::V_int& match_id, std::ext::V_string& sample_id);
00133
00151         void calculate_bin_header();
00152
00157         void printOutputHeader_magee();
00158 };
00159
00167 void conver_eigen_to_arma( SpaMat const& eigenMat, arma::sp_mat& armaMat);
00168
00180 std::ext::V_int match_indices(std::ext::V_string const& original, std::ext::V_opt_string const&
    filtered);
00181
00190 template <typename T>
00191 std::ext::V_opt_string slice(std::vector<T> const& vec, std::ext::V_int const& indices);
00192
00199 bool any_isna(std::ext::V_int vec);
00200
00208 std::ext::V_bool in_op(std::ext::V_string const& vec1, std::ext::V_string const& vec2);
00209
00210
00218 std::ext::V_string filtered_ids(std::ext::V_string const& vec1, std::ext::V_bool const& vec2);
00219
00226 void check_not_empty(std::ext::V_string const& sample_id);
00227
00228
00235 bool any_duplicated(std::ext::V_string const& vec);
00236
00243 std::ext::V_bool list_duplicates_bool(std::ext::V_string const& vec);
00244
00255 DensMat filter_unique_rows(DensMat const& mat, std::ext::V_string& id_include);
00256
00265 std::ext::V_bool apply_on_columns(DensMat const& mat, const std::function<bool(VectorXd const&)>&
    func, int threshold);
00266
00274 bool unique_less_equal(DensVec const& vec, int threshold);
00275
00282 std::ext::Map_str_Vint generate_strata_list(std::ext::V_string const& vec);
00283
00291 void scale(DensMat& mat, bool center = true, bool scale = true);
```

```
00292
00302 DensMat cbind(DensMat const& A, DensMat const& B);
00303
00313 std::ext::V_string match_id_include(std::ext::V_string const& id_include, std::ext::V_string const&
      sample_id);
00314
00321 std::ext::V_int remove_minus_one(std::ext::V_int vec);
00322
00329 std::ext::V_string create_strata(std::ext::VV_string const& Ecat);
```

8.15 include/MatrixUtils.h File Reference

Functions

- void initvec (double *v, int N)
- void matTmatprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matmatprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matNmatNprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matmatTprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matTvecprod (double *A, double *v, double *u, int Nrow, int Ncol)
- void matvecprod (double *A, double *v, double *u, int Nrow, int Ncol)
- void vecmatprod (double *v, double *A, double *u, int N)
- void matInv (double *A, int N)
- void matAdd (double *A, double *B, int N, double alpha)
- void subMatrix (double *A, double *u, int M, int N, int NrowA, int NrowB, int start)
- void matvecSprod (double *A, double *v, double *u, int Nrow, int Ncol, int start)

8.15.1 Function Documentation

8.15.1.1 initvec()

```
void initvec (
            double * v,
            int N )
```

8.15.1.2 matAdd()

```
void matAdd (
            double * A,
            double * B,
            int N,
            double alpha )
```

8.15.1.3 matInv()

```
void matInv (
            double * A,
            int N )
```

8.15.1.4 matmatprod()

```
void matmatprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.15.1.5 matmatTprod()

```
void matmatTprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.15.1.6 matNmatNprod()

```
void matNmatNprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.15.1.7 matTmatprod()

```
void matTmatprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.15.1.8 matTvecprod()

```
void matTvecprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol )
```

8.15.1.9 matvecprod()

```
void matvecprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol )
```

8.15.1.10 matvecSprod()

```
void matvecSprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int start )
```

8.15.1.11 subMatrix()

```
void subMatrix (
            double * A,
            double * u,
            int M,
            int N,
            int NrowA,
            int NrowB,
            int start )
```

8.15.1.12 vecmatprod()

```
void vecmatprod (
            double * v,
            double * A,
            double * u,
            int N )
```

8.16 MatrixUtils.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXUTILS_H
00002 #define MATRIXUTILS_H
00003
00004 void initvec(double* v, int N);
00005
00006 void matTmatprod(double* A, double* v, double* u, int Nrow, int Ncol, int NcolB);
00007
00008 void matmatprod(double* A, double* v, double* u, int Nrow, int Ncol, int NcolB);
00009
00010 void matNmatNprod(double* A, double* v, double* u, int Nrow, int Ncol, int NcolB);
00011
00012 void matmatTprod(double* A, double* v, double* u, int Nrow, int Ncol, int NcolB);
00013
00014 void matTvecprod(double* A, double* v, double* u, int Nrow, int Ncol);
00015
00016 void matvecprod(double* A, double* v, double* u, int Nrow, int Ncol);
00017
00018 void vecmatprod(double* v, double* A, double* u, int N);
00019
00020 void matInv(double* A, int N);
00021
00022 void matAdd(double* A, double* B, int N, double alpha);
00023
00024 void subMatrix(double* A, double* u, int M, int N, int NrowA, int NrowB, int start);
00025
00026 void matvecSprod(double* A, double* v, double* u, int Nrow, int Ncol, int start);
00027 #endif
```

8.17 include/Pheno.h File Reference

```
#include "ReadFiles.h"
```

Classes

- class Pheno

8.18 Pheno.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "ReadFiles.h"
00004
00005
00006 class Pheno
```

```
00007 {
00008     public:
00009         DataFrame m_data_frame;
00010         std::string m_sam_id;
00011         std::ext::V_string m_v_hdrs;
00017         unsigned int size();
00023         void set_path(std::string_view path);
00029         std::string get_path() const;
00035         void read_file(std::string_view, char delim = ',');
00036         std::pair<std::string, std::string> check_binary(std::string pheno_name);
00037     private:
00038         std::string m_path;
00039 };
```

8.19 include/ReadBed.h File Reference

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include "ReadParameters.h"
#include "TimeUtils.h"
```

Classes

- class Bed

Macros

- #define READBED_H

Functions

- void gemBED (int thread_num, double sigma2, double ∗resid, double ∗XinvXTX, vector< double > miu, BinE binE, Bed bed, CommandLine cmd)

8.19.1 Macro Definition Documentation

8.19.1.1 READBED_H

```
#define READBED_H
```

8.19.2 Function Documentation

8.19.2.1 gemBED()

```
void gemBED (
            int thread_num,
            double sigma2,
            double * resid,
            double * XinvXTX,
            vector< double > miu,
            BinE binE,
            Bed bed,
            CommandLine cmd )
```

8.20 ReadBed.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <string>
00003 #include <stdio.h>
00004 #include <stdlib.h>
00005
```

```
00006 #include "ReadParameters.h"
00007 #include "TimeUtils.h"
00008
00009 #ifndef READBED_H
00010 #define READBED_H
00011
00012 class Bed {
00013
00014 public:
00015
00016     uint32_t n_samples = 0;
00017     uint32_t n_variants = 0;
00018     char famDelim;
00019     char bimDelim;
00020     int bimLast;
00021     int new_samSize;
00022     std::vector<string>   sampleID;
00023     std::vector<string>   sampleID_all;
00024     std::vector<double>   new_covdata;
00025     std::vector<double>   new_phenodata;
00026     std::vector<long int> include_idx;
00027     // For check of co-linear relations between covX;
00028     int numIntSelCol_new;
00029     int numExpSelCol_new;
00030     int numSelCol_new;
00031     std::vector<int> excludeCol;
00032     int phenoType;
00033     vector<uint32_t> begin;
00034     vector<uint32_t> end;
00035     uint32_t threads;
00036     bool filterVariants = false;
00037     std::vector<long long unsigned int> bedVariantPos;
00038
00039
00040     void processBed(string bedFile, string bimFile, string famFile);
00041     void processFam(Bed bed, string famFile, unordered_map<string, vector<vector<string>> phenomap,
      string phenoMissingKey, int numSelCol, int samSize);
00042     void getBedVariantPos(Bed bed, CommandLine cmd);
00043 };
00044
00045 void gemBED(int thread_num, double sigma2, double* resid, double* XinvXTX, vector<double> miu, BinE
      binE,  Bed bed, CommandLine cmd);
00046
00047 #endif
```

8.21 include/ReadBGEN.h File Reference

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "ReadParameters.h"
#include "BinaryEUtils.h"
#include "TimeUtils.h"
```

Classes

- class Bgen

  Reader and processor for BGEN genotype files (v1.1–v1.3).

Functions

- void gemBGEN (int thread_num, double sigma2, double ∗resid, double ∗XinvXTX, vector< double > miu, BinE binE, Bgen bgen, CommandLine cmd)
- void Bgen13GetTwoVals (const unsigned char ∗prob_start, uint32_t bit_precision, uintptr_t offset, uintptr_t ∗first_val_ptr, uintptr_t ∗second_val_ptr)

8.21.1 Function Documentation

8.21.1.1 Bgen13GetTwoVals()

```
void Bgen13GetTwoVals (
            const unsigned char * prob_start,
            uint32_t bit_precision,
            uintptr_t offset,
            uintptr_t * first_val_ptr,
            uintptr_t * second_val_ptr )
```

8.21.1.2 gemBGEN()

```
void gemBGEN (
            int thread_num,
            double sigma2,
            double * resid,
            double * XinvXTX,
            vector< double > miu,
            BinE binE,
            Bgen bgen,
            CommandLine cmd )
```

8.22 ReadBGEN.h

Go to the documentation of this file.

```
00001
00002 #include <string>
00003 #include <stdio.h>
00004 #include <stdlib.h>
00005 #include <math.h>
00006 #include "ReadParameters.h"
00007 #include "BinaryEUtils.h"
00008 #include "TimeUtils.h"
00009 #ifndef READBGEN_H
00010 #define READBGEN_H
00011
00023 class Bgen {
00024
00025     public:
00026
00027         // For file
00028         FILE* fin;
00029
00030         // For BGEN offset
00031         uint offset;
00032
00033         // For BGEN header block
00034         uint Mbgen;
00035         uint Nbgen;
00036         uint CompressedSNPBlocks;
00037         uint Layout;
00038
00039         // For BGEN header-flag block;
00040         uint SampleIdentifiers;
00041
00042
00043         // For ID matching
00044         int new_samSize;
00045         std::vector<string>   sampleID;
00046         //AllsampleIDs before matching
00047         std::vector<string>   sampleID_all;
00048         std::vector<double>   new_covdata;
00049         std::vector<double>   new_phenodata;
00050         std::vector<long int> include_idx;
00051         vector <long int> variant_pos;
00052         std::vector<unsigned int> includeVariantIndex;
00053         // For check of co-linear relations between covX;
00054         int numIntSelCol_new;
00055         int numExpSelCol_new;
00056         int numSelCol_new;
00057         std::vector<int> excludeCol;
00058         // For multithreading BGEN file
00059         int phenoType;
00060         uint threads;
00061         bool filterVariants;
00062         std::vector<uint> Mbgen_begin;
```

```
00063         std::vector<uint> Mbgen_end;
00064         std::vector<long long unsigned int> bgenVariantPos;
00065         std::vector<vector<uint» keepVariants;
00066
00067
00068         void processBgenHeaderBlock(string bgenfile);
00069         void processBgenSampleBlock(Bgen bgen, char samplefile[300], bool useSample,
      unordered_map<string, vector<vector<string»> phenomap, string phenoMissingKey, int numSelCol, int
      samSize);
00070         void getPositionOfBgenVariant(Bgen bgen, CommandLine cmd);
00071 };
00072
00073 void gemBGEN(int thread_num, double sigma2, double* resid, double* XinvXTX, vector<double> miu, BinE
      binE, Bgen bgen, CommandLine cmd);
00074 void Bgen13GetTwoVals(const unsigned char* prob_start, uint32_t bit_precision, uintptr_t offset,
      uintptr_t* first_val_ptr, uintptr_t* second_val_ptr);//Extracts two values from a binary-encoded
      probability array
00075
00076 #endif
00077
```

8.23 include/ReadFiles.h File Reference

```
#include <optional>
#include <string>
#include <string_view>
#include <map>
#include <unordered_map>
#include <utility>
#include <set>
#include <vector>
#include <variant>
#include "fmt/core.h"
#include "fmt/format.h"
#include <spdlog/spdlog.h>
```

Classes

- class DataFrame

  A class for storing infromation in the tabular files.

Namespaces

- namespace std
- namespace std::ext

Typedefs

- using std::ext::V_string = std::vector< std::string >
- using std::ext::VV_string = std::vector< V_string >
- using std::ext::Data = unordered_map< string, V_string >
- using std::ext::V_double = std::vector< double >
- using std::ext::V_float = std::vector< float >
- using std::ext::V_int = std::vector< int >
- using std::ext::V_size_t = std::vector< std::size_t >
- using std::ext::VV_int = std::vector< V_int >
- using std::ext::V_bool = std::vector< bool >
- using std::ext::V_lluint = std::vector< long long unsigned int >
- using std::ext::map_str_int = std::map< std::string, int >
- using std::ext::Umap_str_int = std::unordered_map< std::string, int >
- using std::ext::V_opt_string = std::vector< std::optional< std::string > >
- using std::ext::Var_bool_int = std::variant< V_bool, V_int >
- using std::ext::Map_str_Vint = std::map< string, V_int >

8.24 ReadFiles.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <optional>
00003 #include <string>
00004 #include <string_view>
00005 #include <map>
00006 #include <unordered_map>
00007 #include <utility>
00008 #include <set>
00009 #include <vector>
00010 #include <variant>
00011 #include "fmt/core.h"
00012 #include "fmt/format.h"
00013 #include <spdlog/spdlog.h>
00014
00015
00016
00017 namespace std
00018 {
00019     namespace ext
00020     {
00021         using V_string = std::vector<std::string>;
00022         using VV_string = std::vector<V_string>;
00023         using Data = unordered_map<string, V_string>;
00024         using V_double = std::vector<double>;
00025         using V_float = std::vector<float>;
00026         using V_int = std::vector<int>;
00027         using V_size_t = std::vector<std::size_t>;
00028         using VV_int = std::vector<V_int>;
00029         using V_bool = std::vector<bool>;
00030         using V_lluint = std::vector<long long unsigned int>;
00031         using map_str_int = std::map<std::string, int>;
00032         using Umap_str_int = std::unordered_map<std::string, int>;
00033         using V_opt_string = std::vector<std::optional<std::string>>;
00034         using Var_bool_int = std::variant<V_bool, V_int>;
00035         using Map_str_Vint = std::map<string, V_int>;
00036     }
00037 }
00038
00043 class DataFrame
00044 {
00045     public:
00046         std::ext::Data m_data;
00047         int m_nrows = 0;
00048         int m_ncols = 0;
00049         std::ext::V_string m_headers;
00050         std::string m_missing_key;
00051         std::ext::V_string m_geno_ids;//bgenIDs
00052
00053         bool isDataFrameCreated() const;
00054         int n_rows() const;
00055         int n_cols() const;
00056
00062         void head(int n = 5);
00069         DataFrame copy_by_hdrs(std::ext::V_string const& v_hdrs);
00076         void read_file(std::string_view path, char delim = ',');
00084         void read_file(std::string_view path,
00085                        std::ext::V_string const& keep_headers,
00086                        char delim = ',');
00093         std::ext::V_string get_header(std::string const& hdr) const;
00099         void remove_duplicates(std::ext::V_string const& v_hdrs);
00100         size_t size_wo_duplicates(std::string const& hdr);
00101         DataFrame remove_duplicates(std::string const& hdr);
00102         bool any_duplicated(std::string const& hdr);
00103         std::set<std::string> list_duplicates(std::string const& hdr);
00104         void match_genoids(std::string hdr_id, std::ext::V_string const& v_hdrs);
00105         std::ext::V_string list_unique(std::string const& hdr);
00106     private:
00107         std::ext::V_string read_lines(std::string_view path);
00108         void fill_data(std::ext::V_string const& v_strs, char delim = ',');
00109         void fill_data(std::ext::V_string const& v_strs,
00110                        std::ext::V_string const& keep_headers, char delim= ','
00111                        );
00112 };
00113
00114
00115
00116
00117
00118
00119
```

8.25 include/ReadParameters.h File Reference

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
```

Classes

- class CommandLine

8.26 ReadParameters.h

Go to the documentation of this file.

```
00001 #ifndef READPARAMETERS_H
00002 #define READPARAMETERS_H
00003
00004 #include <string>
00005 #include <stdio.h>
00006 #include <stdlib.h>
00007
00008 class CommandLine {
00009
00010 public:
00011     char samplefile[300];
00012
00013     // Genotype files
00014     std::string bgenFile;
00015     std::string sampleFile;
00016     std::string pgenFile;
00017     std::string pvarFile;
00018     std::string psamFile;
00019     std::string bedFile;
00020     std::string bimFile;
00021     std::string famFile;
00022     bool useSampleFile = false;
00023     bool useBgenFile = false;
00024     bool usePgenFile = false;
00025     bool useBedFile = false;
00026
00027     // Phenotype file
00028     std::string pheno_file;
00029     std::string delim;
00030     std::string missing;
00031     char pheno_delim;
00032
00033     // kinship file
00034     std::string kin_file;
00035     std::string delim_k;
00036     bool kin_flag = false;
00037     double kin_diag;
00038 //  std::string missing;
00039     char kin_delim;
00040
00041     // Categorical exposures
00042     int cat_threshold;
00043     std::vector<std::string> cat_names;
00044
00045     // Out file
00046     std::string outFile;
00047     std::string outStyle;
00048
00049     // Inputs
00050     std::string phenoName;
00051     std::string randomSlope;
00052     std::string sampleID;
00053
00054     int numSelCol    = 0;
00055     int numExpSelCol = 0;
00056     int numIntSelCol = 0;
00057     std::vector<std::string> cov;
00058     std::vector<std::string> icov;
00059     std::vector<std::string> exp;
00060     std::string group;
00061     std::unordered_map<std::string, int> covHM;
00062     std::unordered_map<std::string, int> intHM;
00063     std::unordered_map<std::string, int> expHM;
00064
00065     // Filtering options
00066     double MAF;
```

```
00067     double missGenoRate;
00068     std::string includeVariantFile;
00069     bool doFilters;
00070 
00071     // Performance options
00072     int center;
00073     int scale;
00074     double tol;
00075     int robust;
00076     int threads;
00077     int stream_snps;
00078     void processCommandLine(int argc, char* argv[]);
00079 };
00080 
00081 
00082 #endif
```

8.27 include/ReadPGEN.h File Reference

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include "ReadParameters.h"
#include "TimeUtils.h"
```

Classes

- class Pgen

Functions

- void gemPGEN (int thread_num, double sigma2, double ∗resid, double ∗XinvXTX, vector< double > miu, BinE binE, Pgen pgen, CommandLine cmd)

8.27.1 Function Documentation

8.27.1.1 gemPGEN()

```
void gemPGEN (
              int thread_num,
              double sigma2,
              double * resid,
              double * XinvXTX,
              vector< double > miu,
              BinE binE,
              Pgen pgen,
              CommandLine cmd )
```

8.28 ReadPGEN.h

Go to the documentation of this file.

```
00001 #include <string>
00002 #include <stdio.h>
00003 #include <stdlib.h>
00004 
00005 #include "ReadParameters.h"
00006 #include "TimeUtils.h"
00007 
00008 #ifndef READPGEN_H
00009 #define READPGEN_H
00010 
00011 class Pgen {
00012 
00013 public:
00014 
00015     uint32_t raw_variant_ct;
00016     uint32_t raw_sample_ct;
```

```
00017
00018     int new_samSize;
00019     std::vector<string>   sampleID;
00020     std::vector<string>   sampleID_all;
00021     std::vector<double>   new_covdata;
00022     std::vector<double>   new_phenodata;
00023     std::vector<long int> include_idx;
00024
00025     int NumExcludeCol;
00026     int phenoType;
00027     vector<uint32_t> begin;
00028     vector<uint32_t> end;
00029     uint32_t threads;
00030     bool filterVariants = false;
00031     std::vector<long long unsigned int> pgenVariantPos;
00032     vector<int> pvarIndex;
00033     int pvarLast;
00034     vector<int> binE_idx;
00035 // For check of co-linear relations between covX;
00036     int numIntSelCol_new;
00037     int numExpSelCol_new;
00038     int numSelCol_new;
00039     std::vector<int> excludeCol;
00040
00041
00042     void processPgenHeader(string pgenFile);
00043     void processPvar(Pgen pgen, string pvarFile);
00044     void processPsam(Pgen pgen, string psamFile, unordered_map<string, vector<vector<string>>
      phenomap, string phenoMissingKey, int numSelCol, int samSize);
00045     void getPgenVariantPos(Pgen pgen, CommandLine cmd);
00046 };
00047
00048 void gemPGEN(int thread_num, double sigma2, double* resid, double* XinvXTX, vector<double> miu, BinE
      binE, Pgen pgen, CommandLine cmd);
00049
00050 #endif
```

8.29 include/SparseInverse.h File Reference

```
#include <armadillo>
#include <Eigen/Sparse>
#include <Eigen/SparseCholesky>
#include <Eigen/Core>
#include <Eigen/Dense>
#include <cholmod.h>
#include <cs.h>
#include "Pheno.h"
#include "Kinship.h"
#include <tuple>
#include <iostream>
```

Classes

- struct pair_hash
- class SparseInverse

Namespaces

- namespace std
- namespace std::ext

Typedefs

- using std::ext::IndexMapUnq = unordered_map< string, int >
- using std::ext::IndexMap = unordered_map< string, V_int >
- using std::ext::IndexMapRev = unordered_map< int, string >
- using std::ext::Triplet_d = Eigen::Triplet< double >
- using std::ext::Triplet_i = Eigen::Triplet< int >

- using std::ext::VecTuples4spmat = vector< Triplet_d >
- using std::ext::VecTriple_i = vector< Triplet_i >
- using SpaMat = Eigen::SparseMatrix< double, Eigen::ColMajor >
- using SpaMatRow = Eigen::SparseMatrix< double, Eigen::RowMajor, int >
- using DensMat = Eigen::MatrixXd
- using DensVec = Eigen::VectorXd
- using DensVecInt = Eigen::VectorXi
- using DenseMatInt = Eigen::MatrixXi

Functions

- template<typename Tuple , size_t... Indices>
  void std::ext::tuplePrint (const Tuple &t, std::index_sequence< Indices... >)
- template<typename... Args>
  void std::ext::tuplePrint (const std::tuple< Args... > &t)

8.29.1 Typedef Documentation

8.29.1.1 DenseMatInt

```
using DenseMatInt = Eigen::MatrixXi
```

8.29.1.2 DensMat

```
using DensMat = Eigen::MatrixXd
```

8.29.1.3 DensVec

```
using DensVec = Eigen::VectorXd
```

8.29.1.4 DensVecInt

```
using DensVecInt = Eigen::VectorXi
```

8.29.1.5 SpaMat

```
using SpaMat = Eigen::SparseMatrix<double, Eigen::ColMajor>
```

8.29.1.6 SpaMatRow

```
using SpaMatRow = Eigen::SparseMatrix<double, Eigen::RowMajor, int>
```

8.30 SparseInverse.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <armadillo>
00003 #include <Eigen/Sparse>
00004 #include <Eigen/SparseCholesky>
00005 #include <Eigen/Core>
00006 #include <Eigen/Dense>
00007 #include <cholmod.h>
00008 #include <cs.h>
00009 #include "Pheno.h"
00010 #include "Kinship.h"
00011 #include <tuple>
00012 #include <iostream>
00013
00014
00015 namespace std
00016 {
00017     namespace ext
00018     {
00019         using IndexMapUnq = unordered_map<string, int>;
00020         using IndexMap = unordered_map<string, V_int>;
```

```
00021         using IndexMapRev = unordered_map<int, string>;
00022         using Triplet_d = Eigen::Triplet<double>;
00023         using Triplet_i = Eigen::Triplet<int>;
00024         using VecTuples4spmat = vector<Triplet_d>;
00025         using VecTriple_i = vector<Triplet_i>;
00026 
00027 
00028         // a template helper function and functions for printing tuples
00029         template<typename Tuple, size_t... Indices>
00030         void tuplePrint(const Tuple& t, std::index_sequence<Indices...>)
00031         {
00032             ((cout « std::get<Indices>(t) « " "), ...);
00033         }
00034 
00035         template<typename... Args>
00036         void tuplePrint(const std::tuple<Args...>& t)
00037         {
00038             tuplePrint(t, std::index_sequence_for<Args...>());
00039             cout « "\n";
00040         }
00041 
00042     }
00043 }
00044 
00045 struct pair_hash
00046 {
00047     template <class T1, class T2>
00048     std::size_t operator () (std::pair<T1,T2> const& pair) const
00049     {
00050         auto h1 = std::hash<T1>{}(pair.first);
00051         auto h2 = std::hash<T2>{}(pair.second);
00052         return h1 ^ h2;
00053     }
00054 };
00055 
00056 using SpaMat = Eigen::SparseMatrix<double, Eigen::ColMajor>;
00057 using SpaMatRow = Eigen::SparseMatrix<double, Eigen::RowMajor, int>;
00058 using DensMat = Eigen::MatrixXd;
00059 using DensVec = Eigen::VectorXd;
00060 using DensVecInt = Eigen::VectorXi;
00061 using DenseMatInt = Eigen::MatrixXi;
00062 
00063 class SparseInverse
00064 {
00065     public:
00066         Pheno pheno;
00067         Kinship kin;
00068         char pheno_delim;
00069         char kin_delim;
00070         int m_ratio_Nob2N;
00071         int m_thr = 2; // threshold to decide on Woodbury
00072 
00073         SparseInverse() =  default;
00074         SparseInverse(std::string kin_add, std::string pheno_add, char kin_delim,
00075                       double kin_diag, char pheno_delim, std::string &m_sam_id,
00076                       std::ext::V_string &m_v_hdrs, std::ext::V_string &bgen_sample_id,
00077                       std::string phenoMissingKey) ;
00078         void set_idx_mp_unq(std::ext::V_string const& vec);
00079         void set_idx_mp(std::ext::V_string v_strs);
00080         std::ext::IndexMap get_idx_mp();
00081         std::ext::IndexMapUnq get_idx_mp_unq();
00082         void set_kin_Nsize(std::ext::VecTuples4spmat& vt4spmat);
00083         void set_kin_Nobssize(std::ext::VecTuples4spmat& vt4spmat);
00084         void set_spmat();
00085         void set_spmat(SpaMat sm);
00086         SpaMat& get_spmat();
00087         SpaMat inv_spamat();
00088         static SpaMat solve_chol(const SpaMat& sm);
00089         //define static as we might not need an object
00090         static SpaMat inv_spamat(SpaMat const& sm);
00091         static SpaMat inv_spamat_chol(SpaMat const& sm, bool return_full_inverse=false);
00092         static DensMat inv(DensMat const &dm);
00093 
00094     private:
00095        std::ext::IndexMap m_idx_mp;
00096        std::ext::IndexMapUnq m_idx_mp_unq;
00097        SpaMat m_spmat;
00098     //    std::vector<SpaMat> m_vspmat;
00099        //std::ext::V_string get_pheno_samid();
00100         void create_tuple4spmat(std::ext::VecTuples4spmat& vt4spmat);
00101        [[nodiscard]] bool is_missing(int id1, int id2);
00102 };
```

8.31 include/TimeUtils.h File Reference

```
#include <chrono>
```

Classes

- class Time

Macros

- #define TIMEUTILS_H

Functions

- void printExecutionTime (std::chrono::steady_clock::time_point start_time, std::chrono::steady_clock::time_point end_time)
- void printExecutionTime1 (std::chrono::steady_clock::time_point start_time, std::chrono::steady_clock::time_point end_time)

8.31.1 Macro Definition Documentation

8.31.1.1 TIMEUTILS_H

```
#define TIMEUTILS_H
```

8.31.2 Function Documentation

8.31.2.1 printExecutionTime()

```
void printExecutionTime (
            std::chrono::steady_clock::time_point start_time,
            std::chrono::steady_clock::time_point end_time )
```

8.31.2.2 printExecutionTime1()

```
void printExecutionTime1 (
            std::chrono::steady_clock::time_point start_time,
            std::chrono::steady_clock::time_point end_time )
```

8.32 TimeUtils.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #ifndef TIMEUTILS_H
00004 #define TIMEUTILS_H
00005
00006
00007 #include <chrono>
00008
00009 class Time {
00010
00011 public:
00012 };
00013
00014 void printExecutionTime(std::chrono::steady_clock::time_point start_time,
      std::chrono::steady_clock::time_point end_time);
00015 void printExecutionTime1(std::chrono::steady_clock::time_point start_time,
      std::chrono::steady_clock::time_point end_time);
00016 #endif
00017
```

8.33 README.md File Reference

8.34 src/README.md File Reference

8.35 src/BinaryEUtils.cpp File Reference

```
#include "declars.h"
#include "BinaryEUtils.h"
```

Functions

- std::unordered_map< std::string, unsigned int > cartesian_map (vector< vector< string > > &v)
- std::vector< std::string > cartesian_vec_sep (vector< vector< string > > &v)

8.35.1 Function Documentation

8.35.1.1 cartesian_map()

```
std::unordered_map< std::string, unsigned int > cartesian_map (
            vector< vector< string > > & v )
```

8.35.1.2 cartesian_vec_sep()

```
std::vector< std::string > cartesian_vec_sep (
            vector< vector< string > > & v )
```

8.36 src/FitgImm.cpp File Reference

```
#include "MAGEE.h"
#include "declars.h"
#include "ReadBGEN.h"
#include "../thirdparty/zstd-1.5.5/lib/zstd.h"
#include "../thirdparty/libdeflate-1.18/libdeflate.h"
#include "../thirdparty/plink-2.0/plink2_bits.h"
#include "../thirdparty/plink-2.0/plink2_base.h"
#include "../thirdparty/plink-2.0/pgenlib_misc.h"
#include "../thirdparty/plink-2.0/pgenlib_read.h"
#include "../thirdparty/plink-2.0/pgenlib_ffi_support.h"
#include <stdexcept>
#include <fstream>
#include <cmath>
#include <cstring>
#include <cstdio>
#include <zlib.h>
#include <memory>
#include <stdint.h>
#include <limits.h>
```

Macros

- #define DBL_EPSILON 2.2204460492503131e-16

Typedefs

- using uint = unsigned int
- using uchar = unsigned char

- using ushort = unsigned short
- using llui = long long unsigned int

Functions

- template<typename Container , typename T >
  std::vector< int > find_indx (const Container &container, T value)
- template<typename Pred >
  std::vector< int > filter_elements (const arma::vec &container, const arma::Col< int > &condition, Pred pred)
- double chi_square_CDF (double x, double df, bool lower_tail, bool log_p)
- void glmm_gei (std::string snpID, arma::mat &G, arma::uvec &snp_skip, size_t &npbidx, size_t npb, size_t n, int ei, int qi, const Magee_Arma &null_obj, std::ofstream &writefile, bool meta_output, std::ext::V_string &tmpout, uint m, uint end)
- void glmm_gei_bgen13 (Magee_Arma const &null_obj, string const &bgenfile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_list, uint begin, uint end, long long unsigned int byte, uint Nbgen, uint compression, bool meta_output)
- void glmm_gei_pgen13 (Magee_Arma const &null_obj, string const &pgenfile, std::string pvarFile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_list, uint begin, uint end, std::ext::V_lluint pgenPos, bool filterVariants, int pvarLength, int pvarLast, std::ext::V_int pvarIndex, bool meta_output)
- void glmm_gei_bed13 (Magee_Arma const &null_obj, string const &bedfile, std::string bimFile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_↩ list, uint begin, uint end, std::ext::V_lluint bedPos, bool filterVariants, char bimDelim, int bimLast, uint32_t n_samples, bool meta_output)

8.36.1 Macro Definition Documentation

8.36.1.1 DBL_EPSILON

```
#define DBL_EPSILON 2.2204460492503131e-16
```

8.36.2 Typedef Documentation

8.36.2.1 llui

```
using llui = long long unsigned int
```

8.36.2.2 uchar

```
using uchar = unsigned char
```

8.36.2.3 uint

```
using uint = unsigned int
```

8.36.2.4 ushort

```
using ushort = unsigned short
```

8.36.3 Function Documentation

8.36.3.1 chi_square_CDF()

```
double chi_square_CDF (
            double x,
            double df,
            bool lower_tail,
            bool log_p )
```

8.36.3.2 filter_elements()

```
template<typename Pred >
std::vector< int > filter_elements (
            const arma::vec & container,
            const arma::Col< int > & condition,
            Pred pred )
```

8.36.3.3 find_indx()

```
template<typename Container , typename T >
std::vector< int > find_indx (
            const Container & container,
            T value )
```

8.36.3.4 glmm_gei()

```
void glmm_gei (
            std::string snpID,
            arma::mat & G,
            arma::uvec & snp_skip,
            size_t & npbidx,
            size_t npb,
            size_t n,
            int ei,
            int qi,
            const Magee_Arma & null_obj,
            std::ofstream & writefile,
            bool meta_output,
            std::ext::V_string & tmpout,
            uint m,
            uint end )
```

8.36.3.5 glmm_gei_bed13()

```
void glmm_gei_bed13 (
            Magee_Arma const & null_obj,
            string const & bedfile,
            std::string bimFile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            std::ext::V_lluint bedPos,
            bool filterVariants,
            char bimDelim,
            int bimLast,
            uint32_t n_samples,
            bool meta_output )
```

8.36.3.6 glmm_gei_bgen13()

```
void glmm_gei_bgen13 (
            Magee_Arma const & null_obj,
```

```
            string const & bgenfile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            long long unsigned int byte,
            uint Nbgen,
            uint compression,
            bool meta_output )
```

8.36.3.7 glmm_gei_pgen13()

```
void glmm_gei_pgen13 (
            Magee_Arma const & null_obj,
            string const & pgenfile,
            std::string pvarFile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            std::ext::V_lluint pgenPos,
            bool filterVariants,
            int pvarLength,
            int pvarLast,
            std::ext::V_int pvarIndex,
            bool meta_output )
```

8.37 src/GEM.cpp File Reference

```
#include "GEM.h"
```

Functions

- int main (int argc, char ∗argv[ ])
- std::string get_log_name (int argc, char ∗argv[ ])

  Extract the output/log file name from command-line arguments.
- int checkBinary (unordered_map< string, vector< vector< string > > > phenoMap, vector< string > sampleID, double epsilon)

  Determine whether the phenotype is binary or continuous.
- void center (int center, int scale, int samSize, int numSelCol, vector< double > covdata, vector< double > ∗covdata_ret)
- void printCovVarMat (int numCovs, vector< string > covNames, double ∗covVarMat, double ∗beta, int phenoType, int samSize)

  Print regression coefficients and covariance matrix of estimates.
- void printOutputHeader (bool useBgen, int numExpSelCol_new, int Sq1, vector< string > covNames, string output, string outStyle, int robust, double sigma2, BinE binE)

Write the header line for the GWAS output results file.

- void fitNullModel (int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret)

  Fit the null regression model (linear or logistic).
- void fitNullModel2 (int samSize, int numSelCol, int phenoType, double epsilon, int robust, std::vector< string > covSelHeadersName, std::vector< double > phenodata, std::vector< double > covdata, std::vector< double > *XinvXTX_ret, vector< double > *miu_ret, vector< double > *resid_ret, double *sigma2_ret, std::vector< double > &beta_ret, std::vector< double > &Xbeta_ret)

  Fit null regression model and return additional intermediate outputs.

8.37.1 Function Documentation

8.37.1.1 center()

```
void center (
            int center,
            int scale,
            int samSize,
            int numSelCol,
            vector< double > covdata,
            vector< double > * covdata_ret )
```

8.37.1.2 checkBinary()

```
int checkBinary (
            unordered_map< string, vector< vector< string > > > phenoMap,
            vector< string > sampleID,
            double epsilon )
```

Determine whether the phenotype is binary or continuous.
This function inspects the phenotype values across all provided sample IDs. It counts the number of unique phenotype categories:

- If only 2 unique values exist → phenotype is treated as binary.
- If more than 2 unique values exist → phenotype is treated as continuous.

If all phenotype values are identical, an error is printed.
When binary, the logistic regression convergence threshold is also reported.

Parameters

phenoMap	Map from sample ID to phenotype vector.
sampleID	Vector of sample IDs in analysis order.
epsilon	Convergence tolerance used in logistic regression.

Returns

int Returns 1 if phenotype is binary, 0 if continuous.

8.37.1.3 fitNullModel()

```
void fitNullModel (
            int samSize,
            int numSelCol,
            int phenoType,
            double epsilon,
            int robust,
```

```
            std::vector< string > covSelHeadersName,
            std::vector< double > phenodata,
            std::vector< double > covdata,
            std::vector< double > * XinvXTX_ret,
            vector< double > * miu_ret,
            vector< double > * resid_ret,
            double * sigma2_ret )
```

Fit the null regression model (linear or logistic).
This function fits a regression model without genotype effects:

- Linear regression for continuous traits
- Logistic regression for binary traits

Outputs include:

- Residual vector
- Dispersion estimate (sigma^2)
- Mean response (mu)
- Precomputed matrix X If logistic regression fails to converge after MAX_ITER, the program terminates with an error.

Parameters

samSize	
numSelCol	
phenoType	
epsilon	
robust	
covSelHeadersName	
phenodata	
covdata	
XinvXTX_ret	
miu_ret	
resid_ret	
sigma2_ret	

8.37.1.4 fitNullModel2()

```
void fitNullModel2 (
            int samSize,
            int numSelCol,
            int phenoType,
            double epsilon,
            int robust,
            std::vector< string > covSelHeadersName,
            std::vector< double > phenodata,
            std::vector< double > covdata,
            std::vector< double > * XinvXTX_ret,
            vector< double > * miu_ret,
            vector< double > * resid_ret,
            double * sigma2_ret,
            std::vector< double > & beta_ret,
            std::vector< double > & Xbeta_ret )
```

Fit null regression model and return additional intermediate outputs.
This is an extended version of fitNullModel().
In addition to standard null model outputs, it also returns:

- beta estimates (alpha)
- linear predictor X∗beta (eta)

These intermediate quantities are required for downstream GMMAT-style score statistic calculations.
Supports both:

- Linear regression
- Logistic regression

Parameters

samSize	
numSelCol	
phenoType	
epsilon	
robust	
covSelHeadersName	
phenodata	
covdata	
XinvXTX_ret	
miu_ret	
resid_ret	
sigma2_ret	
beta_ret	
Xbeta_ret	

8.37.1.5 get_log_name()

```
std::string get_log_name (
            int argc,
            char * argv[] )
```

Extract the output/log file name from command-line arguments.
This function scans the command-line inputs and looks for the option `--out`. If found, the following argument is returned as the log/output file name.
If the option is not provided, an empty string (fallback) is returned.

Parameters

argc	Number of command-line arguments.
argv	Array of command-line argument strings.

Returns

std::string The file name specified after `--out`, or an empty string.

8.37.1.6 main()

```
int main (
            int argc,
            char * argv[] )
```

8.37.1.7 printCovVarMat()

```
void printCovVarMat (
            int numCovs,
            vector< string > covNames,
            double * covVarMat,
            double * beta,
            int phenoType,
            int samSize )
```

Print regression coefficients and covariance matrix of estimates.
Function outputs:

- Estimated regression coefficients (beta)
- Standard errors
- Z-values
- Chi-square p-values

It also prints the full variance-covariance matrix.
Used after fitting the null model in linear or logistic regression.

Parameters

numCovs	Number of covariates.
covNames	Names of covariates.
covVarMat	Variance-covariance matrix.
beta	Estimated regression coefficients.
phenoType	Phenotype type (1=binary, 0=continuous).
samSize	Number of samples.

8.37.1.8 printOutputHeader()

```
void printOutputHeader (
            bool useBgen,
            int numExpSelCol_new,
            int Sq1,
            vector< string > covNames,
            string output,
            string outStyle,
            int robust,
            double sigma2,
            BinE binE )
```

Write the header line for the GWAS output results file.
This function generates the correct column header depending on:
The header includes:

- Variant information (CHR, POS, alleles, AF)
- Marginal effects
- Interaction effects
- Standard errors
- Covariance terms
- P-values

Parameters

useBgen	
numExpSelCol_new	
Sq1	
covNames	
output	
outStyle	
robust	
sigma2	
binE	

8.38 src/GMMAT.cpp File Reference

```
#include "../include/GMMAT.h"
#include <algorithm>
#include <limits>
#include <cmath>
#include <thread>
#include <mutex>
```

Namespaces

- namespace details

Functions

- void center_dataframe (DataFrame &df, const std::vector< std::string > &headers, bool center, bool scale)
- DensMat apply_wb (Eigen::Ref< const Eigen::MatrixXd > const &mat, DensVec const &diag_sigma_↩ i, SpaMat const &diag_sigma_i_ZPchol)
- template<typename T >
  int sign (T t)
- std::ext::V_int intersect (std::ext::V_int const &vec1, std::ext::V_int const &vec2)
- std::ext::V_int logic_update_fixed_condtion (DensVec dv, double tol)
- DensVec conv_vec_vecXd (std::ext::V_double v_std)
- SpaMat details::diag (DensVec vec)
- SpaMat details::diag (std::ext::V_double vec)
- DensMat create_covdata (const DataFrame &df)
- std::ext::map_str_int createHeaderMap (std::ext::V_string const &headers)
- std::ext::V_int conv_stdvs2stdvi (std::ext::V_string const &strings)
- DensVec linkinv (DensVec const &eta, std::string const &family_t, const std::string &link)
- double calc_variance (DensVec const &dv)
- std::ext::V_double conv_dm2stdV (DensMat const &dm)
- std::ext::V_double conv_dv2stdVd (DensVec const &dv)
- DensVec conv_stdVs2dV (std::ext::V_string const &stdv)
- DensVec conv_stdV2dv (std::ext::V_double v)
- DensVec slice_vec (DensVec const &dv, std::ext::V_int const &indices)

  Slice dense vector based on indexes.
- DensVec slice_vec (DensVec const &dv, std::ext::V_bool const &indices)

  Slice dense vector based on logical vector.
- DensVec slice_vec (DensVec const &dv, std::ext::Var_bool_int const &indices)

  Slice dense vector based on variant vector(either int or bool)

- SpaMat slice_mat (SpaMat const &spm, std::ext::V_int const &indices)

  Slice sparse matrix based on rows index.
- DensMat slice_mat (DensMat const &dm, std::ext::V_int const &indices)
- DensMat slice_mat (DensMat const &dm, std::ext::V_bool const &indices)

  Slice dense matrix based on rows indexes.
- DensMat slice_mat (DensMat const &dm, std::ext::Var_bool_int const &indices)

  Slice dense matrix based on variant row vector(either int or bool)
- SpaMat slice_mat (SpaMat const &spm, std::ext::V_int rows, std::ext::V_int cols, bool check_size)

  Slice sparse matrix based on cols and rows index.
- SpaMat slice_mat (SpaMat const &spm, std::ext::V_bool const &ind1, std::ext::V_bool const &ind2, bool check_size)

  Slice sparse matrix based on column and row logical vectors.
- SpaMat slice_mat (SpaMat const &spm, std::ext::Var_bool_int const &ind1, std::ext::Var_bool_int const &ind2, bool check_size)

  Slice sparse matrix based on cols and rows variant(either int or bool)
- DensMat slice_mat (DensMat const &dm, std::ext::V_int ind1, std::ext::V_int ind2)

  Slice dense matrix based on cols and rows index.
- DensMat slice_mat (DensMat const &dm, std::ext::V_bool const &ind1, std::ext::V_bool const &ind2)

  Slice dense matrix based on column and row logical vectors.
- DensMat slice_mat (DensMat const &dm, std::ext::Var_bool_int const &ind1, std::ext::Var_bool_int const &ind2)

  Slice matrix based on columns and rows variant vectors(either int or bool)
- DensMat slice_mat_cols (DensMat const &dm, std::ext::V_int const &ind2)

  Slice dense matrix based on column index vector.
- bool check_convergence (const DensVec &alpha, const DensVec &alpha0, const DensVec &tau, const DensVec &tau0, double tol, size_t &i, int maxiter)
- double hadamard_sum_block (SpaMat &Z, const SpaMat &K, int n, int rowOff, int colOff)

8.38.1 Function Documentation

8.38.1.1 apply_wb()

```
DensMat apply_wb (
            Eigen::Ref< const Eigen::MatrixXd > const & mat,
            DensVec const & diag_sigma_i,
            SpaMat const & diag_sigma_i_ZPchol )
```

8.38.1.2 calc_variance()

```
double calc_variance (
            DensVec const & dv )
```

8.38.1.3 center_dataframe()

```
void center_dataframe (
            DataFrame & df,
            const std::vector< std::string > & headers,
            bool center,
            bool scale )
```

8.38.1.4 check_convergence()

```
bool check_convergence (
            const DensVec & alpha,
            const DensVec & alpha0,
            const DensVec & tau,
            const DensVec & tau0,
            double tol,
            size_t & i,
            int maxiter )
```

8.38.1.5 conv_dm2stdV()

```
std::ext::V_double conv_dm2stdV (
            DensMat const & dm )
```

8.38.1.6 conv_dv2stdVd()

```
std::ext::V_double conv_dv2stdVd (
            DensVec const & dv )
```

8.38.1.7 conv_stdV2dv()

```
DensVec conv_stdV2dv (
            std::ext::V_double v )
```

8.38.1.8 conv_stdVs2dV()

```
DensVec conv_stdVs2dV (
            std::ext::V_string const & stdv )
```

8.38.1.9 conv_stdvs2stdvi()

```
std::ext::V_int conv_stdvs2stdvi (
            std::ext::V_string const & strings )
```

8.38.1.10 conv_vec_vecXd()

```
DensVec conv_vec_vecXd (
            std::ext::V_double v_std )
```

8.38.1.11 create_covdata()

```
DensMat create_covdata (
            const DataFrame & df )
```

8.38.1.12 createHeaderMap()

```
std::ext::map_str_int createHeaderMap (
            std::ext::V_string const & headers )
```

8.38.1.13 hadamard_sum_block()

```
double hadamard_sum_block (
            SpaMat & Z,
            const SpaMat & K,
            int n,
            int rowOff,
            int colOff )
```

8.38.1.14 intersect()

```
std::ext::V_int intersect (
            std::ext::V_int const & vec1,
            std::ext::V_int const & vec2 )
```

8.38.1.15 linkinv()

```
DensVec linkinv (
            DensVec const & eta,
            std::string const & family_t,
            const std::string & link )
```

8.38.1.16 logic_update_fixed_condtion()

```
std::ext::V_int logic_update_fixed_condtion (
            DensVec dv,
            double tol )
```

8.38.1.17 sign()

```
template<typename T >
int sign (
            T t )
```

8.38.1.18 slice_mat() [1/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_bool const & ind1,
            std::ext::V_bool const & ind2 )
```

Slice dense matrix based on column and row logical vectors.

Parameters

Dense	matrix
std::vector<bool>	
std::vector<bool>	

Returns

Dense matrix

8.38.1.19 slice_mat() [2/10]

```
DensMat slice_mat (
            const DensMat & dm,
            const std::ext::V_bool & indices )
```

Slice dense matrix based on rows indexes.

Parameters

Dense	matrix
std::vector<bool>	

Returns

Dense matrix

8.38.1.20 slice_mat() [3/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_int const & indices )
```

8.38.1.21 slice_mat() [4/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::V_int ind1,
            std::ext::V_int ind2 )
```

Slice dense matrix based on cols and rows index.

Parameters

Dense	matrix
std::vector<int>	
std::vector<int>	

Returns

Dens matrix

8.38.1.22 slice_mat() [5/10]

```
DensMat slice_mat (
            DensMat const & dm,
            std::ext::Var_bool_int const & ind1,
            std::ext::Var_bool_int const & ind2 )
```

Slice matrix based on columns and rows variant vectors(either int or bool)

Parameters

Dense	matrix
std::variant<int,bool>	
std::variant<int,bool>	

Returns

Dense matrix

8.38.1.23 slice_mat() [6/10]

```
DensMat slice_mat (
            const DensMat & dm,
            const std::ext::Var_bool_int & indices )
```

Slice dense matrix based on variant row vector(either int or bool)

Parameters

Dense	matrix
std::variant<int,bool>	

Returns

Dense matrix

8.38.1.24 slice_mat() [7/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_bool const & ind1,
            std::ext::V_bool const & ind2,
            bool check_size = true )
```

Slice sparse matrix based on column and row logical vectors.

Parameters

Sparse	matrix
std::vector<bool>	
std::vector<bool>	

Returns

sparse matrix

8.38.1.25 slice_mat() [8/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_int const & indices )
```

Slice sparse matrix based on rows index.

Parameters

spm	
indices	

Returns

SpaMat

8.38.1.26 slice_mat() [9/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::V_int ind1,
            std::ext::V_int ind2,
            bool check_size = true )
```

Slice sparse matrix based on cols and rows index.

Parameters

Sparse	matrix
std::vector<int>	
std::vector<int>	

Returns

sparse matrix

8.38.1.27 slice_mat() [10/10]

```
SpaMat slice_mat (
            SpaMat const & spm,
            std::ext::Var_bool_int const & ind1,
            std::ext::Var_bool_int const & ind2,
            bool check_size = true )
```

Slice sparse matrix based on cols and rows variant(either int or bool)

Parameters

Sparse	matrix
std::variant<int,bool>	
std::variant<int,bool>	

Returns

sparse matrix

8.38.1.28 slice_mat_cols()

```
DensMat slice_mat_cols (
            DensMat const & dm,
            std::ext::V_int const & ind2 )
```

Slice dense matrix based on column index vector.

Parameters

Dense	matrix
std::vector<int>	

Returns

Dense matrix

8.38.1.29 slice_vec() [1/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::V_bool const & indices )
```

Slice dense vector based on logical vector.

Parameters

Dense	vector
std::vector<bool>	

Returns

Dense vector

8.38.1.30 slice_vec() [2/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::V_int const & indices )
```

Slice dense vector based on indexes.

Parameters

Dense	vector
std::vector<int>	

Returns

Dense vector

8.38.1.31 slice_vec() [3/3]

```
DensVec slice_vec (
            DensVec const & dv,
            std::ext::Var_bool_int const & indices )
```

Slice dense vector based on variant vector(either int or bool)

Parameters

Dense	vector
std::variant<int,bool>	

Returns

Dense vector

8.39 src/Kinship.cpp File Reference

```
#include "../include/Kinship.h"
```

Namespaces

- namespace details

Functions

- void details::add_dquot (std::ext::V_string &v_strs)

8.40 src/Logger.cpp File Reference

```
#include "Logger.h"
```

8.41 src/MAGEE.cpp File Reference

```
#include "MAGEE.h"
```

Functions

- void glmm_gei_bgen13 (Magee_Arma const &null_obj, string const &bgenfile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_list, uint begin, uint end, long long unsigned int byte, uint Nbgen, uint compression, bool meta_output=false)
- void glmm_gei_pgen13 (Magee_Arma const &null_obj, string const &pgenfile, std::string pvarFile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_list, uint begin, uint end, std::ext::V_lluint pgenPos, bool filterVariants, int pvarLength, int pvarLast, std::ext::V_int pvarIndex, bool meta_output=false)
- void glmm_gei_bed13 (Magee_Arma const &null_obj, string const &bedfile, std::string bimFile, string const &outfile, double minmaf, double missrate, size_t npb, int ei, int qi, std::ext::Map_str_Vint const &strata_↩ list, uint begin, uint end, std::ext::V_lluint bedPos, bool filterVariants, char bimDelim, int bimLast, uint32_t n_samples, bool meta_output=false)
- void convert_eigen_to_arma (SpaMat const &eigenMat, arma::sp_mat &armaMat)
- std::ext::V_int match_indices (std::ext::V_string const &original, std::ext::V_opt_string const &filtered)

  Matches indices from the original vector to the filtered vector.
- std::ext::V_string match_id_include (std::ext::V_string const &id_include, std::ext::V_string const &sample_id)

  Matches included IDs with sample IDs.
- bool any_isna (std::ext::V_int vec)

  Checks if any element in the vector is NA (represented as -1).
- std::ext::V_int remove_minus_one (std::ext::V_int vec)

  Removes elements equal to -1 from the vector.
- std::ext::V_string create_strata (std::ext::VV_string const &Ecat)

  Creates strata by concatenating elements in each row of the binary columns of matrix E.
- std::ext::V_bool in_op (const std::ext::V_string &vec1, const std::ext::V_string &vec2)

  Serach vector1 in a unique vector2.
- std::ext::V_int in_op_indices (std::ext::V_bool const &vec)
- std::ext::V_string filtered_ids (std::ext::V_string const &vec1, std::ext::V_bool const &vec2)

  Filters a vector of strings based on a boolean vector.
- void check_not_empty (std::ext::V_string const &sample_id)

  Checks if the input vector of strings is not empty.
- template<typename T >
  std::ext::V_opt_string slice (std::vector< T > const &vec, std::ext::V_int const &indices)

  Extracts elements from a vector based on specified indices.
- bool any_duplicated (std::ext::V_string const &vec)

  Checks if there are any duplicated elements in the vector.
- std::ext::V_bool list_duplicates_bool (std::ext::V_string const &vec)

  Identifies duplicates in a vector of strings.
- DensMat filter_unique_rows (DensMat const &mat, std::ext::V_string &id_include)

  Filters out duplicate rows from a matrix.
- std::ext::V_bool apply_on_columns (DensMat const &mat, const std::function< bool(VectorXd const &, int)> &func, int threshold)
- bool unique_less_equal (DensVec const &vec, int threshold)

  Checks if the number of unique elements in a vector is less than or equal to a threshold.
- std::ext::Map_str_Vint generate_strata_list (std::ext::V_string const &vec)

  Generates a strata list mapping observations to their corresponding strata.
- void scale (DensMat &mat, bool center, bool scale)

  Scales a matrix by centering and/or scaling its columns.

- DensMat cbind (DensMat const &A, DensMat const &B)

  Combines two matrices column-wise.
- void spa_mat_ones (arma::sp_mat &arma_mat)
- SpaMat apply_wb_J (SpaMat const &J, DensVec const &diag_sigma_i, SpaMat const &diag_sigma_i_↩ ZPchol)

8.41.1 Function Documentation

8.41.1.1 any_duplicated()

```
bool any_duplicated (
            std::ext::V_string const & vec )
```

Checks if there are any duplicated elements in the vector.

Parameters

vec	The vector to check.

Returns

True if there are duplicates, otherwise false.

8.41.1.2 any_isna()

```
bool any_isna (
            std::ext::V_int vec )
```

Checks if any element in the vector is NA (represented as -1).

Parameters

vec	The vector to check.

Returns

True if any element is -1, otherwise false.

8.41.1.3 apply_on_columns()

```
std::ext::V_bool apply_on_columns (
            DensMat const & mat,
            const std::function< bool(VectorXd const &, int)> & func,
            int threshold )
```

8.41.1.4 apply_wb_J()

```
SpaMat apply_wb_J (
            SpaMat const & J,
            DensVec const & diag_sigma_i,
            SpaMat const & diag_sigma_i_ZPchol )
```

8.41.1.5 cbind()

```
DensMat cbind (
            DensMat const & A,
            DensMat const & B )
```

Combines two matrices column-wise.

This function concatenates two matrices `A` and `B` horizontally to create a new matrix.

Parameters

A	The first matrix.
B	The second matrix.

Returns

The combined matrix.

8.41.1.6 check_not_empty()

```
void check_not_empty (
            std::ext::V_string const & sample_id )
```

Checks if the input vector of strings is not empty.

Parameters

sample_id	The vector of sample IDs to check.

Exceptions

std::runtime_error	if the vector is empty.

8.41.1.7 convert_eigen_to_arma()

```
void convert_eigen_to_arma (
            SpaMat const & eigenMat,
            arma::sp_mat & armaMat )
```

8.41.1.8 create_strata()

```
std::ext::V_string create_strata (
            std::ext::VV_string const & Ecat )
```

Creates strata by concatenating elements in each row of the binary columns of matrix E.

Parameters

Ecat	A vector of vector of strings representing categorical data.

Returns

A vector of concatenated strings representing strata.

8.41.1.9 filter_unique_rows()

```
DensMat filter_unique_rows (
            DensMat const & mat,
            std::ext::V_string & id_include )
```

Filters out duplicate rows from a matrix.
This function removes duplicate rows from the input matrix and returns a matrix with only unique rows, as well as updating the `id_include` vector accordingly.

Parameters

mat	The input matrix.
id_include	The vector of IDs to filter.

Returns

A matrix with unique rows.

8.41.1.10 filtered_ids()

```
std::ext::V_string filtered_ids (
            std::ext::V_string const & vec1,
            std::ext::V_bool const & vec2 )
```

Filters a vector of strings based on a boolean vector.

Parameters

vec1	The vector to filter.
vec2	The boolean vector indicating which elements to keep.

Returns

A filtered vector of strings.

8.41.1.11 generate_strata_list()

```
std::ext::Map_str_Vint generate_strata_list (
            std::ext::V_string const & vec )
```

Generates a strata list mapping observations to their corresponding strata.

Parameters

vec	The vector of strata identifiers.

Returns

A map from strata identifiers to lists of observation indices.

8.41.1.12 glmm_gei_bed13()

```
void glmm_gei_bed13 (
            Magee_Arma const & null_obj,
            string const & bedfile,
            std::string bimFile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            std::ext::V_lluint bedPos,
```

```
            bool filterVariants,
            char bimDelim,
            int bimLast,
            uint32_t n_samples,
            bool meta_output = false )
```

8.41.1.13 glmm_gei_bgen13()

```
void glmm_gei_bgen13 (
            Magee_Arma const & null_obj,
            string const & bgenfile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            long long unsigned int byte,
            uint Nbgen,
            uint compression,
            bool meta_output = false )  [extern]
```

8.41.1.14 glmm_gei_pgen13()

```
void glmm_gei_pgen13 (
            Magee_Arma const & null_obj,
            string const & pgenfile,
            std::string pvarFile,
            string const & outfile,
            double minmaf,
            double missrate,
            size_t npb,
            int ei,
            int qi,
            std::ext::Map_str_Vint const & strata_list,
            uint begin,
            uint end,
            std::ext::V_lluint pgenPos,
            bool filterVariants,
            int pvarLength,
            int pvarLast,
            std::ext::V_int pvarIndex,
            bool meta_output = false )  [extern]
```

8.41.1.15 in_op()

```
std::ext::V_bool in_op (
            const std::ext::V_string & vec1,
            const std::ext::V_string & vec2 )
```

Serach vector1 in a unique vector2.
Checks if elements of vec1 are in vec2.

Parameters

vec1	
vec2	

Returns

std::ext::V_bool

8.41.1.16 in_op_indices()

```
std::ext::V_int in_op_indices (
            std::ext::V_bool const & vec )
```

8.41.1.17 list_duplicates_bool()

```
std::ext::V_bool list_duplicates_bool (
            std::ext::V_string const & vec )
```

Identifies duplicates in a vector of strings.

Parameters

vec	The vector to check for duplicates.

Returns

A boolean vector indicating the presence of duplicates.

8.41.1.18 match_id_include()

```
std::ext::V_string match_id_include (
            std::ext::V_string const & id_include,
            std::ext::V_string const & sample_id )
```

Matches included IDs with sample IDs.
This function returns the subset of IDs in `id_include` if they are present in `sample_id`.

Parameters

id_include	A vector of IDs to include.
sample_id	A vector of sample IDs.

Returns

A vector of matched IDs.

8.41.1.19 match_indices()

```
std::ext::V_int match_indices (
            std::ext::V_string const & original,
            std::ext::V_opt_string const & filtered )
```

Matches indices from the original vector to the filtered vector.
This function returns a vector of indices that match the elements in the original vector with those in the filtered vector. If an element in the original vector does not have a match, the corresponding index is set to -1.

Parameters

original	The original vector of strings.
filtered	The filtered vector of optional strings.

Returns

A vector of matched indices or -1 for non-matches.

8.41.1.20 remove_minus_one()

```
std::ext::V_int remove_minus_one (
            std::ext::V_int vec )
```

Removes elements equal to -1 from the vector.

Parameters

vec	The vector to filter.

Returns

A vector with all elements that are not equal to -1.

8.41.1.21 scale()

```
void scale (
            DensMat & mat,
            bool center = true,
            bool scale = true )
```

Scales a matrix by centering and/or scaling its columns.

Parameters

mat	The matrix to scale.
center	Whether to center the matrix by subtracting the mean of each column.
scale	Whether to scale the matrix by dividing by the standard deviation of each column.

8.41.1.22 slice()

```
template<typename T >
std::ext::V_opt_string slice (
            std::vector< T > const & vec,
            std::ext::V_int const & indices )
```

Extracts elements from a vector based on specified indices.

Template Parameters

T	The type of elements in the vector.

Parameters

vec	The original vector.
indices	The indices of the elements to extract.

Returns

A vector of optional strings, with nullopt representing NA.

8.41.1.23 spa_mat_ones()

```
void spa_mat_ones (
            arma::sp_mat & arma_mat )
```

8.41.1.24 unique_less_equal()

```
bool unique_less_equal (
            DensVec const & vec,
            int threshold )
```

Checks if the number of unique elements in a vector is less than or equal to a threshold.

Parameters

vec	The vector to check.
threshold	The threshold value.

Returns

True if the number of unique elements is less than or equal to the threshold, otherwise false.

8.42 src/MatrixUtils.cpp File Reference

```
#include <cstdio>
#include <stdlib.h>
```

Functions

- void dgemm_ (char *TRANSA, char *TRANSB, const int *M, const int *N, const int *K, double *alpha, double *A, const int *LDA, double *B, const int *LDB, double *beta, double *C, const int *LDC)
- void dgemv_ (char *TRANS, const int *M, const int *N, double *alpha, double *A, const int *LDA, double *X, const int *INCX, double *beta, double *C, const int *INCY)
- void dgetrf_ (const int *M, const int *N, double *A, const int *LDA, const int *IPIV, const int *INFO)
- void dgetri_ (const int *N, double *A, const int *LDA, const int *IPIV, double *WORK, const int *LWORK, const int *INFO)
- void daxpy_ (const int *N, double *DA, double *DX, const int *INCX, double *DY, const int *INCY)
- void dlacpy_ (char *uplo, const int *m, const int *n, double *a, const int *lda, double *b, const int *ldb)
- void initvec (double *v, int N)
- void matTmatprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matNmatNprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matmatprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matmatTprod (double *A, double *v, double *u, int Nrow, int Ncol, int NcolB)
- void matTvecprod (double *A, double *v, double *u, int Nrow, int Ncol)
- void matvecprod (double *A, double *v, double *u, int Nrow, int Ncol)
- void vecmatprod (double *v, double *A, double *u, int N)
- void matInv (double *A, int N)
- void matAdd (double *A, double *B, int N, double alpha)
- void subMatrix (double *A, double *u, int M, int N, int NrowA, int NrowB, int start)
- void matvecSprod (double *A, double *v, double *u, int Nrow, int Ncol, int start)

8.42.1 Function Documentation

8.42.1.1 daxpy_()

```
void daxpy_ (
             const int * N,
             double * DA,
             double * DX,
             const int * INCX,
             double * DY,
             const int * INCY )
```

8.42.1.2 dgemm_()

```
void dgemm_ (
             char * TRANSA,
             char * TRANSB,
             const int * M,
             const int * N,
             const int * K,
             double * alpha,
             double * A,
             const int * LDA,
             double * B,
             const int * LDB,
             double * beta,
             double * C,
             const int * LDC )
```

8.42.1.3 dgemv_()

```
void dgemv_ (
             char * TRANS,
             const int * M,
             const int * N,
             double * alpha,
             double * A,
             const int * LDA,
             double * X,
             const int * INCX,
             double * beta,
             double * C,
             const int * INCY )
```

8.42.1.4 dgetrf_()

```
void dgetrf_ (
             const int * M,
             const int * N,
             double * A,
             const int * LDA,
             const int * IPIV,
             const int * INFO )
```

8.42.1.5 dgetri_()

```
void dgetri_ (
             const int * N,
             double * A,
             const int * LDA,
```

```
            const int * IPIV,
            double * WORK,
            const int * LWORK,
            const int * INFO )
```

8.42.1.6 dlacpy_()

```
void dlacpy_ (
            char * uplo,
            const int * m,
            const int * n,
            double * a,
            const int * lda,
            double * b,
            const int * ldb )
```

8.42.1.7 initvec()

```
void initvec (
            double * v,
            int N )
```

8.42.1.8 matAdd()

```
void matAdd (
            double * A,
            double * B,
            int N,
            double alpha )
```

8.42.1.9 matInv()

```
void matInv (
            double * A,
            int N )
```

8.42.1.10 matmatprod()

```
void matmatprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.42.1.11 matmatTprod()

```
void matmatTprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.42.1.12 matNmatNprod()

```
void matNmatNprod (
            double * A,
```

```
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.42.1.13 matTmatprod()

```
void matTmatprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int NcolB )
```

8.42.1.14 matTvecprod()

```
void matTvecprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol )
```

8.42.1.15 matvecprod()

```
void matvecprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol )
```

8.42.1.16 matvecSprod()

```
void matvecSprod (
            double * A,
            double * v,
            double * u,
            int Nrow,
            int Ncol,
            int start )
```

8.42.1.17 subMatrix()

```
void subMatrix (
            double * A,
            double * u,
            int M,
            int N,
            int NrowA,
            int NrowB,
            int start )
```

8.42.1.18 vecmatprod()

```
void vecmatprod (
            double * v,
```

```
            double * A,
            double * u,
            int N )
```

8.43 src/Pheno.cpp File Reference

```
#include "../include/Pheno.h"
#include <iostream>
#include <iterator>
#include <unordered_set>
```

8.44 src/ReadBed.cpp File Reference

```
#include "declars.h"
#include "ReadBed.h"
```

Functions

- void gemBED (int thread_num, double sigma2, double *resid, double *XinvXTX, vector< double > miu, BinE binE, Bed bed, CommandLine cmd)

8.44.1 Function Documentation

8.44.1.1 gemBED()

```
void gemBED (
            int thread_num,
            double sigma2,
            double * resid,
            double * XinvXTX,
            vector< double > miu,
            BinE binE,
            Bed bed,
            CommandLine cmd )
```

8.45 src/ReadBGEN.cpp File Reference

```
#include "declars.h"
#include "ReadBGEN.h"
#include "ReadParameters.h"
#include "../thirdparty/zstd-1.5.5/lib/zstd.h"
#include "../thirdparty/libdeflate-1.18/libdeflate.h"
```

Typedefs

- typedef std::numeric_limits< double > dbl

Functions

- void Bgen13GetTwoVals (const unsigned char *prob_start, uint32_t bit_precision, uintptr_t offset, uintptr_t *first_val_ptr, uintptr_t *second_val_ptr)
- void gemBGEN (int thread_num, double sigma2, double *resid, double *XinvXTX, vector< double > miu, BinE binE, Bgen bgen, CommandLine cmd)

8.45.1 Typedef Documentation

8.45.1.1 dbl

```
typedef std::numeric_limits< double > dbl
```

8.45.2 Function Documentation

8.45.2.1 Bgen13GetTwoVals()

```
void Bgen13GetTwoVals (
            const unsigned char * prob_start,
            uint32_t bit_precision,
            uintptr_t offset,
            uintptr_t * first_val_ptr,
            uintptr_t * second_val_ptr )
```

8.45.2.2 gemBGEN()

```
void gemBGEN (
            int thread_num,
            double sigma2,
            double * resid,
            double * XinvXTX,
            vector< double > miu,
            BinE binE,
            Bgen bgen,
            CommandLine cmd )
```

8.46 src/ReadFiles.cpp File Reference

```
#include <algorithm>
#include <cstddef>
#include <fstream>
#include <iostream>
#include <iterator>
#include <sstream>
#include "ReadFiles.h"
#include <unordered_set>
```

8.47 src/ReadParameters.cpp File Reference

```
#include "declars.h"
```

Macros

- #define VERSION "2.2"

Functions

- void print_help ()

8.47.1 Macro Definition Documentation

8.47.1.1 VERSION

```
#define VERSION "2.2"
```

8.47.2 Function Documentation

8.47.2.1 print_help()

```
void print_help ( )
```

8.48 src/ReadPGEN.cpp File Reference

```
#include "declars.h"
#include "ReadPGEN.h"
#include "../thirdparty/plink-2.0/plink2_bits.h"
#include "../thirdparty/plink-2.0/plink2_base.h"
#include "../thirdparty/plink-2.0/pgenlib_misc.h"
#include "../thirdparty/plink-2.0/pgenlib_read.h"
#include "../thirdparty/plink-2.0/pgenlib_ffi_support.h"
```

Functions

- void gemPGEN (int thread_num, double sigma2, double ∗resid, double ∗XinvXTX, vector< double > miu, BinE binE, Pgen pgen, CommandLine cmd)

8.48.1 Function Documentation

8.48.1.1 gemPGEN()

```
void gemPGEN (
            int thread_num,
            double sigma2,
            double * resid,
            double * XinvXTX,
            vector< double > miu,
            BinE binE,
            Pgen pgen,
            CommandLine cmd )
```

8.49 src/SparseInverse.cpp File Reference

```
#include "../include/SparseInverse.h"
#include <fstream>
#include <algorithm>
#include <iterator>
#include <unordered_set>
#include <chrono>
```

Functions

- uint64_t combine_indices (long long a, long long b)
- cholmod_sparse ∗ convertEigenToSuiteSparse (SpaMat const &sm, cholmod_common ∗cm)
- cs_di ∗ cholmodSparseToCsparse (cholmod_sparse ∗L)
- cs_di ∗ createSparseIdentity (int m)

8.49.1 Function Documentation

8.49.1.1 cholmodSparseToCsparse()

```
cs_di * cholmodSparseToCsparse (
            cholmod_sparse * L )
```

8.49.1.2 combine_indices()

```
uint64_t combine_indices (
            long long a,
            long long b )
```

8.49.1.3 convertEigenToSuiteSparse()

```
cholmod_sparse * convertEigenToSuiteSparse (
            SpaMat const & sm,
            cholmod_common * cm )
```

8.49.1.4 createSparseIdentity()

```
cs_di * createSparseIdentity (
            int m )
```

8.50 src/TimeUtils.cpp File Reference

```
#include "declars.h"
#include "TimeUtils.h"
```

Functions

- void printExecutionTime (std::chrono::steady_clock::time_point start_time, std::chrono::steady_clock::time_point end_time)
- void printExecutionTime1 (std::chrono::steady_clock::time_point start_time, std::chrono::steady_clock::time_point end_time)

8.50.1 Function Documentation

8.50.1.1 printExecutionTime()

```
void printExecutionTime (
            std::chrono::steady_clock::time_point start_time,
            std::chrono::steady_clock::time_point end_time )
```

8.50.1.2 printExecutionTime1()

```
void printExecutionTime1 (
            std::chrono::steady_clock::time_point start_time,
            std::chrono::steady_clock::time_point end_time )
```

8.51 testinv.cpp File Reference

```
#include "build-update-nullmodel/_deps/eigen-src/Eigen/Dense"
#include <iostream>
```

Functions

- int main ()

8.51.1 Function Documentation

8.51.1.1 main()

```
int main ( )
```

Index