MuGen

Generated by Doxygen 1.8.20

Chapter 1

Introduction

MuGen is a C++ library that implements a comprehensive approach to Bayesian inference of multi-trait models for quantitative genetics. It enables genome-wide association studies and genome-enabled prediction, allows for complicated and unbalanced experimental designs, outlier observations, and missing data. However, while motivation for the construction of this library was to model genetic data, other types of data that could benefit from hierarchical treatment (see **[gelman04] [gelman07]**) can also be analyzed.

The basic idea in hierarchical modeling is that so-called location parameters (i.e., group means and regression coefficients) can be arranged so that levels are nested within each other. This arrangement is particularly suited for Bayesian inference because model hierarchy fits a hierarchy of prior distributions **[gelman04] [gelman07] [greenberg10]** . In addition to the location parameters, scale parameters (in our case, covariance matrices) are also modeled. Furthermore, some location parameters may not be hierarchically nested within other location parameters. Most parameters of interest can be fit into one of these categories, and given an implementation that can form a part of a model. These individual parameters are represented by objects of various classes in MuGen. Having provided implementations for Gibbs sampling of a variety of types of parameters, we enable the user to quickly build a flexible model that accomodates most applications without having to worry about computational complexity.

The structure of the hierarchy is represented by index classes (RanIndex and its derivatives) that store information about which location parameter is related to which upper or lower member of the model hierarchy. These index parameters can themselves stochastically change to accomodate mixture models where membership of a group is uncertain. However, the latter feature is still experimental.

Describing statistical hierarchical models unfortunately creates a collision of terms, since our classes are also arranged in a hierarchy. The latter is the C++ class derivation hierarchy and is not related to the statistical term. In the documentation we use the term "hierarchy" we to describe the statistical model relationships among variables unless otherwise specified.

Internal implementation of MuGen is multithreaded (using OpenMP, `http://openmp.org/wp/`) and uses GNU Scientific Library and BLAS for computation. These computational details are hidden behind the interface which is designed for users familiar with basic C++ programming. A publication describing and validating the library is in preparation.

1.1 File formats

Unless explicitly specified otherwise, all files are saved and read as binary files created by GSL functions of the class `gsl_matrix_fwrite`. To read and write these files from R, we created a set of functions found in the file GSLRIO.cpp. To compile for R, use

```
R CMD SHLIB GSLRIO.cpp -lgsl -O3 -DHAVE_INLINE -DGSL_RANGE_CHECK_OFF
```

and put the resulting GSLRIO.o where R can find it. It can then be loaded using `dyn.load()`.

1.2 Dependencies

Aside from the C++ standard library, the implementation depends only on the GNU Scienetific Library (GSL) (http://www.gnu.org/software/gsl/) and Basic Linear Algebra Subprograms (BLAS, http://www.netlib.org/blas/). We are currently working with GSL version 1.16, but any version that supports linear algebra routines like Cholesky decomposition will work. Choosing the right BLAS implementation is crucial for performance. Generic BLAS is not recommended. We have successfully compiled MuGen with Apple's Accelerate framework, Intel's MKL (https://software.intel.com/en-us/intel-mkl) and AMD's ACML (http://developer.amd.com/tools-and-sdks/cpu-development/amd-core-math-library-acml/), all with good results. MKL seems to perform similarly to ACML on AMD Opteron processors we use to perform most computationally intensive tasks.

1.3 Compilation instructions

We use the GNU compiler collection (both version 4.7 and 4.9) for compilation. We did not use any C++11 features, so older compilers should work, too. We also successfully compiled the library with Intel icc and Apple's LLVM version 6.0. However, the latter does not support OpenMP, so the resulting code is not multithreaded.

To compile, make sure GSL is where the compiler can see it and is pointing to your preferred implemetation of BLAS. Then, simply run

```
g++ -c MuGen.cpp -lz -lgslcblas -lgsl -lpthread -lm -fopenmp \\
    -DHAVE_INLINE -DGSL_RANGE_CHECK_OFF -O3 -march=native

ar crv libMuGen.a MuGen.o
```

and move libMuGen.a to, say, /usr/local/lib/ and MuGen.h to, say, /usr/local/include/.

MuGen has been successfully compiled under RedHat, Amazon Linux and Mac OS X Mavericks and Yosemite. On Mac OS X, at least when using gcc-4.7 and above, using -march=native does not always work. In that case, try including -m64 -msse4.2 -mfancy-math-387.
Windows compilation has not been attempted.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

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

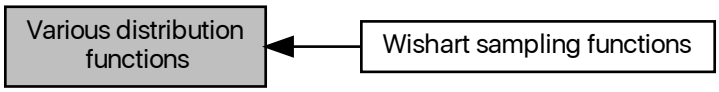
Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 Various distribution functions

Collaboration diagram for Various distribution functions:



Modules

- Wishart sampling functions

Functions

- void MVgauss (const gsl_vector *, const gsl_matrix *, const gsl_rng *, gsl_vector *)
  Multivariate Gaussian sampling function.
- size_t rtgeom (const double &, const size_t &, const gsl_rng *)
  Truncated geometric distribution.

6.1.1 Detailed Description

Sampling functions for distributions not already in GSL.

Warning

These are internal functions that are invoked from within classes. Because they need to be as efficient as possible, no range-checking is performed (especially if GSL range checking is turned off at compilation). They rely on the user to keep track of dimensions.

6.1.2 Function Documentation

6.1.2.1 MVgauss()

```
void MVgauss (
            const gsl_vector * ,
            const gsl_matrix * ,
            const gsl_rng * ,
            gsl_vector *  )
```

Multivariate Gaussian sampling function.

This function gives a sample from the MV Gaussian distribution with a given mean and covariance matrix.

Parameters

in	*gsl_vector*∗	pointer to a vector of means
in	*gsl_matrix*∗	pointer to a matrix with the Cholesky-decomposed covariance matrix
in	*gsl_rng*∗	pointer to a pseudo-random number generator (PNG)
out	*gsl_vector*∗	pointer to a destination vector for the sample

6.1.2.2 rtgeom()

```
size_t rtgeom (
            const double & ,
            const size_t & ,
            const gsl_rng *  )
```

Truncated geometric distribution.

Sampling from the truncated geometric distribution, returning a size_t type index value.

Parameters

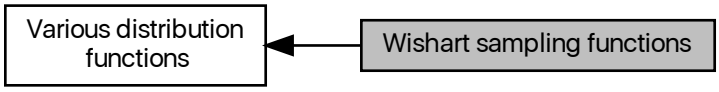
in	*double&*	probability of success
in	*size_t&*	maximum value
in	*gsl_rng*∗	pointer to a PNG

Returns

a value of the type size_t that is a sample from the distribution.

6.2 Wishart sampling functions

Collaboration diagram for Wishart sampling functions:



Functions

- void **Wishart** (const gsl_matrix ∗, const int &, const gsl_rng ∗, gsl_matrix ∗)
- void **Wishart** (const gsl_matrix ∗, const size_t &, const gsl_rng ∗, gsl_matrix ∗)

6.2.1 Detailed Description

These functions give samples from the Wishart distribution, the only difference between them is the type of the degrees of freedom paramter.

Parameters

in	*gsl_matrix*∗	pointer to a matrix with the Cholesky-transformed covariance matrix parameter
in	*int/size_t&*	degrees of freedom parameter
in	*gsl_rng*∗	pointer to a PNG
out	*gsl_matrix*∗	pointer to a matrix with the sample

6.3 Auxiliary functions

Collaboration diagram for Auxiliary functions:

Modules

- Centering functions

Functions

- double mhl (const gsl_vector *beta, const gsl_matrix *SigI)

  Mahalanobis distance.
- void printMat (const gsl_matrix *m)

  Print matrix to screen.
- unsigned long long rdtsc ()

  Accessing the processor RTDSC instruction.

6.3.1 Detailed Description

Various functions that are needed internally.

6.3.2 Function Documentation

6.3.2.1 mhl()

```
double mhl (
            const gsl_vector * beta,
            const gsl_matrix * SigI )
```

Mahalanobis distance.

Calculates the Mahalanobis distance of a vector from zero.

Parameters

in	*gsl_vector∗*	the vector
in	*gsl_matrix∗*	the corresponding inverse-covariance matrix

Returns

a scalar value of type double.

6.3.2.2 printMat()

```
void printMat (
            const gsl_matrix * m )
```

Print matrix to screen.

Printing a GSL matrix to screen, with each row on a line, values seprated by spaces.

Parameters

in	*gsl_matrix∗*	matrix to be displayed
in	*double&*	label for missing data

6.3.2.3 rdtsc()

```
unsigned long long rdtsc ( )
```

Accessing the processor RTDSC instruction.

Returns

a value of type unsigned long long.

This function outputs the processor RTDSC instruction for use in seeding random number generators using assembly code.

Warning

this function is unlikely to work under Windows.

6.4 Centering functions

Collaboration diagram for Centering functions:

Functions

- void colCenter (gsl_matrix ∗inplace)
  Matrix centering in-place.
- void colCenter (const gsl_matrix ∗source, gsl_matrix ∗res)
  Matrix centering with copy.
- void colCenter (gsl_matrix ∗inplace, const double &absLab)
  Matrix centering in-place with missing values.
- void colCenter (const gsl_matrix ∗source, gsl_matrix ∗res, const double &absLab)
  Matrix centering with copy and missing values.
- void vecCenter (gsl_vector ∗inplace)
  Vector centering in-place.
- void vecCenter (const gsl_vector ∗source, gsl_vector ∗res)
  Vector centering with copy.
- void meanImpute (gsl_matrix ∗inplace, const double &absLab)
  Mean imputation without centering.

6.4.1 Detailed Description

Functions that center matrix columns and vectors.

6.4.2 Function Documentation

6.4.2.1 colCenter() [1/4]

```
void colCenter (
            const gsl_matrix * source,
            gsl_matrix * res )
```

Matrix centering with copy.

Parameters

in	*gsl_matrix*∗	the matrix to be centered, left unmodified
out	*gsl_matrix*∗	the modified matrix

6.4.2.2 colCenter() [2/4]

```
void colCenter (
            const gsl_matrix * source,
            gsl_matrix * res,
            const double & absLab )
```

Matrix centering with copy and missing values.

In addition to centering does mean-imputation of missing values.

Parameters

in	*gsl_matrix*∗	matrix to be centered, but not changed
out	*gsl_matrix*∗	centered matrix
in	*double&*	label for missing values

6.4.2.3 colCenter() [3/4]

```
void colCenter (
            gsl_matrix * inplace )
```

Matrix centering in-place.

Parameters

in,out	*gsl_matrix*∗	the matrix to be modified in-place

6.4.2.4 colCenter() [4/4]

```
void colCenter (
            gsl_matrix * inplace,
            const double & absLab )
```

Matrix centering in-place with missing values.

Parameters

in,out	*gsl_matrix∗*	matrix to be modified in-place
in	*double&*	label for missing values

6.4.2.5 meanImpute()

```
void meanImpute (
            gsl_matrix * inplace,
            const double & absLab )
```

Mean imputation without centering.

Parameters

in,out	*gsl_matrix∗*	matrix to be modified in-place
in	*double&*	label for missing values

6.4.2.6 vecCenter() [1/2]

```
void vecCenter (
            const gsl_vector * source,
            gsl_vector * res )
```

Vector centering with copy.

Parameters

in	*gsl_vector∗*	vector to be centered, but not changed
out	*gsl_vector∗*	modified vector

6.4.2.7 vecCenter() [2/2]

```
void vecCenter (
            gsl_vector * inplace )
```

Vector centering in-place.

Parameters

in,out	*gsl_vector**	vector to be modified
in	*double&*	label for missing values

6.5 Individual location parameters

Classes

- class MVnorm

  The abstract base class for location parameter rows.
- class MVnormMu

  Individual vector of means.
- class MVnormMuPEX

  Individual vector of means with parameter expansion.
- class MVnormBetaPEX

  Individual regression with parameter expansion.
- class MVnormMuMiss

  Individual vector of means with missing data.
- class MVnormBeta

  Generic regression.
- class MVnormBetaFt

  Regression with multiple predictors.
- class MVnormMuBlk

  Individual vector of means with blocks of traits.
- class MVnormBetaBlk

  Individual vector of regression coefficients with blocks of traits.
- class MVnormBetaFtBlk

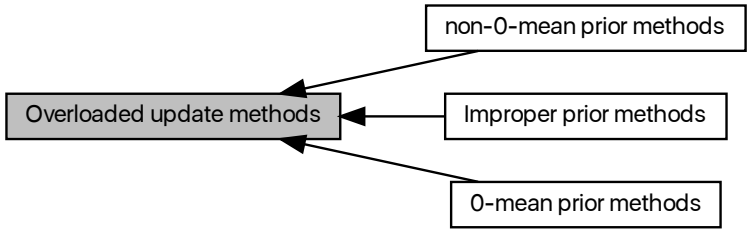
  Individual vector of conditional regression coefficients with blocks of traits.

6.5.1 Detailed Description

A hierarchy of classes that refer to rows of location parameter matrices. These classes are internal to Grp classes and are not directly declared by the user of the library.
They are typically updated with samples from the multivariate normal distribution. Because the methods are used for much of the computation, they are implemented for speed and almost no chacking for errors is done. The assumption is that the encapsualting classes will do all of that correctly. Constructors are used to set initial values.

6.6 Overloaded update methods

Collaboration diagram for Overloaded update methods:



Modules

- Improper prior methods
- 0-mean prior methods
- non-0-mean prior methods

Functions

- void RanIndex::update (const Grp &theta, const Grp &mu, const vector< SigmaI > &SigI, const MixP &p)

  Update mixture model with multiple covariances.
- void RanIndex::update (const Grp &theta, const Grp &mu, const SigmaI &SigI, const MixP &p)

  Update mixture model with a single covariance.
- virtual void RanIndex::update (const Grp &y, const SigmaI &SigIe, BetaGrpBVSR *theta, const SigmaI &SigIp)

  Variable selection update.
- void Apex::update (const Grp &y, const gsl_matrix *xi, const SigmaI &SigIm)

  Unreplicated Gaussian update.
- void Apex::update (const Grp &y, const gsl_matrix *xi, const SigmaI &SigIm, const RanIndex &ind)

  Replicated Gaussian update.
- void Apex::update (const Grp &y, const gsl_matrix *xi, const Qgrp &q, const SigmaI &SigIm, const RanIndex &ind)

  Replicated Student- t update.
- virtual void MuGrpMiss::update (const Grp &mu, const SigmaI &SigIm)

  Standard Gaussian imputation.
- virtual void MuGrpMiss::update (const Grp &mu, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian imputation with a prior.
- virtual void MuGrpEE::update (const Grp &muPr, const SigmaI &SigIm)

  Gaussian prior.
- virtual void MuGrpEE::update (const Grp &muPr, const Qgrp &q, const SigmaI &SigIm)

Student-t prior.

- void BetaGrpSnp::update (const Grp &dat, const SigmaI &SigIm)

  Response update function.
- void BetaGrpSnpMiss::update (const Grp &dat, const SigmaI &SigIm)

  Response update function.
- virtual void SigmaI::update (const Grp &dat)

  Basic Gaussian update.
- virtual void SigmaI::update (const Grp &dat, const Grp &mu)

  Gaussian update with a mean.
- virtual void SigmaI::update (const Grp &dat, const Qgrp &q)

  Basic Student-t update.
- virtual void SigmaI::update (const Grp &dat, const Grp &mu, const Qgrp &q)

  Student-t update with a mean.
- virtual void Qgrp::update (const Grp &dat, const Grp &mu, const SigmaI &SigI)

  Update with a mean.
- virtual void Qgrp::update (const Grp &dat, const SigmaI &SigI)

  Basic update.
- void MixP::update (const RanIndex &Nvec)

  Gibbs update function.

6.6.1 Detailed Description

Update functions generate new stochastic values of a given parameter or set of parameters as we step through the Markov chain iteration. These are mostly Gibbs updates, but occasional Metropolis steps are used in special cases.

6.6.2 Function Documentation

6.6.2.1 update() [1/19]

```
void SigmaI::update (
            const Grp & dat )  [virtual]
```

Basic Gaussian update.

The data are assumed already centered, so the update is based on the simple cross-product of the data.

Parameters

in	*Grp&*	data

Reimplemented in SigmaIblk.

6.6.2.2 update() [2/19]

```
void SigmaI::update (
            const Grp & dat,
            const Grp & mu )  [virtual]
```

Gaussian update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean

Reimplemented in SigmaIblk.

6.6.2.3 update() [3/19]

```
void SigmaI::update (
            const Grp & dat,
            const Grp & mu,
            const Qgrp & q )  [virtual]
```

Student- t update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean
in	*Qgrp&*	scale parameter

Reimplemented in SigmaIpex.

6.6.2.4 update() [4/19]

```
void Qgrp::update (
            const Grp & dat,
            const Grp & mu,
            const SigmaI & SigI )  [virtual]
```

Update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean
in	*Sigma↩ I&*	inverse-covariance

Reimplemented in QgrpPEX.

6.6.2.5 update() [5/19]

```
void SigmaI::update (
            const Grp & dat,
            const Qgrp & q )  [virtual]
```

Basic Student- t update.

The data are assumed already centered, so the update is based on the simple cross-product of the data and the current value of the scale parameter.

Parameters

in	*Grp&*	data
in	*Qgrp&*	scale parameter

Reimplemented in SigmaIpex.

6.6.2.6 update() [6/19]

```
void Qgrp::update (
            const Grp & dat,
            const SigmaI & SigI )  [virtual]
```

Basic update.

The data are assumed already centered, so the update is based on the simple cross-product of the data and the current value of the inverse-covariance.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	inverse-covariance

Reimplemented in QgrpPEX.

6.6.2.7 update() [7/19]

```
void BetaGrpSnp::update (
            const Grp & dat,
            const SigmaI & SigIm )  [virtual]
```

Response update function.

Unlike standard update functions, this one only saves MCMC samples of the response. The covariance provided is ignored.
The actual regression is performed on the point estimates of response values calculated as means of the stored MCMC values, and is done at the end by envoking the dump() function.

Parameters

in	*Grp&*	response variable to be saved
in	*Sigma↩ I&*	inverse-covariance (ignored)

Reimplemented from MuGrp.

6.6.2.8 update() [8/19]

```
void BetaGrpSnpMiss::update (
            const Grp & dat,
            const SigmaI & SigIm )  [virtual]
```

Response update function.

Unlike standard update functions, this one only saves MCMC samples of the response. The covariance provided is ignored. The actual regression is performed on the point estimates of response values calculated as means of the stored MCMC values, and is done at the end by envoking the dump() function.

Parameters

in	*Grp&*	response variable to be saved
in	*Sigma↩ I&*	inverse-covariance (ignored)

Reimplemented from MuGrp.

6.6.2.9 update() [9/19]

```
void MuGrpMiss::update (
            const Grp & mu,
            const SigmaI & SigIm )  [virtual]
```

Standard Gaussian imputation.

If some data are present, performs standard Gaussian marginal imputation (desribed in, e.g., **[chatfield80]**) with the provided Grp object as a mean and the SigmaI object as the inverse-covariance. If no data for a row are present, simply replaces the row values by a Gaussian sample with mean and inverse-covariance provided. Rows with no missing data are ignored.

Parameters

in	*Grp&*	mean
in	*SigmaI&*	inverse-covariance

Reimplemented from MuGrp.

Reimplemented in MuGrpEEmiss.

6.6.2.10 update() [10/19]

```
void MuGrpMiss::update (
            const Grp & mu,
            const SigmaI & SigIm,
            const SigmaI & SigIp )  [virtual]
```

Gaussian imputation with a prior.

If some data are present, performs Gaussian marginal imputation (desribed in, e.g., **[chatfield80]**) with the provided Grp object as a mean and the SigmaI object as the inverse-covariance, but with a 0-mean prior. If no data for a row are present, simply replaces the row values by a Gaussian sample with mean and inverse-covariance provided. Rows with no missing data are ignored.

Warning

Has not been extensively tested

Parameters

in	*Grp&*	mean
in	*SigmaI&*	inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

6.6.2.11 update() [11/19]

```
void MuGrpEE::update (
            const Grp & muPr,
            const Qgrp & q,
            const SigmaI & SigIm )  [virtual]
```

Student- t prior.

The Grp object contains the prior means and the SigmaI object – the prior inverse-covariance for the sampling of data values. The upper index of the object must have the same number of groups as the number of rows in the prior matrix addressed by fMat(). While the sampling is independent, the prior inverse variances for each trait are taken from the diagonal of the inverse-covariance matrix (in the SigmaI object), and are thus influenced by any correlated traits.

Parameters

in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weights
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

6.6.2.12 update() [12/19]

```
void MuGrpEE::update (
            const Grp & muPr,
            const SigmaI & SigIm )  [virtual]
```

Gaussian prior.

The Grp object contains the prior means and the SigmaI object – the prior inverse-covariance for the sampling of data values. The upper index of the object must have the same number of groups as the number of rows in the prior matrix addressed by fMat(). While the sampling is independent, the prior inverse variances for each trait are taken from the diagonal of the inverse-covariance matrix (in the SigmaI object), and are thus influenced by any correlated traits.

Parameters

in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

6.6.2.13 update() [13/19]

```
void RanIndex::update (
            const Grp & theta,
            const Grp & mu,
            const SigmaI & SigI,
            const MixP & p )
```

Update mixture model with a single covariance.

Updates the group relationships for Gaussian mixture models when each group has a different mean but all covrariances are the same.

Parameters

in	*Grp&*	data
in	*Grp&*	group means
in	*SigmaI&*	common inverse-covariance
in	*MixP&*	prior proportions

6.6.2.14 update() [14/19]

```
void RanIndex::update (
            const Grp & theta,
            const Grp & mu,
            const vector< SigmaI > & SigI,
            const MixP & p )
```

Update mixture model with multiple covariances.

Updates the group relationships for Gaussian mixture models when each group has a different mean and covariance.

Parameters

in	*Grp&*	data
in	*Grp&*	group means
in	*vector<SigmaI>&*	vector of inverse-covariances
in	*MixP&*	prior proportions

6.6.2.15 update() [15/19]

```
void Apex::update (
            const Grp & y,
            const gsl_matrix * xi,
            const Qgrp & q,
            const SigmaI & SigIm,
            const RanIndex & ind )
```

Replicated Student- t update.

Student- t data with replication, i.e., the number of rows in $\boldsymbol{Y}$ is larger than the number of rows in $\boldsymbol{\Xi}$.

Parameters

in	*Grp&*	data
in	*gsl_matrix∗*	"raw" location parameter matrix
in	*Qgrp&*	Student- t weights
in	*SigmaI&*	data inverse-covariance matrix
in	*RanIndex&*	index relating data rows to rows in $\boldsymbol{\Xi}$

6.6.2.16 update() [16/19]

```
void Apex::update (
            const Grp & y,
            const gsl_matrix * xi,
            const SigmaI & SigIm )
```

Unreplicated Gaussian update.

Gaussian data with no replication.

Parameters

in	*Grp&*	data
in	*gsl_matrix∗*	"raw" location parameter matrix
in	*SigmaI&*	data inverse-covariance matrix

6.6.2.17 update() [17/19]

```
void Apex::update (
            const Grp & y,
```

```
            const gsl_matrix * xi,
            const SigmaI & SigIm,
            const RanIndex & ind )
```

Replicated Gaussian update.

Gaussian data with replication, i.e., the number of rows in $\boldsymbol{Y}$ is larger than the number of rows in $\boldsymbol{\Xi}$.

Parameters

in	*Grp&*	data
in	*gsl_matrix∗*	"raw" location parameter matrix
in	*SigmaI&*	data inverse-covariance matrix
in	*RanIndex&*	index relating data rows to rows in $\boldsymbol{\Xi}$

6.6.2.18 update() [18/19]

```
virtual void RanIndex::update (
            const Grp & y,
            const SigmaI & SigIe,
            BetaGrpBVSR * theta,
            const SigmaI & SigIp )  [inline], [virtual]
```

Variable selection update.

Update for a mixture that includes point-mass at 0. Only implemented in the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	likelihood inverse-covariance
in	*BetaGrpBVSR∗*	location parameter values
in	*SigmaI&*	prior inverse-covariance

Reimplemented in RanIndexVS.

6.6.2.19 update() [19/19]

```
void MixP::update (
            const RanIndex & Nvec )
```

Gibbs update function.

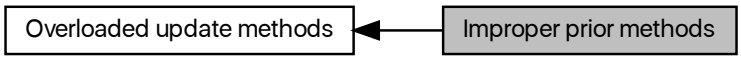
Performs standard Gibbs mixture updating (e.g., **[gelman04]**) with samples from a Dirichlet distribution.

Parameters

in	*RanIndex&*	data index of which elements belong to which category

6.7 Improper prior methods

Collaboration diagram for Improper prior methods:



Functions

- virtual void MVnorm::update (const Grp &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood.
- virtual void MVnorm::update (const Grp &, const Qgrp &, const SigmaI &, const gsl_rng *)=0

  Sudent-t likelihood.
- virtual void Grp::update (const Grp &, const SigmaI &)=0

  Gaussian likelihood, improper prior.
- virtual void Grp::update (const Grp &, const Qgrp &, const SigmaI &)=0

  Student-t likelihood, improper prior.

6.7.1 Detailed Description

6.7.2 Function Documentation

6.7.2.1 update() [1/4]

```
virtual void Grp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [pure virtual]
```

Student-t likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student-t weight parameter for data
in	*SigmaI&*	data inverse-covariance

Implemented in MuGrpEEmiss, MuGrpEE, MuBlk, BetaGrpFt, and MuGrp.

6.7.2.2 update() [2/4]

```
virtual void MVnorm::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.7.2.3 update() [3/4]

```
virtual void Grp::update (
            const Grp & ,
            const SigmaI &  )  [pure virtual]
```

Gaussian likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance

Implemented in MuGrpEEmiss, MuGrpEE, MuGrpMiss, MuBlk, BetaGrpSnpMiss, BetaGrpSnp, BetaGrpFt, and MuGrp.

6.7.2.4 update() [4/4]

```
virtual void MVnorm::update (
            const Grp & ,
```

```
        const SigmaI & ,
        const gsl_rng *  )  [pure virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuMiss, MVnormMuBlk, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.8 Mahalanobis distance functions

Functions

- virtual double MVnorm::mhl (const MVnorm *x, const SigmaI &SigI)

  Mahalanobis distance to a vector.
- virtual double MVnorm::mhl (const MVnorm *x, const SigmaI &SigI) const

  Mahalanobis distance to a vector.
- virtual double MVnorm::mhl (const gsl_vector *x, const SigmaI &SigI)

  Mahalanobis distance to a vector.
- virtual double MVnorm::mhl (const gsl_vector *x, const SigmaI &SigI) const

  Mahalanobis distance to a vector.
- virtual double MVnorm::mhl (const SigmaI &SigI)

  Mahalanobis distance to zero.
- virtual double MVnorm::mhl (const SigmaI &SigI) const

  Mahalanobis distance to zero.

6.8.1 Detailed Description

Overloaded functions to calculate the Mahalanobis distance of the vector of location parameters stored in the class to another vector or to zero.

6.8.2 Function Documentation

6.8.2.1 mhl() [1/6]

```
double MVnorm::mhl (
            const gsl_vector * x,
            const SigmaI & SigI )  [virtual]
```

Mahalanobis distance to a vector.

Calculates the Mahalanobis distance between the vector stored in the class and the provided GSL vector.

Parameters

in	*MVnorm&*	vector to calculate the distance from
in	*SigmaI&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.8.2.2 mhl() [2/6]

```
double MVnorm::mhl (
            const gsl_vector * x,
            const SigmaI & SigI ) const  [virtual]
```

Mahalanobis distance to a vector.

Calculates the Mahalanobis distance between the vector stored in the class and the provided GSL vector. The const version of the above function.

Parameters

in	*MVnorm&*	vector to calculate the distance from
in	*SigmaI&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.8.2.3 mhl() [3/6]

```
double MVnorm::mhl (
            const MVnorm * x,
            const SigmaI & SigI )  [virtual]
```

Mahalanobis distance to a vector.

Calculates the Mahalanobis distance between the vector stored in the class and the provided vector of class MVnorm.

Parameters

in	*MVnorm&*	vector to calculate the distance from
in	*SigmaI&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.8.2.4 mhl() [4/6]

```
double MVnorm::mhl (
            const MVnorm * x,
            const SigmaI & SigI ) const  [virtual]
```

Mahalanobis distance to a vector.

Calculates the Mahalanobis distance between the vector stored in the class and the provided vector of class MVnorm. The const version of the above function.

Parameters

in	*MVnorm&*	vector to calculate the distance from
in	*SigmaI&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.8.2.5 mhl() [5/6]

```
double MVnorm::mhl (
            const SigmaI & SigI )  [virtual]
```

Mahalanobis distance to zero.

Calculates the Mahalanobis distance between the vector stored in the class zero.

Parameters

in	*Sigma↩ I&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.8.2.6 mhl() [6/6]

```
double MVnorm::mhl (
            const SigmaI & SigI ) const  [virtual]
```

Mahalanobis distance to zero.

Calculates the Mahalanobis distance between the vector stored in the class zero. The const version of the above function.

Parameters

in	*Sigma↩ I&*	inverse covariance matrix for scaling

Returns

the distance value of type *double*.

6.9 Multivariate Gaussian density functions

Functions

- double MVnorm::density (const gsl_vector *theta, const SigmaI &SigI)

  Multivariate Gaussian density.
- double MVnorm::density (const gsl_vector *theta, const SigmaI &SigI) const

  Multivariate Gaussian density.
- double MVnorm::density (const MVnorm *theta, const SigmaI &SigI)

  Multivariate Gaussian density.
- double MVnorm::density (const MVnorm *theta, const SigmaI &SigI) const

  Multivariate Gaussian density.

6.9.1 Detailed Description

These functions calculate multivariate Gaussian density given the mean and inverse-covariance provided and the vector of values stored in the class as data.

6.9.2 Function Documentation

6.9.2.1 density() [1/4]

```
double MVnorm::density (
            const gsl_vector * theta,
            const SigmaI & SigI )
```

Multivariate Gaussian density.

Parameters

in	*gsl_vector**	vector of means
in	*SigmaI&*	inverse-covariance matrix

Returns

density value of type *double*.

6.9.2.2 density() [2/4]

```
double MVnorm::density (
            const gsl_vector * theta,
            const SigmaI & SigI ) const
```

Multivariate Gaussian density.

Note

A const version of the above function.

Parameters

in	*gsl_vector**	vector of means
in	*SigmaI&*	inverse-covariance matrix

Returns

density value of type *double*.

6.9.2.3 density() [3/4]

```
double MVnorm::density (
            const MVnorm * theta,
            const SigmaI & SigI )
```

Multivariate Gaussian density.

Parameters

in	*MVnorm**	vector of means
in	*SigmaI&*	inverse-covariance matrix

Returns

density value of type *double*

6.9.2.4 density() [4/4]

```
double MVnorm::density (
            const MVnorm * theta,
            const SigmaI & SigI ) const
```

Multivariate Gaussian density.

Note

A const version of the above function.

Parameters

in	*MVnorm*∗	vector of means
in	*Sigma1&*	inverse-covariance matrix

Returns

density value of type *double*.

6.10 Index classes

Classes

- class RanIndex

  Generic index class.
- class RanIndexVS

  BVSR index class.

6.10.1 Detailed Description

Index classes relate model hierarchy levels to each other through indexes. The relationships can be kept the same throughout the Gibbs sampling process. Alternatively, they can be updated according to various models, implemented in the base and derived class.

Note

The updating of indexes has not yet been extensively tested.

6.11 Arithmetic operators

Functions

- MuGrp operator+ (const Grp &m1, const Grp &m2)
  Addition operator.
- MuGrp operator+ (const MuGrp &m1, const Grp &m2)
  Addition operator.
- MuGrp operator+ (const Grp &m1, const MuGrp &m2)
  Addition operator.
- MuGrp operator- (const Grp &m1, const Grp &m2)
  Subtraction operator.
- MuGrp operator- (const MuGrp &m1, const Grp &m2)
  Subtraction operator.
- MuGrp operator- (const Grp &m1, const MuGrp &m2)
  Subtraction operator.

6.11.1 Detailed Description

Arithmetic operators for Grp type location parameters. The matrices addressed by the fMat() member functions are added. These are essential to build Gibbs samplers with multiple parameters at the same level of hierarchy, such as variable-intercept regressions **[gelman07]** . However, care must be taken because only certain combinations of objects are compatible. These are

1. fMat() matrices have the same number of rows. In this case, they are simply added or subtracted as is.
2. One of the matrices is a single-row matrix. In this case, the row is added to each row of the bigger matrix (or subtracted in the appropriate direction). The resulting MuGrp object will have the same size value matrix as the bigger fMat().
3. The RanIndex to the lower level of the Grp (or MuGrp) object with the smaller fMat() points to the same number of rows as in the fMat() of the larger object. In this case, the RanIndex is used to relate the rows of the larger matrix to those of the smaller matrix, the rows of the smaller matrix are re-used as necessary. The resulting MuGrp object has a value matrix of the same size as the larger fMat().
4. None of the above apply, but the RanIndex to the lower level of each object points to the same number of rows. In this case, a MuGrp object with a value matrix with that number of rows is created, and the addition/subtraction is performed with rows corresponing to the same lower level of the RanIndex.

In all other cases, an error is issued and execution is aborted. The number of traits (columns) has to always be the same. Addition is commutative, subtraction is anti-commutative.

6.11.2 Function Documentation

6.11.2.1 operator+() [1/3]

```
MuGrp operator+ (
            const Grp & m1,
            const Grp & m2 )
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

6.11.2.2 operator+() [2/3]

```
MuGrp operator+ (
            const Grp & m1,
            const MuGrp & m2 )
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*MuGrp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

6.11.2.3 operator+() [3/3]

```
MuGrp operator+ (
            const MuGrp & m1,
            const Grp & m2 )
```

Addition operator.

Parameters

in	*MuGrp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

6.11.2.4 operator-() [1/3]

```
MuGrp operator- (
            const Grp & m1,
            const Grp & m2 )
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

6.11.2.5 operator-() [2/3]

```
MuGrp operator- (
            const Grp & m1,
            const MuGrp & m2 )
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*MuGrp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

6.11.2.6 operator-() [3/3]

```
MuGrp operator- (
            const MuGrp & m1,
            const Grp & m2 )
```

Subtraction operator.

Parameters

in	*MuGrp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

6.12 Groups of location parameters

Classes

- class Grp
  Base location parameter group class.
- class MuGrp
  Hierarchical mean.
- class MuGrpPEX
  "Random effect" with parameter expansion
- class MuGrpMiss
  Data with missing phenotype values.
- class MuGrpEE
  Data with measurement error.
- class MuGrpEEmiss
  Data with measurement error and missing phenotypes.
- class BetaGrpFt
  Multivariate multiple regression.
- class BetaGrpPEX
  Multivariate multiple regression with parameter expansion.
- class BetaGrpPC
  Relationship matrix regression.
- class BetaGrpPCpex
  Multiplicative parameter expansion for PC regression.
- class BetaGrpSnp
  Simple single-SNP regression class.
- class BetaGrpSnpCV
  Single-SNP regression with conditional variance.
- class BetaGrpPSR
  Single-SNP regression with partial effects.
- class BetaGrpSnpMiss
  Simple single-SNP regression class with missing data.
- class BetaGrpSnpMissCV
  Single-SNP regression with conditional variance and missing data.
- class BetaGrpPSRmiss
  Single-SNP regression with partial effects and missing data.
- class BetaGrpBVSR
  Bayesian variable selection regression.
- class MuBlk
  Hierarchical mean with independent blocks of traits.
- class BetaBlk
  Regression with independent blocks of traits.

6.12.1 Detailed Description

Location parameters grouped by common modeling features. They have common priors, common data and prior covariances, and if they are regression coefficients a common set of predictors. Values are stored in a matrix with the columns representing traits (or analogous parameters) and rows – individual location parameters. The latter are targets of MVnorm classes and are typically modified in Markov chains through update functions implemented in these individual-row objects.

6.13 Functions to update fitted values

Functions

- void MuGrpPEX::_updateFitted ()

  Update adjusted values.
- virtual void BetaGrpFt::_updateFitted ()

  Update fitted values.
- void BetaGrpPEX::_updateAfitted ()

  Calculate redundant parameter fitted values.
- void BetaGrpBVSR::_updateFitted ()

  Update fitted values.
- void BetaBlk::_updateFitted ()

  Update fitted values.

6.13.1 Detailed Description

Protected member functions of the derived Grp classes that implement multiple regressions in some form. In Gibbs sampling for multiple regression, it is necessary to calculate partial fitted matrices of the form $\boldsymbol{X}_{\cdot -k}\boldsymbol{B}_{-k\cdot}$ for each k-th predictor (i.e., product for all but the k-th predictor), as well as the complete fitted matrix $\boldsymbol{C} = \boldsymbol{X}\boldsymbol{B}$. Here, $\boldsymbol{X}$ is the predictor matrix and $\boldsymbol{B}$ is the matrix of regression coefficients. This process is typically the bottleneck of the Gibbs sampler when even a moderate number of predictors (≥ 100) are in the model. The key idea in implementing these functions is that the multiplications required to calculate all the partial matrix products are also necessary for computing the complete fitted matrix. Thus, elements of the complete matrix

$$c_{i,j} = \sum_k x_{i,k}\beta_{k,j}$$

are calculated first, then the individual $x_{i,k}\beta_{k,j}$ are subtracted to get the required partial fitted matrices. Further speed-ups are achieved by parallelizing the calculations by row of the regression coefficient matrix (i.e. by individual predictor).

6.13.2 Function Documentation

6.13.2.1 _updateAfitted()

```
void BetaGrpPEX::_updateAfitted ( )  [protected]
```

Calculate redundant parameter fitted values.

Calculates separate $(\boldsymbol{X}\boldsymbol{\Xi}\boldsymbol{A})_{\cdot -p}$ for each trait p.

6.13.2.2 _updateFitted()

```
void MuGrpPEX::_updateFitted ( )  [protected]
```

Update adjusted values.

Creates the adjusted value matrix and the individual $\boldsymbol{\Xi}_{\cdot -m}\boldsymbol{A}_{-m\cdot}$.

6.14 Inverse covariance matrices

Classes

- class SigmaI

  Basic inverse-covariance.
- class SigmaIblk

  Block-diagonal inverse-covariance.
- class SigmaIpex

  PEX inverse-covariance.

6.14.1 Detailed Description

Group of classes that implement inverse-covariance matrix updating. We are dealing with inverse-covarince (a.k.a. concentration) matrices because most algorithms in hierarchical models use these rather than covariance matrices. Thus, we save on matrix inversion operations by keeping track only of inverse-covariances. The save functions, however, output covariance matrices. All the internal algorithms for all the classes use the lower triangles of the underlying symmetric matrix. On saving, the matrices are symmetric and it does not matter for downstream applications. However, if a user wants to manipulate or interact with these in a sampler, addressing the lower triangle ensures stability of the algorithm.

6.15 Student-t weight parameters

Classes

- class Qgrp

  Standard Student-t weights.
- class QgrpPEX

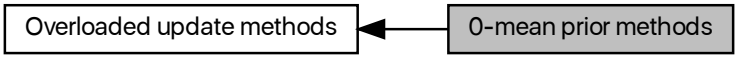
  Student-t weights for PEX.

6.15.1 Detailed Description

A popular way to build a Gibbs sampler for a Student-t model **[dyk01] [gelman04]** is to treat it like mixture of Gaussians, with the covariance of each data point weighted proportionally to its distabce from the mean. The weights are modeled as χ^2_ν scalars, with the degrees of freedom ν equal to the Student-t degrees of freedom. It is possible to further construct Metropolis updating steps for the degrees of freedom, but in our experience it complicates computation without adding much to inference. We therefore leave degrees of freedom as constant, to be varied manually by the user. We encourage trying a few degree of freedom values to see how the values affect inference. All constructors for these classes intitialize all weights deterministically to 1.0. Empirically, this seems the most numerically stable approach, and still results in reasonably well-spread starting points as long the location parameters and inverse-covariances are initiated stochastically.

6.16 0-mean prior methods

Collaboration diagram for 0-mean prior methods:



Functions

- virtual void MVnorm::update (const Grp &, const SigmaI &, const SigmaI &, const gsl_rng *)=0
  Gaussian likelihood, Gaussian prior.
- virtual void MVnorm::update (const Grp &, const SigmaI &, const double &, const SigmaI &, const gsl_rng *)=0
  Gaussian likelihood, Student-t prior.
- virtual void MVnorm::update (const Grp &, const Qgrp &, const SigmaI &, const SigmaI &, const gsl_rng *)=0
  Student-t likelihood, Gaussian prior.
- virtual void MVnorm::update (const Grp &, const Qgrp &, const SigmaI &, const double &, const SigmaI &, const gsl_rng *)=0
  Student-t likelihood, Student-t prior.
- virtual void Grp::update (const Grp &, const SigmaI &, const SigmaI &)=0
  Gaussian likelihood, 0-mean Gaussian prior.
- virtual void Grp::update (const Grp &, const Qgrp &, const SigmaI &, const SigmaI &)=0
  Student-t likelihood, 0-mean Gaussian prior.
- virtual void Grp::update (const Grp &, const SigmaI &, const Qgrp &, const SigmaI &)=0
  Gaussian likelihood, 0-mean Student-t prior.
- virtual void Grp::update (const Grp &, const Qgrp &, const SigmaI &, const Qgrp &, const SigmaI &)=0
  Student-t likelihood, 0-mean Student-t prior.

6.16.1 Detailed Description

6.16.2 Function Documentation

6.16.2.1 update() [1/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Student-t likelihood, Student-t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.16.2.2 update() [2/8]

```
virtual void Grp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [pure virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.16.2.3 update() [3/8]

```
virtual void Grp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [pure virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.16.2.4 update() [4/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.16.2.5 update() [5/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.16.2.6 update() [6/8]

```
virtual void Grp::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [pure virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.16.2.7 update() [7/8]

```
virtual void Grp::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [pure virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Implemented in MuGrpMiss, MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.16.2.8 update() [8/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Gaussian likelihood, Gaussian prior.

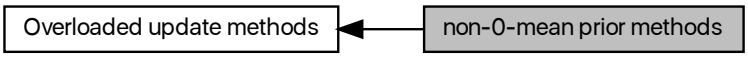
Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuMiss, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.17 non-0-mean prior methods

Collaboration diagram for non-0-mean prior methods:



Functions

- virtual void MVnorm::update (const Grp &, const SigmaI &, const Grp &, const SigmaI &, const gsl_rng $*$)=0

  Gaussian likelihood, Gaussian prior.
- virtual void MVnorm::update (const Grp &, const SigmaI &, const Grp &, const double &, const SigmaI &, const gsl_rng $*$)=0

  Gaussian likelihood, Student-t prior.
- virtual void MVnorm::update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const SigmaI &, const gsl_rng $*$)=0

  Student-t likelihood, Gaussian prior.
- virtual void MVnorm::update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const double &, const SigmaI &, const gsl_rng $*$)=0

  Student-t likelihood, Student-t prior.
- virtual void Grp::update (const Grp &, const SigmaI &, const Grp &, const SigmaI &)=0

  Gaussian likelihood, non-zero mean Gaussian prior.
- virtual void Grp::update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const SigmaI &)=0

  Student-t likelihood, non-zero mean Gaussian prior.
- virtual void Grp::update (const Grp &, const SigmaI &, const Grp &, const Qgrp &, const SigmaI &)=0

  Gaussian likelihood, non-zero mean Student-t prior.
- virtual void Grp::update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const Qgrp &, const SigmaI &)=0

  Student-t likelihood, non-zero mean Student-t prior.

6.17.1 Detailed Description

6.17.2 Function Documentation

6.17.2.1 update() [1/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [pure virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.17.2.2 update() [2/8]

```
virtual void Grp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ) [pure virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.17.2.3 update() [3/8]

```
virtual void Grp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [pure virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.17.2.4 update() [4/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance

Parameters

in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.17.2.5 update() [5/8]

```
virtual void MVnorm::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [pure virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

6.17.2.6 update() [6/8]

```
virtual void Grp::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ) [pure virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.17.2.7 update() [7/8]

```
virtual void Grp::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [pure virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Implemented in MuBlk, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, MuGrpPEX, and MuGrp.

6.17.2.8 update() [8/8]

```
virtual void MVnorm::update (
            const Grp & ,
```

```
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [pure virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implemented in MVnormBetaFt, MVnormMuBlk, MVnormBetaPEX, MVnormMuPEX, MVnormMu, MVnormBetaFtBlk, MVnormBetaBlk, and MVnormBeta.

Chapter 7

Class Documentation

7.1 Apex Class Reference

Multiplicative redundant parameter.

`#include <MuGen.h>`

Public Member Functions

- Apex ()
  Default constructor.
- Apex (const size_t &d, vector< vector< double > > &fE)
  Dimension-only constructor.
- Apex (const double &dV, const size_t &d, vector< vector< double > > &fE)
  Diagonal prior constructor.
- Apex (const SigmaI &SigI, vector< vector< double > > &fE)
  Covariance prior constructor.
- ~Apex ()
  Destructor.
- Apex (const Apex &A)
  Copy constructor.
- Apex & operator= (const Apex &A)
  Assignement constructor.
- gsl_matrix * getMat ()
  Get redundant parameter matrix.
- const gsl_matrix * getMat () const
  Get redundant parameter matrix.
- void setPr (const double &pr)
  Set prior value.
- void update (const Grp &y, const gsl_matrix *xi, const SigmaI &SigIm)
  Unreplicated Gaussian update.
- void update (const Grp &y, const gsl_matrix *xi, const SigmaI &SigIm, const RanIndex &ind)
  Replicated Gaussian update.
- void update (const Grp &y, const gsl_matrix *xi, const Qgrp &q, const SigmaI &SigIm, const RanIndex &ind)
  Replicated Student-t update.

7.1.1 Detailed Description

Multiplicative redundant parameter.

Implements multiplicative redundant parametrization for analogs of frequentist random-effects models. It is a multivariate extension of well-established multiplicative data augmentation schemes **[gelman07] [gelman08]** (Greenberg, unpublished). Here, instead of a scalar redundant parameter, I am using a redundant parameter matrix. This approach introduces additional per-iteration computational burden, but results in much faster model convergence in some cases (Greenberg, unpublished). Therefore, it should only be used when convergence problems are severe and cannot be eliminated with simpler re-parametrizations.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Apex() [1/5]

```
Apex::Apex ( )
```

Default constructor.

All matrices are set to be 1×1 with the element equal to 1.

7.1.2.2 Apex() [2/5]

```
Apex::Apex (
            const size_t & d,
            vector< vector< double > > & fE )
```

Dimension-only constructor.

Initializes a vague prior with diagonal elements set to 10^{-6}.

Parameters

in	*size_t&*	dimension (i.e. number of traits)
in	*vector<vector*	<double> >& fitted values

7.1.2.3 Apex() [3/5]

```
Apex::Apex (
            const double & dV,
```

```
            const size_t & d,
            vector< vector< double > > & fE )
```

Diagonal prior constructor.

Initializes a prior with diagonal elements set to the given value.

Parameters

in	*double&*	inverse-prior value
in	*size_t&*	dimension (i.e. number of traits)
in	*vector<vector*	<double> >& fitted values

7.1.2.4 Apex() [4/5]

```
Apex::Apex (
            const SigmaI & SigI,
            vector< vector< double > > & fE )
```

Covariance prior constructor.

Initializes a prior with the given inverse-covariance matrix. Dimension is set to the dimension of the prior matrix.

Parameters

in	*SigmaI&*	inverse-prior matrix
in	*vector<vector*	<double> >& fitted values

7.1.2.5 Apex() [5/5]

```
Apex::Apex (
            const Apex & A )
```

Copy constructor.

Parameters

in	*Apex&*	object to be copied

7.1.3 Member Function Documentation

7.1.3.1 operator=()

```
Apex & Apex::operator= (
            const Apex & A )
```

Assignement constructor.

Parameters

in	*Apex&*	object to be copied

Returns

Apex object

7.1.3.2 setPr()

```
void Apex::setPr (
            const double & pr )
```

Set prior value.

Sets the diagonal of the inverse-prior matrix to the given value

Parameters

in	*double&*	inverse-prior value

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

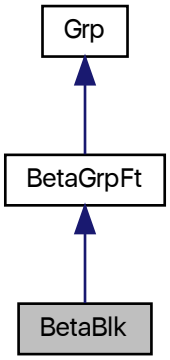
- MuGen.h
- MuGen.cpp

7.2 BetaBlk Class Reference

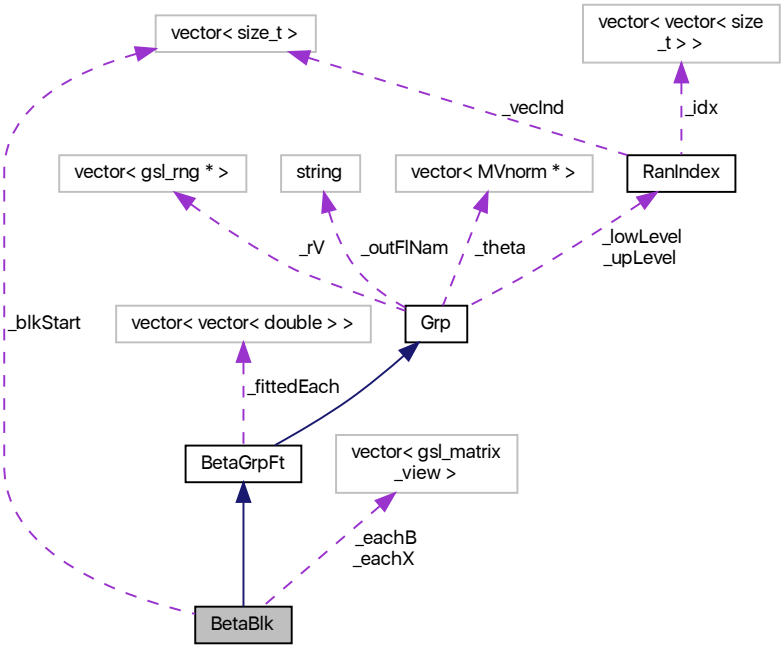
Regression with independent blocks of traits.

```
#include <MuGen.h>
```

Inheritance diagram for BetaBlk:



Collaboration diagram for BetaBlk:



Public Member Functions

- BetaBlk ()

  Default constructor.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const string &blkIndFileNam, const int &nThr)

  Basic constructor.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const string &outFlNam, const string &blkIndFileNam, const int &nThr)

  Constructor with output file name.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, RanIndex &up, const string &blkInd↩ FileNam, const int &nThr)

  Constructor with a prior index.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, RanIndex &up, const string &outFlNam, const string &blkIndFileNam, const int &nThr)

  Constructor with a prior index and output file.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const double &absLab, const string &blkIndFileNam, const int &nThr)

  Constructor with missing predictor data.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const double &absLab, const string &outFlNam, const string &blkIndFileNam, const int &nThr)

  Constructor with missing predictor values and output file name.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const double &absLab, RanIndex &up, const string &blkIndFileNam, const int &nThr)

  Constructor with a prior index and missing predictor data.
- BetaBlk (const Grp &dat, const string &predFlName, const size_t &Npred, const double &absLab, RanIndex &up, const string &outFlNam, const string &blkIndFileNam, const int &nThr)

  Constructor with a prior index, missing predictor data, and output file.
- ~BetaBlk ()

  Destructor.

Protected Member Functions

- void _updateFitted ()

  Update fitted values.

Protected Attributes

- vector< size_t > _blkStart

  Block start indexes.
- vector< gsl_matrix_view > _eachX

  Predictor submatrices.
- vector< gsl_matrix_view > _eachB

  Regression coefficient submatrices.

7.2.1 Detailed Description

Regression with independent blocks of traits.

Blocks of inter-correlated traits with correlations among blocks set to exactly zero. These models arise, for example, when traits measured in different environments are modeled together as different traits. The traits belonging to the same block have to be in adjacent columns. The number of predictors per block must be the same (future devolpment will drop this requirement), and the predictor columns corresponding to the same trait block have to be adjacent in the _Xmat matrix.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BetaBlk() [1/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            const string & blkIndFileNam,
            const int & nThr )
```

Basic constructor.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.2 BetaBlk() [2/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
```

```
            const string & predFlName,
            const size_t & Npred,
            const string & outFlNam,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with output file name.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*string&*	output file name
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.3 BetaBlk() [3/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            RanIndex & up,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with a prior index.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*RanIndex&*	prior (upper-level) index
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.4 BetaBlk() [4/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            RanIndex & up,
            const string & outFlNam,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with a prior index and output file.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*RanIndex&*	prior (upper-level) index
in	*string&*	output file name
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.5 BetaBlk() [5/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            const double & absLab,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with missing predictor data.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*double&*	missing value label
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.6 BetaBlk() [6/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            const double & absLab,
            const string & outFlNam,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with missing predictor values and output file name.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*double&*	missing value label
in	*string&*	output file name
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.7 BetaBlk() [7/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
```

```
            const string & predFlName,
            const size_t & Npred,
            const double & absLab,
            RanIndex & up,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with a prior index and missing predictor data.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block
in	*double&*	missing data label
in	*RanIndex&*	prior (upper-level) index
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.2.8 BetaBlk() [8/8]

```
BetaBlk::BetaBlk (
            const Grp & dat,
            const string & predFlName,
            const size_t & Npred,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const string & blkIndFileNam,
            const int & nThr )
```

Constructor with a prior index, missing predictor data, and output file.

The total number of predictors (columns in *Xmat) is the provided number of predictors times the number of blocks. The file that has the block start indexes should be a white-space delimited text file. The file that has the lower level index information should have a matrix of _int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the predictor file
in	*size_t&*	number of predictors per block

Parameters

in	*double&*	missing data label
in	*RanIndex&*	prior (upper-level) index
in	*string&*	output file name
in	*string&*	name of the file with the block indexes
in	*int&*	number of threads

7.2.3 Member Data Documentation

7.2.3.1 _blkStart

```
vector< size_t > BetaBlk::_blkStart  [protected]
```

Block start indexes.

Vector of indexes that indicate the column that starts each block.

7.2.3.2 _eachB

```
vector< gsl_matrix_view > BetaBlk::_eachB  [protected]
```

Regression coefficient submatrices.

A vector of GSL matrix views of the overall value matrix.

7.2.3.3 _eachX

```
vector< gsl_matrix_view > BetaBlk::_eachX  [protected]
```

Predictor submatrices.

A vector of GSL matrix views of the submatrices of the overall predictor matrix that correspond to each trait block.

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

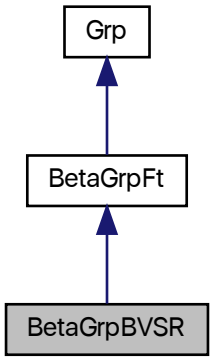
- MuGen.h
- MuGen.cpp

7.3 BetaGrpBVSR Class Reference

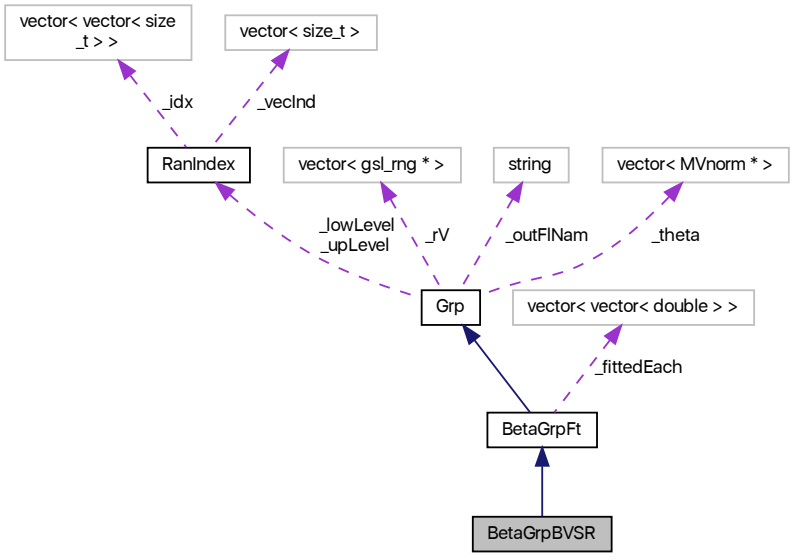
Bayesian variable selection regression.

`#include <MuGen.h>`

Inheritance diagram for BetaGrpBVSR:



Collaboration diagram for BetaGrpBVSR:



Public Member Functions

- BetaGrpBVSR ()

  Default constructor.
- BetaGrpBVSR (const Grp &y, const SigmaI &SigI, const string &predFlNam, const double &Nmul, const double &rSqMax, RanIndex &up, const string &outFlNam, const int &nThr)

  Basic constructor.
- BetaGrpBVSR (const Grp &y, const SigmaI &SigI, const string &predFlNam, const double &Nmul, const double &rSqMax, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Basic constructor with replication.
- BetaGrpBVSR (const Grp &y, const SigmaI &SigI, const string &predFlNam, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &up, const string &outFlNam, const int &nThr)

  Basic constructor with missing data.
- BetaGrpBVSR (const Grp &y, const SigmaI &SigI, const string &predFlNam, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Basic constructor with replication and missing data.
- ~BetaGrpBVSR ()

  Destructor.
- void save (const SigmaI &SigI)

  Regression value store.

Protected Member Functions

- void _updateFitted ()

  Update fitted values.
- void _ldToss (const gsl_vector *var, const gsl_permutation *prm, const double &rSqMax, const size_t &Nsel, const size_t &Npck, vector< vector< size_t > > &idx, vector< vector< size_t > > &rLd, gsl_matrix *Xpck)

  Testing candidates for correlation.
- double _MGkernel (const Grp &dat, const SigmaI &SigI) const

  Gaussian kernel.
- double _MGkernel (const Grp &dat, const SigmaI &SigI, const size_t &prInd) const

  Gaussian kernel dropping one predictor.
- double _MGkernel (const Grp &dat, const SigmaI &SigIe, const SigmaI &SigIpr, const size_t &prInd)

  Gaussian kernel adding one predictor.

Protected Attributes

- gsl_matrix * _selB

  Matrix of selected predictors.
- gsl_matrix * _tmpXb

  $\mathbf{x}_{\cdot l}\beta_l$. *matrix for the candidate dropped/added element*

Friends

- void **RanIndexVS::update** (const Grp &, const SigmaI &, BetaGrpBVSR *, const SigmaI &)

7.3.1 Detailed Description

Bayesian variable selection regression.

A multivariate implementation of Bayesian variable selection regression (BVSR), together with the RanIndexVS class. The model is similar to that described in **[guan11]** , but unlike their method a pre-selection of predictors based on a first-pass single-predictor regression is made. The user has control over the number of predictors to choose, and the cut-off for correlation among predictors (which arises as linkage disequilibrium in SNP regressions). This numebr is expressed as a multiple of genetic sample size. Once the pre-selected group of predictors is formed, variable selection is performed within the smaller group. Predictors discarded for correlation with picked variables are assigned the same probability of retention in the model as their corresponding picked predictors. Rao-Blackwellised values of regression coefficients are saved by the dump() function.

Warning

This class is experimental and has not been extensively tested

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BetaGrpBVSR() [1/4]

```
BetaGrpBVSR::BetaGrpBVSR (
            const Grp & y,
            const SigmaI & SigI,
            const string & predFlNam,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Basic constructor.

Constructor with no missing data in predictors and no replication. Performs the initial pre-selection of predictors.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*string&*	preddictor file name
in	*double&*	number of predictors to keep, in multiples of sample size
in	*double&*	r^2 cut-off for predictor correlation
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.3.2.2 BetaGrpBVSR() [2/4]

```
BetaGrpBVSR::BetaGrpBVSR (
            const Grp & y,
            const SigmaI & SigI,
            const string & predFlNam,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Basic constructor with replication.

Constructor with no missing data in predictors and replication. A RanIndex index is used instead of a design matrix. Performs the initial pre-selection of predictors.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*string&*	preddictor file name
in	*double&*	number of predictors to keep, in multiples of sample size
in	*double&*	r^2 cut-off for predictor correlation
in	*RanIndex&*	replicate (data) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.3.2.3 BetaGrpBVSR() [3/4]

```
BetaGrpBVSR::BetaGrpBVSR (
            const Grp & y,
            const SigmaI & SigI,
            const string & predFlNam,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Basic constructor with missing data.

Constructor with missing data in predictors and no replication. Missing data in the selected predictors are filled in using mean imputation. Performs the initial pre-selection of predictors.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*string&*	preddictor file name
in	*double&*	number of predictors to keep, in multiples of sample size
in	*double&*	r^2 cut-off for predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.3.2.4 BetaGrpBVSR() [4/4]

```
BetaGrpBVSR::BetaGrpBVSR (
            const Grp & y,
            const SigmaI & SigI,
            const string & predFlNam,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Basic constructor with replication and missing data.

Constructor with missing data in predictors and replication. Missing data in the selected predictors are filled in using mean imputation. A RanIndex index is used instead of a design matrix. Performs the initial pre-selection of predictors.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*string&*	preddictor file name
in	*double&*	number of predictors to keep, in multiples of sample size
in	*double&*	r^2 cut-off for predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	replicate (data) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.3.3 Member Function Documentation

7.3.3.1 _ldToss()

```
void BetaGrpBVSR::_ldToss (
            const gsl_vector * var,
            const gsl_permutation * prm,
            const double & rSqMax,
            const size_t & Nsel,
            const size_t & Npck,
            vector< vector< size_t > > & idx,
            vector< vector< size_t > > & rLd,
            gsl_matrix * Xpck )  [protected]
```

Testing candidates for correlation.

Goes through the list of "top" predictors and eliminates lower-ranked candidates correlated with them. If any candidates are eliminated, the list of top predictors is augmented with predictors previously discarded. The identity of the predictors eliminated for correlation is saved. The correlation among predictors arises, for example, when estimating SNP effect in genetics (GWAS).

Warning

Tested only superficially

Parameters

in	*gsl_vector*∗	$\left(\boldsymbol{x}^T\boldsymbol{x}\right)^{-1}$
in	*gsl_permutation*∗	predictor ranks
in	*double&*	r^2 cut-off
in	*size_t&*	number of predictors to pick
out	*vector*<	vector<size_t> >& index of picked predictors
out	*vector*<	vector<size_t> >& index relating dropped correlated predictors to their group-defining predictor
out	*gsl_matrix*∗	matrix of picked predictor values

7.3.3.2 _MGkernel() [1/3]

```
double BetaGrpBVSR::_MGkernel (
            const Grp & dat,
            const SigmaI & SigI ) const  [protected], [virtual]
```

Gaussian kernel.

Calculates the multivariate Gaussian kernel value for all regression coefficients in the model.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	inverse-covariance

Returns

double kernel value

Reimplemented from BetaGrpFt.

7.3.3.3 _MGkernel() [2/3]

```
double BetaGrpBVSR::_MGkernel (
            const Grp & dat,
            const SigmaI & SigI,
            const size_t & prInd ) const  [protected], [virtual]
```

Gaussian kernel dropping one predictor.

Calculates the multivariate Gaussian kernel value for all regression coefficients in the model, except for the one indicated.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	inverse-covariance
in	*size_t&*	index of the dropped predictor

Returns

double kernel value

Reimplemented from BetaGrpFt.

7.3.3.4 _MGkernel() [3/3]

```
double BetaGrpBVSR::_MGkernel (
            const Grp & dat,
            const SigmaI & SigIe,
```

```
            const SigmaI & SigIpr,
            const size_t & prInd )  [protected]
```

Gaussian kernel adding one predictor.

Calculates the multivariate Gaussian kernel value for all regression coefficients in the model, plus the indicated extra predictor with a Gaussian zero-mean prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance
in	*size_t&*	index of the dropped predictor

Returns

double kernel value

7.3.3.5 save()

```
void BetaGrpBVSR::save (
            const SigmaI & SigI )  [virtual]
```

Regression value store.

Stores t-statistic values for each trait and pre-selected (i.e. whether or not currently in the model) regression coefficient in a variable to subsequently save to a file by dump().

Parameters

in	*SigmaI&*	data inverse-covariance

Reimplemented from BetaGrpFt.

7.3.4 Member Data Documentation

7.3.4.1 _selB

```
gsl_matrix* BetaGrpBVSR::_selB  [protected]
```

Matrix of selected predictors.

Contains only the predictors currently in the model.

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

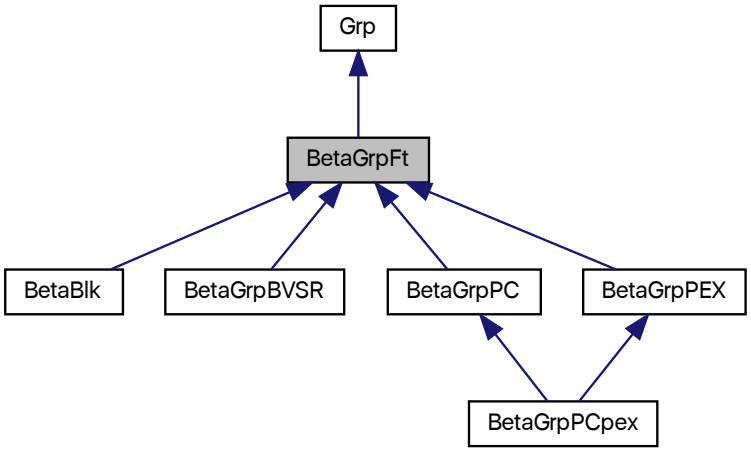
- MuGen.h
- MuGen.cpp

7.4 BetaGrpFt Class Reference

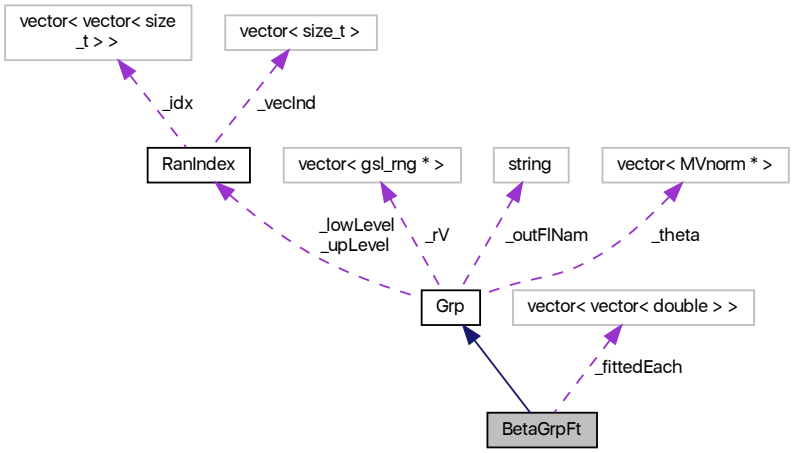
Multivariate multiple regression.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpFt:



Collaboration diagram for BetaGrpFt:



Public Member Functions

- BetaGrpFt ()

  Default constructor.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const int &nThr)

  Simple constructor.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, RanIndex &up, const int &nThr)

  Simple constructor with a prior index.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, RanIndex &low, RanIndex &up, const int &nThr)

  Simple constructor with a prior index and replication.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const string &outFlNam, const int &nThr)

  Simple constructor with output file name.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, RanIndex &up, const string &outFl↩ Nam, const int &nThr)

  Simple constructor with a prior index and output file name.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Simple constructor with a prior index, replication and output file name.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, const int &nThr)

  Missing data constructor.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, RanIndex &up, const int &nThr)

  Missing data constructor with a prior index.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, RanIndex &low, RanIndex &up, const int &nThr)

Missing data constructor with a prior index and replication.

- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, const string &outFlNam, const int &nThr)

  Missing data constructor with output file name.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, RanIndex &up, const string &outFlNam, const int &nThr)

  Missing data constructor with a prior index and output file name.
- BetaGrpFt (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &absLab, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Missing data constructor with a prior index, replication and output file name.
- BetaGrpFt (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Nmul, const double &rSqMax, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor.
- BetaGrpFt (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Nmul, const double &rSqMax, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor with replication.
- BetaGrpFt (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor with missing predictor data.
- BetaGrpFt (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor with missing predictor data and replication.
- virtual ~BetaGrpFt ()

  Destructor.
- BetaGrpFt (const BetaGrpFt &mG)

  Copy constructor.
- BetaGrpFt & operator= (const BetaGrpFt &mG)

  Assignment operator.
- virtual const gsl_matrix * fMat () const

  Access to the fitted value matrix.
- void save (const SigmaI &SigI)

  Store samples.
- void dump ()

  Dump to a file.
- double lnOddsRat (const Grp &y, const SigmaI &SigI, const size_t i) const

  Log-odds ratio.
- void update (const Grp &dat, const SigmaI &SigIm)

  Gaussian likelihood, improper prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm)

  Student- t likelihood, improper prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp)

  Student- t likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Student- t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

Student-t likelihood, 0-mean Student-t prior.

- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Protected Member Functions

- virtual void _updateFitted ()

  Update fitted values.
- void _rankPred (const gsl_matrix ∗y, const SigmaI &SigI, gsl_vector ∗XtX, gsl_permutation ∗prm)

  Rank predictors.
- void _rankPred (const gsl_matrix ∗y, const SigmaI &SigI, const double &absLab, gsl_vector ∗XtX, gsl_permutation ∗prm)

  Rank predictors with missing data.
- void _ldToss (const gsl_vector ∗var, const gsl_permutation ∗prm, const double &rSqMax, const size_t &Npck, vector< vector< size_t > > &idx, vector< vector< size_t > > &rLd, gsl_matrix ∗Xpck)

  Testing candidates for correlation.
- virtual double _MGkernel (const Grp &dat, const SigmaI &SigI) const

  Gaussian kernel.
- virtual double _MGkernel (const Grp &dat, const SigmaI &SigI, const size_t &prInd) const

  Gaussian kernel dropping one predictor.

Protected Attributes

- vector< vector< double > > _fittedEach

  Partial fitted value matrices.
- gsl_matrix ∗ _fittedAll

  Matrix of fitted values.
- gsl_matrix ∗ _valueSum

  Sample storage matrix.
- gsl_matrix ∗ _Xmat

  Predictor matrix.
- int _nThr

  Number of threads.
- double _numSaves

  Number of saves.

7.4.1 Detailed Description

Multivariate multiple regression.

Implements multivariate (multitrait) regression with multiple predictors. A single predictor is treated as a special case internally, the user does not have to do anything different. A variety for penalized regression methods is available, implemented through priors and pre-selection of variables, although the latter is still experimental. As the number of predictors grows, the computational burden increases and updates involving these objects become bottlenecks in the Markov chain computation. Therefore, great care has been taken to optimize computation for this class, sometimes at the expense of error checking.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 BetaGrpFt() [1/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const int & nThr )
```

Simple constructor.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication). No upper index is specified, so this object will implement a regression with an improper flat prior. Caution is advised if the number of predictors approaches the number of rows in the response, since inference will not be well-conditioned.

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*int&*	number of threads

7.4.2.2 BetaGrpFt() [2/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
```

```
            RanIndex & up,
            const int & nThr )
```

Simple constructor with a prior index.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.4.2.3 BetaGrpFt() [3/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            RanIndex & low,
            RanIndex & up,
            const int & nThr )
```

Simple constructor with a prior index and replication.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.4.2.4 BetaGrpFt() [4/17]

```
BetaGrpFt::BetaGrpFt (
```

```
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const string & outFlNam,
            const int & nThr )
```

Simple constructor with output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication). No upper index is specified, so this object will implement a regression with an improper flat prior. Caution is advised if the number of predictors approaches the number of rows in the response, since inference will not be well-conditioned.

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size↩ _t&*	number of predictors
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.5 BetaGrpFt() [5/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Simple constructor with a prior index and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.6 BetaGrpFt() [6/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Simple constructor with a prior index, replication and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.7 BetaGrpFt() [7/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            const int & nThr )
```

Missing data constructor.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication). No upper index is specified, so this object will implement a regression with an improper flat prior. Caution is advised if the number of predictors approaches the number of rows in the response, since inference will not be well-conditioned.

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*int&*	number of threads

7.4.2.8 BetaGrpFt() [8/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            RanIndex & up,
            const int & nThr )
```

Missing data constructor with a prior index.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.4.2.9 BetaGrpFt() [9/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            RanIndex & low,
```

```
            RanIndex & up,
            const int & nThr )
```

Missing data constructor with a prior index and replication.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.4.2.10 BetaGrpFt() [10/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            const string & outFlNam,
            const int & nThr )
```

Missing data constructor with output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication). No upper index is specified, so this object will implement a regression with an improper flat prior. Caution is advised if the number of predictors approaches the number of rows in the response, since inference will not be well-conditioned.

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.11 BetaGrpFt() [11/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Missing data constructor with a prior index and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.12 BetaGrpFt() [12/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Missing data constructor with a prior index, replication and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.13 BetaGrpFt() [13/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Selection constructor.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model.
The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection).

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.14 BetaGrpFt() [14/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Selection constructor with replication.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection).

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.15 BetaGrpFt() [15/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Selection constructor with missing predictor data.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection). Missing predictor data are filled in by mean imputation.

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.16 BetaGrpFt() [16/17]

```
BetaGrpFt::BetaGrpFt (
            const Grp & rsp,
```

```
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Selection constructor with missing predictor data and replication.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection). Missing predictor data are filled in by mean imputation.

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.4.2.17 BetaGrpFt() [17/17]

```
BetaGrpFt::BetaGrpFt (
            const BetaGrpFt & mG )
```

Copy constructor.

Parameters

in	*BetaGrpFt&*	object to be copied

Returns

BetaGrpFt object

7.4.3 Member Function Documentation

7.4.3.1 _ldToss()

```
void BetaGrpFt::_ldToss (
            const gsl_vector * var,
            const gsl_permutation * prm,
            const double & rSqMax,
            const size_t & Npck,
            vector< vector< size_t > > & idx,
            vector< vector< size_t > > & rLd,
            gsl_matrix * Xpck )  [protected]
```

Testing candidates for correlation.

Goes through the list of "top" predictors and eliminates lower-ranked candidates correlated with them. If any candidates are eliminated, the list of top predictors is augmented with predictors previously discarded. The identity of the predictors eliminated for correlation is saved. The correlation among predictors arises, for example, when estimating SNP effect in genetics (GWAS).

Warning

Tested only superficially

Parameters

in	*gsl_vector*∗	$\left(\boldsymbol{x}^T\boldsymbol{x}\right)^{-1}$
in	*gsl_permutation*∗	predictor ranks
in	*double&*	r^2 cut-off
in	*size_t&*	number of predictors to pick
out	*vector*<	vector<size_t> >& index of picked predictors
out	*vector*<	vector<size_t> >& index relating dropped correlated predictors to their group-defining predictor
out	*gsl_matrix*∗	matrix of picked predictor values

7.4.3.2 _MGkernel() [1/2]

```
double BetaGrpFt::_MGkernel (
            const Grp & dat,
            const SigmaI & SigI ) const  [protected], [virtual]
```

Gaussian kernel.

Calculates the multivariate Gaussian kernel value for all regression coefficients in the model.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance

Returns

double kernel value

Reimplemented in BetaGrpBVSR.

7.4.3.3 _MGkernel() [2/2]

```
double BetaGrpFt::_MGkernel (
            const Grp & dat,
            const SigmaI & SigI,
            const size_t & prInd ) const  [protected], [virtual]
```

Gaussian kernel dropping one predictor.

Calculates the multivariate Gaussian kernel value for all regression coefficients in the model, except for the one indicated.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance
in	*size_t&*	index of the dropped predictor

Returns

double kernel value

Reimplemented in BetaGrpBVSR.

7.4.3.4 _rankPred() [1/2]

```
void BetaGrpFt::_rankPred (
            const gsl_matrix * y,
            const SigmaI & SigI,
            const double & absLab,
            gsl_vector * XtX,
            gsl_permutation * prm )  [protected]
```

Rank predictors with missing data.

Ranking the predictors by the size of their effects in preparation for eliminating the ones below a certain rank. Missing predictor values are labeled by a given value.

Parameters

in	*gsl_matrix*∗	data (response)
in	*SigmaI&*	data inverse-covariance
in	*double&*	missing data label
in	*gsl_vector*∗	vector of regression scales $\left(\boldsymbol{x}^T\boldsymbol{x}\right)^{-1}$
out	*gsl_permutation*∗	predictor ranks

7.4.3.5 _rankPred() [2/2]

```
void BetaGrpFt::_rankPred (
            const gsl_matrix * y,
            const SigmaI & SigI,
            gsl_vector * XtX,
            gsl_permutation * prm )  [protected]
```

Rank predictors.

Ranking the predictors by the size of their effects in preparation for eliminating the ones below a certain rank.

Parameters

in	*gsl_matrix*∗	data (response)
in	*SigmaI&*	data inverse-covariance
in	*gsl_vector*∗	vector of regression scales $\left(\boldsymbol{x}^T\boldsymbol{x}\right)^{-1}$
out	*gsl_permutation*∗	predictor ranks

7.4.3.6 dump()

```
void BetaGrpFt::dump ( )  [virtual]
```

Dump to a file.

Dumps paramter values averaged over the MCMC run to a file in the end of the run. Has an effect only in derived classes where it is implemented. In these classes, the name of the file is pre-specified at initialization.

Reimplemented from Grp.

7.4.3.7 fMat()

```
virtual const gsl_matrix* BetaGrpFt::fMat ( ) const  [inline], [virtual]
```

Access to the fitted value matrix.

Pointer to the $\boldsymbol{XB}$ matrix, while dMat() accesses the regression coefficient matrix.

Returns

gsl_matrix* pointer to the matrix of fitted values

Reimplemented from Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.8 lnOddsRat()

```
double BetaGrpFt::lnOddsRat (
            const Grp & y,
            const SigmaI & SigI,
            const size_t i ) const  [virtual]
```

Log-odds ratio.

Log-odds ration between models with and without the i-th regression coefficient (row). Makes sense only in regression classes, everywhere else returns zero.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*size_t*	index of the element (row)

Returns

double log-odds ratio

Reimplemented from Grp.

7.4.3.9 operator=()

```
BetaGrpFt & BetaGrpFt::operator= (
            const BetaGrpFt & mG )
```

Assignment operator.

Parameters

in	*BetaGrpFt&*	object to be copied

Returns

BetaGrpFt& target object

7.4.3.10 save()

```
void BetaGrpFt::save (
            const SigmaI & SigI )  [virtual]
```

Store samples.

Stores samples of predictor effect scores in a matrix to be dumped at the end of the MCMC run.

Parameters

in	*Sigma↩ I&*	data inverse-covariance

Reimplemented from Grp.

Reimplemented in BetaGrpBVSR, BetaGrpPCpex, and BetaGrpPEX.

7.4.3.11 update() [1/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance

Implements Grp.

7.4.3.12 update() [2/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.13 update() [3/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Sigma↩ I&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.14 update() [4/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.15 update() [5/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.16 update() [6/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance

Implements Grp.

7.4.3.17 update() [7/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.18 update() [8/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.19 update() [9/10]

```
void BetaGrpFt::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.3.20 update() [10/10]

```
void BetaGrpFt::update (
            const Grp & ,
```

```
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Implements Grp.

Reimplemented in BetaGrpPCpex, and BetaGrpPEX.

7.4.4 Member Data Documentation

7.4.4.1 _fittedAll

```
gsl_matrix* BetaGrpFt::_fittedAll  [protected]
```

Matrix of fitted values.

The $\boldsymbol{X}\boldsymbol{B}matrix.$ In cases where the index to the lower level is initialized, i.e. there is replication in the response, this matrix has the same number of rows as the number of unique values of the predictor.

7.4.4.2 _fittedEach

```
vector< vector<double> > BetaGrpFt::_fittedEach  [protected]
```

Partial fitted value matrices.

Each member of the vector stores the element-specific fitted matrix ($\boldsymbol{X}_{\cdot -k}\boldsymbol{B}_{-k\cdot}$) as a vector in the row-major format, to be accessed as a matrix_view of an array. If there is only one predictor, this is empty. If there is replication (i.e. the index to the lower level is initialized), these matrices have the same number of rows as the response (equivalent to $\boldsymbol{Z}\boldsymbol{X}_{\cdot -k}\boldsymbol{B}_{-k\cdot}$, where $\boldsymbol{Z}$ is the design matrix).

7.4.4.3 _numSaves

```
double BetaGrpFt::_numSaves  [protected]
```

Number of saves.

Number of saves made, to calculate the mean at the end.

7.4.4.4 _valueSum

```
gsl_matrix* BetaGrpFt::_valueSum  [protected]
```

Sample storage matrix.

Stores Markov chain samples of the value matrix if we want only the point estimates in the end. Stores the sum of all the estimates saved up to now; allocated only if needed (under the condition that _numSaves != 0.0)

7.4.4.5 _Xmat

```
gsl_matrix* BetaGrpFt::_Xmat  [protected]
```

Predictor matrix.

Individual predictors are in columns. If there is replication, the number of rows is expanded to fit the number of rows in the response matrix ($\boldsymbol{ZXB}$).

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

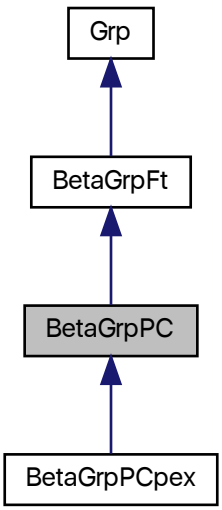
- MuGen.h
- MuGen.cpp

7.5 BetaGrpPC Class Reference

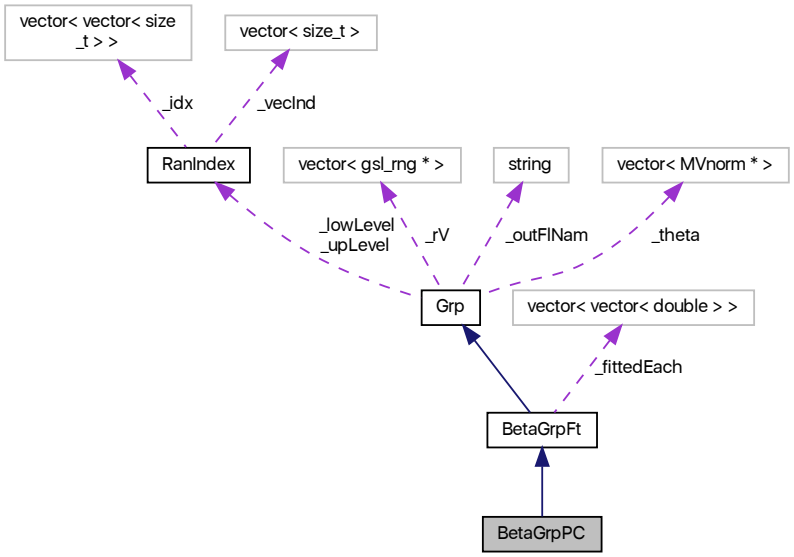
Relationship matrix regression.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpPC:



Collaboration diagram for BetaGrpPC:



Public Member Functions

- BetaGrpPC ()

  Default constructor.
- BetaGrpPC (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, RanIndex &up, const int &nThr)

  Constructor with a prior index.
- BetaGrpPC (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, RanIndex &low, RanIndex &up, const int &nThr)

  Constructor with a prior index and replication.
- BetaGrpPC (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, RanIndex &up, const string &outFlNam, const int &nThr)

  Constructor with a prior index, replication and output file name.
- BetaGrpPC (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Constructor with a prior index and replication.
- virtual ~BetaGrpPC ()

  Destructor.
- BetaGrpPC (const BetaGrpPC &mG)

  Copy constructor.
- BetaGrpPC & operator= (const BetaGrpPC &mG)

  Assignment operator.

Additional Inherited Members

7.5.1 Detailed Description

Relationship matrix regression.

Implements regression models with principal components of a relationship matrix as predictors. Only principal vectors with non-zero eigenvalues are used. If the likelihood and the prior are Gaussian, and the prior mean is zero, this is equivalent to the "mixed model" in packages like EMMA and TASSEL. Because the PC vectors are orthonormal there are some speed-ups in initialization. The updates are handled by the same internal row classes as in BetaGrpFt.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 BetaGrpPC() [1/5]

```
BetaGrpPC::BetaGrpPC (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            RanIndex & up,
            const int & nThr )
```

Constructor with a prior index.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*RanIndex&*	prior index
in	*int&*	number of threads

7.5.2.2 BetaGrpPC() [2/5]

```
BetaGrpPC::BetaGrpPC (
            const Grp & rsp,
```

```
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            RanIndex & low,
            RanIndex & up,
            const int & nThr )
```

Constructor with a prior index and replication.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*int&*	number of threads

7.5.2.3 BetaGrpPC() [3/5]

```
BetaGrpPC::BetaGrpPC (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Constructor with a prior index, replication and output file name.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.5.2.4 BetaGrpPC() [4/5]

```
BetaGrpPC::BetaGrpPC (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )
```

Constructor with a prior index and replication.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.5.2.5 BetaGrpPC() [5/5]

```
BetaGrpPC::BetaGrpPC (
            const BetaGrpPC & mG )
```

Copy constructor.

Parameters

in	*BetaGrpPC&*	object to be copied

7.5.3 Member Function Documentation

7.5.3.1 operator=()

```
BetaGrpPC & BetaGrpPC::operator= (
            const BetaGrpPC & mG )
```

Assignment operator.

Parameters

in	*BetaGrpPC&*	object to be copied

Returns

BetaGrpPC& target object

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

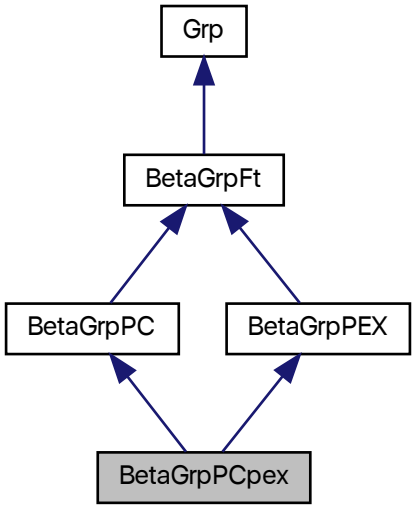
- MuGen.h
- MuGen.cpp

7.6 BetaGrpPCpex Class Reference

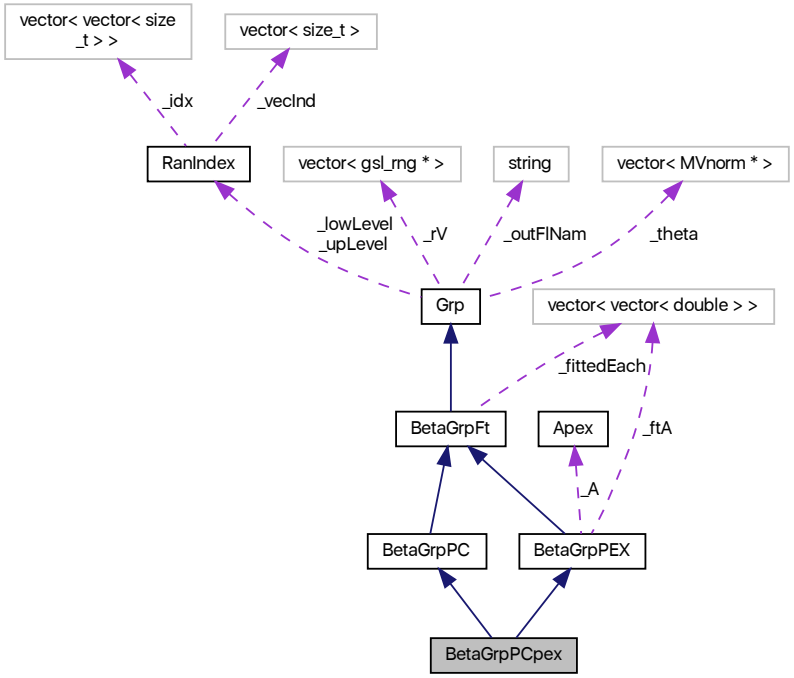
Multiplicative parameter expansion for PC regression.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpPCpex:



Collaboration diagram for BetaGrpPCpex:



Public Member Functions

- BetaGrpPCpex ()

  Default constructor.
- BetaGrpPCpex (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, const double &Spr, RanIndex &up, const int &nThr)

  Constructor with a prior index.
- BetaGrpPCpex (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, const double &Spr, RanIndex &low, RanIndex &up, const int &nThr)

  Constructor with a prior index and replication.
- BetaGrpPCpex (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, const double &Spr, RanIndex &up, const string &outFlNam, const int &nThr)

  Constructor with a prior index, replication and output file name.
- BetaGrpPCpex (const Grp &rsp, const string &predFlNam, const string &evFlNam, const size_t &Npred, const double &Spr, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Constructor with a prior index and replication.
- ~BetaGrpPCpex ()

  Destructor.
- const gsl_matrix * fMat () const

  Access adjusted fitted value matrix.
- void save ()

Save adjusted values.

- void save (const string &outFlNam)

  Save adjusted values to named file.
- void save (const SigmaI &SigI)

  Save adjusted values with the adjusted covariance matrix.
- void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Student-t prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Additional Inherited Members

7.6.1 Detailed Description

Multiplicative parameter expansion for PC regression.

Brings together implementations from BetaGrpPC and BetaGrpPEX.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 BetaGrpPCpex() [1/4]

```
BetaGrpPCpex::BetaGrpPCpex (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & up,
            const int & nThr )  [inline]
```

Constructor with a prior index.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	prior index
in	*int&*	number of threads

7.6.2.2 BetaGrpPCpex() [2/4]

```
BetaGrpPCpex::BetaGrpPCpex (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & low,
            RanIndex & up,
            const int & nThr )  [inline]
```

Constructor with a prior index and replication.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*int&*	number of threads

7.6.2.3 BetaGrpPCpex() [3/4]

```
BetaGrpPCpex::BetaGrpPCpex (
            const Grp & rsp,
```

```
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Constructor with a prior index, replication and output file name.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name
in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.6.2.4 BetaGrpPCpex() [4/4]

```
BetaGrpPCpex::BetaGrpPCpex (
            const Grp & rsp,
            const string & predFlNam,
            const string & evFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Constructor with a prior index and replication.

Reads principal vectors and eigen-values from files. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index.

Parameters

in	*Grp&*	data for initialization
in	*string&*	PC vector file name

Parameters

in	*string&*	eigen-value file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	lower-level (replicate) index
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*int&*	number of threads

7.6.3 Member Function Documentation

7.6.3.1 fMat()

```
const gsl_matrix* BetaGrpPCpex::fMat ( ) const  [inline], [virtual]
```

Access adjusted fitted value matrix.

Returns

gsl_matrix* pointer to the adjusted-value fitted matrix

Reimplemented from BetaGrpPEX.

7.6.3.2 save() [1/3]

```
void BetaGrpPCpex::save ( )  [inline], [virtual]
```

Save adjusted values.

Appends the current adjusted mean values to the file whose name was set during construction.

Reimplemented from BetaGrpPEX.

7.6.3.3 save() [2/3]

```
void BetaGrpPCpex::save (
            const SigmaI & SigI )  [inline], [virtual]
```

Save adjusted values with the adjusted covariance matrix.

Appends the current adjusted mean values and prior covariance matrix to the files whose names were set during construction.

Reimplemented from BetaGrpPEX.

7.6.3.4 save() [3/3]

```
void BetaGrpPCpex::save (
            const string & outFlNam )  [inline], [virtual]
```

Save adjusted values to named file.

Appends the current adjusted mean values to the named file.

Parameters

in	*string&*	file name

Reimplemented from BetaGrpPEX.

7.6.3.5 update() [1/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [inline], [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.6 update() [2/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [inline], [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.7 update() [3/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
```

```
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [inline], [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.8 update() [4/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [inline], [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.9 update() [5/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
```

```
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [inline], [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.10 update() [6/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [inline], [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.11 update() [7/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [inline], [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

7.6.3.12 update() [8/8]

```
void BetaGrpPCpex::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [inline], [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpPEX.

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

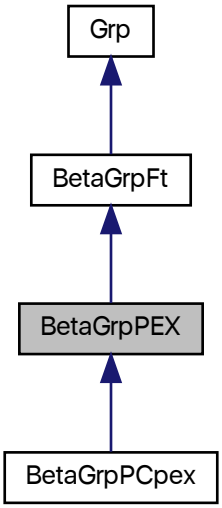
- MuGen.h

7.7 BetaGrpPEX Class Reference

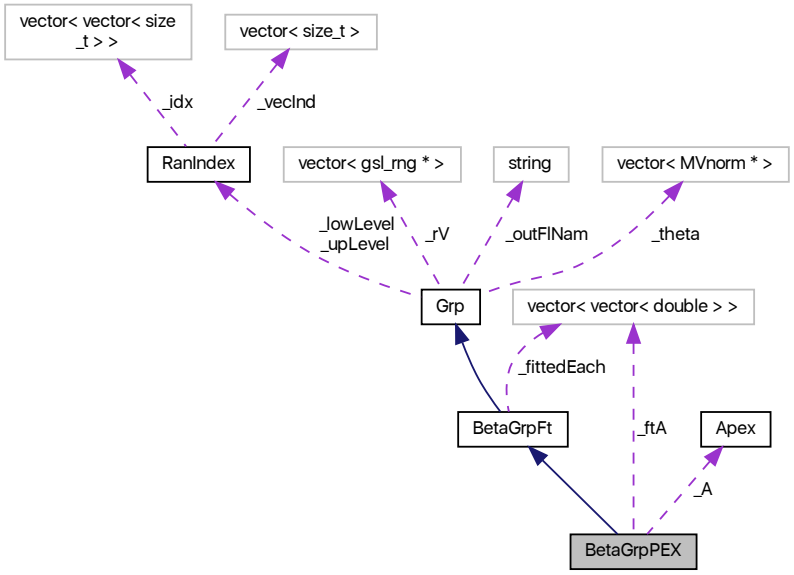
Multivariate multiple regression with parameter expansion.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpPEX:



Collaboration diagram for BetaGrpPEX:



Public Member Functions

- BetaGrpPEX ()

  Default constructor.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, RanIndex &up, const int &nThr)

  Simple constructor with a prior index.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, RanIndex &low, RanIndex &up, const int &nThr)

  Simple constructor with a prior index and replication.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, RanIndex &up, const string &outFlNam, const int &nThr)

  Simple constructor with a prior index and output file name.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Simple constructor with a prior index, replication and output file name.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, const double &absLab, RanIndex &up, const int &nThr)

  Missing data constructor with a prior index.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, const double &absLab, RanIndex &low, RanIndex &up, const int &nThr)

  Missing data constructor with a prior index and replication.
- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, const double &absLab, RanIndex &up, const string &outFlNam, const int &nThr)

Missing data constructor with a prior index and output file name.

- BetaGrpPEX (const Grp &rsp, const string &predFlNam, const size_t &Npred, const double &Spr, const double &absLab, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Missing data constructor with a prior index, replication and output file name.
- BetaGrpPEX (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Spr, const double &Nmul, const double &rSqMax, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor.
- BetaGrpPEX (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Spr, const double &Nmul, const double &rSqMax, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor with replication.
- BetaGrpPEX (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Spr, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &up, const string &outFl↩ Nam, const int &nThr)

  Selection constructor with missing predictor data.
- BetaGrpPEX (const Grp &rsp, const SigmaI &SigI, const string &predFlNam, const size_t &Npred, const double &Spr, const double &Nmul, const double &rSqMax, const double &absLab, RanIndex &low, RanIndex &up, const string &outFlNam, const int &nThr)

  Selection constructor with missing predictor data and replication.
- virtual ∼BetaGrpPEX ()

  Destructor.
- virtual const gsl_matrix ∗ fMat () const

  Access adjusted fitted value matrix.
- virtual void save ()

  Save adjusted values.
- virtual void save (const string &outFlNam)

  Save adjusted values to named file.
- virtual void save (const SigmaI &SigI)

  Save adjusted values with the adjusted covariance matrix.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Protected Member Functions

- void _finishConstruct (const double &Spr)

  Finish construction.
- void _finishFitted ()

  Adjusted matrix calculation.
- void _updateAfitted ()

  Calculate redundant parameter fitted values.
- BetaGrpPEX (const double &Spr)

  Finishing constructor.

Protected Attributes

- gsl_matrix * _tSigIAt

  Scaled inverse-covariance.
- Apex _A

  Multiplicative redundant parameter.
- vector< vector< double > > _ftA

  Fitted values.
- gsl_matrix * _fittedAllAdj

  Adjusted fitted matrix.

7.7.1 Detailed Description

Multivariate multiple regression with parameter expansion.

Implements multiplicative prameter expansion as in MuGrpPEX, but for regression coefficients. The "raw" value matrix Ξ contains regression coefficients on the unadjusted scale (see MuGrpPEX for explanation of these terms). The adjusted regression coefficient matrix is not calculated, but fMat() point to the adjusted fitted value matrix $\boldsymbol{X}\Xi\boldsymbol{A}$. Regression models with this method must have a prior, typically one with mean zero.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 BetaGrpPEX() [1/13]

```
BetaGrpPEX::BetaGrpPEX (
            const double & Spr )  [inline], [protected]
```

Finishing constructor.

For use in BetaGrpPCpex.

Parameters

in	*double&*	prior inverse variance for A

7.7.2.2 BetaGrpPEX() [2/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & up,
            const int & nThr )  [inline]
```

Simple constructor with a prior index.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.7.2.3 BetaGrpPEX() [3/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & low,
            RanIndex & up,
            const int & nThr )  [inline]
```

Simple constructor with a prior index and replication.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.7.2.4 BetaGrpPEX() [4/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Simple constructor with a prior index and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.5 BetaGrpPEX() [5/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
```

```
            const size_t & Npred,
            const double & Spr,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Simple constructor with a prior index, replication and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.6 BetaGrpPEX() [6/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & absLab,
            RanIndex & up,
            const int & nThr )  [inline]
```

Missing data constructor with a prior index.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.7.2.7 BetaGrpPEX() [7/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const int & nThr )  [inline]
```

Missing data constructor with a prior index and replication.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*int&*	number of threads

7.7.2.8 BetaGrpPEX() [8/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Missing data constructor with a prior index and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of rows in the response (no replication).

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.9 BetaGrpPEX() [9/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Missing data constructor with a prior index, replication and output file name.

Reads the predictor from a file and initiates regression coefficients using the provided response data. Predictor has missing data labeled by the provided value. Number of rows of the predictor is equal to the number of groups in the lower (data) index. The number of rows in the response is equal to the number of elements in the low index

Parameters

in	*Grp&*	response
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for A
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.10 BetaGrpPEX() [10/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Selection constructor.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection).

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for $\boldsymbol{A}$
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.11 BetaGrpPEX() [11/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
```

```
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
            const double & Spr,
            const double & Nmul,
            const double & rSqMax,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Selection constructor with replication.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection).

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for $\boldsymbol{A}$
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.12 BetaGrpPEX() [12/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
```

```
            const size_t & Npred,
            const double & Spr,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Selection constructor with missing predictor data.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection). Missing predictor data are filled in by mean imputation.

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for $\boldsymbol{A}$
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.2.13 BetaGrpPEX() [13/13]

```
BetaGrpPEX::BetaGrpPEX (
            const Grp & rsp,
            const SigmaI & SigI,
            const string & predFlNam,
            const size_t & Npred,
```

```
            const double & Spr,
            const double & Nmul,
            const double & rSqMax,
            const double & absLab,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam,
            const int & nThr )  [inline]
```

Selection constructor with missing predictor data and replication.

Unlike the previous constructors, selection-type constructors pre-screen predictors for association with any of the traits (using Hoteling-like multi-trait statistics), and keeps only the specified proportion in the model. The candidate predictors are further screened for between-predictor correlation and only ones with r^2 below the provided threshold are kept in the model. This among-predictor correlation often arises in SNP data, where there is likage disequilbrium (LD) among markers. Once the predictors are picked, the updating proceeds like for other BetaGrpFt objects, i.e. the set remains the same (unlike variable selection). Missing predictor data are filled in by mean imputation.

Warning

This set of models is still experimental and has not been extenisively tested. For example, it seems clear that the rest of the model has to be pre-run to convergence before initializing this kind of object. Furthermore, the initializeing predictor probably has to be a mean of several (~ 50) MCMC samples.

Parameters

in	*Grp&*	response
in	*SigmaI&*	data inverse-covariance
in	*srting&*	predictor file name
in	*size_t&*	number of predictors
in	*double&*	prior inverse variance for $\boldsymbol{A}$
in	*double&*	fraction of predictors to retain, as a fraction of the number of data points
in	*double&*	r^2 cut-off for among-predictor correlation
in	*double&*	missing data label
in	*RanIndex&*	replication index
in	*RanIndex&*	index to the prior
in	*string&*	output file name
in	*int&*	number of threads

7.7.3 Member Function Documentation

7.7.3.1 _finishConstruct()

```
void BetaGrpPEX::_finishConstruct (
            const double & Spr )  [protected]
```

Finish construction.

Most of the construction tasks are handled by the parent class. This function sets up the PEX portion of the class.

Parameters

in	*double&*	prior inverse variance for A

7.7.3.2 _finishFitted()

```
void BetaGrpPEX::_finishFitted ( )  [protected]
```

Adjusted matrix calculation.

Calculates adjusted fitted matrix and all sub-matrices.

7.7.3.3 fMat()

```
virtual const gsl_matrix* BetaGrpPEX::fMat ( ) const  [inline], [virtual]
```

Access adjusted fitted value matrix.

Returns

gsl_matrix* pointer to the adjusted-value fitted matrix

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.4 save() [1/3]

```
void BetaGrpPEX::save ( )  [virtual]
```

Save adjusted values.

Appends the current adjusted mean values to the file whose name was set during construction.

Reimplemented from Grp.

Reimplemented in BetaGrpPCpex.

7.7.3.5 save() [2/3]

```
void BetaGrpPEX::save (
            const SigmaI & SigI )  [virtual]
```

Save adjusted values with the adjusted covariance matrix.

Appends the current adjusted mean values and prior covariance matrix to the files whose names were set during construction.

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.6 save() [3/3]

```
void BetaGrpPEX::save (
            const string & outFlNam )  [virtual]
```

Save adjusted values to named file.

Appends the current adjusted mean values to the named file.

Parameters

in	*string&*	file name

Reimplemented from Grp.

Reimplemented in BetaGrpPCpex.

7.7.3.7 update() [1/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.8 update() [2/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.9 update() [3/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.10 update() [4/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.11 update() [5/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.12 update() [6/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.13 update() [7/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.3.14 update() [8/8]

```
void BetaGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Sigma↩ I&*	prior inverse-covariance

Reimplemented from BetaGrpFt.

Reimplemented in BetaGrpPCpex.

7.7.4 Member Data Documentation

7.7.4.1 _fittedAllAdj

```
gsl_matrix* BetaGrpPEX::_fittedAllAdj  [protected]
```

Adjusted fitted matrix.

The $\boldsymbol{X\Xi A}$ matrix. The inherited _fittedAll matrix is $\boldsymbol{X\Xi}$, i.e. the "raw" version.

7.7.4.2 _ftA

```
vector<vector<double> > BetaGrpPEX::_ftA  [protected]
```

Fitted values.

Vector of vectorized individual fitted matrices $\boldsymbol{\Xi}_{\cdot -m}\boldsymbol{A}_{-m\cdot}$.

7.7.4.3 _tSigIAt

```
gsl_matrix* BetaGrpPEX::_tSigIAt  [protected]
```

Scaled inverse-covariance.

The matrix $\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^T\right)^T$ that is common for sampling each row of the value matrix.

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

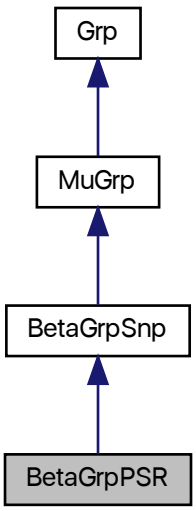
- MuGen.h
- MuGen.cpp

7.8 BetaGrpPSR Class Reference

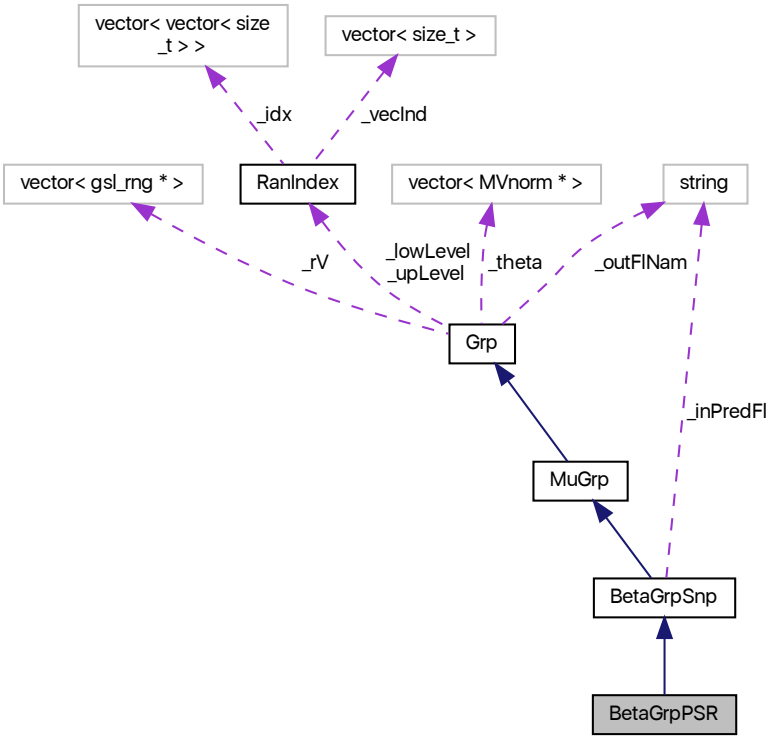
Single-SNP regression with partial effects.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpPSR:



Collaboration diagram for BetaGrpPSR:



Public Member Functions

- BetaGrpPSR ()

  Default constructor.
- BetaGrpPSR (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with no replication and p-values.
- BetaGrpPSR (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with replication and p-values.
- BetaGrpPSR (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with no replication and ABF.
- BetaGrpPSR (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with replication and ABF.
- ~BetaGrpPSR ()

  Destructor.
- BetaGrpPSR (const BetaGrpPSR &mG)

Copy constructor.

- BetaGrpPSR & operator= (const BetaGrpPSR &mG)

  Assignment operator.

- void dump ()

  Dump results to the output file.

Additional Inherited Members

7.8.1 Detailed Description

Single-SNP regression with partial effects.

Implements single-marker GWAS. Each trait is treated as a single response, and the other traits are added to the SNP as predictors. Only the SNP test is reported. No Hotelling-type mutivariate association test is performed, so the number of columns in the results matrix is the same as the number of traits. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 BetaGrpPSR() [1/5]

```
BetaGrpPSR::BetaGrpPSR (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )  [inline]
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads

7.8.2.2 BetaGrpPSR() [2/5]

```
BetaGrpPSR::BetaGrpPSR (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )  [inline]
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads

7.8.2.3 BetaGrpPSR() [3/5]

```
BetaGrpPSR::BetaGrpPSR (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )  [inline]
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name

Parameters

in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.8.2.4 BetaGrpPSR() [4/5]

```
BetaGrpPSR::BetaGrpPSR (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )  [inline]
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.8.2.5 BetaGrpPSR() [5/5]

```
BetaGrpPSR::BetaGrpPSR (
            const BetaGrpPSR & mG )
```

Copy constructor.

Parameters

in	*BetaGrpPSR&*	object to be copied

7.8.3 Member Function Documentation

7.8.3.1 dump()

```
void BetaGrpPSR::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from BetaGrpSnp.

7.8.3.2 operator=()

```
BetaGrpPSR & BetaGrpPSR::operator= (
            const BetaGrpPSR & mG )
```

Assignment operator.

Parameters

in	*BetaGrpPSR&*	object to be copied

Returns

BetaGrpPSR& target object

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

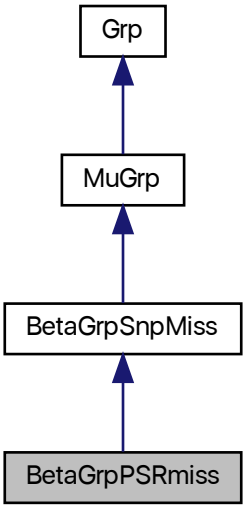
- MuGen.h
- MuGen.cpp

7.9 BetaGrpPSRmiss Class Reference

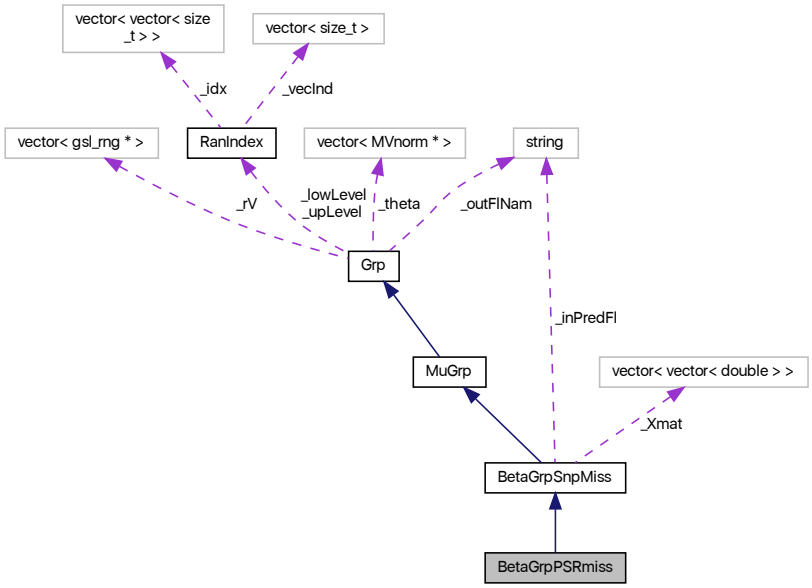
Single-SNP regression with partial effects and missing data.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpPSRmiss:



Collaboration diagram for BetaGrpPSRmiss:



Public Member Functions

- BetaGrpPSRmiss ()

  Default constructor.
- BetaGrpPSRmiss (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with no replication and p-values.
- BetaGrpPSRmiss (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with replication and p-values.
- BetaGrpPSRmiss (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with no replication and ABF.
- BetaGrpPSRmiss (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with replication and ABF.
- ~BetaGrpPSRmiss ()

  Destructor.
- BetaGrpPSRmiss (const BetaGrpPSRmiss &mG)

  Copy constructor.
- BetaGrpPSRmiss & operator= (const BetaGrpPSRmiss &mG)

  Assignment operator.
- void dump ()

  Dump results to the output file.

Additional Inherited Members

7.9.1 Detailed Description

Single-SNP regression with partial effects and missing data.

Implements single-marker GWAS with missing genotype data. Rows with missing genotypes are dropped. Each trait is treated as a single response, and the other traits are added to the SNP as predictors. Only the SNP test is reported. No Hotelling-type mutivariate association test is performed, so the number of columns in the results matrix is the same as the number of traits. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 BetaGrpPSRmiss() [1/5]

```
BetaGrpPSRmiss::BetaGrpPSRmiss (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )  [inline]
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.9.2.2 BetaGrpPSRmiss() [2/5]

```
BetaGrpPSRmiss::BetaGrpPSRmiss (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )  [inline]
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.9.2.3 BetaGrpPSRmiss() [3/5]

```
BetaGrpPSRmiss::BetaGrpPSRmiss (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )  [inline]
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name

Parameters

in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.9.2.4 BetaGrpPSRmiss() [4/5]

```
BetaGrpPSRmiss::BetaGrpPSRmiss (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )  [inline]
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.9.2.5 BetaGrpPSRmiss() [5/5]

```
BetaGrpPSRmiss::BetaGrpPSRmiss (
            const BetaGrpPSRmiss & mG )
```

Copy constructor.

Parameters

in	*BetaGrpPSRmiss&*	object to be copied

7.9.3 Member Function Documentation

7.9.3.1 dump()

```
void BetaGrpPSRmiss::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from BetaGrpSnpMiss.

7.9.3.2 operator=()

```
BetaGrpPSRmiss & BetaGrpPSRmiss::operator= (
            const BetaGrpPSRmiss & mG )
```

Assignment operator.

Parameters

in	*BetaGrpPSRmiss&*	object to be copied

Returns

BetaGrpPSRmiss& target object

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

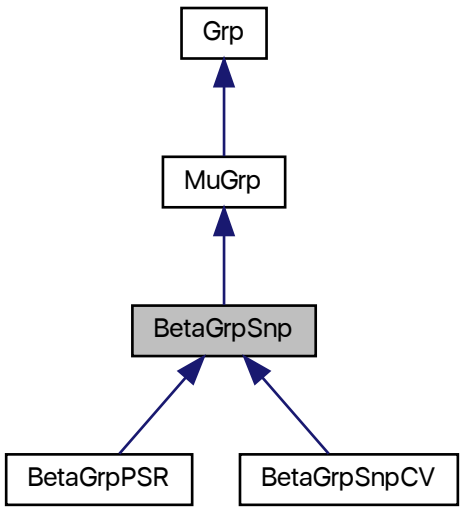
- MuGen.h
- MuGen.cpp

7.10 BetaGrpSnp Class Reference

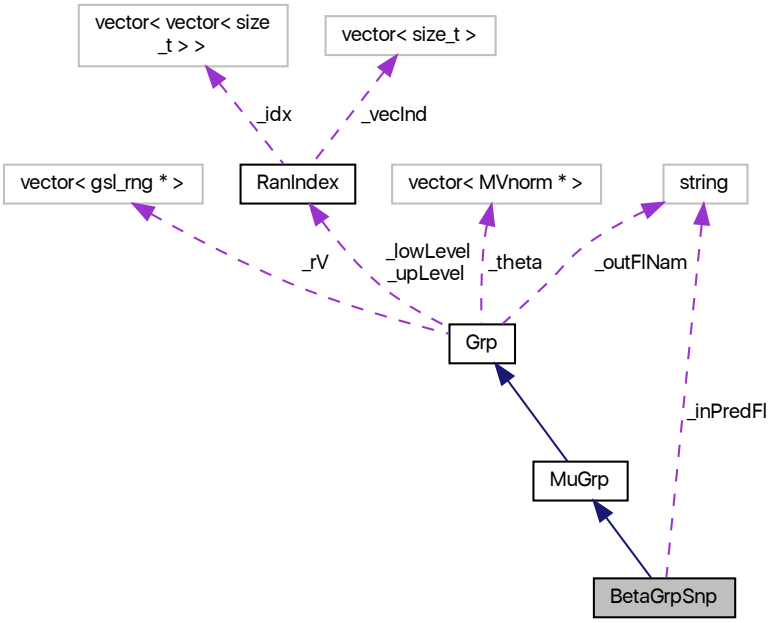
Simple single-SNP regression class.

`#include <MuGen.h>`

Inheritance diagram for BetaGrpSnp:



Collaboration diagram for BetaGrpSnp:



Public Member Functions

- BetaGrpSnp ()

  Default constructor.
- BetaGrpSnp (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with no replication and p-values.
- BetaGrpSnp (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with replication and p-values.
- BetaGrpSnp (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with no replication and ABF.
- BetaGrpSnp (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with replication and ABF.
- ~BetaGrpSnp ()

  Destructor.
- virtual void dump ()

  Dump results to the output file.
- BetaGrpSnp (const BetaGrpSnp &mG)

  Copy constructor.

- BetaGrpSnp & operator= (const BetaGrpSnp &mG)

  Assignment operator.
- const gsl_matrix * fMat () const

  Access adjusted fitted value matrix.
- void update (const Grp &dat, const SigmaI &SigIm)

  Response update function.

Protected Attributes

- gsl_matrix * _Xmat

  SNP predictor matrix.
- gsl_matrix * _fakeFmat

  Matrix for addition operators.
- gsl_matrix * _Ystore

  Response storage.
- size_t _Npred

  Number of predictors (SNPs)
- int _nThr

  Number of threads.
- double _numSaves

  Number of saves.
- double _priorVar

  Prior variance.
- string _inPredFl

  Predictor (SNP) file name.

Additional Inherited Members

7.10.1 Detailed Description

Simple single-SNP regression class.

Implements single-marker GWAS. Each trait is treated separately, including the variances (i.e., $\widehat{\sigma}_p^2 = \widehat{\mathbf{\Sigma}}_{p,p}$). However, in addition to the single-trait tests an extra Hotelling-type mutivariate association test is performed. This multivariate test reflects the distance of the whole vector of trait effects to zero, potentially increasing the power to detect pleiotropic SNPs with moderate effects on mutiple traits. The saved value matrix thus has one extra column for this test. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 BetaGrpSnp() [1/5]

```
BetaGrpSnp::BetaGrpSnp (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads

7.10.2.2 BetaGrpSnp() [2/5]

```
BetaGrpSnp::BetaGrpSnp (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads

7.10.2.3 BetaGrpSnp() [3/5]

```
BetaGrpSnp::BetaGrpSnp (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.10.2.4 BetaGrpSnp() [4/5]

```
BetaGrpSnp::BetaGrpSnp (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.10.2.5 BetaGrpSnp() [5/5]

```
BetaGrpSnp::BetaGrpSnp (
            const BetaGrpSnp & mG )
```

Copy constructor.

Parameters

in	*BetaGrpSnp&*	object to be copied

7.10.3 Member Function Documentation

7.10.3.1 dump()

```
void BetaGrpSnp::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from Grp.

Reimplemented in BetaGrpPSR, and BetaGrpSnpCV.

7.10.3.2 fMat()

```
const gsl_matrix* BetaGrpSnp::fMat ( ) const  [inline], [virtual]
```

Access adjusted fitted value matrix.

Returns

gsl_matrix* pointer to a zero-valued matrix

Reimplemented from Grp.

7.10.3.3 operator=()

```
BetaGrpSnp & BetaGrpSnp::operator= (
            const BetaGrpSnp & mG )
```

Assignment operator.

Parameters

in	*BetaGrpSnp&*	object to be copied

Returns

BetaGrpSnp& target object

7.10.4 Member Data Documentation

7.10.4.1 _fakeFmat

```
gsl_matrix* BetaGrpSnp::_fakeFmat  [protected]
```

Matrix for addition operators.

This matrix is addressed by fMat() and set to all zero values, so that if this object is in an addition/subtraction operator things don't break. Because this object operates outside of the regular Markov chains, on the residuals of the model, it ordinarily makes no sense to add or subtract it from anything.

7.10.4.2 _numSaves

```
double BetaGrpSnp::_numSaves  [protected]
```

Number of saves.

Number of times the response samples have been saved.

7.10.4.3 _priorVar

```
double BetaGrpSnp::_priorVar  [protected]
```

Prior variance.

Prior variance for Wakefield's **[wakefield07]** ABF calculation (Wakefield's notation for it is W). If it zero, $-\log_{10} p$ are calculated.

7.10.4.4 _Xmat

```
gsl_matrix* BetaGrpSnp::_Xmat  [protected]
```

SNP predictor matrix.

Read in only at the end of the run in the dump() function.

7.10.4.5 _Ystore

```
gsl_matrix* BetaGrpSnp::_Ystore  [protected]
```

Response storage.

MCMC samples from a response variable (representing a residual of the model) are stored here and divided by the total number of saves at the end before the regression is performed.

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

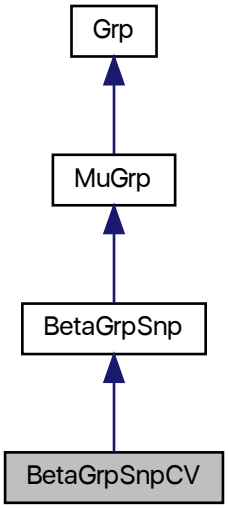
- MuGen.h
- MuGen.cpp

7.11 BetaGrpSnpCV Class Reference

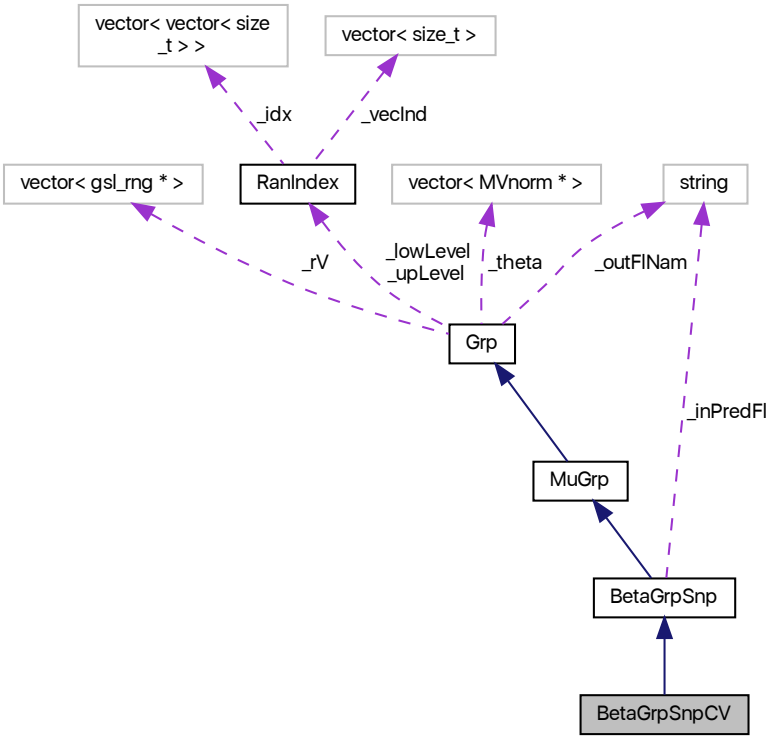
Single-SNP regression with conditional variance.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpSnpCV:



Collaboration diagram for BetaGrpSnpCV:



Public Member Functions

- BetaGrpSnpCV ()

  Default constructor.
- BetaGrpSnpCV (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with no replication and p-values.
- BetaGrpSnpCV (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr)

  Constructor with replication and p-values.
- BetaGrpSnpCV (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with no replication and ABF.
- BetaGrpSnpCV (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar)

  Constructor with replication and ABF.
- ~BetaGrpSnpCV ()

  Destructor.
- BetaGrpSnpCV (const BetaGrpSnpCV &mG)

Copy constructor.

- BetaGrpSnpCV & operator= (const BetaGrpSnpCV &mG)

  Assignment operator.

- void dump ()

  Dump results to the output file.

Additional Inherited Members

7.11.1 Detailed Description

Single-SNP regression with conditional variance.

Implements single-marker GWAS. Each trait is treated separately for coefficient calculation. In contrast, the variance estimates are taken from the inverse-covariance: $\widehat{\sigma}_p^2 = \left[\widehat{\mathbf{\Sigma}}_{p,p}^{-1}\right]^{-1} \leq \widehat{\mathbf{\Sigma}}_{p,p}$. In addition to the single-trait tests an extra Hotelling-type mutivariate association test is performed. This multivariate test reflects the distance of the whole vector of trait effects to zero, potentially increasing the power to detect pleiotropic SNPs with moderate effects on mutiple traits. The saved value matrix thus has one extra column for this test. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 BetaGrpSnpCV() [1/5]

```
BetaGrpSnpCV::BetaGrpSnpCV (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )  [inline]
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits

7.11.2.2 BetaGrpSnpCV() [2/5]

```
BetaGrpSnpCV::BetaGrpSnpCV (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr )  [inline]
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads

7.11.2.3 BetaGrpSnpCV() [3/5]

```
BetaGrpSnpCV::BetaGrpSnpCV (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )  [inline]
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name

Parameters

in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.11.2.4 BetaGrpSnpCV() [4/5]

```
BetaGrpSnpCV::BetaGrpSnpCV (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar )  [inline]
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance

7.11.2.5 BetaGrpSnpCV() [5/5]

```
BetaGrpSnpCV::BetaGrpSnpCV (
            const BetaGrpSnpCV & mG )
```

Copy constructor.

Parameters

in	*BetaGrpSnpCV&*	object to be copied

7.11.3 Member Function Documentation

7.11.3.1 dump()

```
void BetaGrpSnpCV::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from BetaGrpSnp.

7.11.3.2 operator=()

```
BetaGrpSnpCV & BetaGrpSnpCV::operator= (
            const BetaGrpSnpCV & mG )
```

Assignment operator.

Parameters

in	*BetaGrpSnp&*	object to be copied

Returns

BetaGrpSnp& target object

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

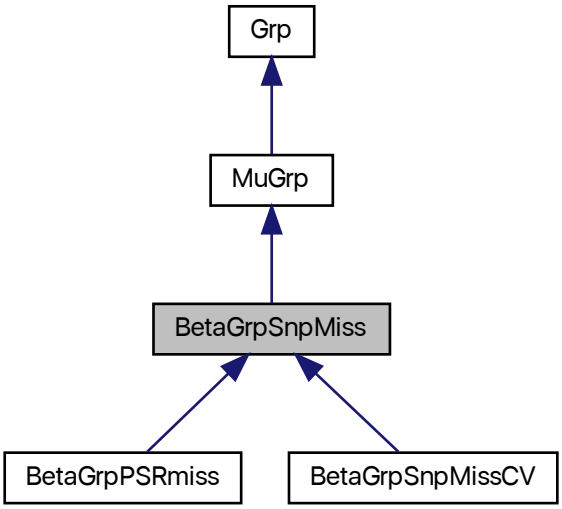
- MuGen.h
- MuGen.cpp

7.12 BetaGrpSnpMiss Class Reference

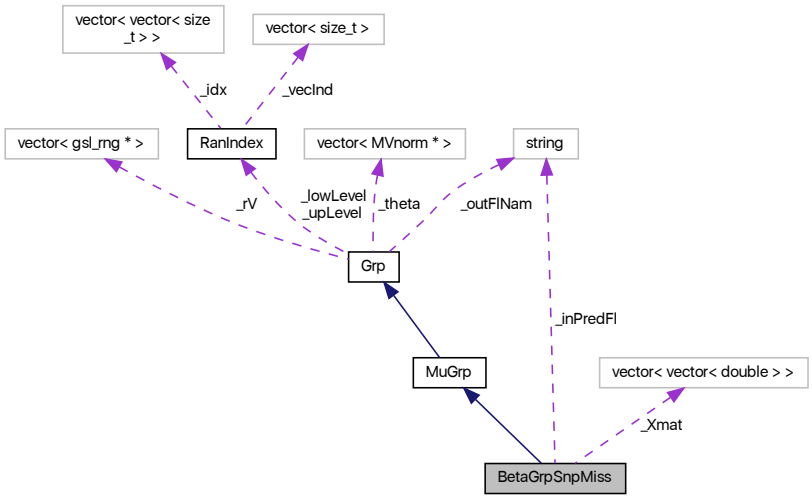
Simple single-SNP regression class with missing data.

`#include <MuGen.h>`

Inheritance diagram for BetaGrpSnpMiss:



Collaboration diagram for BetaGrpSnpMiss:



Public Member Functions

- BetaGrpSnpMiss ()

  Default constructor.
- BetaGrpSnpMiss (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with no replication and p-values.
- BetaGrpSnpMiss (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with replication and p-values.
- BetaGrpSnpMiss (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with no replication and ABF.
- BetaGrpSnpMiss (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with replication and ABF.
- ~BetaGrpSnpMiss ()

  Destructor.
- BetaGrpSnpMiss (const BetaGrpSnpMiss &mG)

  Copy constructor.
- BetaGrpSnpMiss & operator= (const BetaGrpSnpMiss &mG)

  Assignment operator.
- virtual void dump ()

  Dump results to the output file.
- const gsl_matrix * fMat () const

  Access adjusted fitted value matrix.
- void update (const Grp &dat, const SigmaI &SigIm)

  Response update function.

Protected Attributes

- vector< vector< double > > _Xmat

  Predictor array.
- gsl_matrix * _fakeFmat

  Matrix for addition operators.
- gsl_matrix * _Ystore

  Response storage.
- size_t _Npred

  Number of predictors (SNPs)
- int _nThr

  Number of threads to use.
- double _numSaves

  Number of saves.
- double _absLab

  Missing value label.
- double _priorVar

  Prior variance.
- string _inPredFl

  Predictor (SNP) file name.

Additional Inherited Members

7.12.1 Detailed Description

Simple single-SNP regression class with missing data.

Implements single-marker GWAS with missing genotypes. Rows with missing genotypes are dropped. Each trait is treated separately, including the variances (i.e., $\widehat{\sigma}_p^2 = \widehat{\mathbf{\Sigma}}_{p,p}$). However, in addition to the single-trait tests an extra Hotelling-type mutivariate association test is performed. This multivariate test reflects the distance of the whole vector of trait effects to zero, potentially increasing the power to detect pleiotropic SNPs with moderate effects on mutiple traits. The saved value matrix thus has one extra column for this test. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 BetaGrpSnpMiss() [1/5]

```
BetaGrpSnpMiss::BetaGrpSnpMiss (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.12.2.2 BetaGrpSnpMiss() [2/5]

```
BetaGrpSnpMiss::BetaGrpSnpMiss (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.12.2.3 BetaGrpSnpMiss() [3/5]

```
BetaGrpSnpMiss::BetaGrpSnpMiss (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name

Parameters

in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.12.2.4 BetaGrpSnpMiss() [4/5]

```
BetaGrpSnpMiss::BetaGrpSnpMiss (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.12.2.5 BetaGrpSnpMiss() [5/5]

```
BetaGrpSnpMiss::BetaGrpSnpMiss (
            const BetaGrpSnpMiss & mG )
```

Copy constructor.

Parameters

in	*BetaGrpSnpMiss&*	object to be copied

7.12.3 Member Function Documentation

7.12.3.1 dump()

```
void BetaGrpSnpMiss::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from Grp.

Reimplemented in BetaGrpPSRmiss, and BetaGrpSnpMissCV.

7.12.3.2 fMat()

```
const gsl_matrix* BetaGrpSnpMiss::fMat ( ) const  [inline], [virtual]
```

Access adjusted fitted value matrix.

Returns

gsl_matrix∗ pointer to a zero-valued matrix

Reimplemented from Grp.

7.12.3.3 operator=()

```
BetaGrpSnpMiss & BetaGrpSnpMiss::operator= (
            const BetaGrpSnpMiss & mG )
```

Assignment operator.

Parameters

in	*BetaGrpSnpMiss&*	object to be copied

Returns

BetaGrpSnpMiss& target object

7.12.4 Member Data Documentation

7.12.4.1 _absLab

```
double BetaGrpSnpMiss::_absLab  [protected]
```

Missing value label.

Value that labels the missing genotypes (predictors). Provided by the user.

7.12.4.2 _fakeFmat

```
gsl_matrix* BetaGrpSnpMiss::_fakeFmat  [protected]
```

Matrix for addition operators.

This matrix is addressed by fMat() and set to all zero values, so that if this object is in an addition/subtraction operator things don't break. Because this object operates outside of the regular Markov chains, on the residuals of the model, it ordinarily makes no sense to add or subtract it from anything.

7.12.4.3 _numSaves

```
double BetaGrpSnpMiss::_numSaves  [protected]
```

Number of saves.

Number of times the response samples have been saved.

7.12.4.4 _priorVar

```
double BetaGrpSnpMiss::_priorVar  [protected]
```

Prior variance.

Prior variance for Wakefield's **[wakefield07]** ABF calculation (Wakefield's notation for it is W). If it zero, $-\log_{10} p$ are calculated.

7.12.4.5 _Xmat

```
vector< vector<double> > BetaGrpSnpMiss::_Xmat  [protected]
```

Predictor array.

The outer vector is the same length as the number of predictors. Each constituent vector stores only the non-missing genotypes.

7.12.4.6 _Ystore

```
gsl_matrix* BetaGrpSnpMiss::_Ystore  [protected]
```

Response storage.

MCMC samples from a response variable (representing a residual of the model) are stored here and divided by the total number of saves at the end before the regression is performed.

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

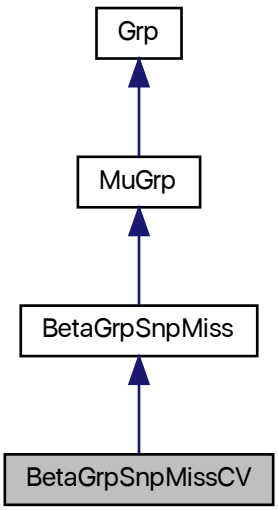
- MuGen.h
- MuGen.cpp

7.13 BetaGrpSnpMissCV Class Reference

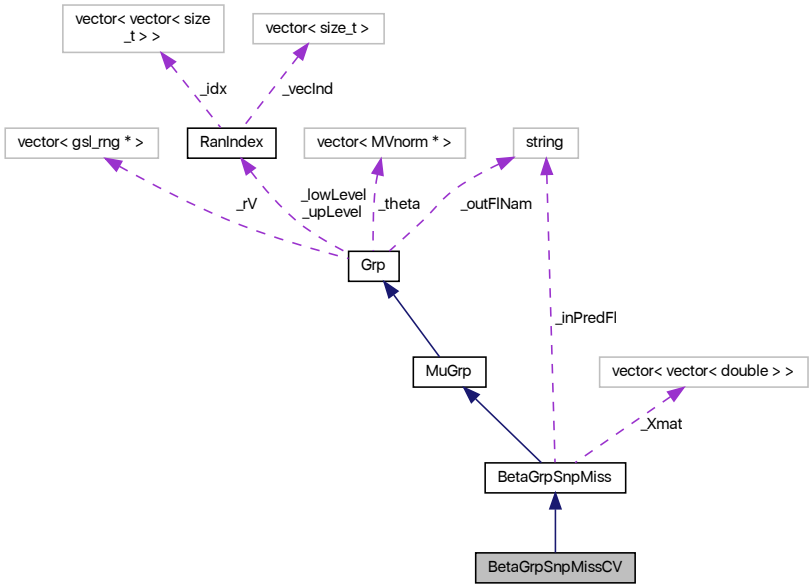
Single-SNP regression with conditional variance and missing data.

```
#include <MuGen.h>
```

Inheritance diagram for BetaGrpSnpMissCV:



Collaboration diagram for BetaGrpSnpMissCV:



Public Member Functions

- BetaGrpSnpMissCV ()

  Default constructor.
- BetaGrpSnpMissCV (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with no replication and p-values.
- BetaGrpSnpMissCV (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &absLab)

  Constructor with replication and p-values.
- BetaGrpSnpMissCV (const string &predFlNam, const string &outFlNam, const size_t &Ndat, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with no replication and ABF.
- BetaGrpSnpMissCV (const string &predFlNam, const string &outFlNam, RanIndex &low, const size_t &Npred, const size_t &d, const int &Nthr, const double &prVar, const double &absLab)

  Constructor with replication and ABF.
- ~BetaGrpSnpMissCV ()

  Destructor.
- BetaGrpSnpMissCV (const BetaGrpSnpMissCV &mG)

  Copy constructor.
- BetaGrpSnpMissCV & operator= (const BetaGrpSnpMissCV &mG)

  Assignment operator.
- void dump ()

  Dump results to the output file.

Additional Inherited Members

7.13.1 Detailed Description

Single-SNP regression with conditional variance and missing data.

Implements single-marker GWAS with missing genotype data. Rows with missing genotypes are dropped. Each trait is treated separately for coefficient calculation. In contrast, the variance estimates are taken from the inverse-covariance: $\widehat{\sigma}_p^2 = \left[\widehat{\mathbf{\Sigma}}_{p,p}^{-1}\right]^{-1} \leq \widehat{\mathbf{\Sigma}}_{p,p}$. In addition to the single-trait tests an extra Hotelling-type mutivariate association test is performed. This multivariate test reflects the distance of the whole vector of trait effects to zero, potentially increasing the power to detect pleiotropic SNPs with moderate effects on mutiple traits. The saved value matrix thus has one extra column for this test. The output is either $-\log_{10} p$, (although this statistic is not strictly a frequentist p-value, it performs very similarly in simulations), or Wakefield's **[wakefield07]** approximation of $-lnBF$ (log-Bayes factor ratio). In the latter case, the user can set the prior variance manually. The SNP regression is performed on the point estimate of a response, as described in documentation of the update() and dump() functions. The latter can be, say, a residual of a mixed-model type GEBV estimate done to control population structure.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 BetaGrpSnpMissCV() [1/5]

```
BetaGrpSnpMissCV::BetaGrpSnpMissCV (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )  [inline]
```

Constructor with no replication and p-values.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.13.2.2 BetaGrpSnpMissCV() [2/5]

```
BetaGrpSnpMissCV::BetaGrpSnpMissCV (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & absLab )  [inline]
```

Constructor with replication and p-values.

Initializes the object with replication, encoded with the given index, and leaves the prior variance for SNP regressions at zero, thereby resulting in a dump of $-\log_{10} p$ values.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	missing value label

7.13.2.3 BetaGrpSnpMissCV() [3/5]

```
BetaGrpSnpMissCV::BetaGrpSnpMissCV (
            const string & predFlNam,
            const string & outFlNam,
            const size_t & Ndat,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )  [inline]
```

Constructor with no replication and ABF.

Initializes the object with no replication (i.e., the number of rows in the predictor is the same as in the response) and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*size_t&*	number of rows in the response matrix
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.13.2.4 BetaGrpSnpMissCV() [4/5]

```
BetaGrpSnpMissCV::BetaGrpSnpMissCV (
            const string & predFlNam,
            const string & outFlNam,
            RanIndex & low,
            const size_t & Npred,
            const size_t & d,
            const int & Nthr,
            const double & prVar,
            const double & absLab )  [inline]
```

Constructor with replication and ABF.

Initializes the object with replication, encoded with the given index, and sets the prior variance for SNP regressions to the given value, therefore dumping Wakefield's log of approximate Bayes factor (ABF) ratios to the output file.

Parameters

in	*string&*	predictor file name
in	*string&*	output file name
in	*RanIndex&*	replicate index
in	*size_t&*	number of predictors
in	*size_t&*	number of traits
in	*int&*	number of threads
in	*double&*	prior variance
in	*double&*	missing value label

7.13.2.5 BetaGrpSnpMissCV() [5/5]

```
BetaGrpSnpMissCV::BetaGrpSnpMissCV (
            const BetaGrpSnpMissCV & mG )
```

Copy constructor.

Parameters

in	*BetaGrpSnpMissCV&*	object to be copied

7.13.3 Member Function Documentation

7.13.3.1 dump()

```
void BetaGrpSnpMissCV::dump ( )  [virtual]
```

Dump results to the output file.

The predictor (SNP) file is read by this function, and the single-predictor regression (i.e., each predictor is tested separately, ignoring all others) is performed. The results are saved to the pre-specified output file.

Reimplemented from BetaGrpSnpMiss.

7.13.3.2 operator=()

```
BetaGrpSnpMissCV & BetaGrpSnpMissCV::operator= (
            const BetaGrpSnpMissCV & mG )
```

Assignment operator.

Parameters

in	*BetaGrpSnpMissCV&*	object to be copied

Returns

BetaGrpSnpMissCV& target object

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

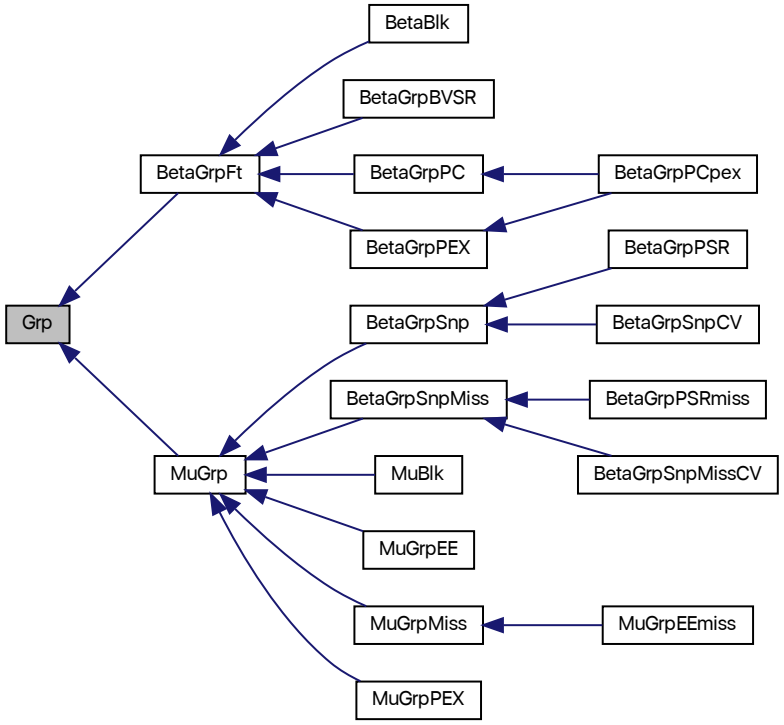
- MuGen.h
- MuGen.cpp

7.14 Grp Class Reference

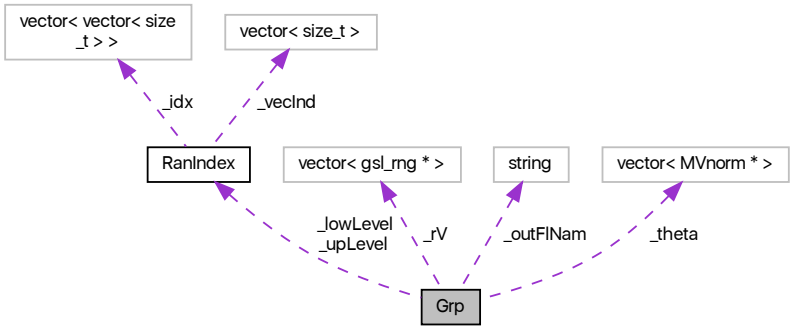
Base location parameter group class.

`#include <MuGen.h>`

Inheritance diagram for Grp:



Collaboration diagram for Grp:



Public Member Functions

- virtual ∼Grp ()

  Destructor.
- virtual void update (const Grp &, const SigmaI &)=0

  Gaussian likelihood, improper prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &)=0

  Student-t likelihood, improper prior.
- virtual void update (const Grp &, const SigmaI &, const SigmaI &)=0

  Gaussian likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const SigmaI &)=0

  Student-t likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &, const SigmaI &, const Qgrp &, const SigmaI &)=0

  Gaussian likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const Qgrp &, const SigmaI &)=0

  Student-t likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &, const SigmaI &, const Grp &, const SigmaI &)=0

  Gaussian likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const SigmaI &)=0

  Student-t likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &, const SigmaI &, const Grp &, const Qgrp &, const SigmaI &)=0

  Gaussian likelihood, non-zero mean Student-t prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const Qgrp &, const SigmaI &)=0

  Student-t likelihood, non-zero mean Student-t prior.
- virtual void save ()

  Save to pre-specified file.
- virtual void save (const string &outFlNam)

  Save to file.
- virtual void save (const string &outMuFlNam, const string &outSigFlNam, const SigmaI &SigI)

  Joint save.
- virtual void save (const SigmaI &SigI)

  Save with inverse-covariance.
- virtual void save (const Grp &y, const SigmaI &SigI)

  Save with data and inverse-covariance.
- void mhlSave (const string &outFlNam, const SigmaI SigI)

  Save Mahalanobis distance.
- virtual void dump ()

  Dump to a file.
- const vector< MVnorm * > & dataVec () const

  Get vector of row pointers.
- virtual const gsl_matrix * dMat () const

  Access the value matrix.
- virtual const gsl_matrix * fMat () const

  Access the value matrix.
- const size_t Ndata () const

  Get number of rows.
- const size_t phenD () const

Get number of traits.

- virtual double lnOddsRat (const Grp &y, const Sigmal &Sigl, const size_t i) const

  Log-odds ratio.

- const MVnorm * operator[ ] (const size_t i) const

  Subscript operator.

- MVnorm * operator[ ] (const size_t i)

  Subscript operator.

- virtual MuGrp mean (RanIndex &grp)

  Group mean.

- virtual const MuGrp mean (RanIndex &grp) const

  Group mean.

- virtual MuGrp mean (RanIndex &grp, const Qgrp &q)

  Group weighted mean.

- virtual const MuGrp mean (RanIndex &grp, const Qgrp &q) const

  Group weighted mean.

- void center ()

  Center the value matrix.

Protected Member Functions

- Grp ()

Protected Attributes

- vector< MVnorm * > _theta

  Vector of pointers to value rows.

- gsl_matrix * _valueMat

  Value matrix.

- RanIndex * _lowLevel

  Lower level index.

- RanIndex * _upLevel

  Upper level index.

- vector< gsl_rng * > _rV

  Vector of PNG pointers.

- string _outFlNam

  Name of the output file.

Friends

- class **MuGrp**
- MuGrp operator+ (const Grp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const MuGrp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const Grp &m1, const MuGrp &m2)

  Addition operator.
- MuGrp operator- (const Grp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const MuGrp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const Grp &m1, const MuGrp &m2)

  Subtraction operator.

7.14.1 Detailed Description

Base location parameter group class.

The abstract base location parameter group class. Cannot be initialized directly.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Grp()

```
Grp::Grp ( )  [protected]
```

Default constructor

7.14.3 Member Function Documentation

7.14.3.1 center()

```
void Grp::center ( )  [inline]
```

Center the value matrix.

Centering is performed by subtracting the mean for each column from each column in place.

7.14.3.2 dataVec()

```
const vector<MVnorm *>& Grp::dataVec ( ) const  [inline]
```

Get vector of row pointers.

Returns

vector<MVnorm *> vector of objects addressing rows of the value matrix

7.14.3.3 dMat()

```
virtual const gsl_matrix* Grp::dMat ( ) const  [inline], [virtual]
```

Access the value matrix.

Gives access to the value matrix. The internal structure this points to depends on the derived class.

Returns

gsl_matrix* pointer to a GSL matrix

7.14.3.4 dump()

```
virtual void Grp::dump ( )  [inline], [virtual]
```

Dump to a file.

Dumps paramter values averaged over the MCMC run to a file in the end of the run. Has an effect only in derived classes where it is implemented. In these classes, the name of the file is pre-specified at initialization.

Reimplemented in BetaGrpPSRmiss, BetaGrpSnpMissCV, BetaGrpSnpMiss, BetaGrpPSR, BetaGrpSnpCV, BetaGrpSnp, and BetaGrpFt.

7.14.3.5 fMat()

```
virtual const gsl_matrix* Grp::fMat ( ) const  [inline], [virtual]
```

Access the value matrix.

Gives access to the value matrix. The internal structure this points to depends on the derived class, and may be different from dMat().

Returns

gsl_matrix* pointer to a GSL matrix

Reimplemented in MuBlk, BetaGrpSnpMiss, BetaGrpSnp, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, and MuGrpPEX.

7.14.3.6 lnOddsRat()

```
virtual double Grp::lnOddsRat (
            const Grp & y,
            const SigmaI & SigI,
            const size_t i ) const  [inline], [virtual]
```

Log-odds ratio.

Log-odds ration between models with and without the i-th regression coefficient (row). Makes sense only in regression classes, everywhere else returns zero.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*size_t*	index of the element (row)

Returns

double log-odds ratio

Reimplemented in BetaGrpFt.

7.14.3.7 mean() [1/4]

```
MuGrp Grp::mean (
            RanIndex & grp )  [virtual]
```

Group mean.

Calculates within-group mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix.

Parameters

in	*RanIndex&*	grouping index

Returns

MuGrp object with the matrix of means

Reimplemented in MuGrpPEX.

7.14.3.8 mean() [2/4]

```
const MuGrp Grp::mean (
            RanIndex & grp ) const  [virtual]
```

Group mean.

Calculates within-group mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix.

Parameters

in	*RanIndex&*	grouping index

Returns

MuGrp object with the matrix of means

Reimplemented in MuGrpPEX.

7.14.3.9 mean() [3/4]

```
MuGrp Grp::mean (
            RanIndex & grp,
            const Qgrp & q )  [virtual]
```

Group weighted mean.

Calculates within-group weighted mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix. The weights are in the provided Qgrp object. Typically, they are from a Student-t model.

Parameters

in	*RanIndex&*	grouping index
in	*Qgrp&*	weights

Returns

MuGrp object with the matrix of means

Reimplemented in MuGrpPEX.

7.14.3.10 mean() [4/4]

```
const MuGrp Grp::mean (
            RanIndex & grp,
            const Qgrp & q ) const  [virtual]
```

Group weighted mean.

Calculates within-group weighted mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix. The weights are in the provided Qgrp object. Typically, they are from a Student-t model.

Parameters

in	*RanIndex&*	grouping index
in	*Qgrp&*	weights

Returns

MuGrp object with the matrix of means

Reimplemented in MuGrpPEX.

7.14.3.11 mhlSave()

```
void Grp::mhlSave (
            const string & outFlNam,
            const SigmaI SigI )
```

Save Mahalanobis distance.

Saves Mahalanobis distances of each row to a vector of zeros, given the provided inverse-covariance.

Parameters

in	*string&*	file name
in	*SigmaI&*	inverse-covariance

7.14.3.12 Ndata()

```
const size_t Grp::Ndata ( ) const  [inline]
```

Get number of rows.

Returns

size_t number of rows in the value matrix, corresponds to the number of mean values or regression coefficients

7.14.3.13 operator[]() [1/2]

```
MVnorm* Grp::operator[] (
            const size_t i )  [inline]
```

Subscript operator.

Get a MVnorm object pointer to a row of the value matrix.

Parameters

in	*size_t*	row index

Returns

MVnorm* pointer to an element in the value matrix

7.14.3.14 operator[]() [2/2]

```
const MVnorm* Grp::operator[] (
            const size_t i ) const  [inline]
```

Subscript operator.

Get a MVnorm object pointer to a row of the value matrix.

Parameters

in	*size_t*	row index

Returns

MVnorm* pointer to an element in the value matrix

7.14.3.15 phenD()

```
const size_t Grp::phenD ( ) const  [inline]
```

Get number of traits.

Returns

size_t number of columns in the value matrix, corresponds to the number of traits

7.14.3.16 save() [1/5]

```
void Grp::save ( )  [virtual]
```

Save to pre-specified file.

Saves a sample from the Markov chain for the object by appending to a file. Unless re-implemented, the current contents of _valueMat are saved. File name is either generic or pre-specified at construction.

Reimplemented in BetaGrpPCpex, BetaGrpPEX, and MuGrpPEX.

7.14.3.17 save() [2/5]

```
virtual void Grp::save (
            const Grp & y,
            const SigmaI & SigI )  [inline], [virtual]
```

Save with data and inverse-covariance.

Reserved for saving to a variable rather than a file. Has an effect only in classes where it is re-implemented.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance

7.14.3.18 save() [3/5]

```
virtual void Grp::save (
            const SigmaI & SigI )  [inline], [virtual]
```

Save with inverse-covariance.

Reserved for saving to a variable rather than a file. Has an effect only in classes where it is re-implemented.

Parameters

in	*SigmaI&*	inverse-covariance

Reimplemented in BetaGrpBVSR, BetaGrpPCpex, BetaGrpPEX, BetaGrpFt, and MuGrpPEX.

7.14.3.19 save() [4/5]

```
void Grp::save (
            const string & outFlNam )  [virtual]
```

Save to file.

Saves a sample from the Markov chain for the object by appending to the named file. Unless re-implemented, the current contents of _valueMat are saved.

Parameters

in	*string&*	file name

Reimplemented in BetaGrpPCpex, BetaGrpPEX, and MuGrpPEX.

7.14.3.20 save() [5/5]

```
void Grp::save (
            const string & outMuFlNam,
            const string & outSigFlNam,
            const SigmaI & SigI )  [virtual]
```

Joint save.

Saves the current value of _valueMat to one file and the covariance matrix corresponding to the given inverse-covariance to the other. Both files are appended. Has more practical importance in some derived classes.

Parameters

in	*string&*	mean file name
in	*string&*	covariance file name
in	*SigmaI&*	inverse-covariance

7.14.4 Friends And Related Function Documentation

7.14.4.1 operator+ [1/3]

```
MuGrp operator+ (
            const Grp & m1,
            const Grp & m2 )  [friend]
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.14.4.2 operator+ [2/3]

```
MuGrp operator+ (
            const Grp & m1,
            const MuGrp & m2 )  [friend]
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*MuGrp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.14.4.3 operator+ [3/3]

```
MuGrp operator+ (
            const MuGrp & m1,
            const Grp & m2 )  [friend]
```

Addition operator.

Parameters

in	*MuGrp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.14.4.4 operator- [1/3]

```
MuGrp operator- (
            const Grp & m1,
            const Grp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

7.14.4.5 operator- [2/3]

```
MuGrp operator- (
            const Grp & m1,
            const MuGrp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*MuGrp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

7.14.4.6 operator- [3/3]

```
MuGrp operator- (
            const MuGrp & m1,
            const Grp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*MuGrp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

7.14.5 Member Data Documentation

7.14.5.1 _lowLevel

```
RanIndex* Grp::_lowLevel  [protected]
```

Lower level index.

If initialized, points to the lower (i.e., data) level in the hierarchy. Otherwise, set to 0. For regressions, plays the role of the design matrix Z.

7.14.5.2 _rV

```
vector<gsl_rng *> Grp::_rV  [protected]
```

Vector of PNG pointers.

Typically is of unit length. However, for derived classes that use multi-threading its length is equal to the number of threads, each thread having its own PNG. PNGS are seeded on construction with the sum of *time(NULL)* and RTDSC.

7.14.5.3 _theta

```
vector<MVnorm *> Grp::_theta  [protected]
```

Vector of pointers to value rows.

Each individual MVnorm pointer corresponds to a row in _valueMat, which can be modified by invoking update functions that MVnorm class members.

7.14.5.4 _upLevel

```
RanIndex* Grp::_upLevel  [protected]
```

Upper level index.

If initialized, points to the upper (i.e., prior) level in the hierarchy. Otherwise, set to 0.

7.14.5.5 _valueMat

```
gsl_matrix* Grp::_valueMat  [protected]
```

Value matrix.

Matrix of location parameter values. The columns are traits, and rows are individual replicates or regression coefficients corresponding to a given predictor. Each row is addressed by an individual element of _theta for most classes. Exceptions are noted in descriptions of those derived classes.

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

- MuGen.h
- MuGen.cpp

7.15 MixP Class Reference

Dirichlet-multinomial mixture prior.

```
#include <MuGen.h>
```

Public Member Functions

- MixP ()
  Default constructor.
- MixP (const vector< double > &initP)
  Constructor with a vector of proportions.
- MixP (const gsl_vector *initP)
  Constructor with a GSL vector of proportions.
- MixP (const vector< double > &initP, const double &a)
  Constructor with a vector of proportions and a single prior class size.
- MixP (const gsl_vector *initP, const double &a)
  Constructor with a GSL vector of proportions and a single prior class size.
- MixP (const vector< double > &initP, const double *a, const size_t &len)
  Constructor with a vector of proportions and an array of prior class sizes.
- MixP (const gsl_vector *initP, const double *a)
  Constructor with a GSL vector of proportions and an array of prior class sizes.
- MixP (const vector< double > &initP, const vector< double > &a)
  Constructor with a vector of proportions and a vector of prior class sizes.
- MixP (const gsl_vector *initP, const vector< double > &a)
  Constructor with a GSL vector of proportions and a vector of prior class sizes.
- MixP (const RanIndex &ind, const double &a)
  Constructor with a RanIndex of group IDs and a vector of prior class sizes.
- ~MixP ()
  Destructor.
- MixP (const MixP &)
  Copy constructor.
- MixP & operator= (const MixP &)
  Assignement operator.
- const double & operator[ ] (const size_t i) const
  Subscript operator.
- void update (const RanIndex &Nvec)
  Gibbs update function.

7.15.1 Detailed Description

Dirichlet-multinomial mixture prior.

Implements the Dirichlet-multinomial prior on proportions in a mixture of Gaussians.

Warning

This has not been extenisively tested

7.15.2 Constructor & Destructor Documentation

7.15.2.1 MixP() [1/10]

```
MixP::MixP (
            const vector< double > & initP )  [inline]
```

Constructor with a vector of proportions.

The prior is set to all ones, i.e. the vaguest possible.

Parameters

in	*vector<double>&*	vector of initial proportions

7.15.2.2 MixP() [2/10]

```
MixP::MixP (
            const gsl_vector * initP )
```

Constructor with a GSL vector of proportions.

The prior is set to all ones, i.e. the vaguest possible.

Parameters

in	*gsl_vector**	vector of initial proportions

7.15.2.3 MixP() [3/10]

```
MixP::MixP (
            const vector< double > & initP,
            const double & a )
```

Constructor with a vector of proportions and a single prior class size.

The prior for each mixture category is set to the same provided value.

Parameters

in	*vector<double>&*	vector of initial proportions
in	*double&*	prior category size

7.15.2.4 MixP() [4/10]

```
MixP::MixP (
            const gsl_vector * initP,
            const double & a )
```

Constructor with a GSL vector of proportions and a single prior class size.

The prior for each mixture category is set to the same provided value.

Parameters

in	*gsl_vector∗*	vector of initial proportions
in	*double&*	prior category size

7.15.2.5 MixP() [5/10]

```
MixP::MixP (
            const vector< double > & initP,
            const double * a,
            const size_t & len )
```

Constructor with a vector of proportions and an array of prior class sizes.

The prior for each mixture category is set to the corresponding value in the array.

Parameters

in	*vector<double>&*	vector of initial proportions
in	*double**	prior category size
in	*size_t&*	prior array length

7.15.2.6 MixP() [6/10]

```
MixP::MixP (
            const gsl_vector * initP,
            const double * a )
```

Constructor with a GSL vector of proportions and an array of prior class sizes.

The prior for each mixture category is set to the corresponding value in the array.

Parameters

in	*gsl_vector**	vector of initial proportions
in	*double**	prior category size

7.15.2.7 MixP() [7/10]

```
MixP::MixP (
            const vector< double > & initP,
            const vector< double > & a )
```

Constructor with a vector of proportions and a vector of prior class sizes.

The prior for each mixture category is set to the corresponding value in the vector.

Parameters

in	*vector<double>&*	vector of initial proportions
in	*vector<double>&*	prior category size

7.15.2.8 MixP() [8/10]

```
MixP::MixP (
```

```
            const gsl_vector * initP,
            const vector< double > & a )
```

Constructor with a GSL vector of proportions and a vector of prior class sizes.

The prior for each mixture category is set to the corresponding value in the vector.

Parameters

in	*gsl_vector∗*	vector of initial proportions
in	*vector<double>*	prior category size

7.15.2.9 MixP() [9/10]

```
MixP::MixP (
            const RanIndex & ind,
            const double & a )
```

Constructor with a RanIndex of group IDs and a vector of prior class sizes.

The prior for each mixture category is set to the same provided value. The proportions are calculated using the data in the RanIndex object.

Parameters

in	*gsl_vector∗*	vector of initial proportions
in	*double&*	prior category size

7.15.2.10 MixP() [10/10]

```
MixP::MixP (
            const MixP & mp )
```

Copy constructor.

Parameters

in	*MixP&*	object to be copied

7.15.3 Member Function Documentation

7.15.3.1 operator=()

```
MixP & MixP::operator= (
            const MixP & mp )
```

Assignement operator.

Parameters

in	*Mixp&*	object to be copied

Returns

MixP target object

7.15.3.2 operator[]()

```
const double& MixP::operator[] (
            const size_t i ) const  [inline]
```

Subscript operator.

Parameters

in	*size↩ _t&*	index of the category

Returns

double& proprtion of the elements in the given category

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

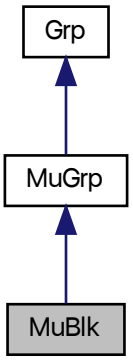
- MuGen.h
- MuGen.cpp

7.16 MuBlk Class Reference

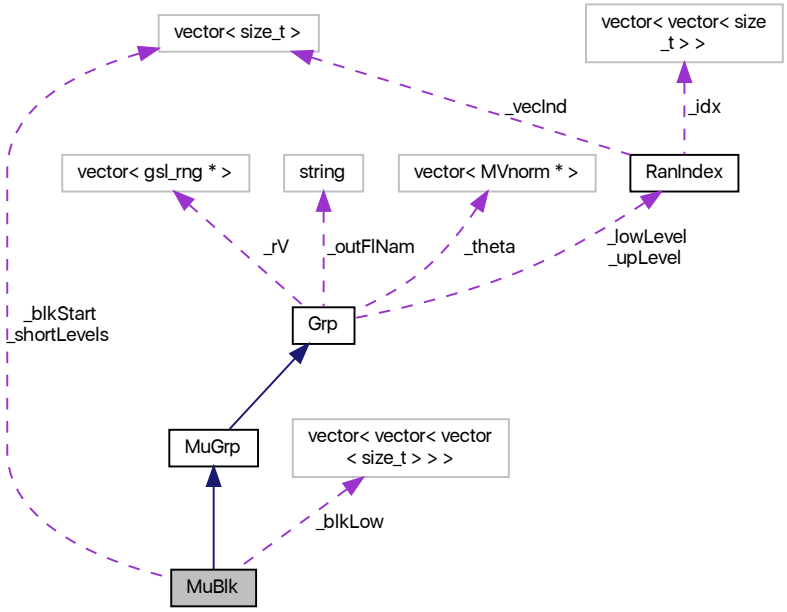
Hierarchical mean with independent blocks of traits.

`#include <MuGen.h>`

Inheritance diagram for MuBlk:



Collaboration diagram for MuBlk:



Public Member Functions

- MuBlk ()

  Default constructor.
- MuBlk (const Grp &dat, const string &lowIndFlName, const size_t &Nval, RanIndex &up, const string &blkInd↩ FileNam)

  Basic constructor.
- MuBlk (const Grp &dat, const string &lowIndFlName, const size_t &Nval, RanIndex &up, const string &outFlNam, const string &blkIndFileNam)

  Constructor with output file name.
- ∼MuBlk ()

  Destructor.
- const gsl_matrix ∗ fMat () const

  Access to the expanded value matrix.
- void update (const Grp &dat, const SigmaI &SigIm)

  Gaussian likelihood, improper prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm)

  Student-t likelihood, improper prior.
- void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Student-t prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Protected Member Functions

- void _updateExp ()

  Update the _expandedVM matrix.
- void _fillIn ()

  Fill in blocks with few levels.
- void _fillInUp ()

  Fill in blocks with few levels and a prior grouping.

Protected Attributes

- vector< size_t > _blkStart

  Block start indexes.
- vector< vector< vector< size_t > > > _blkLow

  Vector of low level indexes.
- gsl_matrix * _expandedVM

  Expanded _valueMat.
- vector< size_t > _shortLevels

  Blocks with few levels.

7.16.1 Detailed Description

Hierarchical mean with independent blocks of traits.

Blocks of inter-correlated traits with correlations among blocks set to exactly zero. These models arise, for example, when traits measured in different environments are modeled together as different traits. Replication structure and the number of replicates and levels (rows in the would-be _valueMat) can be different among blocks. The only limitation is that the traits belonging to the same block have to be in adjacent columns.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 MuBlk() [1/2]

```
MuBlk::MuBlk (
            const Grp & dat,
            const string & lowIndFlName,
            const size_t & Nval,
            RanIndex & up,
            const string & blkIndFileNam )
```

Basic constructor.

The provided number of values is the number of rows in the value matrix, i.e. the maximum number of levels of the lower level (data) index. The file that has the lower level index information should have a matrix of *int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the file with the low-level index
in	*size_t&*	maximum number of levels
in	*RanIndex&*	prior index
in	*string&*	name of the file with the block indexes

7.16.2.2 MuBlk() [2/2]

```
MuBlk::MuBlk (
            const Grp & dat,
            const string & lowIndFlName,
            const size_t & Nval,
            RanIndex & up,
            const string & outFlNam,
            const string & blkIndFileNam )
```

Constructor with output file name.

The provided number of values is the number of rows in the value matrix, i.e. the maximum number of levels of the lower level (data) index. The file that has the lower level index information should have a matrix of *int* with the number of rows the same as the number of rows in the data, and the number of columns equal to the number of blocks.

Parameters

in	*Grp&*	data for initialization
in	*string&*	name of the file with the low-level index
in	*size_t&*	maximum number of levels
in	*RanIndex&*	prior index
in	*string&*	output file name
in	*string&*	name of the file with the block indexes

7.16.3 Member Function Documentation

7.16.3.1 _fillIn()

```
void MuBlk::_fillIn ( )  [protected]
```

Fill in blocks with few levels.

For blocks with the number of levels smaller than the maximum, fill in the value matrix with the overall mean among levels. This insures that the overall mean is equal to what it would be if only the available rows were used.

7.16.3.2 _fillInUp()

```
void MuBlk::_fillInUp ( )  [protected]
```

Fill in blocks with few levels and a prior grouping.

The same as _fillIn(), but the means are calculated according to the grouping in the prior (upper) level in the hierarchy.

7.16.3.3 update() [1/10]

```
void MuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance

Reimplemented from MuGrp.

7.16.3.4 update() [2/10]

```
void MuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.5 update() [3/10]

```
void MuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.6 update() [4/10]

```
void MuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.7 update() [5/10]

```
void MuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.8 update() [6/10]

```
void MuBlk::update (
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance

Reimplemented from MuGrp.

7.16.3.9 update() [7/10]

```
void MuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.10 update() [8/10]

```
void MuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.11 update() [9/10]

```
void MuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.3.12 update() [10/10]

```
void MuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.16.4 Member Data Documentation

7.16.4.1 _blkLow

```
vector< vector< vector<size_t> > > MuBlk::_blkLow  [protected]
```

Vector of low level indexes.

Indexes to data, similar to _lowLevel. These are separate for each block, not necessarily the same number of levels or replicates in each.

7.16.4.2 _blkStart

```
vector< size_t > MuBlk::_blkStart  [protected]
```

Block start indexes.

Vector of indexes that indicate the column that starts each block.

7.16.4.3 _expandedVM

```
gsl_matrix* MuBlk::_expandedVM  [protected]
```

Expanded _valueMat.

Expanded value matrix, with the rows repeated according to _blkLow, separately for each block. To be used for arithmetic operators.

7.16.4.4 _shortLevels

```
vector<size_t> MuBlk::_shortLevels  [protected]
```

Blocks with few levels.

For cases where the number of levels of _blkLow differ among blocks, this vector stores the index of the first element past the available one.
For blocks that have the same number of levels as the largest among blocks, the value is zero. Each object has to have at least one zero in this vector, which is for the block with the largest number of levels.

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

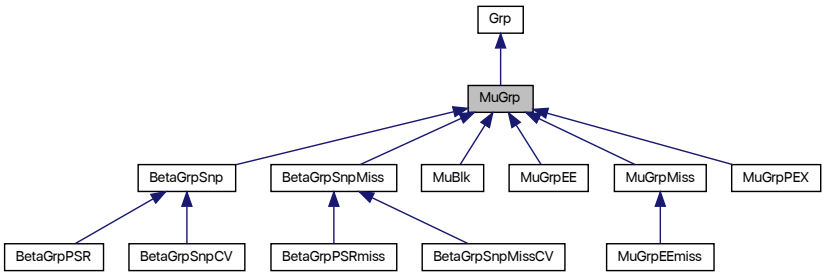
- MuGen.h
- MuGen.cpp

7.17 MuGrp Class Reference

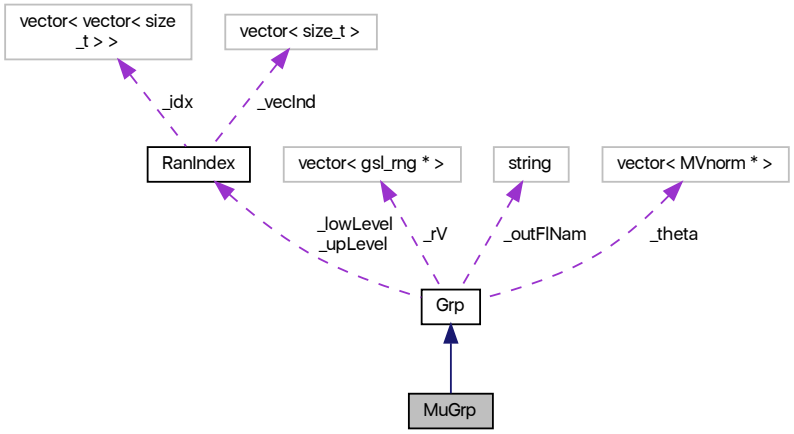
Hierarchical mean.

`#include <MuGen.h>`

Inheritance diagram for MuGrp:



Collaboration diagram for MuGrp:



Public Member Functions

- MuGrp ()

  Default constructor.
- MuGrp (RanIndex &low, const size_t &d)

  Deterministic zero-value constructor.
- MuGrp (const string &datFlNam, RanIndex &low, RanIndex &up, const size_t &d)

Constructor with data from file.

- MuGrp (const string &datFlNam, RanIndex &up, const size_t &d)

  Constructor with data from file and no lower level.
- MuGrp (const vector< MVnorm * > &dat, RanIndex &low, RanIndex &up)

  Constructor with a vector of MVnorm pointers.
- MuGrp (const Grp &dat, RanIndex &low, RanIndex &up)

  Constructor with a Grp object.
- MuGrp (const vector< MVnorm * > &dat, RanIndex &low, RanIndex &up, const string &outFlNam)

  Constructor with a vector of MVnorm pointers and output file name.
- MuGrp (const Grp &dat, RanIndex &low, RanIndex &up, const string &outFlNam)

  Constructor with a Grp object and output file name.
- MuGrp (const Grp &dat, RanIndex &low)

  Deterministic mean constructor.
- MuGrp (const Grp &dat, const Qgrp &q, RanIndex &low)

  Deterministic weighted mean constructor.
- MuGrp (const gsl_matrix *dat)

  Deterministic constructor with a GSL matrix.
- MuGrp (const gsl_matrix *dat, RanIndex &low)

  Deterministic GSL matrix mean constructor.
- MuGrp (const gsl_matrix *dat, const Qgrp &q, RanIndex &low)

  Deterministic GSL matrix weighted mean constructor.
- virtual ~MuGrp ()

  Destructor.
- MuGrp (const MuGrp &mG)

  Copy constructor.
- MuGrp (const Grp &g)

  Copy constructor.
- MuGrp & operator= (const MuGrp &mG)

  Assignemnt operator.
- virtual void update (const Grp &dat, const Sigmal &SigIm)

  Gaussian likelihood, improper prior.
- virtual void update (const Grp &dat, const Qgrp &q, const Sigmal &SigIm)

  Student-t likelihood, improper prior.
- virtual void update (const Grp &dat, const Sigmal &SigIm, const Sigmal &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const Sigmal &SigIm, const Sigmal &SigIp)

  Student-t likelihood, 0-mean Gaussian prior.
- virtual void update (const Grp &dat, const Sigmal &SigIm, const Qgrp &qPr, const Sigmal &SigIp)

  Gaussian likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const Sigmal &SigIm, const Qgrp &qPr, const Sigmal &SigIp)

  Student-t likelihood, 0-mean Student-t prior.
- virtual void update (const Grp &dat, const Sigmal &SigIm, const Grp &muPr, const Sigmal &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const Sigmal &SigIm, const Grp &muPr, const Sigmal &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.
- virtual void update (const Grp &dat, const Sigmal &SigIm, const Grp &muPr, const Qgrp &qPr, const Sigmal &SigIp)

Gaussian likelihood, non-zero mean Student-t prior.

- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Friends

- MuGrp operator+ (const Grp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const MuGrp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const Grp &m1, const MuGrp &m2)

  Addition operator.
- MuGrp operator- (const Grp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const MuGrp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const Grp &m1, const MuGrp &m2)

  Subtraction operator.

Additional Inherited Members

7.17.1 Detailed Description

Hierarchical mean.

Standard hierarchical mean parameter of Bayesian hierarchical models **[gelman04] [gelman07]** . Can also be used to store raw data (i.e., the lowest level in the hierarchy or the repsonse in a mixed model; in which case it is not updated), and to implement a Bayesian analog of a frequentist random effect (in which case the prior is zero).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 MuGrp() [1/14]

```
MuGrp::MuGrp (
            RanIndex & low,
            const size_t & d )
```

Deterministic zero-value constructor.

Sets all elements of the value matrix to zero. Can be used to construct a 0-mean prior. No upper level is set and thus this is not typically updated.

Parameters

in	*RanIndex&*	index to the lower level
in	*size_t&*	number of traits

7.17.2.2 MuGrp() [2/14]

```
MuGrp::MuGrp (
            const string & datFlNam,
            RanIndex & low,
            RanIndex & up,
            const size_t & d )
```

Constructor with data from file.

Reading in data from a file. The resulting value matrix has the number of rows equal to the number of groups in the low index and the number of elements in the upper index.
Index consistency is required, otherwise an error is issued and the program aborted. The constructor is deterministic.

Parameters

in	*string&*	data file name
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level
in	*size_t&*	number of traits

7.17.2.3 MuGrp() [3/14]

```
MuGrp::MuGrp (
            const string & datFlNam,
            RanIndex & up,
            const size_t & d )
```

Constructor with data from file and no lower level.

Reading in data from a file. The resulting value matrix has the number of rows equal to the number of elements in the upper index. This constructor is deterministic and can be used to set up the lowest (i.e. data) level of the hierarchy, and not update it.

Parameters

in	*string&*	data file name
in	*RanIndex&*	index to the upper (prior) level
in	*size_t&*	number of traits

7.17.2.4 MuGrp() [4/14]

```
MuGrp::MuGrp (
            const vector< MVnorm * > & dat,
            RanIndex & low,
            RanIndex & up )
```

Constructor with a vector of MVnorm pointers.

Initializes a value matrix with means among the elements of the vector of MVnorm pointers, according to the low index (i.e., the given vector contains the data). Values for the matrix are sampled around the mean values. The number returned by low.getNgrp() has to be equal to the number returned by up.getNtot(), and is equal to the number of rows in the value matrix. Violation of any of these conditions results in an error.

Parameters

in	*vector<MVnorm*	*>& vector of pointers to rows of the value matrix
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level

7.17.2.5 MuGrp() [5/14]

```
MuGrp::MuGrp (
            const Grp & dat,
            RanIndex & low,
            RanIndex & up )
```

Constructor with a Grp object.

Initializes a value matrix with means among the rows of the matrix in Grp addressed by the dMat() member, according to the low index (i.e., the Grp object contains the data). Values for the matrix are sampled around the mean values. The number returned by low.getNgrp() has to be equal to the number returned by up.getNtot(), and is equal to the number of rows in the value matrix. Violation of any of these conditions results in an error.

Parameters

in	*Grp&*	vector of pointers to rows of the value matrix
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level

7.17.2.6 MuGrp() [6/14]

```
MuGrp::MuGrp (
            const vector< MVnorm * > & dat,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam )
```

Constructor with a vector of MVnorm pointers and output file name.

Initializes a value matrix with means among the elements of the vector of MVnorm pointers, according to the low index (i.e., the given vector contains the data). Values for the matrix are sampled around the mean values. The number returned by low.getNgrp() has to be equal to the number returned by up.getNtot(), and is equal to the number of rows in the value matrix. Violation of any of these conditions results in an error.

Parameters

in	*vector<MVnorm*	*>& vector of pointers to rows of the value matrix
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level
in	*string&*	name of the output file

7.17.2.7 MuGrp() [7/14]

```
MuGrp::MuGrp (
            const Grp & dat,
            RanIndex & low,
            RanIndex & up,
            const string & outFlNam )
```

Constructor with a Grp object and output file name.

Initializes a value matrix with means among the rows of the matrix in Grp addressed by the dMat() member, according to the low index (i.e., the Grp object contains the data). Values for the matrix are sampled around the mean values. The number returned by low.getNgrp() has to be equal to the number returned by up.getNtot(), and is equal to the number of rows in the value matrix. Violation of any of these conditions results in an error.

Parameters

in	*Grp&*	vector of pointers to rows of the value matrix
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level
in	*string&*	name of the output file

7.17.2.8 MuGrp() [8/14]

```
MuGrp::MuGrp (
            const Grp & dat,
            RanIndex & low )
```

Deterministic mean constructor.

The value matrix of the resulting object has low.getNgrp() rows and is deterministically set to within-group means.

Parameters

in	*Grp&*	data object
in	*RanIndex&*	index that defines groups

7.17.2.9 MuGrp() [9/14]

```
MuGrp::MuGrp (
            const Grp & dat,
            const Qgrp & q,
            RanIndex & low )
```

Deterministic weighted mean constructor.

The value matrix of the resulting object has low.getNgrp() rows and is deterministically set to within-group weighted means. Weghts are in the Qgrp object. They are typically Student- t model weights.

Parameters

in	*Grp&*	data object
in	*Qgrp&*	weights
in	*RanIndex&*	index that defines groups

7.17.2.10 MuGrp() [10/14]

```
MuGrp::MuGrp (
            const gsl_matrix * dat )
```

Deterministic constructor with a GSL matrix.

Indexes not set. The given matrix is copied to the value matrix.

Parameters

in	*gsl_matrix∗*	data matrix

7.17.2.11 MuGrp() [11/14]

```
MuGrp::MuGrp (
            const gsl_matrix * dat,
            RanIndex & low )
```

Deterministic GSL matrix mean constructor.

The value matrix of the resulting object has low.getNgrp() rows and is deterministically set to within-group means. Mostly used internally in other derived classes.

Parameters

in	*gsl_matrix∗*	data matrix
in	*RanIndex&*	index that defines groups

7.17.2.12 MuGrp() [12/14]

```
MuGrp::MuGrp (
            const gsl_matrix * dat,
            const Qgrp & q,
            RanIndex & low )
```

Deterministic GSL matrix weighted mean constructor.

The value matrix of the resulting object has low.getNgrp() rows and is deterministically set to within-group weighted means. Mostly used internally in other derived classes.

Parameters

in	*gsl_matrix∗*	data matrix
in	*Qgrp&*	weights
in	*RanIndex&*	index that defines groups

7.17.2.13 MuGrp() [13/14]

```
MuGrp::MuGrp (
            const MuGrp & mG )
```

Copy constructor.

Parameters

in	*MuGrp&*	object to be copied

Returns

MuGrp& copy of the object

7.17.2.14 MuGrp() [14/14]

```
MuGrp::MuGrp (
            const Grp & g )
```

Copy constructor.

Parameters

in	*Grp&*	object to be copied

Returns

MuGrp& copy of the object

7.17.3 Member Function Documentation

7.17.3.1 operator=()

```
MuGrp & MuGrp::operator= (
            const MuGrp & mG )
```

Assignemnt operator.

Parameters

in	*MuGrp&*	object to be copied

Returns

MuGrp& target object

7.17.3.2 update() [1/10]

```
void MuGrp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance

Implements Grp.

Reimplemented in MuGrpEEmiss, MuGrpEE, and MuBlk.

7.17.3.3 update() [2/10]

```
void MuGrp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.4 update() [3/10]

```
void MuGrp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.5 update() [4/10]

```
void MuGrp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.6 update() [5/10]

```
void MuGrp::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.7 update() [6/10]

```
void MuGrp::update (
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance

Implements Grp.

Reimplemented in MuGrpEEmiss, MuGrpEE, MuGrpMiss, MuBlk, BetaGrpSnpMiss, and BetaGrpSnp.

7.17.3.8 update() [7/10]

```
void MuGrp::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Sigma↩ I&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*Sigma↩ I&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.9 update() [8/10]

```
void MuGrp::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.10 update() [9/10]

```
void MuGrp::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuBlk, and MuGrpPEX.

7.17.3.11 update() [10/10]

```
void MuGrp::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Implements Grp.

Reimplemented in MuGrpMiss, MuBlk, and MuGrpPEX.

7.17.4 Friends And Related Function Documentation

7.17.4.1 operator+ [1/3]

```
MuGrp operator+ (
            const Grp & m1,
            const Grp & m2 )  [friend]
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.17.4.2 operator+ [2/3]

```
MuGrp operator+ (
            const Grp & m1,
            const MuGrp & m2 )  [friend]
```

Addition operator.

Parameters

in	*Grp&*	first summand
in	*MuGrp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.17.4.3 operator+ [3/3]

```
MuGrp operator+ (
            const MuGrp & m1,
            const Grp & m2 )  [friend]
```

Addition operator.

Parameters

in	*MuGrp&*	first summand
in	*Grp&*	second summand

Returns

MuGrp object that is the sum of the two input objects

7.17.4.4 operator- [1/3]

```
MuGrp operator- (
            const Grp & m1,
            const Grp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

7.17.4.5 operator- [2/3]

```
MuGrp operator- (
            const Grp & m1,
            const MuGrp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*Grp&*	minuend
in	*MuGrp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

7.17.4.6 operator- [3/3]

```
MuGrp operator- (
            const MuGrp & m1,
            const Grp & m2 )  [friend]
```

Subtraction operator.

Parameters

in	*MuGrp&*	minuend
in	*Grp&*	subtrahend

Returns

MuGrp object that is the difference between the two input objects

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

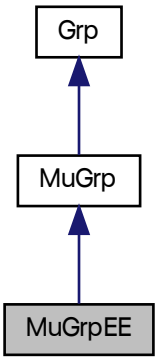
- MuGen.h
- MuGen.cpp

7.18 MuGrpEE Class Reference

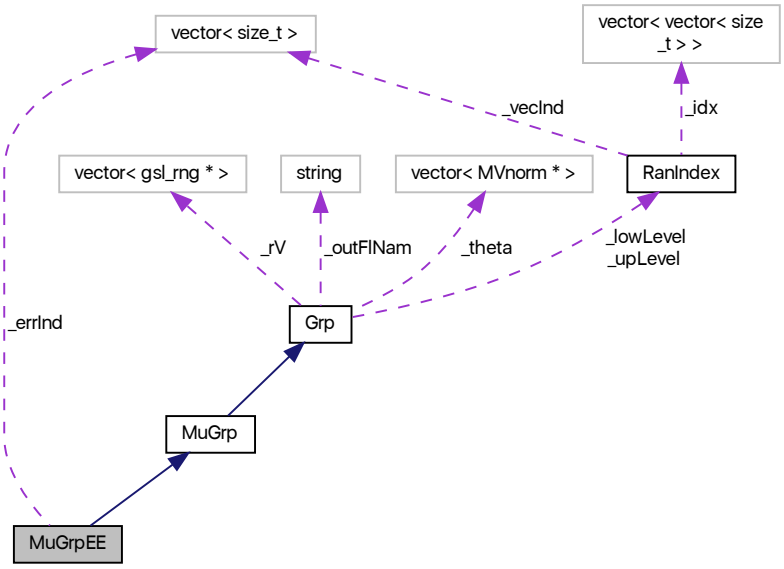
Data with measurement error.

```
#include <MuGen.h>
```

Inheritance diagram for MuGrpEE:



Collaboration diagram for MuGrpEE:



Public Member Functions

- MuGrpEE ()

  Default constructor.
- MuGrpEE (const string &datFlNam, const string &varFlNam, const string &indFlNam, RanIndex &up, const size_t &d)

  Constructor with index read from a file.
- MuGrpEE (const string &datFlNam, const string &varFlNam, const vector< size_t > &varInd, RanIndex &up, const size_t &d)

  Constructor with index vector.
- ~MuGrpEE ()

  Destructor.
- MuGrpEE (const MuGrpEE &mG)

  Copy constructor.
- MuGrpEE & operator= (const MuGrpEE &mG)

  Assignment operator.
- virtual void update (const Grp &muPr, const SigmaI &SigIm)

  Gaussian prior.
- virtual void update (const Grp &muPr, const Qgrp &q, const SigmaI &SigIm)

  Student-t prior.

Protected Attributes

- gsl_matrix * _errorInvVar

  Matrix of error inverse-variances.
- gsl_matrix * _meanVal

  Matrix of means.
- vector< size_t > _errInd

  Trait index.

Additional Inherited Members

7.18.1 Detailed Description

Data with measurement error.

If some phenotypes are measured with known error (e.g., technical replicates that are not idividually available), these can be sampled rather than used as point estimates. A mix of traits with and without measurement error is allowed. This class has to be on the bottom of the model hierarchy.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 MuGrpEE() [1/3]

```
MuGrpEE::MuGrpEE (
            const string & datFlNam,
            const string & varFlNam,
            const string & indFlNam,
            RanIndex & up,
            const size_t & d )
```

Constructor with index read from a file.

The data, error variances and the index identifying traits with errors are read from files. The index file must be a white-space separated text file. Variances are inverted after reading and are checked for sanity (i.e. not smaller than machine epsilon).

Parameters

in	*string&*	data file name
in	*string&*	variance file name
in	*string&*	index values file name
in	*RainIndex&*	upper level (prior) index
in	*size_t&*	number of traits

7.18.2.2 MuGrpEE() [2/3]

```
MuGrpEE::MuGrpEE (
            const string & datFlNam,
            const string & varFlNam,
            const vector< size_t > & varInd,
            RanIndex & up,
            const size_t & d )
```

Constructor with index vector.

The data and error variances are read from files, but the index identifying traits with errors is provided in a vector. Variances are inverted after reading and are checked for sanity (i.e. not smaller than machine epsilon).

Parameters

in	*string&*	data file name
in	*string&*	variance file name
in	*vector<size_t>&*	index values file name
in	*RainIndex&*	upper level (prior) index
in	*size_t&*	number of traits

7.18.2.3 MuGrpEE() [3/3]

```
MuGrpEE::MuGrpEE (
            const MuGrpEE & mG )
```

Copy constructor.

Parameters

in	*MuGrpEE&*	object to be copied

Returns

MuGrpEE object

7.18.3 Member Function Documentation

7.18.3.1 operator=()

```
MuGrpEE & MuGrpEE::operator= (
            const MuGrpEE & mG )
```

Assignment operator.

Parameters

in	*MuGrpEE&*	object to be copied

Returns

MuGrpEE& target object

7.18.4 Member Data Documentation

7.18.4.1 _errInd

```
vector<size_t> MuGrpEE::_errInd  [protected]
```

Trait index.

Contains indexes of the traits that have measurment errors. Only one index for all rows, i.e. once a trait has measurment error all samples must have a non-zero error variance.

7.18.4.2 _errorInvVar

```
gsl_matrix* MuGrpEE::_errorInvVar  [protected]
```

Matrix of error inverse-variances.

The matrix has the same number of rows as the value matrix, and the number of columns no larger than the value matrix. Variances are read from a file and inverted on initialization.

7.18.4.3 _meanVal

```
gsl_matrix* MuGrpEE::_meanVal  [protected]
```

Matrix of means.

Has the same dimensions as the _errorVar matrix, and stores mean values for the variables that will be sampled.

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

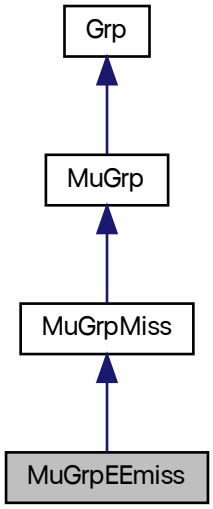
- MuGen.h
- MuGen.cpp

7.19 MuGrpEEmiss Class Reference

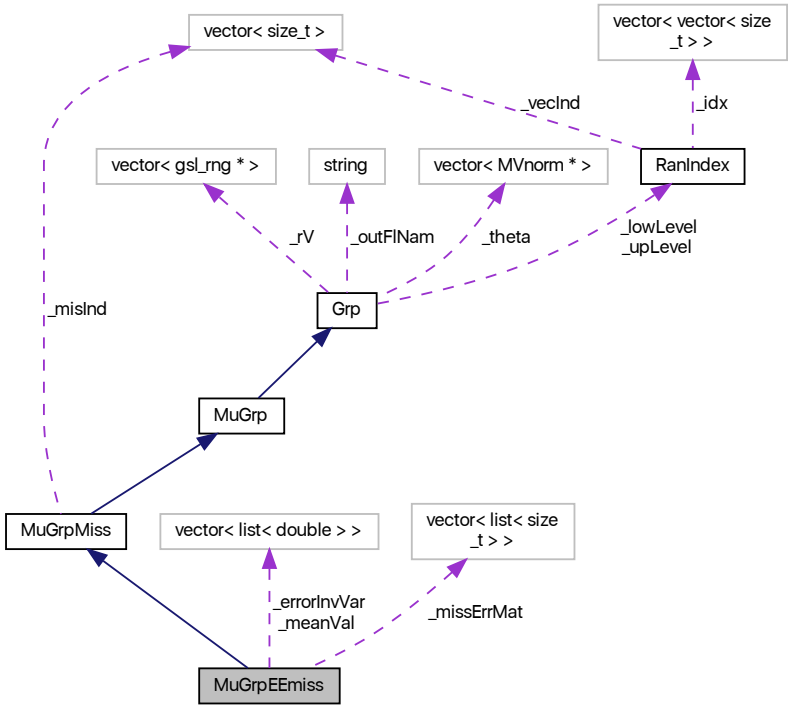
Data with measurement error and missing phenotypes.

```
#include <MuGen.h>
```

Inheritance diagram for MuGrpEEmiss:



Collaboration diagram for MuGrpEEmiss:



Public Member Functions

- MuGrpEEmiss ()

  Default constructor.
- MuGrpEEmiss (const string &datFlNam, const string &varFlNam, const string &indFlNam, const string &mis↩ MatFlNam, const string &misVecFlNam, RanIndex &up, const size_t &d)

  Constructor with index read from a file.
- MuGrpEEmiss (const string &datFlNam, const string &varFlNam, const vector< size_t > &varInd, const string &misMatFlNam, const string &misVecFlNam, RanIndex &up, const size_t &d)

  Constructor with index vector.
- ~MuGrpEEmiss ()

  Destructor.
- MuGrpEEmiss (const MuGrpEEmiss &mG)

  Copy constructor.
- MuGrpEEmiss & operator= (const MuGrpEEmiss &mG)

  Assignment operator.
- void update (const Grp &muPr, const SigmaI &SigIm)

  Standard Gaussian imputation.
- void update (const Grp &muPr, const Qgrp &q, const SigmaI &SigIm)

  Student-t likelihood, improper prior.

Protected Attributes

- vector< list< double > > _errorInvVar

  Measurement error inverse-variances.
- vector< list< double > > _meanVal

  Means for values with errors.
- vector< list< size_t > > _missErrMat

  Error index.

Additional Inherited Members

7.19.1 Detailed Description

Data with measurement error and missing phenotypes.

Some traits have measurment errors, some (possibly an overlapping set) are missing. Sampling with error is done only for the present phenotypes, thus not all rows have the same number of traits with error variances.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 MuGrpEEmiss() [1/3]

```
MuGrpEEmiss::MuGrpEEmiss (
            const string & datFlNam,
            const string & varFlNam,
            const string & indFlNam,
            const string & misMatFlNam,
            const string & misVecFlNam,
            RanIndex & up,
            const size_t & d )
```

Constructor with index read from a file.

The data, error variances and the index identifying traits with errors are read from files. The index file must be a white-space separated text file. The index is initially the same for each row, but then the values corresponding to missing phenotypes are dropped for relevant rows.

Parameters

in	*string&*	data file name
in	*string&*	variance file name
in	*string&*	index values file name
in	*string&*	matrix missing data index file name
in	*string&*	vector missing data index file name
in	*RanIndex&*	upper level (prior) index
in	*size_t&*	number of traits

7.19.2.2 MuGrpEEmiss() [2/3]

```
MuGrpEEmiss::MuGrpEEmiss (
            const string & datFlNam,
            const string & varFlNam,
            const vector< size_t > & varInd,
            const string & misMatFlNam,
            const string & misVecFlNam,
            RanIndex & up,
            const size_t & d )
```

Constructor with index vector.

The data and error variances are read from files, but the index identifying traits with errors is provided in a vector. The index is initially the same for each row, but then the values corresponding to missing phenotypes are dropped for relevant rows.

Parameters

in	*string&*	data file name
in	*string&*	variance file name
in	*vector<size_t>&*	index values
in	*string&*	matrix missing data index file name
in	*string&*	vector missing data index file name
in	*RainIndex&*	upper level (prior) index
in	*size_t&*	number of traits

7.19.2.3 MuGrpEEmiss() [3/3]

```
MuGrpEEmiss::MuGrpEEmiss (
            const MuGrpEEmiss & mG )
```

Copy constructor.

Parameters

in	*MuGrpEEmiss&*	object to be copied

Returns

MuGrpEEmiss object

7.19.3 Member Function Documentation

7.19.3.1 operator=()

```
MuGrpEEmiss & MuGrpEEmiss::operator= (
            const MuGrpEEmiss & mG )
```

Assignment operator.

Parameters

in	*MuGrpEEmiss&*	object to be copied

Returns

MuGrpEEmiss& target object

7.19.3.2 update() [1/2]

```
void MuGrpEEmiss::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, improper prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance

Reimplemented from MuGrp.

7.19.3.3 update() [2/2]

```
void MuGrpEEmiss::update (
            const Grp & mu,
            const SigmaI & SigIm )  [virtual]
```

Standard Gaussian imputation.

If some data are present, performs standard Gaussian marginal imputation (desribed in, e.g., **[chatfield80]**) with the provided Grp object as a mean and the SigmaI object as the inverse-covariance. If no data for a row are present, simply replaces the row values by a Gaussian sample with mean and inverse-covariance provided. Rows with no missing data are ignored.

Parameters

in	*Grp&*	mean
in	*SigmaI&*	inverse-covariance

Reimplemented from MuGrpMiss.

7.19.4 Member Data Documentation

7.19.4.1 _errorInvVar

```
vector< list<double> > MuGrpEEmiss::_errorInvVar  [protected]
```

Measurement error inverse-variances.

The same configuration as the variance vector of lists. This is the subset of _valueMat that corresponds to the values to be updated with experimental error.

7.19.4.2 _meanVal

```
vector< list<double> > MuGrpEEmiss::_meanVal  [protected]
```

Means for values with errors.

The length of the vector is equal to the number of rows of the value matrix. For each row, the error variances are in a list that has as many members as there are non-missing phenotypes with errors.

7.19.4.3 _missErrMat

```
vector< list<size_t> > MuGrpEEmiss::_missErrMat  [protected]
```

Error index.

The same configuration as the variance vector of lists, but containing indexes of the elements of the value matrix that have measurement errors.

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

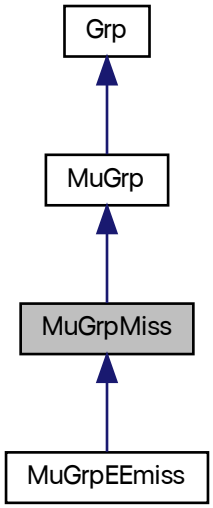
- MuGen.h
- MuGen.cpp

7.20 MuGrpMiss Class Reference

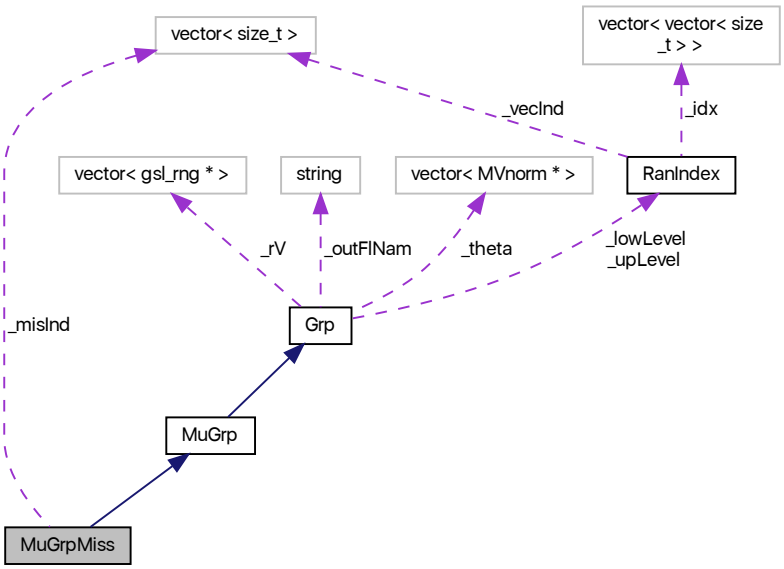
Data with missing phenotype values.

```
#include <MuGen.h>
```

Inheritance diagram for MuGrpMiss:



Collaboration diagram for MuGrpMiss:



Public Member Functions

- MuGrpMiss ()

  Default constructor.
- MuGrpMiss (const string &datFlNam, const string &misMatFlNam, const string &misVecFlNam, RanIndex &up, const size_t &d)

  Full constructor.
- ~MuGrpMiss ()

  Destructor.
- MuGrpMiss (const MuGrpMiss &mG)

  Copy constructor.
- MuGrpMiss & operator= (const MuGrpMiss &mG)

  Assignment operator.
- virtual void update (const Grp &mu, const SigmaI &SigIm)

  Standard Gaussian imputation.
- virtual void update (const Grp &mu, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian imputation with a prior.
- size_t **nMis** () const
- size_t **nMis** ()

Protected Attributes

- vector< size_t > _misInd

  Index of the rows with missing data.

Additional Inherited Members

7.20.1 Detailed Description

Data with missing phenotype values.

Implements missing phenotypic (responce) data imputation. This object has to be at the bottom of the hierarchy, so there is no index to the lower level.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 MuGrpMiss() [1/2]

```
MuGrpMiss::MuGrpMiss (
            const string & datFlNam,
            const string & misMatFlNam,
            const string & misVecFlNam,
            RanIndex & up,
            const size_t & d )
```

Full constructor.

Reads data and indexes that show which data are missing from files. Matrix index has the same dimensions as the value matrix and has "1" in places where the data are missing and "0" otherwise. The vector missing data index indicates the number of data points missing in each row (zero for rows with all data present). The data file can have any double-precision value in place of missing data. These values are ignored and replaced by the overall mean at construction, but subsequently updated.

Parameters

in	*string&*	data file name
in	*string&*	matrix missing data index file name
in	*string&*	vector missing data index file name
in	*RanIndex&*	index pointing to the prior (next level in the hierarchy)
in	*size_t&*	number of traits

7.20.2.2 MuGrpMiss() [2/2]

```
MuGrpMiss::MuGrpMiss (
            const MuGrpMiss & mG )
```

Copy constructor.

Parameters

in	*MuGrpMiss&*	object to be copied

Returns

MuGrpMiss object

7.20.3 Member Function Documentation

7.20.3.1 operator=()

```
MuGrpMiss & MuGrpMiss::operator= (
            const MuGrpMiss & mG )
```

Assignment operator.

Parameters

in	*MuGrpMiss&*	object to be copied

Returns

MuGrpMiss& target object

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

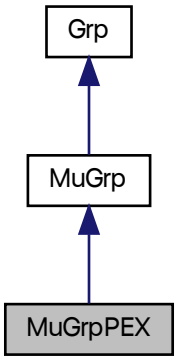
- MuGen.h
- MuGen.cpp

7.21 MuGrpPEX Class Reference

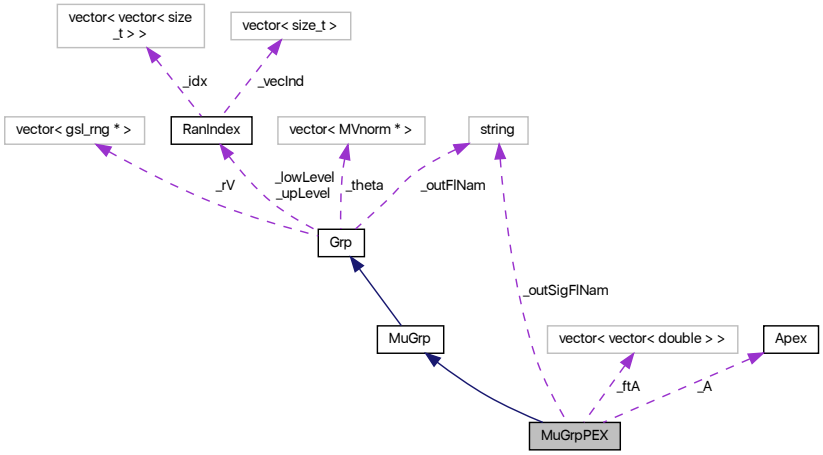
"Random effect" with parameter expansion

```
#include <MuGen.h>
```

Inheritance diagram for MuGrpPEX:



Collaboration diagram for MuGrpPEX:



Public Member Functions

- MuGrpPEX ()

  Default constructor.
- MuGrpPEX (const Grp &dat, RanIndex &low, RanIndex &up, const double &Spr, const int &nThr)

  Full constructor.
- MuGrpPEX (const Grp &dat, RanIndex &low, RanIndex &up, const string outMuFlNam, const double &Spr, const int &nThr)

Full constructor with location parameter file name.

- ~MuGrpPEX ()

  Destructor.

- MuGrpPEX (const MuGrpPEX &mGp)

  Copy constructor.

- MuGrpPEX & operator= (const MuGrpPEX &mGp)

  Assignment operator.

- const gsl_matrix * fMat () const

  Access the adjusted value matrix.

- void save ()

  Save the adjusted values.

- void save (const string &outFlNam)

  Save adjusted values to named file.

- void save (const SigmaI &SigI)

  Save with the covariance matrix.

- Apex & getA ()

  Access the redundant parameter.

- void setApr (const double &pr)

  Set the inverse-prior for the redundant parameter matrix.

- MuGrp mean (RanIndex &grp)

  Group mean.

- const MuGrp mean (RanIndex &grp) const

  Group mean.

- MuGrp mean (RanIndex &grp, const Qgrp &q)

  Group weighted mean.

- const MuGrp mean (RanIndex &grp, const Qgrp &q) const

  Group weighted mean.

- void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Gaussian prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Gaussian prior.

- void update (const Grp &dat, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, 0-mean Student-t prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, 0-mean Student-t prior.

- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Gaussian prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Gaussian prior.

- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Gaussian likelihood, non-zero mean Student-t prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const Qgrp &qPr, const SigmaI &SigIp)

  Student-t likelihood, non-zero mean Student-t prior.

Protected Member Functions

- void _updateFitted ()

  Update adjusted values.

Protected Attributes

- gsl_matrix * _adjValMat

  Adjusted value matrix.
- gsl_matrix * _tSigIAt

  Scaled inverse-covariance.
- Apex _A

  Redundant parameter matrix.
- vector< vector< double > > _ftA

  Fitted values.
- int _nThr

  number of threads
- string _outSigFlNam

  Covariance output file name.

7.21.1 Detailed Description

"Random effect" with parameter expansion

An implementation of a multivariate extension (Greenberg, unpublished) of the multiplicative PEX scheme for "random effects" model **[gelman07] [gelman08]** . This object has to be outside the model hierarchy, i.e. have the same prior for all rows. Variables are divided into "raw" and "adjusted" as in **[gelman07]** . "Adjusted" variables are on the correct scale for interpretation. The "raw" value matrix is denoted Ξ and the redundant parameter matrix (implemented in the Apex class) is $\boldsymbol{A}$.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 MuGrpPEX() [1/3]

```
MuGrpPEX::MuGrpPEX (
            const Grp & dat,
            RanIndex & low,
            RanIndex & up,
            const double & Spr,
            const int & nThr )
```

Full constructor.

Initializing constructor that provides starting values for the location parameter and redundant parameter matrices.

Parameters

in	*Grp&*	data
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level
in	*double&*	inverse-prior for the redundant parameter matrix
in	*int&*	number of threads

7.21.2.2 MuGrpPEX() [2/3]

```
MuGrpPEX::MuGrpPEX (
            const Grp & dat,
            RanIndex & low,
            RanIndex & up,
            const string outMuFlNam,
            const double & Spr,
            const int & nThr )
```

Full constructor with location parameter file name.

Initializing constructor that provides starting values for the location parameter and redundant parameter matrices. Sets the name of the file where the adjusted location parameter chains will be saved.

Parameters

in	*Grp&*	data
in	*RanIndex&*	index to the lower (data) level
in	*RanIndex&*	index to the upper (prior) level
in	*string&*	output file name
in	*double&*	inverse-prior for the redundant parameter matrix
in	*int&*	number of threads

7.21.2.3 MuGrpPEX() [3/3]

```
MuGrpPEX::MuGrpPEX (
            const MuGrpPEX & mGp )
```

Copy constructor.

Parameters

in	*MuGrpPEX&*	object to be copied

Returns

MuGrpPEX object

7.21.3 Member Function Documentation

7.21.3.1 fMat()

```
const gsl_matrix* MuGrpPEX::fMat ( ) const  [inline], [virtual]
```

Access the adjusted value matrix.

Returns

gsl_matrix* pointer to the adjusted value matrix

Reimplemented from Grp.

7.21.3.2 getA()

```
Apex& MuGrpPEX::getA ( )  [inline]
```

Access the redundant parameter.

Returns

Apex& the redundant parameter matrix

7.21.3.3 mean() [1/4]

```
MuGrp MuGrpPEX::mean (
            RanIndex & grp )  [virtual]
```

Group mean.

Calculates within-group mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix.

Parameters

in	*RanIndex&*	grouping index

Returns

MuGrp object with the matrix of means

Reimplemented from Grp.

7.21.3.4 mean() [2/4]

```
const MuGrp MuGrpPEX::mean (
            RanIndex & grp ) const  [virtual]
```

Group mean.

Calculates within-group mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix.

Parameters

in	*RanIndex&*	grouping index

Returns

MuGrp object with the matrix of means

Reimplemented from Grp.

7.21.3.5 mean() [3/4]

```
MuGrp MuGrpPEX::mean (
            RanIndex & grp,
            const Qgrp & q )  [virtual]
```

Group weighted mean.

Calculates within-group weighted mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix. The weights are in the provided Qgrp object. Typically, they are from a Student-t model.

Parameters

in	*RanIndex&*	grouping index
in	*Qgrp&*	weights

Returns

MuGrp object with the matrix of means

Reimplemented from Grp.

7.21.3.6 mean() [4/4]

```
const MuGrp MuGrpPEX::mean (
            RanIndex & grp,
            const Qgrp & q ) const  [virtual]
```

Group weighted mean.

Calculates within-group weighted mean values of the value matrix (by row). The groups are defined by the upper-level of the given RanIndex variable. The lower-level number of elements of the index has to be the same as the number of rows in the current object's value matrix. The weights are in the provided Qgrp object. Typically, they are from a Student-t model.

Parameters

in	*RanIndex&*	grouping index
in	*Qgrp&*	weights

Returns

MuGrp object with the matrix of means

Reimplemented from Grp.

7.21.3.7 operator=()

```
MuGrpPEX & MuGrpPEX::operator= (
            const MuGrpPEX & mGp )
```

Assignment operator.

Parameters

in	*MuGrpPEX&*	object to be copied

Returns

MuGrpPEX& target object

7.21.3.8 save() [1/3]

```
void MuGrpPEX::save ( )  [virtual]
```

Save the adjusted values.

Appends the current adjusted mean values to the file whose name was set during construction.

Reimplemented from Grp.

7.21.3.9 save() [2/3]

```
void MuGrpPEX::save (
            const SigmaI & SigI )  [virtual]
```

Save with the covariance matrix.

Appends the current adjusted mean values and prior covariance matrix to the files whose names were set during construction.

Reimplemented from Grp.

7.21.3.10 save() [3/3]

```
void MuGrpPEX::save (
            const string & outFlNam )  [virtual]
```

Save adjusted values to named file.

Appends the current adjusted mean values to the named file.

Parameters

in	*string&*	file name

Reimplemented from Grp.

7.21.3.11 setApr()

```
void MuGrpPEX::setApr (
            const double & pr )  [inline]
```

Set the inverse-prior for the redundant parameter matrix.

Parameters

in	*double&*	inverse-prior value

7.21.3.12 update() [1/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.13 update() [2/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.14 update() [3/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for the data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.15 update() [4/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Student- t likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*Qgrp&*	Student- t weight parameter for data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.16 update() [5/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Student- t prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.17 update() [6/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, non-zero mean Gaussian prior.

For the relationship to work properly, the _upLevel index of the focal object has to point to the rows of the prior mean matrix. This is the matrix addressed by dMat(). The relationship to the data is dependent on the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.18 update() [7/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Qgrp & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Student- t prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*Qgrp&*	Student- t weight parameter for the prior
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.3.19 update() [8/8]

```
void MuGrpPEX::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI &  )  [virtual]
```

Gaussian likelihood, 0-mean Gaussian prior.

Parameters

in	*Grp&*	data
in	*SigmaI&*	data inverse-covariance
in	*SigmaI&*	prior inverse-covariance

Reimplemented from MuGrp.

7.21.4 Member Data Documentation

7.21.4.1 _adjValMat

```
gsl_matrix* MuGrpPEX::_adjValMat  [protected]
```

Adjusted value matrix.

The location paramter of interest, i.e. ΞA.

7.21.4.2 _ftA

```
vector<vector<double> > MuGrpPEX::_ftA  [protected]
```

Fitted values.

Vector of vectorized individual fitted matrices $\Xi_{\cdot -m} A_{-m \cdot}$.

7.21.4.3 _outSigFlNam

```
string MuGrpPEX::_outSigFlNam  [protected]
```

Covariance output file name.

This is to save the adjusted prior covariance matrix associated with this object. Saving from the SigmaI object gives the raw matrix.

7.21.4.4 _tSigIAt

```
gsl_matrix* MuGrpPEX::_tSigIAt  [protected]
```

Scaled inverse-covariance.

The matrix $\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^T\right)^T$ that is common for sampling each row of the value matrix.

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

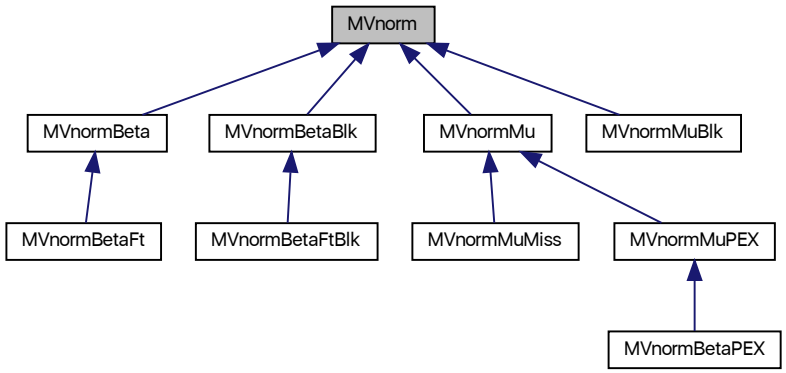
- MuGen.h
- MuGen.cpp

7.22 MVnorm Class Reference

The abstract base class for location parameter rows.

```
#include <MuGen.h>
```

Inheritance diagram for MVnorm:



Public Member Functions

- MVnorm (const MVnorm &)

  Copy constructor.
- MVnorm & operator= (const MVnorm &)

  Assignement operator.
- virtual ~MVnorm ()

  Virtual destructor.
- virtual void update (const Grp &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const gsl_rng *)=0

  Sudent- t likelihood.
- virtual void update (const Grp &, const SigmaI &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &, const SigmaI &, const double &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood, Student- t prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const SigmaI &, const gsl_rng *)=0

  Student- t likelihood, Gaussian prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const double &, const SigmaI &, const gsl_rng *)=0

  Student- t likelihood, Student- t prior.
- virtual void update (const Grp &, const SigmaI &, const Grp &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &, const SigmaI &, const Grp &, const double &, const SigmaI &, const gsl_rng *)=0

  Gaussian likelihood, Student- t prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const SigmaI &, const gsl_rng *)=0

  Student- t likelihood, Gaussian prior.
- virtual void update (const Grp &, const Qgrp &, const SigmaI &, const Grp &, const double &, const SigmaI &, const gsl_rng *)=0

  Student- t likelihood, Student- t prior.
- virtual double mhl (const MVnorm *x, const SigmaI &SigI)

  Mahalanobis distance to a vector.
- virtual double mhl (const MVnorm *x, const SigmaI &SigI) const

  Mahalanobis distance to a vector.
- virtual double mhl (const gsl_vector *x, const SigmaI &SigI)

  Mahalanobis distance to a vector.
- virtual double mhl (const gsl_vector *x, const SigmaI &SigI) const

  Mahalanobis distance to a vector.
- virtual double mhl (const SigmaI &SigI)

  Mahalanobis distance to zero.
- virtual double mhl (const SigmaI &SigI) const

  Mahalanobis distance to zero.
- double density (const gsl_vector *theta, const SigmaI &SigI)

  Multivariate Gaussian density.
- double density (const gsl_vector *theta, const SigmaI &SigI) const

  Multivariate Gaussian density.
- double density (const MVnorm *theta, const SigmaI &SigI)

Multivariate Gaussian density.

- double density (const MVnorm *theta, const Sigmal &SigI) const

  Multivariate Gaussian density.

- void save (const string &fileNam, const char *how="a")

  Save function.

- void save (FILE *fileStr)

  Save function.

- double operator[ ] (const size_t i) const

  Subscript operator.

- void valSet (const size_t i, const double x)

  Setting an element to a value.

- const gsl_vector * getVec () const

  Access the location vector.

- size_t len () const

  Length of the location vector.

- virtual size_t nMissP () const

  Number of missing values.

- virtual const vector< size_t > getMisPhen () const

  Indexes of missing values.

- virtual const vector< size_t > * down () const

  Points to the corresponding data.

- virtual const size_t * up () const

  Points to the prior.

- virtual double scalePar () const

  Scale parameter.

Protected Member Functions

- MVnorm ()

  Default constructor.

- MVnorm (const size_t &d)

  Dimension-only constructor.

- MVnorm (gsl_vector *mn)

  Dimension and vector value constructor.

- MVnorm (gsl_vector *mn, const gsl_vector *sd, const gsl_rng *r)

  Univariate Gaussian constructor.

- MVnorm (gsl_vector *mn, const gsl_matrix *Sig, const gsl_rng *r)

  Multivariate Gaussian constructor.

- MVnorm (gsl_matrix *mn, const size_t &iRw)

  Dimension and vector value constructor.

- MVnorm (gsl_matrix *mn, const size_t &iRw, const gsl_vector *sd, const gsl_rng *r)

  Univariate Gaussian constructor with a matrix.

- MVnorm (gsl_matrix *mn, const size_t &iRw, const gsl_matrix *Sig, const gsl_rng *r)

  Multivariate Gaussian constructor with a matrix.

Protected Attributes

- gsl_vector_view _vec

  Data vector.
- size_t _d

  Length of the data vector.

7.22.1 Detailed Description

The abstract base class for location parameter rows.

This is the generic location parameter class and cannot be envoked directly. All the constructors are protected.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 MVnorm() [1/9]

```
MVnorm::MVnorm ( )  [inline], [protected]
```

Default constructor.

This is a deterministic constructor that results in a pointer to nowhere and the dimension member set to zero.

7.22.2.2 MVnorm() [2/9]

```
MVnorm::MVnorm (
            const size_t & d )  [inline], [protected]
```

Dimension-only constructor.

Sets the dimension to a value, but the *vector_view* is not assigned a target.

Parameters

in	*size_t&*	dimension value

7.22.2.3 MVnorm() [3/9]

```
MVnorm::MVnorm (
```

```
            gsl_vector * mn )  [protected]
```

Dimension and vector value constructor.

A deterministic constructor that sets the _vec member to point to the provided *gsl_vector* and the dimension member to the size of the provided *gsl_vector*.

Parameters

in	*gsl_vector*∗	target vector

7.22.2.4 MVnorm() [4/9]

```
MVnorm::MVnorm (
            gsl_vector * mn,
            const gsl_vector * sd,
            const gsl_rng * r )  [protected]
```

Univariate Gaussian constructor.

Sets the _vec member to point to the provided *gsl_vector* and ads univariate Gaussian noise independtly to each element, modifying the target.

Parameters

in,out	*gsl_vector*∗	vector of means
in	*gsl_vector*∗	vector of standard deviations for the univariate Gaussian, has to be no shorter than the mean vector. If it is longer, the extra values are ignored.
in	*gsl_rng*∗	pointer to a PNG

7.22.2.5 MVnorm() [5/9]

```
MVnorm::MVnorm (
            gsl_vector * mn,
            const gsl_matrix * Sig,
            const gsl_rng * r )  [protected]
```

Multivariate Gaussian constructor.

Sets the _vec member to point to the provided *gsl_vector* and ads multivariate Gaussian noise, modifying the target.

Parameters

in,out	*gsl_vector*∗	vector of means
in	*gsl_matrix*∗	covariance matrix for the multivariate Gaussian, has to match the mean vector in dimensions and has to be symmetric positive-definite
in	*gsl_rng*∗	pointer to a PNG

7.22.2.6 MVnorm() [6/9]

```
MVnorm::MVnorm (
            gsl_matrix * mn,
            const size_t & iRw )  [protected]
```

Dimension and vector value constructor.

A deterministic constructor that sets the _vec member to point to a row (indicated by the row index) of the provided *gsl_matrix* and the dimension member to the number of columns in the provided *gsl_matrix*.

Parameters

in	*gsl_matrix*∗	target matrix
in	*size_t&*	row index of the matrix

7.22.2.7 MVnorm() [7/9]

```
MVnorm::MVnorm (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_vector * sd,
            const gsl_rng * r )  [protected]
```

Univariate Gaussian constructor with a matrix.

Sets the _vec member to point to a row (indicated by the row index) of the provided *gsl_matrix* and ads univariate Gaussian noise independtly to each element, modifying the target.

Parameters

in,out	*gsl_matrix*∗	target matrix
in	*size_t&*	row index of the target matrix
in	*gsl_vector*∗	vector of standard deviations for the univariate Gaussian, has to be no shorter than a row of the mean matrix. If it is longer, the extra values are ignored.
in	*gsl_rng*∗	pointer to a PNG

7.22.2.8 MVnorm() [8/9]

```
MVnorm::MVnorm (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_matrix * Sig,
            const gsl_rng * r )  [protected]
```

Multivariate Gaussian constructor with a matrix.

Sets the _vec member to point to a row (indicated by the row index) of the provided *gsl_matrix* and ads multivariate Gaussian noise, modifying the target.

Parameters

in,out	*gsl_matrix*∗	target matrix
in	*size_t&*	row index of the target matrix
in	*gsl_matrix*∗	covariance matrix for the multivariate Gaussian, has to be symmetric positive-definite, with diemsions equal to the number of columns in the target matrix
in	*gsl_rng*∗	pointer to a PNG

7.22.2.9 MVnorm() [9/9]

```
MVnorm::MVnorm (
            const MVnorm & mu )
```

Copy constructor.

Provided for completeness and has not been extensively tested. The constructor is deterministic, i.e. the values are copied exactly, with no stochastic perturbation.

Parameters

in	*MVnorm&*	object to be copied

Returns

object of class *MVnorm*.

7.22.2.10 ~MVnorm()

```
virtual MVnorm::~MVnorm ( )  [inline], [virtual]
```

Virtual destructor.

The destructor has nothing to do since members of this class cannot be de-allocated.

7.22.3 Member Function Documentation

7.22.3.1 down()

```
virtual const vector<size_t>* MVnorm::down ( ) const  [inline], [virtual]
```

Points to the corresponding data.

Access to the pointer to a vector that indexes the data used to update an instance of the class. Is non-zero only for classes which can be part of a hierarchical model.

Returns

Pointer to a vector of *size_t*.

Reimplemented in MVnormMu.

7.22.3.2 getMisPhen()

```
virtual const vector<size_t> MVnorm::getMisPhen ( ) const  [inline], [virtual]
```

Indexes of missing values.

Accesses a vector with indexes of missing phenotypes.

Returns

Vector of *size_t* that contains indexes that correspond to missing values. For classes that do not allow missing values it is empty.

Reimplemented in MVnormMuMiss, and MVnormMu.

7.22.3.3 getVec()

```
const gsl_vector* MVnorm::getVec ( ) const  [inline]
```

Access the location vector.

Provides access to the internal parameter vector.

Returns

GSL vector pointing to the corresponding location parameter.

7.22.3.4 len()

```
size_t MVnorm::len ( ) const  [inline]
```

Length of the location vector.

Returns

Length of the parameter vector, of type *size_t*. Corresponds to the number of traits.

7.22.3.5 nMissP()

```
virtual size_t MVnorm::nMissP ( ) const  [inline], [virtual]
```

Number of missing values.

Accesses the number of missing phenotype values. Non-zero only for classes that implement treatment of missing values.

Returns

Number of missing phenotypes, of type *size_t*.

Reimplemented in MVnormMuMiss, and MVnormMu.

7.22.3.6 operator=()

```
MVnorm & MVnorm::operator= (
            const MVnorm & mu )
```

Assignement operator.

Provided for completeness and has not been extensively tested. The operator is deterministic, i.e. the values are copied exactly, with no stochastic perturbation.

Parameters

in	*MVnorm&*	object to be assigned

Returns

reference to an object of class *MVnorm*.

7.22.3.7 operator[]()

```
double MVnorm::operator[] (
            const size_t i ) const  [inline]
```

Subscript operator.

Overloaded subscript operator for the vector of location parameter values.

Parameters

in	*size_t*	index value

Returns

value of the member vector corresponding to the index value, of type *double*.

7.22.3.8 save() [1/2]

```
void MVnorm::save (
            const string & fileNam,
            const char * how = "a" )
```

Save function.

Saves the current value of the location parameter to a file, appending by default.

Parameters

in	*string&*	file name
in	*char∗*	save mode, "a" for append by default. The allowable values are as for a standard C file stream

7.22.3.9 save() [2/2]

```
void MVnorm::save (
            FILE * fileStr )
```

Save function.

Saves the current value of the location parameter to a file.

Parameters

in	*FILE*∗	C file stream

7.22.3.10 scalePar()

```
virtual double MVnorm::scalePar ( ) const  [inline], [virtual]
```

Scale parameter.

Access to a scale parameter member, defined only for some derived regression-type classes. If not defined, returns 1.0.

Returns

Scale paramter value of type *double*.

Reimplemented in MVnormBeta.

7.22.3.11 up()

```
virtual const size_t* MVnorm::up ( ) const  [inline], [virtual]
```

Points to the prior.

Access to the pointer to the correspoding vector of priors. Is non-zero only for classes where a prior is implemented.

Returns

Pointer to *size_t*.

Reimplemented in MVnormBetaBlk, MVnormMuBlk, MVnormBeta, and MVnormMu.

7.22.3.12 valSet()

```
void MVnorm::valSet (
            const size_t i,
            const double x )  [inline]
```

Setting an element to a value.

Sets the *i*-th element of the location vector to a value deterministically

Parameters

in	*size_t*	index
in	*double*	new value of the element

7.22.4 Member Data Documentation

7.22.4.1 _vec

```
gsl_vector_view MVnorm::_vec  [protected]
```

Data vector.

This is actually a pointer (implemented as a GSL *vector_view*) to a row in the corresponding matrix of location parameters.

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

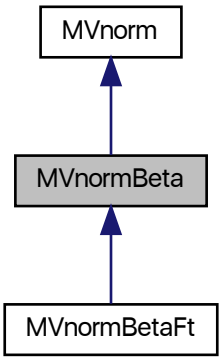
- MuGen.h
- MuGen.cpp

7.23 MVnormBeta Class Reference

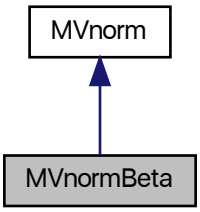
Generic regression.

```
#include <MuGen.h>
```

Inheritance diagram for MVnormBeta:



Collaboration diagram for MVnormBeta:



Public Member Functions

- MVnormBeta ()

  Default constructor.
- MVnormBeta (const size_t d)

  Dimension-only constructor.
- MVnormBeta (gsl_vector ∗b, const gsl_vector ∗sd, gsl_matrix ∗pred, const size_t &iCl, const gsl_rng ∗r)

  Univariate random constructor with vector.
- MVnormBeta (gsl_vector ∗b, const gsl_vector ∗sd, gsl_matrix ∗pred, const size_t &iCl, const gsl_rng ∗r, const size_t &up)

  Univariate random constructor with vector and pointer to prior.
- MVnormBeta (gsl_vector ∗b, const gsl_matrix ∗Sig, gsl_matrix ∗pred, const size_t &iCl, const gsl_rng ∗r)

  Multivariate random constructor with vector.
- MVnormBeta (gsl_vector ∗b, const gsl_matrix ∗Sig, gsl_matrix ∗pred, const size_t &iCl, const gsl_rng ∗r, const size_t &up)

  Multivariate random constructor with vector and pointer to prior.
- MVnormBeta (const gsl_matrix ∗resp, gsl_matrix ∗pred, const size_t &iCl, const gsl_matrix ∗Sig, const gsl_rng ∗r, gsl_matrix ∗bet, const size_t &iRw)

  Multivariate constructor with response matrix.
- MVnormBeta (const gsl_matrix ∗resp, gsl_matrix ∗pred, const size_t &iCl, const gsl_matrix ∗Sig, const gsl_rng ∗r, const size_t &up, gsl_matrix ∗bet, const size_t &iRw)

  Multivariate constructor with response matrix and index for a prior.
- MVnormBeta (const Grp &resp, gsl_matrix ∗pred, const size_t &iCl, const gsl_matrix ∗Sig, const gsl_rng ∗r, gsl_matrix ∗bet, const size_t &iRw)

  Multivariate constructor with a Grp type response.
- MVnormBeta (const Grp &resp, gsl_matrix ∗pred, const size_t &iCl, const gsl_matrix ∗Sig, const gsl_rng ∗r, const size_t &up, gsl_matrix ∗bet, const size_t &iRw)

  Multivariate constructor with a Grp type response and index for a prior.
- MVnormBeta (gsl_matrix ∗pred, const size_t &iCl, gsl_matrix ∗bet, const size_t &iRw)

  Deterministic constructor.
- MVnormBeta (gsl_matrix ∗pred, const size_t &iCl, const size_t &up, gsl_matrix ∗bet, const size_t &iRw)

  Deterministic constructor with prior index.

- MVnormBeta (const MVnormBeta &)

  Deterministic copy constructor.
- MVnormBeta & operator= (const MVnormBeta &)

  Deterministic assignement operator.
- virtual ~MVnormBeta ()

  Destructor.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const gsl_rng *r)

  Gaussian likelihood.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const gsl_rng *r)

  Sudent-t likelihood.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- const size_t * up () const

  Points to the prior.
- double scalePar () const

  Scale parameter.

Protected Attributes

- gsl_vector_view _X

  Predictor.
- double _scale

  Scale parameter.
- size_t _N

  Length of the predictor.
- const size_t * _upLevel

  Row index of the prior.

Additional Inherited Members

7.23.1 Detailed Description

Generic regression.

Generic implementation of a regression. By itself to be used strictly for regressions with a single predictor because it ignores the effects of other predictors.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 MVnormBeta() [1/12]

```
MVnormBeta::MVnormBeta (
            const size_t d )
```

Dimension-only constructor.

Sets the dimension to a value, but the _vec is not assigned a target.

7.23.2.2 MVnormBeta() [2/12]

```
MVnormBeta::MVnormBeta (
            gsl_vector * b,
            const gsl_vector * sd,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_rng * r )
```

Univariate random constructor with vector.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors.

Parameters

in	*gsl_vector*∗	vector of values
in	*gsl_vector*∗	vector of standard deviations
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_rng*∗	pointer to a PNG

7.23.2.3 MVnormBeta() [3/12]

```
MVnormBeta::MVnormBeta (
            gsl_vector * b,
            const gsl_vector * sd,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_rng * r,
            const size_t & up )
```

Univariate random constructor with vector and pointer to prior.

Sets up _vec to point to the provided vector of values, the predictor to a column in the provided matrix of predictors, and the pointer to the corresponding row in the prior matrix.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_vector∗*	vector of standard deviations
in	*gsl_matrix∗*	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	index of the appropriate row in the prior matrix

7.23.2.4 MVnormBeta() [4/12]

```
MVnormBeta::MVnormBeta (
            gsl_vector * b,
            const gsl_matrix * Sig,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_rng * r )
```

Multivariate random constructor with vector.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_matrix∗*	covariance matrix
in	*gsl_matrix∗*	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_rng∗*	pointer to a PNG

7.23.2.5 MVnormBeta() [5/12]

```
MVnormBeta::MVnormBeta (
            gsl_vector * b,
            const gsl_matrix * Sig,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_rng * r,
            const size_t & up )
```

Multivariate random constructor with vector and pointer to prior.

Sets up _vec to point to the provided vector of values, the predictor to a column in the provided matrix of predictors, and the pointer to the corresponding row in the prior matrix.

Parameters

in	*gsl_vector*∗	vector of values
in	*gsl_matrix*∗	covariance matrix
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_rng*∗	pointer to a PNG
in	*size_t&*	index of the appropriate row in the prior matrix

7.23.2.6 MVnormBeta() [6/12]

```
MVnormBeta::MVnormBeta (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )
```

Multivariate constructor with response matrix.

Sets up _vec to point to a row in the provided matrix of regression coefficients, and the predictor to a column of the matrix of predictors. Initializes regression coefficients using the provided response matrix.

Parameters

in	*gsl_matrix*∗	response matrix
in	*gsl_matrix*∗	predictor martix

Parameters

in	*size_t&*	column index of the predictor matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.7 MVnormBeta() [7/12]

```
MVnormBeta::MVnormBeta (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )
```

Multivariate constructor with response matrix and index for a prior.

Sets up _vec to point to a row in the provided matrix of regression coefficients, and the predictor to a column of the matrix of predictors. Initializes regression coefficients using the provided response matrix.
Initializes _upLevel to point to the appropriate row in the matrix of priors.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor martix
in	*size_t&*	column index of the predictor matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.8 MVnormBeta() [8/12]

```
MVnormBeta::MVnormBeta (
            const Grp & resp,
```

```
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )
```

Multivariate constructor with a Grp type response.

Sets up _vec to point to a row in the provided matrix of regression coefficients, and the predictor to a column of the matrix of predictors. Initializes regression coefficients using the provided response matrix.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor martix
in	*size_t&*	column index of the predictor matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.9 MVnormBeta() [9/12]

```
MVnormBeta::MVnormBeta (
            const Grp & resp,
            gsl_matrix * pred,
            const size_t & iCl,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )
```

Multivariate constructor with a Grp type response and index for a prior.

Sets up _vec to point to a row in the provided matrix of regression coefficients, and the predictor to a column of the matrix of predictors. Initializes regression coefficients using the provided response matrix. Initializes _upLevel to point to the appropriate row in the matrix of priors.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor martix
in	*size_t&*	column index of the predictor matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.10 MVnormBeta() [10/12]

```
MVnormBeta::MVnormBeta (
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * bet,
            const size_t & iRw )
```

Deterministic constructor.

Does not initialize the regression coefficients, but simply points to the already-initialized matrix of values.

Parameters

in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the predictor matrix
in	*gsl_matrix∗*	regression coefficient matrix
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.11 MVnormBeta() [11/12]

```
MVnormBeta::MVnormBeta (
            gsl_matrix * pred,
            const size_t & iCl,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )
```

Deterministic constructor with prior index.

Does not initialize the regression coefficients, but simply points to the already-initialized matrix of values. Sets the _upLevel to point to the corresponding row of the prior matrix

Parameters

in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the predictor matrix
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	regression coefficient matrix
in	*size_t&*	row index of the regression coefficient matrix

7.23.2.12 MVnormBeta() [12/12]

```
MVnormBeta::MVnormBeta (
            const MVnormBeta & b )
```

Deterministic copy constructor.

Parameters

in	*MVnormBeta&*	object to be copied

7.23.3 Member Function Documentation

7.23.3.1 operator=()

```
MVnormBeta & MVnormBeta::operator= (
            const MVnormBeta & b )
```

Deterministic assignement operator.

Parameters

in	*MVnormBeta&*	object to be assigned

Returns

A reference to a MVnormBeta object

7.23.3.2 scalePar()

```
double MVnormBeta::scalePar ( ) const  [inline], [virtual]
```

Scale parameter.

Access to a scale parameter member, defined only for some derived regression-type classes. If not defined, returns 1.0.

Returns

Scale paramter value of type *double*.

Reimplemented from MVnorm.

7.23.3.3 up()

```
const size_t* MVnormBeta::up ( ) const  [inline], [virtual]
```

Points to the prior.

Access to the pointer to the correspoding vector of priors. Is non-zero only for classes where a prior is implemented.

Returns

Pointer to *size_t*.

Reimplemented from MVnorm.

7.23.3.4 update() [1/10]

```
void MVnormBeta::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.5 update() [2/10]

```
void MVnormBeta::update (
            const Grp & ,
```

```
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.6 update() [3/10]

```
void MVnormBeta::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.7 update() [4/10]

```
void MVnormBeta::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.8 update() [5/10]

```
void MVnormBeta::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.9 update() [6/10]

```
void MVnormBeta::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.10 update() [7/10]

```
void MVnormBeta::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.11 update() [8/10]

```
void MVnormBeta::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.12 update() [9/10]

```
void MVnormBeta::update (
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.3.13 update() [10/10]

```
void MVnormBeta::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFt.

7.23.4 Member Data Documentation

7.23.4.1 _scale

```
double MVnormBeta::_scale  [protected]
```

Scale parameter.

Typically $x^{\intercal}x$, but some exceptions exist in special cases.

7.23.4.2 _upLevel

```
const size_t* MVnormBeta::_upLevel  [protected]
```

Row index of the prior.

Points to the index of a row in the prior matrix

7.23.4.3 _X

```
gsl_vector_view MVnormBeta::_X  [protected]
```

Predictor.

Points to a vector of predictor values, typically a column of the predictor matrix that is in the encapsulating Grp class.

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

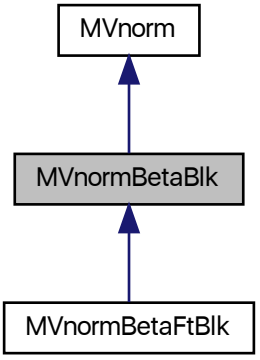
- MuGen.h
- MuGen.cpp

7.24 MVnormBetaBlk Class Reference

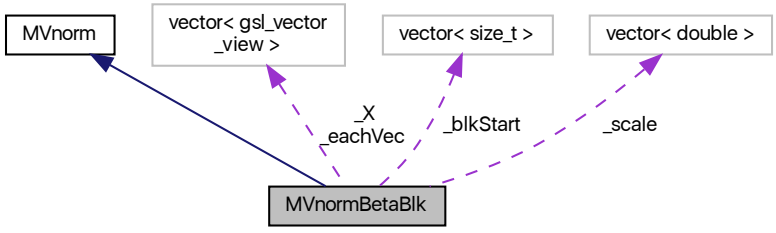
Individual vector of regression coefficients with blocks of traits.

```
#include <MuGen.h>
```

Inheritance diagram for MVnormBetaBlk:



Collaboration diagram for MVnormBetaBlk:



Public Member Functions

- MVnormBetaBlk ()

  Default constructor.
- MVnormBetaBlk (const Grp &resp, gsl_matrix *pred, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Constructor with no prior index.
- MVnormBetaBlk (const Grp &resp, gsl_matrix *pred, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Constructor with a prior index.
- MVnormBetaBlk (const gsl_matrix *resp, gsl_matrix *pred, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Constructor with no prior index.
- MVnormBetaBlk (const gsl_matrix *resp, gsl_matrix *pred, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Constructor with a prior index.
- virtual ~MVnormBetaBlk ()

  Destructor.
- const size_t * up () const

  Points to the prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const gsl_rng *r)

  Gaussian likelihood.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const gsl_rng *r)

  Sudent-t likelihood.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- const vector< size_t > * _blkStart

  Start positions of blocks.
- vector< gsl_vector_view > _eachVec

  Trait blocks.
- vector< gsl_vector_view > _X

  Separate predictors.
- vector< double > _scale

  Vector of scales.
- const size_t * _upLevel

  Pointer to a row index of the prior.

Additional Inherited Members

7.24.1 Detailed Description

Individual vector of regression coefficients with blocks of traits.

Implements separate regression models for blocks of traits. The likelihood covariance matrix is block-diagonal. Traits of the same block have to be contiguous within the vector.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 MVnormBetaBlk() [1/5]

```
MVnormBetaBlk::MVnormBetaBlk ( )  [inline]
```

Default constructor.

Simply calls the MVnorm default constructor.

7.24.2.2 MVnormBetaBlk() [2/5]

```
MVnormBetaBlk::MVnormBetaBlk (
            const Grp & resp,
            gsl_matrix * pred,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )
```

Constructor with no prior index.

Stochastic constructor for a regression with an improper prior.

Parameters

in	*Grp&*	response variable
in	*gsl_matrix∗*	predictor matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.24.2.3 MVnormBetaBlk() [3/5]

```
MVnormBetaBlk::MVnormBetaBlk (
            const Grp & resp,
            gsl_matrix * pred,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )
```

Constructor with a prior index.

Stochastic constructor for a regression with a proper prior.

Parameters

in	*Grp&*	response variable
in	*gsl_matrix∗*	predictor matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.24.2.4 MVnormBetaBlk() [4/5]

```
MVnormBetaBlk::MVnormBetaBlk (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )
```

Constructor with no prior index.

Stochastic constructor for a regression with an improper prior.

Parameters

in	*gsl_matrix*∗	response matrix
in	*gsl_matrix*∗	predictor matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix*∗	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng*∗	pointer to a PNG
in	*gsl_matrix*∗	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.24.2.5 MVnormBetaBlk() [5/5]

```
MVnormBetaBlk::MVnormBetaBlk (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )
```

Constructor with a prior index.

Stochastic constructor for a regression with a proper prior.

Parameters

in	*gsl_matrix*∗	response matrix

Parameters

in	*gsl_matrix∗*	predictor matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.24.3 Member Function Documentation

7.24.3.1 up()

```
const size_t* MVnormBetaBlk::up ( ) const  [inline], [virtual]
```

Points to the prior.

Access to the pointer to the correspoding vector of priors. Is non-zero only for classes where a prior is implemented.

Returns

Pointer to *size_t*.

Reimplemented from MVnorm.

7.24.3.2 update() [1/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.3 update() [2/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.4 update() [3/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
```

```
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.5 update() [4/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.6 update() [5/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.7 update() [6/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.8 update() [7/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng*$*$	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.9 update() [8/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng*$*$	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.10 update() [9/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.3.11 update() [10/10]

```
void MVnormBetaBlk::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaFtBlk.

7.24.4 Member Data Documentation

7.24.4.1 _blkStart

```
const vector< size_t >* MVnormBetaBlk::_blkStart  [protected]
```

Start positions of blocks.

Pointer to the vector that's in the corresponding Grp class.

7.24.4.2 _eachVec

```
vector< gsl_vector_view > MVnormBetaBlk::_eachVec  [protected]
```

Trait blocks.

Vector views of vector pieces, each corresponding to a block of variables. Pointer to the vector that's in the corresponding Grp class.

7.24.4.3 _scale

```
vector<double> MVnormBetaBlk::_scale  [protected]
```

Vector of scales.

Typically $X^T X$, but can bet something else in special cases. Each _X has a correspodning _scale.

7.24.4.4 _upLevel

```
const size_t* MVnormBetaBlk::_upLevel  [protected]
```

Pointer to a row index of the prior.

This has to be the same for all the blocks.

7.24.4.5 _X

```
vector< gsl_vector_view > MVnormBetaBlk::_X  [protected]
```

Separate predictors.

Vector views of predictor vectors, each corresponding to a block of variables. The predictor matrix is in the encompassing Grp class.

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

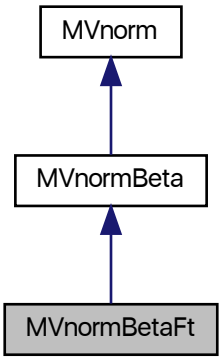
- MuGen.h
- MuGen.cpp

7.25 MVnormBetaFt Class Reference

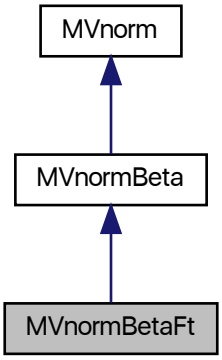
Regression with multiple predictors.

`#include <MuGen.h>`

Inheritance diagram for MVnormBetaFt:



Collaboration diagram for MVnormBetaFt:



Public Member Functions

- MVnormBetaFt ()

  Default constructor.
- MVnormBetaFt (gsl_vector *b, const gsl_vector *sd, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_rng *r)

  Univariate random constructor with vector.
- MVnormBetaFt (gsl_vector *b, const gsl_vector *sd, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_rng *r, const size_t &up)

  Univariate random constructor with vector and prior index.
- MVnormBetaFt (gsl_vector *b, const gsl_matrix *Sig, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_rng *r)

  Multivariate random constructor with vector.
- MVnormBetaFt (gsl_vector *b, const gsl_matrix *Sig, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_rng *r, const size_t &up)

  Multivariate random constructor with vector and prior index.
- MVnormBetaFt (const gsl_matrix *resp, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_matrix *Sig, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with response matrix.
- MVnormBetaFt (const gsl_matrix *resp, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_matrix *Sig, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with response matrix and prior index.
- MVnormBetaFt (const gsl_matrix *resp, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const gsl_matrix *Sig, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with response matrix.
- MVnormBetaFt (const gsl_matrix *resp, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const gsl_matrix *Sig, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with response matrix and prior index.
- MVnormBetaFt (const Grp &resp, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_matrix *Sig, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with Grp type response.
- MVnormBetaFt (const Grp &resp, gsl_matrix *pred, const size_t &iCl, gsl_matrix *allFt, const size_t &begRw, const gsl_matrix *Sig, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with Grp type response and prior index.
- MVnormBetaFt (const Grp &resp, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const gsl_matrix *Sig, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with Grp type response.
- MVnormBetaFt (const Grp &resp, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const gsl_matrix *Sig, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Multivariate random constructor with Grp type response and prior index.
- MVnormBetaFt (gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, gsl_matrix *bet, const size_t &iRw)

  Deterministic constructor.
- MVnormBetaFt (gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Deterministic constructor with a prior index.
- MVnormBetaFt (const MVnormBetaFt &)

  Deterministic copy constructor.
- MVnormBetaFt & operator= (const MVnormBetaFt &)

  Deterministic assignment operator.

- virtual ~MVnormBetaFt ()

  Destructor.
- void update (const Grp &resp, const SigmaI &SigIb, const gsl_rng *r)

  Gaussian likelihood.
- void update (const Grp &resp, const Qgrp &q, const SigmaI &SigIb, const gsl_rng *r)

  Sudent-t likelihood.
- void update (const Grp &resp, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &resp, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- void update (const Grp &resp, const Qgrp &q, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- void update (const Grp &resp, const Qgrp &q, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- void update (const Grp &resp, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &resp, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- void update (const Grp &resp, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- void update (const Grp &resp, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- gsl_matrix_view _fitted

  Matrix of already fitted values.

Additional Inherited Members

7.25.1 Detailed Description

Regression with multiple predictors.

Full implementation of a multiple regression using one-at-a-time updating. Updates regression coefficients for the current predictor, while accounting for the effects of other predictors

7.25.2 Constructor & Destructor Documentation

7.25.2.1 MVnormBetaFt() [1/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_vector * b,
            const gsl_vector * sd,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_rng * r )  [inline]
```

Univariate random constructor with vector.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices.

Parameters

in	*gsl_vector*∗	vector of values
in	*gsl_vector*∗	vector of standard deviations
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix*∗	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_rng*∗	pointer to a PNG

7.25.2.2 MVnormBetaFt() [2/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_vector * b,
            const gsl_vector * sd,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_rng * r,
            const size_t & up )  [inline]
```

Univariate random constructor with vector and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*gsl_vector*∗	vector of values

Parameters

in	*gsl_vector*∗	vector of standard deviations
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix*∗	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_rng*∗	pointer to a PNG
in	*size_t&*	row index of the prior matrix

7.25.2.3 MVnormBetaFt() [3/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_vector * b,
            const gsl_matrix * Sig,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_rng * r )  [inline]
```

Multivariate random constructor with vector.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices.

Parameters

in	*gsl_vector*∗	vector of values
in	*gsl_matrix*∗	covariance matrix
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix*∗	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_rng*∗	pointer to a PNG

7.25.2.4 MVnormBetaFt() [4/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_vector * b,
            const gsl_matrix * Sig,
```

```
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_rng * r,
            const size_t & up )  [inline]
```

Multivariate random constructor with vector and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*gsl_vector*∗	vector of values
in	*gsl_matrix*∗	covariance matrix
in	*gsl_matrix*∗	matrix of predictors
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix*∗	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_rng*∗	pointer to a PNG
in	*size_t&*	row index of the prior matrix

7.25.2.5 MVnormBetaFt() [5/15]

```
MVnormBetaFt::MVnormBetaFt (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with response matrix.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices.

Parameters

in	*gsl_matrix*∗	response matrix
in	*gsl_matrix*∗	predictor matrix

Parameters

in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix∗*	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.6 MVnormBetaFt() [6/15]

```
MVnormBetaFt::MVnormBetaFt (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with response matrix and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated sub-matrix of a matrix that contains all the partial fitted matrices. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix∗*	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.7 MVnormBetaFt() [7/15]

```
MVnormBetaFt::MVnormBetaFt (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with response matrix.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated vector that has the appropriate partial fitted values.

Parameters

in	*gsl_matrix*∗	response matrix
in	*gsl_matrix*∗	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*vector*<*double*>*&*	vectorized partial fitted matrix
in	*gsl_matrix*∗	covariance matrix
in	*gsl_rng*∗	pointer to a PNG
in	*gsl_matrix*∗	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.8 MVnormBetaFt() [8/15]

```
MVnormBetaFt::MVnormBetaFt (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with response matrix and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated vector that has the appropriate partial fitted values. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.9 MVnormBetaFt() [9/15]

```
MVnormBetaFt::MVnormBetaFt (
            const Grp & resp,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with Grp type response.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated submatrix of a matrix that contains all the partial fitted matrices.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix∗*	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.10 MVnormBetaFt() [10/15]

```
MVnormBetaFt::MVnormBetaFt (
            const Grp & resp,
            gsl_matrix * pred,
            const size_t & iCl,
            gsl_matrix * allFt,
            const size_t & begRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with Grp type response and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated sub-matrix of a matrix that contains all the partial fitted matrices. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*gsl_matrix∗*	matrix with all the partial fitted matrices stacked
in	*size_t&*	index of the row where the relevant fitted matrix begins
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.11 MVnormBetaFt() [11/15]

```
MVnormBetaFt::MVnormBetaFt (
            const Grp & resp,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with Grp type response.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated vector that has the appropriate partial fitted values.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.12 MVnormBetaFt() [12/15]

```
MVnormBetaFt::MVnormBetaFt (
            const Grp & resp,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Multivariate random constructor with Grp type response and prior index.

Sets up _vec to point to the provided vector of values, and the predictor to a column in the provided matrix of predictors. Initializes regression coefficients using the provided response matrix. The _fitted member points to the indicated vector that has the appropriate partial fitted values. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*Grp&*	response
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index for the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.13 MVnormBetaFt() [13/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Deterministic constructor.

Does not initialize the regression coefficients, but simply points to the already-initialized matrix of values. The _fitted member points to the indicated vector that has the appropriate partial fitted values.

Parameters

in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*gsl_matrix∗*	regression coefficient matrix
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.14 MVnormBetaFt() [14/15]

```
MVnormBetaFt::MVnormBetaFt (
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Deterministic constructor with a prior index.

Does not initialize the regression coefficients, but simply points to the already-initialized matrix of values. The _fitted member points to the indicated vector that has the appropriate partial fitted values. The _upLevel member is set to point to a row in the prior matrix.

Parameters

in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	regression coefficient matrix
in	*size_t&*	row index of the regression coefficient matrix

7.25.2.15 MVnormBetaFt() [15/15]

```
MVnormBetaFt::MVnormBetaFt (
            const MVnormBetaFt & b )
```

Deterministic copy constructor.

Parameters

in	*MVnormBetaFt&*	object to be copied

7.25.3 Member Function Documentation

7.25.3.1 operator=()

```
MVnormBetaFt & MVnormBetaFt::operator= (
            const MVnormBetaFt & b )
```

Deterministic assignment operator.

Parameters

in	*MVnormBetaFt&*	object to be copied

Returns

Refernce to an object of class MVnormBetaFt

7.25.3.2 update() [1/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.3 update() [2/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.4 update() [3/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const Qgrp & ,
```

```
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.5 update() [4/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.6 update() [5/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const Qgrp & ,
```

```
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.7 update() [6/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.8 update() [7/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const SigmaI & ,
```

```
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.9 update() [8/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.10 update() [9/10]

```
void MVnormBetaFt::update (
            const Grp & ,
```

```
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.3.11 update() [10/10]

```
void MVnormBetaFt::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBeta.

7.25.4 Member Data Documentation

7.25.4.1 _fitted

```
gsl_matrix_view MVnormBetaFt::_fitted  [protected]
```

Matrix of already fitted values.

$\boldsymbol{XB}$ matrix for all predictors but the current one (i.e., $\boldsymbol{X}_{\cdot -j}\boldsymbol{B}_{-j\cdot}$). It is subtracted from the response and the regression is performed on the residual.

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

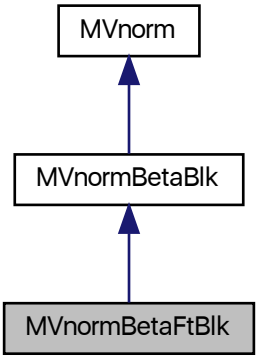
- MuGen.h
- MuGen.cpp

7.26 MVnormBetaFtBlk Class Reference

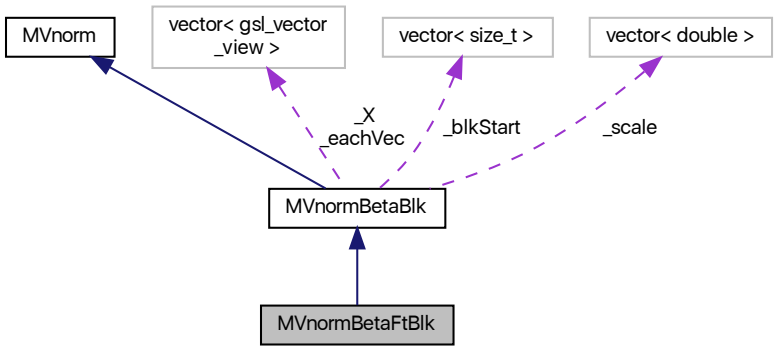
Individual vector of conditional regression coefficients with blocks of traits.

`#include <MuGen.h>`

Inheritance diagram for MVnormBetaFtBlk:



Collaboration diagram for MVnormBetaFtBlk:



Public Member Functions

- MVnormBetaFtBlk ()

  Default constructor.
- MVnormBetaFtBlk (const Grp &resp, gsl_matrix *pred, vector< double > &eaFt, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Constructor with no prior index.
- MVnormBetaFtBlk (const Grp &resp, gsl_matrix *pred, vector< double > &eaFt, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Constructor with a prior index.
- MVnormBetaFtBlk (const gsl_matrix *resp, gsl_matrix *pred, vector< double > &eaFt, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, gsl_matrix *bet, const size_t &iRw)

  Constructor with no prior index.
- MVnormBetaFtBlk (const gsl_matrix *resp, gsl_matrix *pred, vector< double > &eaFt, const vector< size_t > &iCl, const gsl_matrix *Sig, const vector< size_t > &blkStart, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw)

  Constructor with a prior index.
- ~MVnormBetaFtBlk ()

  Destructor.
- void update (const Grp &dat, const SigmaI &SigIb, const gsl_rng *r)

  Gaussian likelihood.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const gsl_rng *r)

  Sudent-t likelihood.
- void update (const Grp &dat, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIb, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- gsl_matrix_view _fitted

  Matrix of fitted values.

Additional Inherited Members

7.26.1 Detailed Description

Individual vector of conditional regression coefficients with blocks of traits.

Implements separate regression models for blocks of traits. The likelihood covariance matrix is block-diagonal. Traits of the same block have to be contiguous within the vector. The regression is conditional on other regression coefficient estimates within a group (i.e., one-at-a-time updating).

7.26.2 Constructor & Destructor Documentation

7.26.2.1 MVnormBetaFtBlk() [1/5]

```
MVnormBetaFtBlk::MVnormBetaFtBlk ( )  [inline]
```

Default constructor.

Simply calls the default constructor of the MVnormBetaBlk class. Pointer to the fitted matrix is not set to anything.

7.26.2.2 MVnormBetaFtBlk() [2/5]

```
MVnormBetaFtBlk::MVnormBetaFtBlk (
            const Grp & resp,
            gsl_matrix * pred,
            vector< double > & eaFt,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Constructor with no prior index.

Stochastic constructor for a regression with an improper prior.

Parameters

in	*Grp&*	response variable
in	*gsl_matrix∗*	predictor matrix
in	*vector<double>&*	vectorized fitted matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.26.2.3 MVnormBetaFtBlk() [3/5]

```
MVnormBetaFtBlk::MVnormBetaFtBlk (
            const Grp & resp,
            gsl_matrix * pred,
            vector< double > & eaFt,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Constructor with a prior index.

Stochastic constructor for a regression with a proper prior.

Parameters

in	*Grp&*	response variable
in	*gsl_matrix∗*	predictor matrix
in	*vector<double>&*	vectorized fitted matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.26.2.4 MVnormBetaFtBlk() [4/5]

```
MVnormBetaFtBlk::MVnormBetaFtBlk (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            vector< double > & eaFt,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Constructor with no prior index.

Stochastic constructor for a regression with an improper prior.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*vector<double>&*	vectorized fitted matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.26.2.5 MVnormBetaFtBlk() [5/5]

```
MVnormBetaFtBlk::MVnormBetaFtBlk (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            vector< double > & eaFt,
            const vector< size_t > & iCl,
            const gsl_matrix * Sig,
            const vector< size_t > & blkStart,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw )  [inline]
```

Constructor with a prior index.

Stochastic constructor for a regression with a proper prior.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*vector<double>&*	vectorized fitted matrix
in	*vector<size_t>&*	column indexes of the predictor matrix
in	*gsl_matrix∗*	covariance marix for stochastic initiation
in	*vector<size_t>&*	vector of start indixes of each block
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index of the prior matrix
in	*gsl_matrix∗*	matrix of regression coefficients
in	*size_t&*	row index of the regression coefficient matrix

7.26.3 Member Function Documentation

7.26.3.1 update() [1/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.2 update() [2/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood

Parameters

in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.3 update() [3/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.4 update() [4/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.5 update() [5/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.6 update() [6/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.7 update() [7/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.8 update() [8/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.9 update() [9/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.3.10 update() [10/10]

```
void MVnormBetaFtBlk::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormBetaBlk.

7.26.4 Member Data Documentation

7.26.4.1 _fitted

```
gsl_matrix_view MVnormBetaFtBlk::_fitted  [protected]
```

Matrix of fitted values.

$\boldsymbol{XB}$ matrix for all predictors but the current one (i.e., $\boldsymbol{X}_{\cdot -j}\boldsymbol{B}_{-j\cdot}$). It is subtracted from the response and the regression is performed on the residual.

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

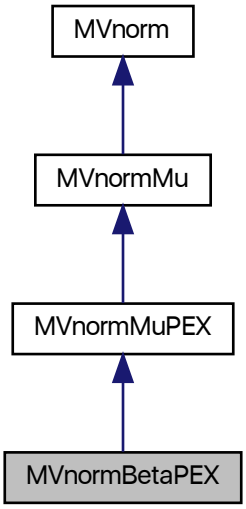
- MuGen.h
- MuGen.cpp

7.27 MVnormBetaPEX Class Reference

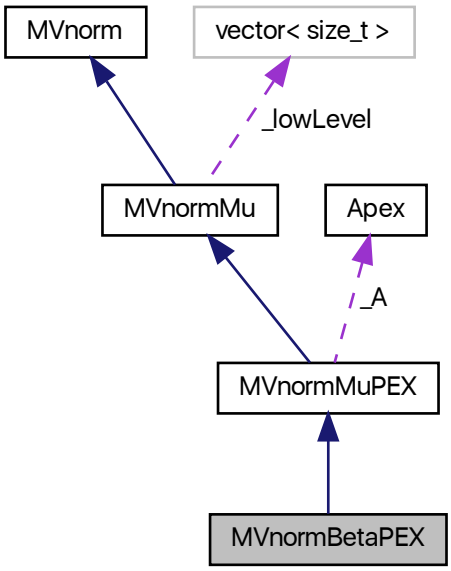
Individual regression with parameter expansion.

```
#include <MuGen.h>
```

Inheritance diagram for MVnormBetaPEX:



Collaboration diagram for MVnormBetaPEX:



Public Member Functions

- MVnormBetaPEX ()

  Default constructor.
- MVnormBetaPEX (const gsl_matrix *resp, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const gsl_matrix *Sig, const gsl_rng *r, const size_t &up, gsl_matrix *bet, const size_t &iRw, Apex &A, gsl_matrix *tSigIAt)

  Random constructor.
- MVnormBetaPEX (const size_t &d, gsl_matrix *pred, const size_t &iCl, vector< double > &eaFt, const size_t &up, gsl_matrix *bet, const size_t &iRw, Apex &A, gsl_matrix *tSigIAt)

  Deterministic constructor.
- MVnormBetaPEX (const MVnormBetaPEX &bet)

  Deterministic copy constructor.
- MVnormBetaPEX & operator= (const MVnormBetaPEX &bet)

  Assignment operator.
- ~MVnormBetaPEX ()

  Destructor.
- void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

Student-t likelihood, Gaussian prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.

- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- gsl_vector_view _X

  Predictor.

- double _scale

  Scale parameter.

- size_t _N

  Length of the predictor.

- gsl_matrix_view _fitted

  Fitted values.

Additional Inherited Members

7.27.1 Detailed Description

Individual regression with parameter expansion.

Implements handling of one-at-a-time regressions, as in MVnormBetaFt but with parameter expansion.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 MVnormBetaPEX() [1/3]

```
MVnormBetaPEX::MVnormBetaPEX (
            const gsl_matrix * resp,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw,
            Apex & A,
            gsl_matrix * tSigIAt )
```

Random constructor.

Calculates initial values for the regression using the provided responce and covariance matrix. The initial values modify the corresponding row of the value matrix, stored in the encapsulating class.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the current predictor in the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	row index for the prior matrix
in	*gsl_matrix∗*	value matrix
in	*size_t&*	row index of the value matrix
in	*Apex&*	object storing the redundant parameter matrix
in	*gsl_matrix∗*	pre-computed auxiliary matrix

7.27.2.2 MVnormBetaPEX() [2/3]

```
MVnormBetaPEX::MVnormBetaPEX (
            const size_t & d,
            gsl_matrix * pred,
            const size_t & iCl,
            vector< double > & eaFt,
            const size_t & up,
            gsl_matrix * bet,
            const size_t & iRw,
            Apex & A,
            gsl_matrix * tSigIAt )
```

Deterministic constructor.

Sets up the proper pointers to parameters of the regression that have already been calculated in the encapsulating class. The corresponding value matrix is not modified.

Parameters

in	*gsl_matrix∗*	response matrix
in	*gsl_matrix∗*	predictor matrix
in	*size_t&*	column index of the current predictor in the predictor matrix
in	*vector<double>&*	vectorized partial fitted matrix
in	*size_t&*	row index for the prior matrix
in	*gsl_matrix∗*	value matrix
in	*size_t&*	row index of the value matrix
in	*Apex&*	object storing the redundant parameter matrix
in	*gsl_matrix∗*	pre-computed auxiliary matrix

7.27.2.3 MVnormBetaPEX() [3/3]

```
MVnormBetaPEX::MVnormBetaPEX (
            const MVnormBetaPEX & bet )
```

Deterministic copy constructor.

Parameters

in	*MVnormBetaPEX&*	object of type MVnormBetaPEX

7.27.3 Member Function Documentation

7.27.3.1 operator=()

```
MVnormBetaPEX & MVnormBetaPEX::operator= (
            const MVnormBetaPEX & bet )
```

Assignment operator.

Deterministic assignement operator

Parameters

in	*MVnormBetaPEX&*	object of type MVnormBetaPEX

Returns

A reference to an object of type MVnormBetaPEX

7.27.3.2 update() [1/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.3 update() [2/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.4 update() [3/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.5 update() [4/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
```

```
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.6 update() [5/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.7 update() [6/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
```

```
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.8 update() [7/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.3.9 update() [8/8]

```
void MVnormBetaPEX::update (
            const Grp & ,
```

```
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMuPEX.

7.27.4 Member Data Documentation

7.27.4.1 _fitted

```
gsl_matrix_view MVnormBetaPEX::_fitted  [protected]
```

Fitted values.

Points to a matrix of partial fitted values that exclude the current regression ($\boldsymbol{X}_{\cdot -i}\boldsymbol{B}_{-i\cdot}$).

7.27.4.2 _scale

```
double MVnormBetaPEX::_scale  [protected]
```

Scale parameter.

Typically $x^{\mathsf{T}}x$, but some exceptions exist in special cases.

7.27.4.3 _X

```
gsl_vector_view MVnormBetaPEX::_X  [protected]
```

Predictor.

Points to a vector of predictor values, typically a column of the predictor matrix that is in the encapsulating Grp class.

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

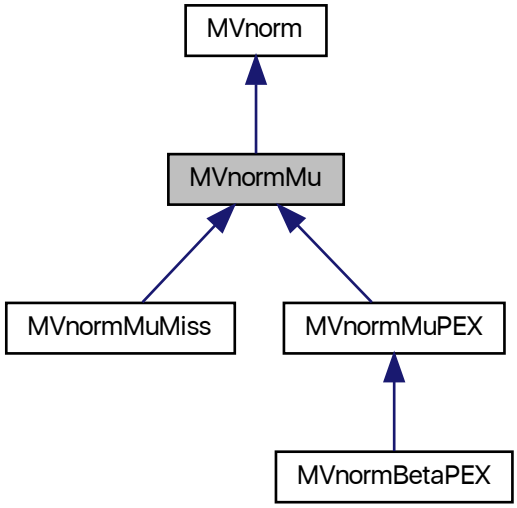
- MuGen.h
- MuGen.cpp

7.28 MVnormMu Class Reference

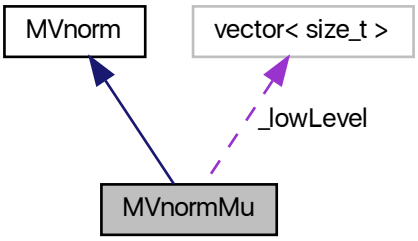
Individual vector of means.

`#include <MuGen.h>`

Inheritance diagram for MVnormMu:



Collaboration diagram for MVnormMu:



Public Member Functions

- MVnormMu ()

  Default constructor.
- MVnormMu (const size_t &d)

  Zero vector constructor.
- MVnormMu (const size_t &d, const vector< size_t > &low, const size_t &up)

  Zero vector with pointers.
- MVnormMu (gsl_vector *mn, const vector< size_t > &low, const size_t &up)

  Deterministic constructor.
- MVnormMu (gsl_vector *mn, const size_t &up)

  Deterministic constructor, prior index only.
- MVnormMu (gsl_matrix *mn, const size_t &iRw)

  Deterministic constructor with a matrix.
- MVnormMu (gsl_matrix *mn, const size_t &iRw, const vector< size_t > &low)

  Deterministic constructor with a matrix and an index to data.
- MVnormMu (gsl_matrix *mn, const size_t &iRw, const vector< size_t > &low, const size_t &up)

  Deterministic constructor with a matrix and indexes to data and a prior.
- MVnormMu (gsl_matrix *mn, const size_t &iRw, const size_t &up)

  Deterministic constructor with a matrix and an index to a prior.
- MVnormMu (gsl_vector *mn, const gsl_vector *sd, const gsl_rng *r, const vector< size_t > &low, const size_t &up)

  Univariate random constructor with a vector and indexes to data and a prior.
- MVnormMu (gsl_vector *mn, const gsl_matrix *Sig, const gsl_rng *r, const vector< size_t > &low, const size_t &up)

  Multivariate random constructor with a vector and indexes to data and a prior.
- MVnormMu (gsl_matrix *mn, const size_t &iRw, const gsl_vector *sd, const gsl_rng *r, const vector< size_t > &low, const size_t &up)

  Univariate random constructor with a matrix and indexes to data and a prior.
- MVnormMu (gsl_matrix *mn, const size_t &iRw, const gsl_matrix *Sig, const gsl_rng *r, const vector< size_t > &low, const size_t &up)

  Multivariate random constructor with a matrix and indexes to data and a prior.
- MVnormMu (const MVnormMu &)

  Copy constructor.
- MVnormMu & operator= (const MVnormMu &)

  Assignment operator.
- virtual ~MVnormMu ()

  Destructor.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const gsl_rng *r)

  Gaussian likelihood.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const gsl_rng *r)

  Sudent- t likelihood.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student- t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

Student-t likelihood, Gaussian prior.

- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.

- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.

- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.

- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

- virtual size_t nMissP () const

  Number of missing values.

- virtual const vector< size_t > getMisPhen () const

  Indexes of missing values.

- const vector< size_t > * down () const

  Points to the corresponding data.

- const size_t * up () const

  Points to the prior.

Protected Attributes

- const vector< size_t > * _lowLevel

  Data indexes.

- const size_t * _upLevel

  Prior index.

Additional Inherited Members

7.28.1 Detailed Description

Individual vector of means.

Refers to a row of a matrix of location parameters that are updated with a likelihood that is an inverse-covariance weighted mean of the data. Variables of this class are typically imbedded in a model hierarchy, and in that case are updated with non-0-mean priors. This class allows us to keep track of which rows in data and prior matrices to use through pointers to index vectors (for data, where there may be more than one row) and to a single index (for the prior mean, since there is only one).
We use pointers because they allow us to modify the indexes in other parts of the model, thus allowing for things like mixture models and model selection schemes.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 MVnormMu() [1/14]

```
MVnormMu::MVnormMu ( )
```

Default constructor.

The _vec member is unassigned and the low-level and upper-level pointers are set to zero.

7.28.2.2 MVnormMu() [2/14]

```
MVnormMu::MVnormMu (
            const size_t & d )  [inline]
```

Zero vector constructor.

Sets _vec to point to a vector of zeros, of length *d*.

Parameters

in	*size_t&*	size of the vector

7.28.2.3 MVnormMu() [3/14]

```
MVnormMu::MVnormMu (
            const size_t & d,
            const vector< size_t > & low,
            const size_t & up )
```

Zero vector with pointers.

Sets _vec to point to a vector of zeros, of length *d*. Pointers to the lower and upper level of the model hierarchy are set.

Parameters

in	*size_t&*	size of the vector
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.4 MVnormMu() [4/14]

```
MVnormMu::MVnormMu (
            gsl_vector * mn,
            const vector< size_t > & low,
            const size_t & up )
```

Deterministic constructor.

Sets _vec to point to a GSL vector of values, without modifying the target during the invocation of the constructor. The pointers to the data and the prior indexes are set.

Parameters

in	*gsl_vector∗*	vector of values
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.5 MVnormMu() [5/14]

```
MVnormMu::MVnormMu (
            gsl_vector * mn,
            const size_t & up )
```

Deterministic constructor, prior index only.

Sets _vec to point to a GSL vector of values, without modifying the target during the invocation of the constructor. Only the pointer to the prior index is set. This constructor can be used for the lowest hierarchy level.

Parameters

in	*gsl_vector∗*	vector of values
in	*size_t&*	index in the matrix of priors

7.28.2.6 MVnormMu() [6/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw )
```

Deterministic constructor with a matrix.

Sets _vec to point to a row of the GSL matrix of values, without modifying the target during the invocation of the constructor.

Parameters

in	*gsl_matrix∗*	matrix of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*

7.28.2.7 MVnormMu() [7/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw,
            const vector< size_t > & low )
```

Deterministic constructor with a matrix and an index to data.

Sets _vec to point to a row of the GSL matrix of values, without modifying the target during the invocation of the constructor. None of the index pointers are set.

Parameters

in	*gsl_matrix∗*	matrix of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*
in	*vector<size_t>&*	vector of indexes of rows in the data matrix

7.28.2.8 MVnormMu() [8/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw,
            const vector< size_t > & low,
            const size_t & up )
```

Deterministic constructor with a matrix and indexes to data and a prior.

Sets _vec to point to a row of the GSL matrix of values, without modifying the target during the invocation of the constructor.

Parameters

in	*gsl_matrix∗*	vector of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.9 MVnormMu() [9/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw,
            const size_t & up )
```

Deterministic constructor with a matrix and an index to a prior.

Sets _vec to point to a row of the GSL matrix of values, without modifying the target during the invocation of the constructor.

Parameters

in	*gsl_matrix∗*	vector of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*
in	*size_t&*	index in the matrix of priors

7.28.2.10 MVnormMu() [10/14]

```
MVnormMu::MVnormMu (
            gsl_vector * mn,
            const gsl_vector * sd,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up )
```

Univariate random constructor with a vector and indexes to data and a prior.

Sets _vec to point to a GSL vector of values, modifying the target during the invocation of the constructor. Modification is by adding independent samples from a univariate normal distribution with the provided standard deviations.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_vector∗*	vector of standard deviations
in	*gsl_rng∗*	pointer to a PNG
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*

7.28.2.11 MVnormMu() [11/14]

```
MVnormMu::MVnormMu (
            gsl_vector * mn,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up )
```

Multivariate random constructor with a vector and indexes to data and a prior.

Sets _vec to point to a GSL vector of values, modifying the target during the invocation of the constructor. Modification is by adding samples from a multivariate normal distribution with the provided covariance matrix.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.12 MVnormMu() [12/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_vector * sd,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up )
```

Univariate random constructor with a matrix and indexes to data and a prior.

Sets _vec to point to a row of the GSL matrix of values, modifying the target during the invocation of the constructor. Modification is by adding independent samples from a univariate normal distribution with the provided standard deviations.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*
in	*gsl_vector*∗	vector of standard deviations
in	*gsl_rng*∗	pointer to a PNG
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.13 MVnormMu() [13/14]

```
MVnormMu::MVnormMu (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up )
```

Multivariate random constructor with a matrix and indexes to data and a prior.

Sets _vec to point to a row of the GSL matrix of values, modifying the target during the invocation of the constructor. Modification is by adding samples from a multivariate normal distribution with the provided covariance matrix.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values, has to be no smaller than 0 and strictly smaller than the number of rows in *mn*
in	*gsl_matrix*∗	covariance matrix
in	*gsl_rng*∗	pointer to a PNG
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors

7.28.2.14 MVnormMu() [14/14]

```
MVnormMu::MVnormMu (
            const MVnormMu & mu )
```

Copy constructor.

Deterministic copy constructor

Parameters

in	*MVnormMu&*	object of type MVnormMu

7.28.3 Member Function Documentation

7.28.3.1 down()

```
const vector<size_t>* MVnormMu::down ( ) const  [inline], [virtual]
```

Points to the corresponding data.

Access to the pointer to a vector that indexes the data used to update an instance of the class. Is implemented for this class.

Returns

Pointer to a vector of *size_t*.

Reimplemented from MVnorm.

7.28.3.2 getMisPhen()

```
virtual const vector<size_t> MVnormMu::getMisPhen ( ) const  [inline], [virtual]
```

Indexes of missing values.

Accesses a vector with indexes of missing phenotypes.

Returns

Vector of *size_t* that contains indexes that correspond to missing values. For classes that do not allow missing values it is empty.

Reimplemented from MVnorm.

Reimplemented in MVnormMuMiss.

7.28.3.3 nMissP()

```
virtual size_t MVnormMu::nMissP ( ) const  [inline], [virtual]
```

Number of missing values.

Accesses the number of missing phenotype values. Non-zero only for classes that implement treatment of missing values.

Returns

Number of missing phenotypes, of type *size_t*.

Reimplemented from MVnorm.

Reimplemented in MVnormMuMiss.

7.28.3.4 operator=()

```
MVnormMu & MVnormMu::operator= (
            const MVnormMu & mu )
```

Assignment operator.

Deterministic assignement operator

Parameters

in	*MVnormMu&*	object of type MVnormMu

Returns

A reference to an object of type MVnormMu

7.28.3.5 up()

```
const size_t* MVnormMu::up ( ) const  [inline], [virtual]
```

Points to the prior.

Access to the pointer to the correspoding vector of priors. Is implemented for this class.

Returns

Pointer to *size_t*.

Reimplemented from MVnorm.

7.28.3.6 update() [1/10]

```
void MVnormMu::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.7 update() [2/10]

```
void MVnormMu::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.8 update() [3/10]

```
void MVnormMu::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.9 update() [4/10]

```
void MVnormMu::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.28.3.10 update() [5/10]

```
void MVnormMu::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.11 update() [6/10]

```
void MVnormMu::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.12 update() [7/10]

```
void MVnormMu::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.13 update() [8/10]

```
void MVnormMu::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormBetaPEX, and MVnormMuPEX.

7.28.3.14 update() [9/10]

```
void MVnormMu::update (
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormMuMiss.

7.28.3.15 update() [10/10]

```
void MVnormMu::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

Reimplemented in MVnormMuMiss, MVnormBetaPEX, and MVnormMuPEX.

7.28.4 Member Data Documentation

7.28.4.1 _lowLevel

```
const vector<size_t>* MVnormMu::_lowLevel  [protected]
```

Data indexes.

A pointer to a vector of *size_t*. The elements are row indexes of the data matrix. Means among these rows are used in the likelihood of the *udpate* functions. The pointer is *const* to make sure we are not modifying it from this class.

7.28.4.2 _upLevel

```
const size_t* MVnormMu::_upLevel  [protected]
```

Prior index.

Pointer to a value of type *size_t* that is a row index of the prior location matrix. The pointer is *const* to make sure we are not modifying it from this class.

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

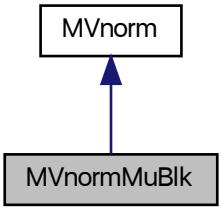
- MuGen.h
- MuGen.cpp

7.29 MVnormMuBlk Class Reference

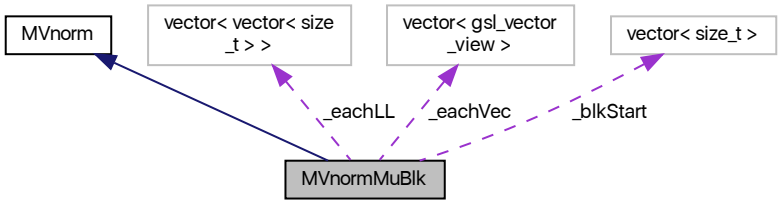
Individual vector of means with blocks of traits.

```
#include <MuGen.h>
```

Inheritance diagram for MVnormMuBlk:



Collaboration diagram for MVnormMuBlk:



Public Member Functions

- MVnormMuBlk ()

  Default constructor.
- MVnormMuBlk (gsl_matrix ∗mn, const size_t &iRw, const vector< size_t > &blkStart, const size_t &up)

  Deterministic constructor.
- MVnormMuBlk (gsl_matrix ∗mn, const size_t &iRw, const gsl_vector ∗sd, const gsl_rng ∗r, const vector< size_t > &blkStart, const vector< vector< size_t > > &eachLL, const size_t &up)

  Univariate stochastic constructor.
- virtual ∼MVnormMuBlk ()

  Destructor.
- const size_t ∗ up () const

  Points to the prior.
- void update (const Grp &dat, const SigmaI &SigIm, const gsl_rng ∗r)

  Gaussian likelihood.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const gsl_rng ∗r)

  Sudent- t likelihood.
- void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng ∗r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng ∗r)

  Gaussian likelihood, Student- t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng ∗r)

  Student- t likelihood, Gaussian prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng ∗r)

  Student- t likelihood, Student- t prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng ∗r)

  Gaussian likelihood, Gaussian prior.
- void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng ∗r)

  Gaussian likelihood, Student- t prior.
- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng ∗r)

Student-t likelihood, Gaussian prior.

- void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- const size_t * _upLevel

  Pointer to a row index of the prior.
- const vector< size_t > * _blkStart

  Start positions of blocks.
- vector< gsl_vector_view > _eachVec

  Trait blocks.
- const vector< vector< size_t > > * _eachLL

  Data matrix row indexes.

Additional Inherited Members

7.29.1 Detailed Description

Individual vector of means with blocks of traits.

Implements separate models for blocks of traits. The likelihood covariance matrix is block-diagonal. Traits of the same block have to be contiguous within the vector.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 MVnormMuBlk() [1/2]

```
MVnormMuBlk::MVnormMuBlk (
            gsl_matrix * mn,
            const size_t & iRw,
            const vector< size_t > & blkStart,
            const size_t & up )
```

Deterministic constructor.

Sets up the member variables to point to blocks of traits in the matrix *mn*, but does not perform stochastic intitialization.

Parameters

in	*gsl_matrix**	mean values matrix
in	*size_t&*	row index of the mean values matrix
in	*vector<size_t>&*	vector of start indexes for each block
in	*size_t&*	row index of the prior matrix

7.29.2.2 MVnormMuBlk() [2/2]

```
MVnormMuBlk::MVnormMuBlk (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_vector * sd,
            const gsl_rng * r,
            const vector< size_t > & blkStart,
            const vector< vector< size_t > > & eachLL,
            const size_t & up )
```

Univariate stochastic constructor.

Sets initial values independently for each trait, modifying the matrix row that corresponds to this vector.

Parameters

in	*gsl_matrix*∗	mean values matrix
in	*size_t&*	row index of the mean values matrix
in	*gsl_vector*∗	vector of standard deviations
in	*gsl_rng*∗	pointer to a PNG
in	*vector*<*size_t*>*&*	vector of start indexes for each block
in	*vector*<	vector<size_t> >& vector of lower-level index vectors
in	*size_t&*	row index of the prior matrix

7.29.3 Member Function Documentation

7.29.3.1 up()

```
const size_t* MVnormMuBlk::up ( ) const  [inline], [virtual]
```

Points to the prior.

Access to the pointer to the correspoding vector of priors. Is non-zero only for classes where a prior is implemented.

Returns

Pointer to *size_t*.

Reimplemented from MVnorm.

7.29.3.2 update() [1/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.3 update() [2/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.4 update() [3/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.5 update() [4/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Sudent- t likelihood.

Parameters

in	*Grp&*	data
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.6 update() [5/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng*∗	pointer to a PNG

Implements MVnorm.

7.29.3.7 update() [6/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng*∗	pointer to a PNG

Implements MVnorm.

7.29.3.8 update() [7/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.9 update() [8/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.10 update() [9/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.3.11 update() [10/10]

```
void MVnormMuBlk::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Implements MVnorm.

7.29.4 Member Data Documentation

7.29.4.1 _blkStart

```
const vector< size_t >* MVnormMuBlk::_blkStart  [protected]
```

Start positions of blocks.

Pointer to the vector that's in the corresponding Grp class.

7.29.4.2 _eachLL

```
const vector< vector<size_t> >* MVnormMuBlk::_eachLL  [protected]
```

Data matrix row indexes.

Each block of traits can have a different number and identity of rows in the data matrix it refers to. Each vector of *size_t* in this vector corresponds to a block, so the size of this should be the same as the size of *_eachVec*. This member is a pointer to the vector that's in the corresponding Grp class.

7.29.4.3 _eachVec

```
vector< gsl_vector_view > MVnormMuBlk::_eachVec  [protected]
```

Trait blocks.

Vector views of vector pieces, each corresponding to a block of variables. Pointer to the vector that's in the corresponding Grp class.

7.29.4.4 _upLevel

```
const size_t* MVnormMuBlk::_upLevel  [protected]
```

Pointer to a row index of the prior.

This has to be the same for all the blocks.

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

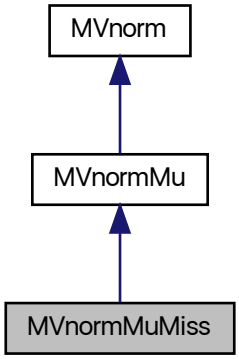
- MuGen.h
- MuGen.cpp

7.30 MVnormMuMiss Class Reference

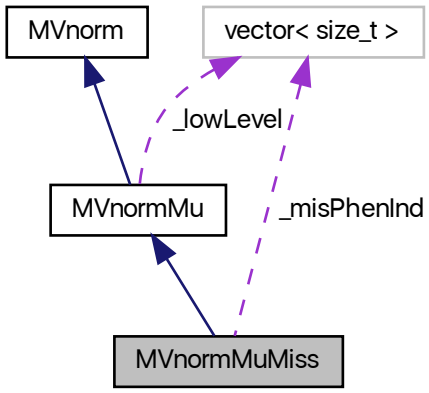
Individual vector of means with missing data.

`#include <MuGen.h>`

Inheritance diagram for MVnormMuMiss:



Collaboration diagram for MVnormMuMiss:



Public Member Functions

- MVnormMuMiss ()

  Default constructor.
- MVnormMuMiss (const size_t &d)

  0-mean deterministic constructor
- MVnormMuMiss (const size_t &d, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Deterministic constructor.
- MVnormMuMiss (gsl_vector *mn, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Deterministic constructor with a vector.
- MVnormMuMiss (gsl_vector *mn, const size_t &up, const vector< size_t > &mis)

  Deterministic constructor with a vector.
- MVnormMuMiss (gsl_vector *mn, const gsl_vector *sd, const gsl_rng *r, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Univariate random constructor with a vector.
- MVnormMuMiss (gsl_vector *mn, const gsl_matrix *Sig, const gsl_rng *r, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Multivariate random constructor with a vector.
- MVnormMuMiss (gsl_matrix *mn, const size_t &iRw, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Deterministic constructor with a matrix.
- MVnormMuMiss (gsl_matrix *mn, const size_t &iRw, const size_t &up, const vector< size_t > &mis)

  Deterministic constructor with a matrix.
- MVnormMuMiss (gsl_matrix *mn, const size_t &iRw, const gsl_vector *sd, const gsl_rng *r, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Univariate random constructor with a matrix.
- MVnormMuMiss (gsl_matrix *mn, const size_t &iRw, const gsl_matrix *Sig, const gsl_rng *r, const vector< size_t > &low, const size_t &up, const vector< size_t > &mis)

  Multivariate random constructor with a matrix.
- MVnormMuMiss (const MVnormMuMiss &)

  Deterministic copy constructor.
- MVnormMuMiss & operator= (const MVnormMuMiss &)

  Assignment operator.
- ~MVnormMuMiss ()

  Destructor.
- void update (const Grp &mu, const SigmaI &SigIm, const gsl_rng *r)

  Gaussian likelihood.
- void update (const Grp &mu, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian 0-mean prior.
- size_t nMissP () const

  Number of missing values.
- const vector< size_t > getMisPhen () const

  Indexes of missing values.

Protected Attributes

- vector< size_t > _misPhenInd

  Missing data index.
- size_t _myInd

  own index

Additional Inherited Members

7.30.1 Detailed Description

Individual vector of means with missing data.

Implements missing phenotype data imputation. Some parameters for update methods have a different meaning than for other MVnorm classes.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 MVnormMuMiss() [1/11]

```
MVnormMuMiss::MVnormMuMiss (
            const size_t & d )  [inline]
```

0-mean deterministic constructor

Sets up the instance to point to a vector of zeros

Parameters

in	*size_t&*	dimension of the vector

7.30.2.2 MVnormMuMiss() [2/11]

```
MVnormMuMiss::MVnormMuMiss (
            const size_t & d,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Deterministic constructor.

Parameters

in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.3 MVnormMuMiss() [3/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_vector * mn,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Deterministic constructor with a vector.

Parameters

in	*gsl_vector∗*	vector of values
in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.4 MVnormMuMiss() [4/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_vector * mn,
            const size_t & up,
            const vector< size_t > & mis )
```

Deterministic constructor with a vector.

Parameters

in	*gsl_vector∗*	vector of values
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.5 MVnormMuMiss() [5/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_vector * mn,
            const gsl_vector * sd,
            const gsl_rng * r,
```

```
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Univariate random constructor with a vector.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_vector∗*	vector of standard deviations
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.6 MVnormMuMiss() [6/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_vector * mn,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Multivariate random constructor with a vector.

Parameters

in	*gsl_vector∗*	vector of values
in	*gsl_matrix∗*	covariance matrix
in	*gsl_rng∗*	pointer to a PNG
in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.7 MVnormMuMiss() [7/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_matrix * mn,
```

```
            const size_t & iRw,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Deterministic constructor with a matrix.

Parameters

in	*gsl_matrix∗*	vector of values
in	*size_t&*	row index of the matrix of values
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.8 MVnormMuMiss() [8/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_matrix * mn,
            const size_t & iRw,
            const size_t & up,
            const vector< size_t > & mis )
```

Deterministic constructor with a matrix.

Parameters

in	*gsl_matrix∗*	vector of values
in	*size_t&*	row index of the matrix of values
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.9 MVnormMuMiss() [9/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_vector * sd,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Univariate random constructor with a matrix.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values
in	*gsl_vector*∗	vector of standard deviations
in	*gsl_rng*∗	pointer to a PNG
in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.10 MVnormMuMiss() [10/11]

```
MVnormMuMiss::MVnormMuMiss (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_matrix * Sig,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up,
            const vector< size_t > & mis )
```

Multivariate random constructor with a matrix.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values
in	*gsl_matrix*∗	covariance matrix
in	*gsl_rng*∗	pointer to a PNG
in	*size_t&*	dimension of the vector
in	*vector<size_t>&*	vector of row indexes for data
in	*size_t&*	row index for the prior matrix
in	*vector<size_t>&*	indexes of missing data

7.30.2.11 MVnormMuMiss() [11/11]

```
MVnormMuMiss::MVnormMuMiss (
            const MVnormMuMiss &  )
```

Deterministic copy constructor.

Parameters

in	*MVnormMuMiss&*	object of type MVnormMuMiss

7.30.3 Member Function Documentation

7.30.3.1 getMisPhen()

```
const vector<size_t> MVnormMuMiss::getMisPhen ( ) const  [inline], [virtual]
```

Indexes of missing values.

Accesses a vector with indexes of missing phenotypes. Implemented for this class.

Returns

Vector of *size_t* that contains indexes that correspond to missing values.

Reimplemented from MVnormMu.

7.30.3.2 nMissP()

```
size_t MVnormMuMiss::nMissP ( ) const  [inline], [virtual]
```

Number of missing values.

Accesses the number of missing phenotype values. Implemented in this class.

Returns

Number of missing phenotypes, of type *size_t*.

Reimplemented from MVnormMu.

7.30.3.3 operator=()

```
MVnormMuMiss& MVnormMuMiss::operator= (
            const MVnormMuMiss &  )
```

Assignment operator.

Deterministic assignement operator

Parameters

in	*MVnormMuMiss&*	object of type MVnormMuMiss

Returns

A reference to an object of type MVnormMuMiss

7.30.3.4 update() [1/2]

```
void MVnormMuMiss::update (
            const Grp & mu,
            const SigmaI & SigIm,
            const gsl_rng * r )  [virtual]
```

Gaussian likelihood.

Here, the Grp variable contains the means for imputation rather than the data

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

7.30.3.5 update() [2/2]

```
void MVnormMuMiss::update (
            const Grp & mu,
            const SigmaI & SigIm,
            const SigmaI & SigIp,
            const gsl_rng * r )  [virtual]
```

Gaussian likelihood, Gaussian 0-mean prior.

Here, the Grp variable contains the means for imputation rather than the data

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the prior
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

7.30.4 Member Data Documentation

7.30.4.1 _misPhenInd

```
vector<size_t> MVnormMuMiss::_misPhenInd  [protected]
```

Missing data index.

Vector of indexes tagging missing data.

7.30.4.2 _myInd

```
size_t MVnormMuMiss::_myInd  [protected]
```

own index

Index of the row the current instance of this class is pointing to. Only defined with constructors that take *iRw*.

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

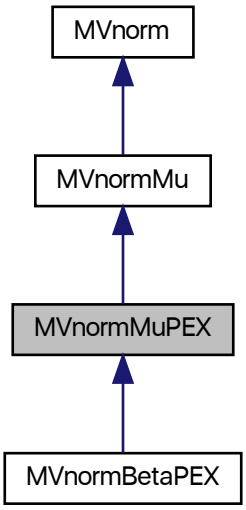
- MuGen.h
- MuGen.cpp

7.31 MVnormMuPEX Class Reference

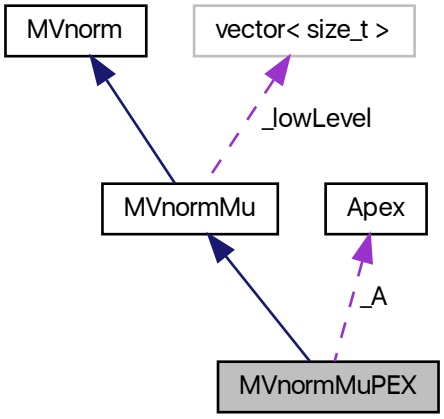
Individual vector of means with parameter expansion.

```
#include <MuGen.h>
```

Inheritance diagram for MVnormMuPEX:



Collaboration diagram for MVnormMuPEX:



Public Member Functions

- MVnormMuPEX ()

Default constructor.

- MVnormMuPEX (gsl_matrix *mn, const size_t &iRw, const vector< size_t > &low, const size_t &up, Apex &A, gsl_matrix *tSigIAt)

  Deterministic constructor.
- MVnormMuPEX (gsl_matrix *mn, const size_t &iRw, const gsl_vector *sd, const gsl_rng *r, const vector< size_t > &low, const size_t &up, Apex &A, gsl_matrix *tSigIAt)

  Univariate random constructor.
- MVnormMuPEX (const MVnormMuPEX &mu)

  Deterministic copy constructor.
- MVnormMuPEX & operator= (const MVnormMuPEX &mu)

  Assignment operator.
- virtual ~MVnormMuPEX ()

  Destructor.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Gaussian likelihood, Student-t prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Gaussian prior.
- virtual void update (const Grp &dat, const Qgrp &q, const SigmaI &SigIm, const Grp &muPr, const double &qPr, const SigmaI &SigIp, const gsl_rng *r)

  Student-t likelihood, Student-t prior.

Protected Attributes

- Apex * _A

  Pointer to the redundant parameter.
- gsl_matrix_view _tSAprod

  Pre-computed auxiliary matrix.

Additional Inherited Members

7.31.1 Detailed Description

Individual vector of means with parameter expansion.

This class is the same as MVnormMu, but implements multivariate parameter expansion for updating (Greenberg, in prep.)

7.31.2 Constructor & Destructor Documentation

7.31.2.1 MVnormMuPEX() [1/3]

```
MVnormMuPEX::MVnormMuPEX (
            gsl_matrix * mn,
            const size_t & iRw,
            const vector< size_t > & low,
            const size_t & up,
            Apex & A,
            gsl_matrix * tSigIAt )
```

Deterministic constructor.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values
in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors
in	*Apex&*	object storing the redundant parameter matrix
in	*gsl_matrix*∗	pre-computed auxiliary matrix

7.31.2.2 MVnormMuPEX() [2/3]

```
MVnormMuPEX::MVnormMuPEX (
            gsl_matrix * mn,
            const size_t & iRw,
            const gsl_vector * sd,
            const gsl_rng * r,
            const vector< size_t > & low,
            const size_t & up,
            Apex & A,
            gsl_matrix * tSigIAt )
```

Univariate random constructor.

Parameters

in	*gsl_matrix*∗	matrix of values
in	*size_t&*	row index of the matrix of values
in	*gsl_vector*∗	vector of standard deviations for the univariate Gaussian
in	*gsl_rng*∗	pointer to a PNG

Parameters

in	*vector<size_t>&*	vector of indexes of rows in the data matrix
in	*size_t&*	index in the matrix of priors
in	*Apex&*	object storing the redundant parameter matrix
in	*gsl_matrix**	pre-computed auxiliary matrix

7.31.2.3 MVnormMuPEX() [3/3]

```
MVnormMuPEX::MVnormMuPEX (
            const MVnormMuPEX & mu )
```

Deterministic copy constructor.

Parameters

in	*MVnormMuPEX&*	object of type MVnormMuPEX

7.31.3 Member Function Documentation

7.31.3.1 operator=()

```
MVnormMuPEX & MVnormMuPEX::operator= (
            const MVnormMuPEX & mu )
```

Assignment operator.

Deterministic assignement operator

Parameters

in	*MVnormMuPEX&*	object of type MVnormMuPEX

Returns

A reference to an object of type MVnormMuPEX

7.31.3.2 update() [1/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.3 update() [2/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Student- t likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.4 update() [3/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector Student- t covariance scale parameter for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.5 update() [4/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const Qgrp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng * ) [virtual]
```

Student- t likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*Qgrp&*	vector of Student- t covariance scale parameters for the likelihood covariance
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.6 update() [5/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const SigmaI & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.7 update() [6/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const double & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Student- t prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*double&*	Student- t scale parameter for the prior covariance
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.8 update() [7/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const SigmaI & ,
            const Grp & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*Grp&*	prior mean
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.3.9 update() [8/8]

```
void MVnormMuPEX::update (
            const Grp & ,
            const SigmaI & ,
            const SigmaI & ,
            const gsl_rng *  )  [virtual]
```

Gaussian likelihood, Gaussian prior.

Parameters

in	*Grp&*	data for the likelihood
in	*SigmaI&*	inverse-covariance matrix for the likelihood
in	*SigmaI&*	prior inverse-covariance matrix
in	*gsl_rng∗*	pointer to a PNG

Reimplemented from MVnormMu.

Reimplemented in MVnormBetaPEX.

7.31.4 Member Data Documentation

7.31.4.1 _A

```
Apex* MVnormMuPEX::_A  [protected]
```

Pointer to the redundant parameter.

A pointer to an object of class Apex, that stores the matrix of redundant parameters.

7.31.4.2 _tSAprod

```
gsl_matrix_view MVnormMuPEX::_tSAprod  [protected]
```

Pre-computed auxiliary matrix.

Vector view of a matrix that stores a pre-computed $(SA)^{\mathrm{T}}$ matrix that is common for all members of the encapsulating Grp class.

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

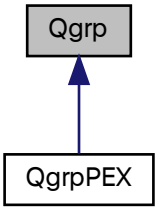
- MuGen.h
- MuGen.cpp

7.32 Qgrp Class Reference

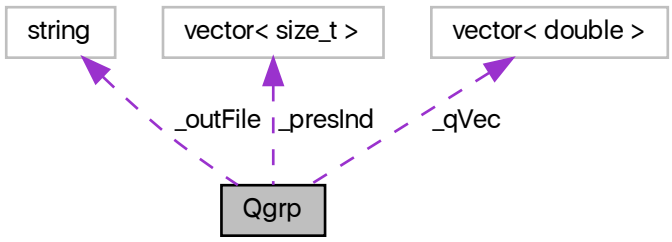
Standard Student- t weights.

```
#include <MuGen.h>
```

Inheritance diagram for Qgrp:



Collaboration diagram for Qgrp:



Public Member Functions

- Qgrp ()
  Default constructor.
- Qgrp (const size_t &N)
  Deterministic size-only constructor.
- Qgrp (const size_t &N, const double &nu)
  Deterministic constructor with degrees of freedom.
- Qgrp (const size_t &N, const double &nu, const string &misVecFlNam)
  Deterministic constructor with degrees of freedom and missing values.
- Qgrp (const size_t &N, const string &outFileNam)
  Deterministic size-only constructor with output file name.
- Qgrp (const size_t &N, const string &outFileNam, const double &nu)
  Deterministic constructor with degrees of freedom and output file name.
- Qgrp (const size_t &N, const string &outFileNam, const double &nu, const string &misVecFlNam)
  Deterministic constructor with degrees of freedom, missing values and output file name.
- virtual ~Qgrp ()
  Destructor.
- double operator[ ] (const size_t i) const
  Const subscript operator.
- double operator[ ] (const size_t i)
  Subscript operator.
- virtual double alpha () const
  Const access to α.
- virtual double alpha ()
  Access to α.
- size_t size () const
  Const length of the weight vector.
- size_t size ()
  Length of the weight vector.

- void save (const char *how="a")

  Save to a stored file name.
- void save (const string &fileNam, const char *how="a")

  Save to a given file name.
- virtual void update (const Grp &dat, const Grp &mu, const Sigmal &SigI)

  Update with a mean.
- virtual void update (const Grp &dat, const Sigmal &SigI)

  Basic update.

Protected Attributes

- vector< double > _qVec

  Vector of weights.
- vector< size_t > _presInd

  Vector of indexes of present data.
- double _nu

  Student-t degrees of freedom.
- string _outFile

  Output file name.
- gsl_rng * _r

  Pseudo-random number generator.

7.32.1 Detailed Description

Standard Student-t weights.

Mostly a standard implementation of the weight parameter sampling for Student-t modeling. The only deviation of the standard approach concerns the situation where some values in the data are missing. If we impute values for these elements, the weights become confounded with means. So the weights are kept strictly equal to 1.0 and not updated.

Note

The weights are relatively small for values far from the mean, i.e. outliers have small q.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 Qgrp() [1/7]

```
Qgrp::Qgrp ( )  [inline]
```

Default constructor.

Results in an empty vector of weights.

7.32.2.2 Qgrp() [2/7]

```
Qgrp::Qgrp (
            const size_t & N )
```

Deterministic size-only constructor.

Creates a vector of weights that all equal to 1.0, and three degrees of freedom (the smallest possible that still gives a destribution with defined mean and variance).

Parameters

in	*size_t&*	number of weights

7.32.2.3 Qgrp() [3/7]

```
Qgrp::Qgrp (
            const size_t & N,
            const double & nu )
```

Deterministic constructor with degrees of freedom.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value.

Parameters

in	*size_t&*	number of weights
in	*double&*	degrees of freedom

7.32.2.4 Qgrp() [4/7]

```
Qgrp::Qgrp (
            const size_t & N,
            const double & nu,
            const string & misVecFlNam )
```

Deterministic constructor with degrees of freedom and missing values.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value. The given file is read to determine which rows of data have missing values.

Parameters

in	*size_t&*	number of weights
in	*double&*	degrees of freedom
in	*string&*	missing value file name

7.32.2.5 Qgrp() [5/7]

```
Qgrp::Qgrp (
            const size_t & N,
            const string & outFileNam )
```

Deterministic size-only constructor with output file name.

Creates a vector of weights that all equal to 1.0, and three degrees of freedom (the smallest possible that still gives a destribution with defined mean and variance).

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name

7.32.2.6 Qgrp() [6/7]

```
Qgrp::Qgrp (
            const size_t & N,
            const string & outFileNam,
            const double & nu )
```

Deterministic constructor with degrees of freedom and output file name.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value.

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name
in	*double&*	degrees of freedom

7.32.2.7 Qgrp() [7/7]

```
Qgrp::Qgrp (
            const size_t & N,
            const string & outFileNam,
            const double & nu,
            const string & misVecFlNam )
```

Deterministic constructor with degrees of freedom, missing values and output file name.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value. The given file is read to determine which rows of data have missing values.

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name
in	*double&*	degrees of freedom
in	*string&*	missing value file name

7.32.3 Member Function Documentation

7.32.3.1 alpha() [1/2]

```
virtual double Qgrp::alpha ( )  [inline], [virtual]
```

Access to α.

Access to the van Dyk and Meng α multiplicative redundant parameter. Is equal to 1.0 in this class.

Returns

double 1.0

Reimplemented in QgrpPEX.

7.32.3.2 alpha() [2/2]

```
virtual double Qgrp::alpha ( ) const  [inline], [virtual]
```

Const access to α.

Access to the van Dyk and Meng α multiplicative redundant parameter. Is equal to 1.0 in this class.

Returns

double 1.0

Reimplemented in QgrpPEX.

7.32.3.3 operator[]() [1/2]

```
double Qgrp::operator[] (
            const size_t i )  [inline]
```

Subscript operator.

Parameters

in	*size_t&*	index

Returns

double weight value

7.32.3.4 operator[]() [2/2]

```
double Qgrp::operator[] (
            const size_t i ) const  [inline]
```

Const subscript operator.

Parameters

in	*size_t&*	index

Returns

double weight value

7.32.3.5 save() [1/2]

```
void Qgrp::save (
            const char * how = "a" )
```

Save to a stored file name.

Parameters

in	*char*∗	saving mode, appending by default

7.32.3.6 save() [2/2]

```
void Qgrp::save (
            const string & fileNam,
            const char * how = "a" )
```

Save to a given file name.

Parameters

in	*string&*	output file name
in	*char*∗	saving mode, appending by default

7.32.3.7 size() [1/2]

```
size_t Qgrp::size ( )  [inline]
```

Length of the weight vector.

Returns

size_t& vector length

7.32.3.8 size() [2/2]

```
size_t Qgrp::size ( ) const  [inline]
```

Const length of the weight vector.

Returns

size_t& vector length

7.32.4 Member Data Documentation

7.32.4.1 _presInd

```
vector<size_t> Qgrp::_presInd  [protected]
```

Vector of indexes of present data.

Only _qVec elements corresponding to rows of data with no missing values are updated.

7.32.4.2 _r

```
gsl_rng* Qgrp::_r  [protected]
```

Pseudo-random number generator.

Initialized the same way as Grp type PNGs, with a sum of time and RTDSC.

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

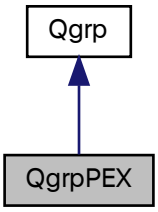
- MuGen.h
- MuGen.cpp

7.33 QgrpPEX Class Reference

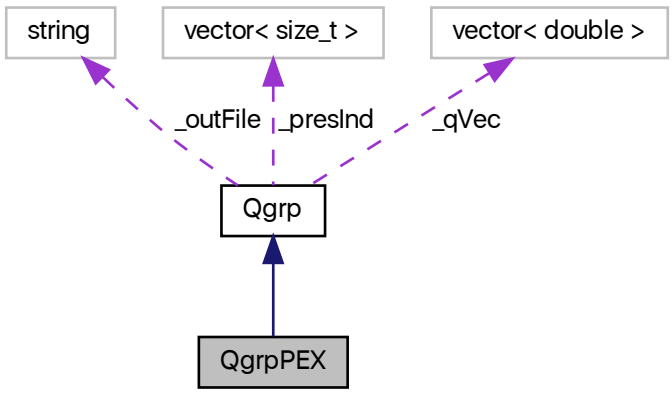
Student- t weights for PEX.

`#include <MuGen.h>`

Inheritance diagram for QgrpPEX:



Collaboration diagram for QgrpPEX:



Public Member Functions

- QgrpPEX ()

  Default constructor.
- QgrpPEX (const size_t &N)

Deterministic size-only constructor.

- QgrpPEX (const size_t &N, const double &nu)

  Deterministic constructor with degrees of freedom.
- QgrpPEX (const size_t &N, const double &nu, const string &misVecFlNam)

  Deterministic constructor with degrees of freedom and missing values.
- QgrpPEX (const size_t &N, const string &outFileNam)

  Deterministic size-only constructor with output file name.
- QgrpPEX (const size_t &N, const string &outFileNam, const double &nu)

  Deterministic constructor with degrees of freedom and output file name.
- QgrpPEX (const size_t &N, const string &outFileNam, const double &nu, const string &misVecFlNam)

  Deterministic constructor with degrees of freedom, missing values and output file name.
- double alpha () const

  Const access to α.
- double alpha ()

  Access to α.
- void update (const Grp &dat, const Grp &mu, const SigmaI &SigI)

  Update with a mean.
- void update (const Grp &dat, const SigmaI &SigI)

  Basic update.

Protected Attributes

- double _alpha

  Redundant parameter α.

7.33.1 Detailed Description

Student- t weights for PEX.

Implements the weight updating for van Dyk and Meng's **[dyk01]** PEX scheme. The multiplicative redundant parameter α is a member of this class and is updated internally. α is initialized deterministically to 1.0.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 QgrpPEX() [1/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N )  [inline]
```

Deterministic size-only constructor.

Creates a vector of weights that all equal to 1.0, and three degrees of freedom (the smallest possible that still gives a destribution with defined mean and variance).

Parameters

in	*size↩ _t&*	number of weights

7.33.2.2 QgrpPEX() [2/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N,
            const double & nu )  [inline]
```

Deterministic constructor with degrees of freedom.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value.

Parameters

in	*size_t&*	number of weights
in	*double&*	degrees of freedom

7.33.2.3 QgrpPEX() [3/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N,
            const double & nu,
            const string & misVecFlNam )  [inline]
```

Deterministic constructor with degrees of freedom and missing values.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value. The given file is read to determine which rows of data have missing values.

Parameters

in	*size_t&*	number of weights
in	*double&*	degrees of freedom
in	*string&*	missing value file name

7.33.2.4 QgrpPEX() [4/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N,
            const string & outFileNam )  [inline]
```

Deterministic size-only constructor with output file name.

Creates a vector of weights that all equal to 1.0, and three degrees of freedom (the smallest possible that still gives a destribution with defined mean and variance).

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name

7.33.2.5 QgrpPEX() [5/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N,
            const string & outFileNam,
            const double & nu )  [inline]
```

Deterministic constructor with degrees of freedom and output file name.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value.

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name
in	*double&*	degrees of freedom

7.33.2.6 QgrpPEX() [6/6]

```
QgrpPEX::QgrpPEX (
            const size_t & N,
            const string & outFileNam,
            const double & nu,
            const string & misVecFlNam )  [inline]
```

Deterministic constructor with degrees of freedom, missing values and output file name.

Creates a vector of weights that all equal to 1.0, and degrees of freedom set to the given value. The given file is read to determine which rows of data have missing values.

Parameters

in	*size_t&*	number of weights
in	*string&*	output file name
in	*double&*	degrees of freedom
in	*string&*	missing value file name

7.33.3 Member Function Documentation

7.33.3.1 alpha() [1/2]

```
double QgrpPEX::alpha ( )  [inline], [virtual]
```

Access to α.

Access to the van Dyk and Meng α multiplicative redundant parameter. Is equal to 1.0 in this class.

Returns

double α

Reimplemented from Qgrp.

7.33.3.2 alpha() [2/2]

```
double QgrpPEX::alpha ( ) const  [inline], [virtual]
```

Const access to α.

Access to the van Dyk and Meng α multiplicative redundant parameter. Is equal to 1.0 in this class.

Returns

double α

Reimplemented from Qgrp.

7.33.3.3 update() [1/2]

```
void QgrpPEX::update (
            const Grp & dat,
            const Grp & mu,
            const SigmaI & SigI )  [virtual]
```

Update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean
in	*SigmaI&*	inverse-covariance

Reimplemented from Qgrp.

7.33.3.4 update() [2/2]

```
void QgrpPEX::update (
            const Grp & dat,
            const SigmaI & SigI )  [virtual]
```

Basic update.

The data are assumed already centered, so the update is based on the simple cross-product of the data and the current value of the inverse-covariance.

Parameters

in	*Grp&*	data
in	*SigmaI&*	inverse-covariance

Reimplemented from Qgrp.

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

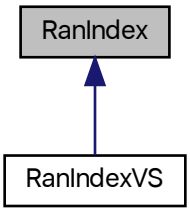
- MuGen.h
- MuGen.cpp

7.34 RanIndex Class Reference

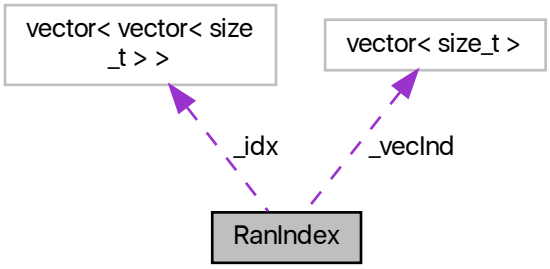
Generic index class.

```
#include <MuGen.h>
```

Inheritance diagram for RanIndex:



Collaboration diagram for RanIndex:



Public Member Functions

- RanIndex ()
  Default constructor.
- RanIndex (const gsl_vector_int *lInd, const size_t &Ntot, const size_t &Nup)
  Vector-based constructor.
- RanIndex (const size_t &Ntot)
  Constructor with one upper level.
- RanIndex (const size_t &Ntot, const size_t &Nup)
  Constructor for single-element lower levels.
- RanIndex (const size_t &Ntot, const size_t &Nup, const string &fileNam)
  Constructor with file input.
- RanIndex (const size_t &Ntot, const size_t &Nup, FILE *fileStr)
  Constructor with file-stream input.

- virtual ~RanIndex ()

  Destructor.
- const vector< size_t > & operator[ ] (const size_t i) const

  Subscript operator.
- vector< size_t > & operator[ ] (const size_t i)

  Subscript operator.
- const vector< size_t > & getIndVec () const

  Retrieve the index vector.
- vector< size_t > & getIndVec ()

  Retrieve the index vector.
- const size_t & priorInd (const size_t i) const

  Upper-level index.
- size_t & priorInd (const size_t i)

  Upper-level index.
- size_t getNtot () const

  Number of lower-level elements.
- size_t getNtot ()

  Number of lower-level elements.
- size_t getNgrp () const

  Number of upper-level elements.
- size_t getNgrp ()

  Number of upper-level elements.
- virtual const vector< double > props () const

  Proportions of each group.
- virtual void init (const vector< vector< size_t > > &idx, const vector< vector< size_t > > &rLD)

  Initialization function.
- virtual void save (const Grp &y, const BetaGrpBVSR *theta, const SigmaI &SigIe)

  Variable selection save.
- virtual void save (const Grp &y, const Grp &theta, const SigmaI &SigIe)

  Probability save.
- virtual void dump ()

  Dump results to a file.
- void update (const Grp &theta, const Grp &mu, const vector< SigmaI > &SigI, const MixP &p)

  Update mixture model with multiple covariances.
- void update (const Grp &theta, const Grp &mu, const SigmaI &SigI, const MixP &p)

  Update mixture model with a single covariance.
- virtual void update (const Grp &y, const SigmaI &SigIe, BetaGrpBVSR *theta, const SigmaI &SigIp)

  Variable selection update.

Protected Attributes

- vector< vector< size_t > > _idx

  Indexes of lower levels.
- vector< size_t > _vecInd

  Indexes of upper levels.
- gsl_rng * _r

  Pointer to a PNG.

7.34.1 Detailed Description

Generic index class.

Establishes relationships of location parameter levels in a hierarchy. If the constructor is deterministic and there is no updating, this class is used simply to relate levels. Updating methods are provided that implement Gaussian mixture models.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 RanIndex() [1/6]

```
RanIndex::RanIndex ( )
```

Default constructor.

The PNG is initialized, _idx is set to length 0, and _vecInd has only one element set to 0.

7.34.2.2 RanIndex() [2/6]

```
RanIndex::RanIndex (
            const gsl_vector_int * lInd,
            const size_t & Ntot,
            const size_t & Nup )
```

Vector-based constructor.

The GSL vector of integers has the indexes of the upper level (number of unique values is *Nup*), and is the same length as the number of rows in the lower-level location parameter matrix (*Ntot*).

Parameters

in	*gsl_vector_int∗*	GSL vector of index information
in	*size_t&*	number of elements in the lower level
in	*size_t&*	number of elements in the upper level

7.34.2.3 RanIndex() [3/6]

```
RanIndex::RanIndex (
            const size_t & Ntot )
```

Constructor with one upper level.

For cases where all the lower-level elemants have the same prior.

Parameters

in	*size_t&*	number of lower-level elements

7.34.2.4 RanIndex() [4/6]

```
RanIndex::RanIndex (
            const size_t & Ntot,
            const size_t & Nup )
```

Constructor for single-element lower levels.

Each element of the lower level has a separate corresponding upper level (prior). Typically the two numbers passed will be the same, but the one-argument constructor is already used for setting up the single-prior index. If the number of elements in the upper level is bigger than that in the lower level, an error is issued. If the number of upper elements is smaller than the *Ntot*, only the first *Nup* elements of the lower level are used and the rest are ignored with a warning.

7.34.2.5 RanIndex() [5/6]

```
RanIndex::RanIndex (
            const size_t & Ntot,
            const size_t & Nup,
            const string & fileNam )
```

Constructor with file input.

Reads the index information from a file. Base-0 indexing is generally assumed, but checks are attempted. Since there are cases when indexes that look non-base-0 are needed (such as when only a subset of lower-level rows are accessed), these generate warnings rather than errors, unless there are upper-level index values outside the set range.

Parameters

in	*size_t&*	number of lower-level elements
in	*size_t&*	number of upper-level elements
in	*string&*	file name

7.34.2.6 RanIndex() [6/6]

```
RanIndex::RanIndex (
            const size_t & Ntot,
            const size_t & Nup,
            FILE * fileStr )
```

Constructor with file-stream input.

Reads the index information from a file. Base-0 indexing is generally assumed, but checks are attempted. Since there are cases when indexes that look non-base-0 are needed (such as when only a subset of lower-level rows are accessed), these generate warnings rather than errors, unless there are upper-level index values outside the set range.

Parameters

in	*size_t&*	number of lower-level elements
in	*size_t&*	number of upper-level elements
in	*FILE**	file stream name

7.34.3 Member Function Documentation

7.34.3.1 dump()

```
virtual void RanIndex::dump ( )  [inline], [virtual]
```

Dump results to a file.

Dumping the mean values for group probabilites to a file. For now, only implemented in the derived class.

Reimplemented in RanIndexVS.

7.34.3.2 getIndVec() [1/2]

```
vector<size_t>& RanIndex::getIndVec ( )  [inline]
```

Retrieve the index vector.

Provides access to the vector of indexes into the upper level.

Returns

const vector<size_t> vector of upper-level indexes

7.34.3.3 getIndVec() [2/2]

```
const vector<size_t>& RanIndex::getIndVec ( ) const  [inline]
```

Retrieve the index vector.

Provides access to the vector of indexes into the upper level. This is the *const* version.

Returns

const vector<size_t> vector of upper-level indexes

7.34.3.4 getNgrp() [1/2]

```
size_t RanIndex::getNgrp ( )  [inline]
```

Number of upper-level elements.

Returns the number of rows in the upper-level (prior) matrix.

Returns

size_t number of upper-level elements

7.34.3.5 getNgrp() [2/2]

```
size_t RanIndex::getNgrp ( ) const  [inline]
```

Number of upper-level elements.

Returns the number of rows in the upper-level (prior) matrix. This is the *const* version.

Returns

size_t number of upper-level elements

7.34.3.6 getNtot() [1/2]

```
size_t RanIndex::getNtot ( )  [inline]
```

Number of lower-level elements.

Returns the number of rows in the lower-level location matrix.

Returns

size_t number of lower-level elements

7.34.3.7 getNtot() [2/2]

```
size_t RanIndex::getNtot ( ) const  [inline]
```

Number of lower-level elements.

Returns the number of rows in the lower-level location matrix. This is the *const* version.

Returns

size_t number of lower-level elements

7.34.3.8 init()

```
void RanIndex::init (
            const vector< vector< size_t > > & idx,
            const vector< vector< size_t > > & rLD )  [virtual]
```

Initialization function.

Deterministically re-initializes the index with a new vector of vectors. Used for some mixture models.

Parameters

in	*vector*<	vector<size_t> >& vector to replace the current _idx
in	*vector*<	vector<size_t> >& a relationship vector; ignored in this class

Reimplemented in RanIndexVS.

7.34.3.9 operator[]() [1/2]

```
vector<size_t>& RanIndex::operator[] (
            const size_t i )  [inline]
```

Subscript operator.

Returns a vector of lower-level indexes for a given upper-level index.

Parameters

in	*size_t*	upper-level index value

Returns

vector<size_t>& vector of lower-level indexes

7.34.3.10 operator[]() [2/2]

```
const vector<size_t>& RanIndex::operator[] (
            const size_t i ) const  [inline]
```

Subscript operator.

Returns a vector of lower-level indexes for a given upper-level index. This is the *const* version.

Parameters

in	*size_t*	upper-level index value

Returns

const vector<size_t>& vector of lower-level indexes

7.34.3.11 priorInd() [1/2]

```
size_t& RanIndex::priorInd (
            const size_t i )  [inline]
```

Upper-level index.

Returns the upper-level index that corresponds to the given lower-level index.

Parameters

in	*size_t*	lower-level index

Returns

const size_t upper-level index

7.34.3.12 priorInd() [2/2]

```
const size_t& RanIndex::priorInd (
            const size_t i ) const  [inline]
```

Upper-level index.

Returns the upper-level index that corresponds to the given lower-level index. This is the *const* version.

Parameters

in	*size_t*	lower-level index

Returns

const size_t upper-level index

7.34.3.13 props()

```
const vector< double > RanIndex::props ( ) const  [virtual]
```

Proportions of each group.

Returns the fraction of the total number of lower-level elements that fall into each of the upper-level groups.

Returns

vector<double> a vector of proportions

7.34.3.14 save() [1/2]

```
virtual void RanIndex::save (
            const Grp & y,
            const BetaGrpBVSR * theta,
            const SigmaI & SigIe )  [inline], [virtual]
```

Variable selection save.

Saves probablities of belonging to a group. For now, only implemented in the derived class.

Parameters

in	*Grp&*	data
in	*BetaGrpBVSR**	location parameters
in	*SigmaI&*	inverse-covariance matrix

Reimplemented in RanIndexVS.

7.34.3.15 save() [2/2]

```
virtual void RanIndex::save (
            const Grp & y,
            const Grp & theta,
            const SigmaI & SigIe )  [inline], [virtual]
```

Probability save.

Saves probablities of belonging to a group. For now, only implemented in the derived class.

Parameters

in	*Grp&*	data
in	*Grp&*	location parameters
in	*Sigma↩ I&*	inverse-covariance matrix

Reimplemented in RanIndexVS.

7.34.4 Member Data Documentation

7.34.4.1 _idx

```
vector< vector<size_t> > RanIndex::_idx  [protected]
```

Indexes of lower levels.

This vector of vectors relates a top level of a hierarchy to the corresponding lower level. The total number of elements is the number of elements in the top level. Each element is a vector of row indexes into the lower level location parameter matrix. Constituent vectors may not be of the same length (corresponding to an unbalanced design).

7.34.4.2 _r

```
gsl_rng* RanIndex::_r  [protected]
```

Pointer to a PNG.

The PNG is initialized whether or not updating is going to be performed. We use the sum of *time(NULL)* and RTDSC for seeding.

7.34.4.3 _vecInd

```
vector<size_t> RanIndex::_vecInd  [protected]
```

Indexes of upper levels.

Stores the row indexes of the upper-level location parameter matrix corresponding to each lower-level index. The length of this vector is the same as the sum of all the vectors in ∗_idx∗, and the number of unique values is equal to the length of ∗_idx∗.

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

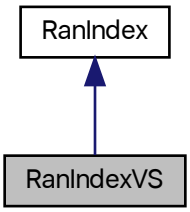
- MuGen.h
- MuGen.cpp

7.35 RanIndexVS Class Reference

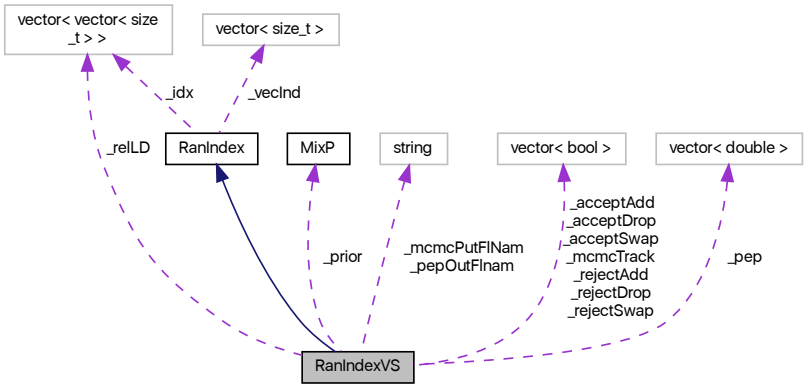
BVSR index class.

`#include <MuGen.h>`

Inheritance diagram for RanIndexVS:



Collaboration diagram for RanIndexVS:



Public Member Functions

- RanIndexVS ()

  Default constructor.
- RanIndexVS (const size_t &Ntot, const string &outFlNam, MixP &pr)

  Constructor with a prior.
- RanIndexVS (const size_t &Ntot, const string &outFlNam)

Constructor without a prior.

- ~RanIndexVS ()

  Destructor.

- void **props** (vector< double > &prp) const
- void save (const Grp &y, const BetaGrpBVSR *theta, const SigmaI &SigIe)

  Save PEP.

- void save (const Grp &y, const Grp &theta, const SigmaI &SigIe)

  Save PEP.

- void dump ()

  Save PEP to a file.

- void init (const vector< vector< size_t > > &idx, const vector< vector< size_t > > &rLD)

  Complete initialization.

- void update (const Grp &y, const SigmaI &SigIe, BetaGrpBVSR *theta, const SigmaI &SigIp)

  Variable selection update.

Protected Member Functions

- size_t _proposal (const size_t &Ntot, const int &Nmn, const gsl_rng *r)

  Rank proposal-generating function.

Protected Attributes

- vector< double > _pep

  Posterior exclusion probabilities.

- vector< bool > _mcmcTrack

  Transition tracking variable.

- vector< vector< size_t > > _relLD

  Mapping selected to original predictors.

- size_t _Ntp

  Original number of predictors.

- MixP * _prior

  Prior proportions.

- vector< bool > _acceptDrop

  Accept a drop.

- vector< bool > _rejectDrop

  Reject a drop.

- vector< bool > _acceptAdd

  Accept an addition.

- vector< bool > _rejectAdd

  Rject an addition.

- vector< bool > _acceptSwap

  Accept a swap.

- vector< bool > _rejectSwap

  Reject a swap.

- double _numSaves

  Number of calls to save()

- string _pepOutFlnam

  PEP output file name.

- string _mcmcPutFlNam

  Accept/reject file name.

7.35.1 Detailed Description

BVSR index class.

Keeps track of the variable selection process for a multivariate version of the Bayesian Variable Selection Regression (BVSR) multi-marker regression method of Guan and Stephens **[guan11]** (Greenberg, unpublished). This clas is a friend of BetaGrpBVSR.

Warning

This class is experimental and has not been extensively tested.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 RanIndexVS() [1/2]

```
RanIndexVS::RanIndexVS (
            const size_t & Ntot,
            const string & outFlNam,
            MixP & pr )
```

Constructor with a prior.

This is a preliminary set-up. The initiation is finished with the init() member function, after pre-screening of predictors is performed.

7.35.2.2 RanIndexVS() [2/2]

```
RanIndexVS::RanIndexVS (
            const size_t & Ntot,
            const string & outFlNam )
```

Constructor without a prior.

This is a preliminary set-up. The initiation is finished with the init() member function, after pre-screening of predictors is performed.

7.35.3 Member Function Documentation

7.35.3.1 _proposal()

```
size_t RanIndexVS::_proposal (
            const size_t & Ntot,
            const int & Nmn,
            const gsl_rng * r )  [protected]
```

Rank proposal-generating function.

The rank-based Metropolis proposal-generating function proposed in Guan and Stephens **[guan11]** .

Warning

This function is not presently used in my implementation of BVSR because I am not sure it is a symmetric proposal. If it is not, then we would need to come up with a Metropolis-Hastings updating scheme.

7.35.3.2 dump()

```
void RanIndexVS::dump ( )  [virtual]
```

Save PEP to a file.

Saves posterior exclusion probabilities (PEP) to a file. Name of the file is stored in the _pepOutFlnam variable.

Reimplemented from RanIndex.

7.35.3.3 init()

```
void RanIndexVS::init (
            const vector< vector< size_t > > & idx,
            const vector< vector< size_t > > & rLD )  [virtual]
```

Complete initialization.

Initializes the index instance after preliminary screening of predictors in performed by the corresponding BetaGrpBVSR variable.

Parameters

in	*vector<*	vector<size_t> >& vector to replace the current _idx
in	*vector<*	vector<size_t> >& a relationship vector; ignored in this class

Reimplemented from RanIndex.

7.35.3.4 save() [1/2]

```
void RanIndexVS::save (
            const Grp & y,
            const BetaGrpBVSR * theta,
            const SigmaI & SigIe )  [virtual]
```

Save PEP.

Stores current PEP values and saves Metropolis proposal statistics to a file.

Parameters

in	*Grp&*	data
in	*BetaGrpBVSR**	location parameters
in	*SigmaI&*	inverse-covariance matrix

Reimplemented from RanIndex.

7.35.3.5 save() [2/2]

```
void RanIndexVS::save (
            const Grp & y,
            const Grp & theta,
            const SigmaI & SigIe )  [virtual]
```

Save PEP.

Stores current PEP values and saves Metropolis proposal statistics to a file.

Parameters

in	*Grp&*	data
in	*Grp&*	location parameters
in	*SigmaI&*	inverse-covariance matrix

Reimplemented from RanIndex.

7.35.3.6 update()

```
void RanIndexVS::update (
            const Grp & y,
            const SigmaI & SigIe,
            BetaGrpBVSR * theta,
            const SigmaI & SigIp )  [virtual]
```

Variable selection update.

Update for a mixture that includes point-mass at 0. Only implemented in the derived class.

Parameters

in	*Grp&*	data
in	*SigmaI&*	likelihood inverse-covariance
in	*BetaGrpBVSR**	location parameter values
in	*SigmaI&*	prior inverse-covariance

Reimplemented from RanIndex.

7.35.4 Member Data Documentation

7.35.4.1 _acceptAdd

```
vector<bool> RanIndexVS::_acceptAdd  [protected]
```

Accept an addition.

The state of _mcmcTrack that corresponds to accepting an addition of a predictor. Pre-set so that it is easy to modify _mcmcTrack.

7.35.4.2 _acceptDrop

```
vector<bool> RanIndexVS::_acceptDrop  [protected]
```

Accept a drop.

The state of _mcmcTrack that corresponds to accepting a drop of a predictor. Pre-set so that it is easy to modify _mcmcTrack.

7.35.4.3 _acceptSwap

```
vector<bool> RanIndexVS::_acceptSwap  [protected]
```

Accept a swap.

The state of _mcmcTrack that corresponds to accepting a swap of predictors. Pre-set so that it is easy to modify _mcmcTrack.

7.35.4.4 _mcmcPutFlNam

```
string RanIndexVS::_mcmcPutFlNam  [protected]
```

Accept/reject file name.

Name of the file where the accept/reject and proposal kind statistics are stored. This file is appended at each save. These statistics are useful for checking and trouble-shooting Metropolis chain behavior.

7.35.4.5 _mcmcTrack

```
vector<bool> RanIndexVS::_mcmcTrack  [protected]
```

Transition tracking variable.

Used to track the Metropolis moves made after the current MCMC step. There are four elements, three last ones being mutually exclusive: [0]: accepted? [1]: proposed adding a predictor? [2]: proposed dropping a predictor? [3]: proposed swapping predictors?

7.35.4.6 _numSaves

```
double RanIndexVS::_numSaves  [protected]
```

Number of calls to save()

Tracks the number of times a save function is called. Before dumping the PEP results, divide each element of _pep by this number.

7.35.4.7 _pep

```
vector<double> RanIndexVS::_pep  [protected]
```

Posterior exclusion probabilities.

Posterior *exclusion* probabilities. This is the opposite if the implementation in Guan and Stephens, where the inclusion probabilities are exported. My implementation results in output that is analogous to $-\log_{10} p$, making it more intuitive to people used to frequentist approaches. The probabilities stored in this variable are in the order of the SNPs picked for initial inclusion in the BetaGrpBVSR class. We keep adding PEP values to this at each save, then dump the mean into a file in the end, giving a Rao-Blackwellized point estimate of PEP.

7.35.4.8 _prior

```
MixP* RanIndexVS::_prior  [protected]
```

Prior proportions.

Prior proportion of predictors in selected to be in the model.

7.35.4.9 _rejectAdd

```
vector<bool> RanIndexVS::_rejectAdd  [protected]
```

Rject an addition.

The state of _mcmcTrack that corresponds to rejecting an addition of a predictor. Pre-set so that it is easy to modify _mcmcTrack.

7.35.4.10 _rejectDrop

```
vector<bool> RanIndexVS::_rejectDrop  [protected]
```

Reject a drop.

The state of _mcmcTrack that corresponds to rejecting a drop of a predictor. Pre-set so that it is easy to modify _↩ mcmcTrack.

7.35.4.11 _rejectSwap

```
vector<bool> RanIndexVS::_rejectSwap  [protected]
```

Reject a swap.

The state of _mcmcTrack that corresponds to rejecting a swap of predictors. Pre-set so that it is easy to modify _↩ mcmcTrack.

7.35.4.12 _relLD

```
vector< vector<size_t> > RanIndexVS::_relLD  [protected]
```

Mapping selected to original predictors.

Relates the new positions of selected predictors to the original positions. Its length is the same as the selected number of predictors. *relLD[i][0] is the original position of the selected predictor _i*; the rest of _relLD[i] (if any) are positions of linked SNPs (or, in general, correlated predictors)

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

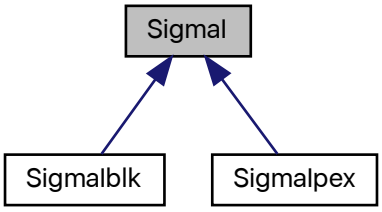
- MuGen.h
- MuGen.cpp

7.36 Sigmal Class Reference

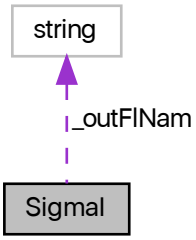
Basic inverse-covariance.

`#include <MuGen.h>`

Inheritance diagram for Sigmal:



Collaboration diagram for Sigmal:



Public Member Functions

- Sigmal ()
  Default constructor.
- Sigmal (const size_t &d, const double &invVar)
  Deterministic diagonal matrix constructor.
- Sigmal (const size_t &d, const double &invVar, const double &df)
  Deterministic diagonal matrix constructor.
- Sigmal (const size_t &d, const double &invVar, const double &df, const string &outFlNam)

Deterministic diagonal matrix constructor with output file name.

- Sigmal (const gsl_matrix *mat)

  Deterministic matrix constructor.

- Sigmal (const gsl_matrix *S, const size_t &d, const size_t &df, const gsl_matrix *LamPr, const double &nu0)

  Constructor with a matrix prior.

- Sigmal (const gsl_matrix *S, const size_t &d, const size_t &df, const double &diagPr, const double &nu0)

  Constructor with a diagonal prior.

- Sigmal (const Grp &dat, const double &prDiag, const double &nu0)

  Location data-based constructor.

- Sigmal (const Grp &dat, const string &outFlNam, const double &prDiag, const double &nu0)

  Location data-based constructor with output file name.

- Sigmal (const Sigmal &)

  Copy constructor.

- Sigmal & operator= (const Sigmal &)

  Assignment operator.

- virtual ~Sigmal ()

  Destructor.

- virtual void update (const Grp &dat)

  Basic Gaussian update.

- virtual void update (const Grp &dat, const Grp &mu)

  Gaussian update with a mean.

- virtual void update (const Grp &dat, const Qgrp &q)

  Basic Student-t update.

- virtual void update (const Grp &dat, const Grp &mu, const Qgrp &q)

  Student-t update with a mean.

- virtual void save (const char *how="a")

  Save to a stored file name.

- virtual void save (const string &fileNam, const char *how="a")

  Save to a given file name.

- void save (const string &fileNam, const Apex &A, const char *how="a")

  Save adjusted matrix.

- void save (FILE *fileStr)

  Save to file stream.

- string getOutFile () const

  Access the output file.

- const gsl_matrix * getMat () const

  Access to the inverse-covariance matrix.

- void srDetUpdate ()

  Update square-root of the determinant.

- double getSrDet () const

  Access the square-root of the determinant.

Protected Attributes

- gsl_matrix * _mat
  Inverse-covariance matrix.
- size_t _d
  Dimension of the matrix.
- double _srDet
  Square root of the determinant.
- gsl_matrix * _LamSc
  Prior inverse-covariance.
- double _n0
  Prior degrees of freedom.
- string _outFlNam
  Output file name.
- gsl_rng * _r
  Pseudo-random number generator.

7.36.1 Detailed Description

Basic inverse-covariance.

Standard model for inverse-covariance matrices with a diagonal Wishart prior (i.e., pre-supposing no correlations among traits). The values on the diagonal of the prior matrix and the prior degrees of freedom can be varied to reflect the vagueness of prior information. These conjugate priors allow for Gibbs sampling even when the number of traits exceeds the number of data points. However, in the latter case the estimation of correlations will likely be inaccurate. Algorithms involving direct penalties on correlations should perform better and are inder development.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 SigmaI() [1/10]

```
SigmaI::SigmaI ( )
```

Default constructor.

Creates a 2×2 identity matrix.

7.36.2.2 SigmaI() [2/10]

```
SigmaI::SigmaI (
            const size_t & d,
            const double & invVar )
```

Deterministic diagonal matrix constructor.

Creates a diagonal matrix with the provided inverse-variance and the degrees of freedom parameter equal to one. Principally used to create vague large-variance Gaussian priors for location parameters.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance

7.36.2.3 SigmaI() [3/10]

```
SigmaI::SigmaI (
            const size_t & d,
            const double & invVar,
            const double & df )
```

Deterministic diagonal matrix constructor.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom

7.36.2.4 SigmaI() [4/10]

```
SigmaI::SigmaI (
            const size_t & d,
            const double & invVar,
            const double & df,
            const string & outFlNam )
```

Deterministic diagonal matrix constructor with output file name.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom
in	*string&*	output file name

7.36.2.5 SigmaI() [5/10]

```
SigmaI::SigmaI (
            const gsl_matrix * mat )
```

Deterministic matrix constructor.

Used to initialize the inverse-covariance with a given matrix deterministically, i.e. the initial values are not sampled. Only the lower-triangular part of the matrix is used, so if it is not symmetric no error is produced.

Parameters

in	*gsl_matrix*∗	value matrix

7.36.2.6 SigmaI() [6/10]

```
SigmaI::SigmaI (
            const gsl_matrix * S,
            const size_t & d,
            const size_t & df,
            const gsl_matrix * LamPr,
            const double & nu0 )
```

Constructor with a matrix prior.

Initializes the matrix with a Wishart sample with the provided matrix as data and another matrix as a prior.

Parameters

in	*gsl_matrix*∗	data matrix
in	*size_t&*	dimension parameter
in	*size_t&*	Wishart sample degrees of freedom
in	*gsl_matrix*∗	prior matrix
in	*double&*	prior degrees of freedom

7.36.2.7 SigmaI() [7/10]

```
SigmaI::SigmaI (
            const gsl_matrix * S,
            const size_t & d,
            const size_t & df,
```

```
            const double & diagPr,
            const double & nu0 )
```

Constructor with a diagonal prior.

Initializes the matrix with a Wishart sample with the provided matrix as data and a diagonal prior with all elements set to the provided value.

Parameters

in	*gsl_matrix∗*	data matrix
in	*size_t&*	dimension parameter
in	*size_t&*	Wishart sample degrees of freedom
in	*double∗*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.36.2.8 Sigmal() [8/10]

```
SigmaI::SigmaI (
            const Grp & dat,
            const double & prDiag,
            const double & nu0 )
```

Location data-based constructor.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.36.2.9 Sigmal() [9/10]

```
SigmaI::SigmaI (
            const Grp & dat,
            const string & outFlNam,
            const double & prDiag,
            const double & nu0 )
```

Location data-based constructor with output file name.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*string&*	output file name
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.36.2.10 Sigmal() [10/10]

```
SigmaI::SigmaI (
            const SigmaI & S )
```

Copy constructor.

Parameters

in	*SigmaI&*	oject to be copied

7.36.3 Member Function Documentation

7.36.3.1 getMat()

```
const gsl_matrix* SigmaI::getMat ( ) const  [inline]
```

Access to the inverse-covariance matrix.

Returns

gsl_matrix* pointer to the inverse-covariance matrix

7.36.3.2 getOutFile()

```
string SigmaI::getOutFile ( ) const  [inline]
```

Access the output file.

Returns

string file name

7.36.3.3 getSrDet()

```
double SigmaI::getSrDet ( ) const  [inline]
```

Access the square-root of the determinant.

Returns

double square root of the determinant of the inverse-covariance

7.36.3.4 operator=()

```
SigmaI & SigmaI::operator= (
            const SigmaI & S )
```

Assignment operator.

Parameters

in	*SigmaI&*	object to be copied

Returns

SigmaI& target object

7.36.3.5 save() [1/4]

```
void SigmaI::save (
            const char * how = "a" )  [virtual]
```

Save to a stored file name.

Parameters

in	*char**	saving mode, appending by default

Reimplemented in SigmaIpex.

7.36.3.6 save() [2/4]

```
void SigmaI::save (
            const string & fileNam,
            const Apex & A,
            const char * how = "a" )
```

Save adjusted matrix.

Saving the adjusted matrix for the multiplicative parameter expansion models.

Parameters

in	*string&*	output file name
in	*Apex&*	redundant parameter matrix
in	*char∗*	saving mode, appending by default

7.36.3.7 save() [3/4]

```
void SigmaI::save (
            const string & fileNam,
            const char * how = "a" )  [virtual]
```

Save to a given file name.

Parameters

in	*string&*	output file name
in	*char∗*	saving mode, appending by default

Reimplemented in SigmaIpex.

7.36.3.8 save() [4/4]

```
void SigmaI::save (
            FILE * fileStr )  [inline]
```

Save to file stream.

Parameters

in	*FILE∗*	file stream name

7.36.4 Member Data Documentation

7.36.4.1 _d

```
size_t SigmaI::_d  [protected]
```

Dimension of the matrix.

The number of rows and columns of _mat.

7.36.4.2 _LamSc

```
gsl_matrix* SigmaI::_LamSc  [protected]
```

Prior inverse-covariance.

Diagonal matrix with zeros on the off-diagonal and inverse variance scaled by degrees of freedom on the diagonal.

7.36.4.3 _mat

```
gsl_matrix* SigmaI::_mat  [protected]
```

Inverse-covariance matrix.

Square symmetric matrix.

7.36.4.4 _r

```
gsl_rng* SigmaI::_r  [protected]
```

Pseudo-random number generator.

Initialized the same way as Grp type PNGs, with a sum of time and RTDSC.

7.36.4.5 _srDet

```
double SigmaI::_srDet  [protected]
```

Square root of the determinant.

Square root of the determinant of the corrent value of the inverse-covariance matrix. Is calculated only if necessary, otherwise is set to -1.0 to provide an easy way to find out if it had been calculated at least once.

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

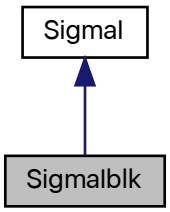
- MuGen.h
- MuGen.cpp

7.37 SigmaIblk Class Reference

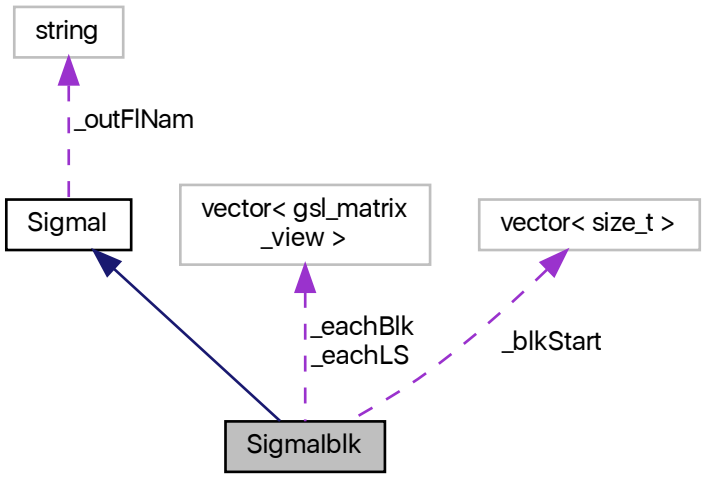
Block-diagonal inverse-covariance.

`#include <MuGen.h>`

Inheritance diagram for SigmaIblk:



Collaboration diagram for SigmaIblk:



Public Member Functions

- SigmaIblk ()

Default constructor.

- SigmaIblk (const size_t &d, const double &invVar, const double &df, const string &blkIndFileNam)

  Deterministic diagonal matrix constructor.
- SigmaIblk (const size_t &d, const double &invVar, const double &df, const string &blkIndFileNam, const string &outFlNam)

  Deterministic diagonal matrix constructor with output file.
- SigmaIblk (const Grp &dat, const string &blkIndFileNam, const double &prDiag, const double &nu0)

  Location data-based constructor.
- SigmaIblk (const Grp &dat, const string &blkIndFileNam, const string &outFlNam, const double &prDiag, const double &nu0)

  Location data-based constructor with output file.
- ~SigmaIblk ()

  Destructor.
- void update (const Grp &dat)

  Basic Gaussian update.
- void update (const Grp &dat, const Grp &mu)

  Gaussian update with a mean.

Protected Attributes

- vector< size_t > _blkStart

  Block start indexes.
- vector< gsl_matrix_view > _eachBlk

  Inverse-covariance matrix submatrices.
- vector< gsl_matrix_view > _eachLS

  Prior matrix submatrices.

7.37.1 Detailed Description

Block-diagonal inverse-covariance.

Implements block-diagonal inverse-covariance, to go with MuBlk and BetaBlk classes. All covariances among blocks are set to exactly zero and are never modified. All sampling is done block by block.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 SigmaIblk() [1/4]

```
SigmaIblk::SigmaIblk (
            const size_t & d,
            const double & invVar,
            const double & df,
            const string & blkIndFileNam )
```

Deterministic diagonal matrix constructor.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom
in	*string&*	name of the file storing block indexes

7.37.2.2 SigmaIblk() [2/4]

```
SigmaIblk::SigmaIblk (
            const size_t & d,
            const double & invVar,
            const double & df,
            const string & blkIndFileNam,
            const string & outFlNam )
```

Deterministic diagonal matrix constructor with output file.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom
in	*string&*	name of the file storing block indexes
in	*string&*	output file name

7.37.2.3 SigmaIblk() [3/4]

```
SigmaIblk::SigmaIblk (
            const Grp & dat,
            const string & blkIndFileNam,
            const double & prDiag,
            const double & nu0 )
```

Location data-based constructor.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*string&*	name of the file storing block indexes
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.37.2.4 SigmaIblk() [4/4]

```
SigmaIblk::SigmaIblk (
            const Grp & dat,
            const string & blkIndFileNam,
            const string & outFlNam,
            const double & prDiag,
            const double & nu0 )
```

Location data-based constructor with output file.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*string&*	name of the file storing block indexes
in	*string&*	output file name
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.37.3 Member Function Documentation

7.37.3.1 update() [1/2]

```
void SigmaIblk::update (
            const Grp & dat )  [virtual]
```

Basic Gaussian update.

The data are assumed already centered, so the update is based on the simple cross-product of the data.

Parameters

in	*Grp&*	data

Reimplemented from SigmaI.

7.37.3.2 update() [2/2]

```
void SigmaIblk::update (
            const Grp & dat,
            const Grp & mu )  [virtual]
```

Gaussian update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean

Reimplemented from SigmaI.

7.37.4 Member Data Documentation

7.37.4.1 _blkStart

```
vector< size_t > SigmaIblk::_blkStart  [protected]
```

Block start indexes.

Vector of indexes that indicate the column that starts each block.

7.37.4.2 _eachBlk

```
vector< gsl_matrix_view > SigmaIblk::_eachBlk  [protected]
```

Inverse-covariance matrix submatrices.

Vector of submatrices of the current sample of the inverse-covariance matrix (_mat). Is the same length as the number of blocks.

7.37.4.3 _eachLS

```
vector< gsl_matrix_view > SigmaIblk::_eachLS  [protected]
```

Prior matrix submatrices.

Vector of submatrices of the prior inverse-covariance matrix (_LamSc). Is the same length as the number of blocks.

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

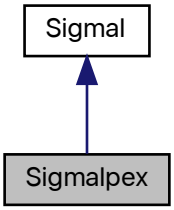
- MuGen.h
- MuGen.cpp

7.38 SigmaIpex Class Reference

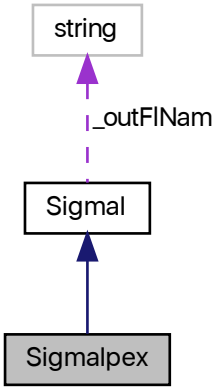
PEX inverse-covariance.

```
#include <MuGen.h>
```

Inheritance diagram for SigmaIpex:



Collaboration diagram for Sigmalpex:



Public Member Functions

- Sigmalpex ()
  Default constructor.
- Sigmalpex (const size_t &d, const double &invVar, const double &df)
  Deterministic diagonal matrix constructor.
- Sigmalpex (const size_t &d, const double &invVar, const double &df, const string &outFlNam)
  Deterministic diagonal matrix constructor with output file name.
- Sigmalpex (const Grp &dat, const double &prDiag, const double &nu0)
  Location data-based constructor.
- Sigmalpex (const Grp &dat, const string &outFlNam, const double &prDiag, const double &nu0)
  Location data-based constructor with output file name.
- ~Sigmalpex ()
  Destructor.
- void save (const char *how="a")
  Save to a stored file name.
- void save (const string &fileNam, const char *how="a")
  Save to a given file name.
- void update (const Grp &dat, const Qgrp &q)
  Basic Student-t update.
- void update (const Grp &dat, const Grp &mu, const Qgrp &q)
  Student-t update with a mean.

Protected Attributes

- double _alpha
  The PEX α parameter.

7.38.1 Detailed Description

PEX inverse-covariance.

Implementation of the van Dyk and Meng's **[dyk01]** parameter expansion scheme from multivariate Student- t sampling. Note that it only works when all the scale values in Qgrp are sampled. So, for example, when we have missing data with a Student- t model for the rest (see Qgrp documentation for details), this class should not be used, it will cause divergence and crashing.

Note

Not to be confused with location parameter PEX such as MuGrpPEX

7.38.2 Constructor & Destructor Documentation

7.38.2.1 SigmaIpex() [1/4]

```
SigmaIpex::SigmaIpex (
            const size_t & d,
            const double & invVar,
            const double & df )  [inline]
```

Deterministic diagonal matrix constructor.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom

7.38.2.2 SigmaIpex() [2/4]

```
SigmaIpex::SigmaIpex (
            const size_t & d,
            const double & invVar,
            const double & df,
            const string & outFlNam )  [inline]
```

Deterministic diagonal matrix constructor with output file name.

Creates a diagonal matrix with the provided inverse-variance and degrees of freedom parameter.

Parameters

in	*size_t&*	number of rows and columns
in	*double&*	inverse variance
in	*double&*	degrees of freedom
in	*string&*	output file name

7.38.2.3 SigmaIpex() [3/4]

```
SigmaIpex::SigmaIpex (
            const Grp & dat,
            const double & prDiag,
            const double & nu0 )  [inline]
```

Location data-based constructor.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.38.2.4 SigmaIpex() [4/4]

```
SigmaIpex::SigmaIpex (
            const Grp & dat,
            const string & outFlNam,
            const double & prDiag,
            const double & nu0 )  [inline]
```

Location data-based constructor with output file name.

Initializes the object with an inverse-covariance of the provided location data. The dimension of the object os equal to the number of columns in the data.

Parameters

in	*Grp&*	data
in	*string&*	output file name
in	*double&*	prior inverse-variance
in	*double&*	prior degrees of freedom

7.38.3 Member Function Documentation

7.38.3.1 save() [1/2]

```
void SigmaIpex::save (
            const char * how = "a" )  [virtual]
```

Save to a stored file name.

Saves the "adjusted" value, i.e. scaled by the multiplicative parameter.

Parameters

in	*char∗*	saving mode, appending by default

Reimplemented from SigmaI.

7.38.3.2 save() [2/2]

```
void SigmaIpex::save (
            const string & fileNam,
            const char * how = "a" )  [virtual]
```

Save to a given file name.

Saves the "adjusted" value, i.e. scaled by the multiplicative parameter.

Parameters

in	*string&*	output file name
in	*char∗*	saving mode, appending by default

Reimplemented from SigmaI.

7.38.3.3 update() [1/2]

```
void SigmaIpex::update (
            const Grp & dat,
```

```
            const Grp & mu,
            const Qgrp & q )  [virtual]
```

Student- t update with a mean.

The mean value is subtracted from the data according to the up-pointing RanIndex in the data object.

Parameters

in	*Grp&*	data
in	*Grp&*	mean
in	*Qgrp&*	scale parameter

Reimplemented from SigmaI.

7.38.3.4 update() [2/2]

```
void SigmaIpex::update (
            const Grp & dat,
            const Qgrp & q )  [virtual]
```

Basic Student- t update.

The data are assumed already centered, so the update is based on the simple cross-product of the data and the current value of the scale parameter.

Parameters

in	*Grp&*	data
in	*Qgrp&*	scale parameter

Reimplemented from SigmaI.

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

- MuGen.h
- MuGen.cpp

7.39 twoVec Struct Reference

Public Attributes

- gsl_vector ∗ **xi**

- gsl_vector * **etaSq**

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

- blupEMMAX.cpp
- myEMMAX.cpp

Chapter 8

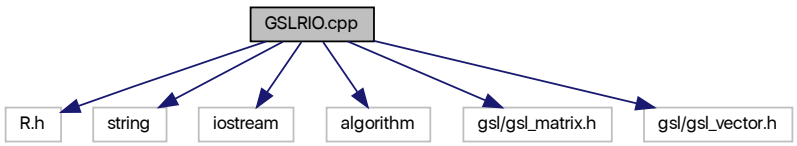
File Documentation

8.1 GSLRIO.cpp File Reference

R interface for GSL binary file format.

```
#include <R.h>
#include <string>
#include <iostream>
#include <algorithm>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
```

Include dependency graph for GSLRIO.cpp:



Functions

- void GSLmatSave (const char **fileNam, const double *vec, const int *nRows, const int *nCols)

  Save a double-precision matrix.
- void GSLvecSaveInt (const char **fileNam, const int *vec, const int *len)

  Save a vector of integers.
- void GSLvecSave (const char **fileNam, const double *vec, const int *len)

  Save a double-precision vector.
- void GSLvecAppend (const char **fileNam, const double *vec, const int *len)

  Appending a double-precision vector to an existing file.
- void GSLiVecLoad (const char **fileNam, const int *len, int *arr)

  Read a vector of integers.
- void GSLmatLoad (const char **fileNam, const int *nRows, const int *nCols, double *arr)

  Read a double-precision array or matrix.

8.1.1 Detailed Description

R interface for GSL binary file format.

Author

Anthony J. Greenberg

Copyright

Version

0.9.1

Functions to read and write binary files saved in GSL binary format from R.

To compile, run on command line:

R CMD SHLIB GSLRIO.cpp -lgsl -O3 -DHAVE_INLINE -DGSL_RANGE_CHECK_OFF

and put the resulting GSLRIO.o where it can be found by `dyn.load()` in R. The functions are then ready to use with the `.C()` command.

Note

To pass double-quoted strings from `.C()`, the corresponding arguments (here they are file names) must be in the form ∗∗fileNam.

8.1.2 Function Documentation

8.1.2.1 GSLiVecLoad()

```
void GSLiVecLoad (
            const char ** fileNam,
            const int * len,
            int * arr )
```

Read a vector of integers.

Reading a file into an array of integers.

Warning

While file existence is checked, there currently is no way to check file size. If the file is too small to hold an array of integers of the required length, the function will crash and likely take the R session with it.

Parameters

in	*char*∗∗	output file name
in	*int*∗	pointer to array length
out	*int*∗	array of integers to be passed to R

8.1.2.2 GSLmatLoad()

```
void GSLmatLoad (
            const char ** fileNam,
            const int * nRows,
            const int * nCols,
            double * arr )
```

Read a double-precision array or matrix.

Reads a double-precision array (vector) or matrix. If a vector is desired, one of the dimensions should be set to one. The matrix is stored by row, so make sure to use matrix(..., byrow = T) in the R code.

Warning

While file existence is checked, there currently is no way to check file size. If the file is too small to hold an array of doubles of the required length, the function will crash and likely take the R session with it.

Parameters

in	*char*∗∗	output file name
in	*int*∗	pointer to the number of rows
in	*int*∗	pointer to the number of columns
out	*double*∗	array to be passed to R and converted to a matrix if necessary

8.1.2.3 GSLmatSave()

```
void GSLmatSave (
            const char ** fileNam,
            const double * vec,
            const int * nRows,
            const int * nCols )
```

Save a double-precision matrix.

The fact that R matrices are column-major, while in C/C++ they are row-major is dealt with internally. The R user does not have to worry about that.

Parameters

in	*char***	output file name
in	*double**	vectorized matrix to be saved
in	*int**	pointer to the number of rows
in	*int**	pointer to the number of columns

8.1.2.4 GSLvecAppend()

```
void GSLvecAppend (
            const char ** fileNam,
            const double * vec,
            const int * len )
```

Appending a double-precision vector to an existing file.

Parameters

in	*char***	output file name
in	*double**	double-precision array
in	*int**	pointer to the array size

8.1.2.5 GSLvecSave()

```
void GSLvecSave (
            const char ** fileNam,
            const double * vec,
            const int * len )
```

Save a double-precision vector.

Parameters

in	*char***	output file name
in	*double**	double-precision array
in	*int**	pointer to the array size

8.1.2.6 GSLvecSaveInt()

```
void GSLvecSaveInt (
            const char ** fileNam,
            const int * vec,
            const int * len )
```

Save a vector of integers.

Typically used to save index (factor) vectors to read into RanIndex objects.

Parameters

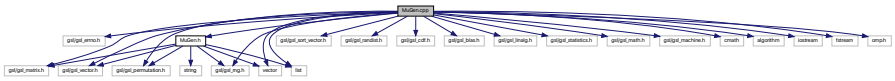
in	*char***	output file name
in	*int**	array of integers
in	*int**	pointer to the array length

8.2 MuGen.cpp File Reference

Implementation of C++ classes for Hierarchical Bayesian Multi-trait quantitative-genetic models.

```
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_permutation.h>
#include <gsl/gsl_sort_vector.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_cdf.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_statistics.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_machine.h>
#include <cmath>
#include <algorithm>
#include <vector>
#include <list>
#include <iostream>
#include <fstream>
#include <omp.h>
#include <MuGen.h>
```

Include dependency graph for MuGen.cpp:



Functions

- void MVgauss (const gsl_vector *mn, const gsl_matrix *SigChl, const gsl_rng *r, gsl_vector *samp)

  Multivariate Gaussian sampling function.
- void **Wishart** (const gsl_matrix *SigC, const int &df, const gsl_rng *r, gsl_matrix *SigIout)
- void **Wishart** (const gsl_matrix *SigC, const size_t &df, const gsl_rng *r, gsl_matrix *SigIout)
- size_t rtgeom (const double &p, const size_t &limit, const gsl_rng *r)

  Truncated geometric distribution.
- double mhl (const gsl_vector *beta, const gsl_matrix *SigI)

  Mahalanobis distance.
- void colCenter (gsl_matrix *inplace)

  Matrix centering in-place.
- void colCenter (const gsl_matrix *source, gsl_matrix *res)

  Matrix centering with copy.
- void colCenter (gsl_matrix *inplace, const double &absLab)

  Matrix centering in-place with missing values.
- void colCenter (const gsl_matrix *source, gsl_matrix *res, const double &absLab)

  Matrix centering with copy and missing values.
- void vecCenter (gsl_vector *inplace)

  Vector centering in-place.
- void vecCenter (const gsl_vector *source, gsl_vector *res)

  Vector centering with copy.
- void meanImpute (gsl_matrix *inplace, const double &absLab)

  Mean imputation without centering.
- void printMat (const gsl_matrix *m)

  Print matrix to screen.
- unsigned long long rdtsc ()

  Accessing the processor RTDSC instruction.
- MuGrp operator+ (const Grp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const MuGrp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const Grp &m1, const MuGrp &m2)

  Addition operator.
- MuGrp operator- (const Grp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const MuGrp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const Grp &m1, const MuGrp &m2)

  Subtraction operator.

8.2.1 Detailed Description

Implementation of C++ classes for Hierarchical Bayesian Multi-trait quantitative-genetic models.

Author

Anthony J. Greenberg

Copyright

Version

0.9.1

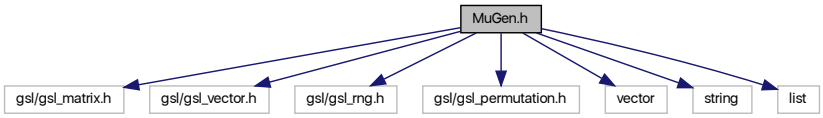
This file contains the implementation of the methods documented in the MuGen.h file.

8.3 MuGen.h File Reference

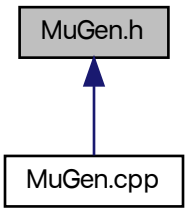
C++ classes for hierarchical Bayesian Multi-trait quantitative-genetic models.

```
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_permutation.h>
#include <vector>
#include <string>
#include <list>
```

Include dependency graph for MuGen.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MVnorm

  The abstract base class for location parameter rows.
- class MVnormMu

  Individual vector of means.
- class MVnormMuPEX

  Individual vector of means with parameter expansion.
- class MVnormBetaPEX

  Individual regression with parameter expansion.
- class MVnormMuMiss

  Individual vector of means with missing data.
- class MVnormBeta

  Generic regression.
- class MVnormBetaFt

  Regression with multiple predictors.
- class MVnormMuBlk

  Individual vector of means with blocks of traits.
- class MVnormBetaBlk

  Individual vector of regression coefficients with blocks of traits.
- class MVnormBetaFtBlk

  Individual vector of conditional regression coefficients with blocks of traits.
- class RanIndex

  Generic index class.
- class RanIndexVS

  BVSR index class.
- class Apex

  Multiplicative redundant parameter.
- class Grp

  Base location parameter group class.
- class MuGrp

Hierarchical mean.

- class MuGrpPEX

  "Random effect" with parameter expansion
- class MuGrpMiss

  Data with missing phenotype values.
- class MuGrpEE

  Data with measurement error.
- class MuGrpEEmiss

  Data with measurement error and missing phenotypes.
- class BetaGrpFt

  Multivariate multiple regression.
- class BetaGrpPEX

  Multivariate multiple regression with parameter expansion.
- class BetaGrpPC

  Relationship matrix regression.
- class BetaGrpPCpex

  Multiplicative parameter expansion for PC regression.
- class BetaGrpSnp

  Simple single-SNP regression class.
- class BetaGrpSnpCV

  Single-SNP regression with conditional variance.
- class BetaGrpPSR

  Single-SNP regression with partial effects.
- class BetaGrpSnpMiss

  Simple single-SNP regression class with missing data.
- class BetaGrpSnpMissCV

  Single-SNP regression with conditional variance and missing data.
- class BetaGrpPSRmiss

  Single-SNP regression with partial effects and missing data.
- class BetaGrpBVSR

  Bayesian variable selection regression.
- class MuBlk

  Hierarchical mean with independent blocks of traits.
- class BetaBlk

  Regression with independent blocks of traits.
- class SigmaI

  Basic inverse-covariance.
- class SigmaIblk

  Block-diagonal inverse-covariance.
- class SigmaIpex

  PEX inverse-covariance.
- class Qgrp

  Standard Student-t weights.
- class QgrpPEX

  Student-t weights for PEX.
- class MixP

  Dirichlet-multinomial mixture prior.

Functions

- void MVgauss (const gsl_vector *, const gsl_matrix *, const gsl_rng *, gsl_vector *)

  Multivariate Gaussian sampling function.
- void **Wishart** (const gsl_matrix *, const int &, const gsl_rng *, gsl_matrix *)
- void **Wishart** (const gsl_matrix *, const size_t &, const gsl_rng *, gsl_matrix *)
- size_t rtgeom (const double &, const size_t &, const gsl_rng *)

  Truncated geometric distribution.
- double mhl (const gsl_vector *beta, const gsl_matrix *SigI)

  Mahalanobis distance.
- void colCenter (gsl_matrix *inplace)

  Matrix centering in-place.
- void colCenter (const gsl_matrix *source, gsl_matrix *res)

  Matrix centering with copy.
- void colCenter (gsl_matrix *inplace, const double &absLab)

  Matrix centering in-place with missing values.
- void colCenter (const gsl_matrix *source, gsl_matrix *res, const double &absLab)

  Matrix centering with copy and missing values.
- void vecCenter (gsl_vector *inplace)

  Vector centering in-place.
- void vecCenter (const gsl_vector *source, gsl_vector *res)

  Vector centering with copy.
- void meanImpute (gsl_matrix *inplace, const double &absLab)

  Mean imputation without centering.
- void printMat (const gsl_matrix *m)

  Print matrix to screen.
- unsigned long long rdtsc ()

  Accessing the processor RTDSC instruction.
- MuGrp operator+ (const Grp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const MuGrp &m1, const Grp &m2)

  Addition operator.
- MuGrp operator+ (const Grp &m1, const MuGrp &m2)

  Addition operator.
- MuGrp operator- (const Grp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const MuGrp &m1, const Grp &m2)

  Subtraction operator.
- MuGrp operator- (const Grp &m1, const MuGrp &m2)

  Subtraction operator.

8.3.1 Detailed Description

C++ classes for hierarchical Bayesian Multi-trait quantitative-genetic models.

Author

Anthony J. Greenberg

Copyright

Version

0.9.1

This is the project header file containing class definitions and interface documentation.

Index