LiMMBo Documentation

Release b

Hannah V. Meyer

Mar 14, 2018

Table of contents

CHAPTER 1

Install

LiMMBo is currently available via pip and will in the future be available through conda-forge . The latter platform provides the recommended installation of LIMIX, which LiMMBo heavily relies on.

The recommended way of installing both packages is the following: Install LIMIX via conda:

```
conda install -c conda-forge limix
```

After successful installation of LIMIX, simply install LiMMBo via pip:

```
pip install limmbo
```

CHAPTER 2

Data input

2.1 Data input

2.1.1 Parse command-line arguments

limmbo.io.parser.**getGWASargs**()

A detailed list of the command-line options for *getGWASargs* can be found in runGWAS.

limmbo.io.parser.**getVarianceEstimationArgs**()

A detailed list of the command-line options for *getVarianceEstimationArgs* can be found in *Variance decomposition*.

2.1.2 Read data

class limmbo.io.reader.**ReadData**(*verbose=True*)

Generate object containing all datasets relevant for the analysis. For variance decomposition, at least phenotypes and relatedness estimates need to be specified. For association testing with LMM, at least phenotype, relatedness estimates and genotypes need to be read.

getCovariates(*file_covariates=None*, *delim=', '*)

Reads [*N* x *K*] covariate matrix with [*N*] samples and [*K*] covariates.

Parameters

- **file_covariates** (*string*) – [*N* x (*K* +1)] covariates file with [*N*] sample IDs in the first column
- **delim** (*string*) – delimiter of covariates file, one of " ", ",", "t"

Returns

updated the following attributes of the ReadData instance:

- **self.covariates** (np.array): [*N* x *K*] covariates matrix

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.utils import file_type
>>> data = ReadData(verbose=False)
>>> file_covs = resource_filename('limmbo',
...                               'io/test/data/covs.csv')
>>> data.getCovariates(file_covariates=file_covs)
>>> data.covariates.index[:3]
Index([u'ID_1', u'ID_2', u'ID_3'], dtype='object')
>>> data.covariates.values[:3,:3]
array([[ 0.92734699,  1.59767659, -0.67263682],
       [ 0.57061985, -0.84679736, -1.11037123],
       [ 0.44201204, -1.61499228,  0.23302345]])
```

getGenotypes (*file_genotypes=None*, *delim=', '*)

Reads genotype file in the following formats: plink (.bed, .bim, .fam), gen (.gen, .sample) or comma-separated values (.csv) file.

Parameters

- **file_geno** (*string*) – path to phenotype file in .plink or .csv format - **plink format**:

  as specified in the plink user manual, binary plink format with .bed, .fam and .bim file

  - **.csv format**:
    - [(*NrSNP* + 1) x (*N'+1)] .csv file with: ['N*] sample IDs in the first row and [*NrSNP*] genotype IDs in the first column
    - sample IDs should be of type: chrom-pos-rsID for instance 22-50714616-rs6010226

- **delim** (*string*) – delimiter of genotype file (when text format), one of " ", ",", "t"

Returns

updated the following attributes of the ReadData instance:

- **self.genotypes** (np.array): [*N* x *NrSNPs*] genotype matrix
- **self.genotypes_info** (pd.dataframe): [*NrSNPs* x 2] dataframe with columns 'chrom' and 'pos', and rsIDs as index

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io.utils import file_type
>>> data = reader.ReadData(verbose=False)
>>> # Read genotypes in delim-format
>>> file_geno = resource_filename('limmbo',
...     'io/test/data/genotypes.csv')
>>> data.getGenotypes(file_genotypes=file_geno)
>>> data.genotypes.index[:4]
```

(continues on next page)

(continued from previous page)

```
Index([u'ID_1', u'ID_2', u'ID_3', u'ID_4'], dtype='object')
>>> data.genotypes.shape
(1000, 20)
>>> data.genotypes.values[:5,:5]
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [2., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
>>> data.genotypes_info[:5]
           chrom       pos
rs1601111      3  88905003
rs13270638     8  20286021
rs75132935     8  76564608
rs72668606     8  79733124
rs55770986     7   2087823
>>> ### read genotypes in plink format
>>> file_geno = resource_filename('limmbo',
...     'io/test/data/genotypes')
>>> data.getGenotypes(file_genotypes=file_geno)
>>> data.genotypes.values[:5,:5]
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [2., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
>>> data.genotypes_info[:5]
           chrom       pos
rs1601111      3  88905003
rs13270638     8  20286021
rs75132935     8  76564608
rs72668606     8  79733124
rs55770986     7   2087823
```

getPCs (*file_pcs=None*, *nrpcs=None*, *delim=', '*)

Reads file with [*N* x *PC*] matrix of [*PC*] principal components from the genotypes of [*N*] samples.

Parameters

- **file_pcs** (*string*) – [*N* x (*PC* +1)] PCA file with [*N*] sample IDs in the first column
- **delim** (*string*) – delimiter of PCA file, one of " ", ",", "t"
- **nrpcs** (*integer*) – Number of PCs to use (uses first nrpcs principal components)

Returns

updated the following attributes of the ReadData instance:

- **self.pcs** (np.array): [*N* x *PC*] principal component matrix

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.utils import file_type
>>> data = ReadData(verbose=False)
>>> file_pcs = resource_filename('limmbo',
...                     'io/test/data/pcs.csv')
>>> data.getPCs(file_pcs=file_pcs, nrpcs=10, delim=" ")
>>> data.pcs.index[:3]
Index([u'ID_1', u'ID_2', u'ID_3'], dtype='object', name=0)
>>> data.pcs.values[:3,:3]
array([[-0.02632738, -0.05791269, -0.03619099],
       [ 0.00761785,  0.00538956,  0.00624196],
       [ 0.01069307, -0.0205066 ,  0.02299996]])
```

getPhenotypes (*file_pheno=None*, *delim=', '*)

Reads [*N* x *P*] phenotype file; file ending must be either .txt or .csv

Parameters

- **file_pheno** (*string*) – path to [*(N'+1) x ('P'+1)] phenotype file with: ['N*] sample IDs in the first column and [*P*] phenotype IDs in the first row
- **delim** (*string*) – delimiter of phenotype file, one of " ", ",", "t"

Returns

updated the following attributes of the ReadData instance:

- **self.phenotypes** (np.array): [*N* x *P*] phenotype matrix

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.utils import file_type
>>> data = ReadData(verbose=False)
>>> file_pheno = resource_filename('limmbo',
...                                'io/test/data/pheno.csv')
>>> data.getPhenotypes(file_pheno=file_pheno)
>>> data.phenotypes.index[:3]
Index([u'ID_1', u'ID_2', u'ID_3'], dtype='object')
>>> data.phenotypes.columns[:3]
Index([u'trait_1', u'trait_2', u'trait_3'], dtype='object')
>>> data.phenotypes.values[:3,:3]
array([[-1.56760036, -1.5324513 ,  1.17789321],
       [-0.85655034,  0.48358151,  1.35664966],
       [ 0.10772832, -0.02262884, -0.27963328]])
```

getRelatedness (*file_relatedness*, *delim=', '*)

Read file of [*N* x *N*] pairwise relatedness estimates of [*N*] samples.

Parameters

- **file_relatedness** (*string*) – [(*N* + *1*) x N] .csv file with: [*N*] sample IDs in the first row
- **delim** (*string*) – delimiter of covariate file, one of " ", ",", " "

Returns

updated the following attributes of the ReadData instance:

- **self.relatedness** (np.array): [*N* x *N*] relatedness matrix

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.utils import file_type
>>> data = ReadData(verbose=False)
>>> file_relatedness = resource_filename('limmbo',
...                     'io/test/data/relatedness.csv')
>>> data.getRelatedness(file_relatedness=file_relatedness)
>>> data.relatedness.index[:3]
Index([u'ID_1', u'ID_2', u'ID_3'], dtype='object')
>>> data.relatedness.columns[:3]
Index([u'ID_1', u'ID_2', u'ID_3'], dtype='object')
>>> data.relatedness.values[:3,:3]
array([[1.00892922e+00, 2.00758504e-04, 4.30499103e-03],
       [2.00758504e-04, 9.98944885e-01, 4.86487318e-03],
       [4.30499103e-03, 4.86487318e-03, 9.85787665e-01]])
```

getSampleSubset (*file_samplelist=None*, *samplelist=None*)

Read file or string with subset of sample IDs to analyse.

Parameters

- **file_samplelist** (*string*) – "path/to/file_samplelist": file contains subset sample IDs with one ID per line, no header.
- **samplestring** (*string*) – comma-separated list of sample IDs e.g. "ID1,ID2,ID5,ID10".

Returns

(numpy array) array containing list of sample IDs

getTraitSubset (*traitstring=None*)

Limit analysis to specific subset of traits

Parameters **traitstring** (*string*) – comma-separated trait numbers (for single traits) or hyphen- separated trait numbers (for trait ranges) or combination of both for trait selection (1-based)

Returns

(numpy array) array containing list of trait IDs

Examples

```
>>> from limmbo.io import reader
>>> data = reader.ReadData(verbose=False)
>>> traitlist = data.getTraitSubset("1,3,5,7-10")
>>> print traitlist
[0 2 4 6 7 8 9]
```

getVarianceComponents (*file_Cg=None*, *file_Cn=None*, *delim_cg=', '*, *delim_cn=', '*)
Reads a comma-separated files with [*P* x *P*] matrices of [*P*] trait covariance estimates.

Parameters

- **file_Cg** (*string*) – [*P* x *P*] .csv file with [*P*] trait covariance estimates of the genetic component
- **file_Cn** (*string*) – [*P* x *P*] .csv file with [*P*] trait covariance estimates of the non-genetic (noise) component

Returns

updated the following attributes of the ReadData instance:

- **self.Cg** (np.array): [*P* x *P*] matrix with trait covariance of the genetic component
- **self.Cn** (np.array): [*P* x *P*] matrix with trait covariance of the non-genetic (noise) component

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> data = ReadData()
>>> file_Cg = resource_filename('limmbo',
...                     'io/test/data/Cg.csv')
>>> file_Cn = resource_filename('limmbo',
...                     'io/test/data/Cn.csv')
>>> data.getVarianceComponents(file_Cg=file_Cg, file_Cn=file_Cn)
>>> data.Cg.shape
(10, 10)
>>> data.Cn.shape
(10, 10)
>>> data.Cg[:3,:3]
array([[ 0.45446454, -0.21084613,  0.01440468],
       [-0.21084613,  0.11443656,  0.01250233],
       [ 0.01440468,  0.01250233,  0.02347906]])
>>> data.Cn[:3,:3]
array([[ 0.53654803, -0.14392748, -0.45483001],
       [-0.14392748,  0.88793093,  0.30539822],
       [-0.45483001,  0.30539822,  0.97785614]])
```

2.1.3 Check data

class limmbo.io.input.**InputData**(*verbose=True*)
Generate object containing all datasets relevant for variance decomposition (phenotypes, relatedness estimates) and pre-processing steps (check for common samples and sample order, covariates regression and phenotype transformation)

Parameters **verbose** (*bool*) – initialise verbose: should progress messages be printed to stdout

addCovariates (*covariates*, *covs_samples=None*)
Add [*N* x *K*] covariate data with [*N*] samples and [*K*] covariates to InputData instance.

Parameters

- **covariates** (*array-like*) – [*N x 'K*] covariate matrix of *N* individuals and *K* covariates; if pandas.DataFrame with covs_samples as index, covs_samples do not have to specified separately.
- **covs_samples** (*array-like*) – [*N*] sample ID

Returns

updated the following attributes of the InputData instance:

- **self.covariates** (pd.DataFrame): [*N* x *K*] covariates matrix
- **self.covs_samples** (np.array): [*N*] sample IDs

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy as np
>>> import pandas as pd
>>> covariates = [(1,2,4),(1,1,6),(0,4,8)]
>>> covs_samples = ['S1','S2', 'S3']
>>> covariates = pd.DataFrame(covariates, index=covs_samples)
>>> indata = input.InputData(verbose=False)
>>> indata.addCovariates(covariates = covariates,
...     covs_samples = covs_samples)
>>> print indata.covariates.shape
(3, 3)
>>> print indata.covs_samples.shape
(3,)
```

addGenotypes(*genotypes*, *geno_samples=None*, *genotypes_info=None*)

Add [*N* x *NrSNP*] genotype array of [*N*] samples and [*NrSNP*] genotypes, [*N*] array of sample IDs and [*NrSNP* x 2] dataframe of genotype description to InputData instance.

Parameters

- **genotypes** (*array-like*) – [*N* x *NrSNP*] genotype array of [*N*] samples and [*NrSNP*] genotypes; if pandas.DataFrame with geno_samples as index, geno_samples do not have to specified separately.
- **geno_samples** (*array-like*) – [*N*] vector of *N* sample IDs
- **genotypes_info** (*dataframe*) – [*NrSNPs* x 2] dataframe with columns 'chrom' and 'pos', and rsIDs as index

Returns

updated the following attributes of the InputData instance:

- **self.genotypes** (pd.DataFrame): [*N* x *NrSNPs*] genotype matrix
- **self.geno_samples** (np.array): [*N*] sample IDs
- **self.genotypes_info** (pd.DataFrame): [*NrSNPs* x 2] dataframe with columns 'chrom' and 'pos', and rsIDs as index

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io import input
>>> data = reader.ReadData(verbose=False)
>>> file_geno = resource_filename('limmbo',
...                               'io/test/data/genotypes.csv')
>>> data.getGenotypes(file_genotypes=file_geno)
>>> indata = input.InputData(verbose=False)
>>> indata.addGenotypes(genotypes=data.genotypes,
...                     genotypes_info=data.genotypes_info)
>>> indata.geno_samples[:5]
array(['ID_1', 'ID_2', 'ID_3', 'ID_4', 'ID_5'], dtype=object)
>>> indata.genotypes.shape
(1000, 20)
>>> indata.genotypes.values[:5,:5]
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [2., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
>>> indata.genotypes_info[:5]
           chrom       pos
rs1601111      3  88905003
rs13270638     8  20286021
rs75132935     8  76564608
rs72668606     8  79733124
rs55770986     7   2087823
```

addPCs (*pcs*, *pc_samples=None*)

Add [*N* x *PC*] matrix of [*PC*] principal components from the genotypes of [*N*] samples to InputData instance.

Parameters

- **pcs** (*array-like*) – [*N x 'PCs*] principal component matrix of *N* individuals and *PCs* principal components; if pandas.DataFrame with pc_samples as index, covs_samples do not have to specified separately.
- **pc_samples** (*array-like*) – [*N*] sample IDs

Returns

updated the following attributes of the InputData instance:

- **self.pcs** (pd.DataFrame): [*N* x *PCs*] principal component matrix
- **self.pc_samples** (np.array): [*N*] sample IDs

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io import input
>>> data = reader.ReadData(verbose=False)
```

(continues on next page)

(continued from previous page)

```
>>> file_pcs = resource_filename('limmbo',
...                      'io/test/data/pcs.csv')
>>> data.getPCs(file_pcs=file_pcs, nrpcs=10, delim=" ")
>>> indata = input.InputData(verbose=False)
>>> indata.addPCs(pcs = data.pcs)
>>> print indata.pcs.shape
(1000, 10)
>>> print indata.pc_samples.shape
(1000,)
```

addPhenotypes (*phenotypes*, *pheno_samples=None*, *phenotype_ID=None*)

Add phenotypes, their phenotype ID and their sample IDs to InputData instance

Parameters

- **phenotypes** (*array-like*) – [*N x 'P*] phenotype matrix of *N* individuals and *P* phenotypes; if pandas.DataFrame with pheno_samples as index and phenotypes_ID as columns, pheno_samples and phenotype_ID do not have to specified separately.
- **pheno_samples** (*array-like*) – [*N*] sample ID
- **phenotype_ID** (*array-like*) – [*P*] phenotype IDs

Returns

updated the following attributes of the InputData instance:

- **self.phenotypes** (pd.DataFrame): [*N* x *P*] phenotype array
- **self.pheno_samples** (np.array): [*N*] sample IDs
- **self.phenotype_ID** (np.array): [*P*] phenotype IDs

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy as np
>>> import pandas as pd
>>> pheno = np.array(((1,2),(7,1),(3,4)))
>>> pheno_samples = ['S1','S2', 'S3']
>>> phenotype_ID = ['ID1','ID2']
>>> phenotypes = pd.DataFrame(pheno, index=pheno_samples,
...     columns = phenotype_ID)
>>> indata = input.InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = phenotypes)
>>> print indata.phenotypes.shape
(3, 2)
>>> print indata.pheno_samples.shape
(3,)
>>> print indata.phenotype_ID.shape
(2,)
```

addRelatedness (*relatedness*, *relatedness_samples=None*)

Add [*N* x *N*] pairwise relatedness estimates of [*N*] samples to the InputData instance

Parameters

- **relatedness** (*array-like*) – [*N x 'N*] relatedness matrix of *N* individuals; if pandas.DataFrame with relatedness_samples as index, relatedness_samples do not have to specified separately.
- **relatedness_samples** (*array-like*) – [*N*] sample IDs

Returns

updated the following attributes of the InputData instance:

- **self.relatedness** (pd.DataFrame): [*N* x *N*] relatedness matrix
- **self.relatedness_samples** (np.array): [*N*] sample IDs

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy
>>> import pandas as pd
>>> from numpy.random import RandomState
>>> from numpy.linalg import cholesky as chol
>>> random = RandomState(5)
>>> N = 100
>>> SNP = 1000
>>> X = (random.rand(N, SNP) < 0.3).astype(float)
>>> relatedness = numpy.dot(X, X.T)/float(SNP)
>>> relatedness_samples = numpy.array(
...     ['S{}'.format(x+1) for x in range(N)])
>>> relatedness = pd.DataFrame(relatedness,
...     index=relatedness_samples)
>>> indata = input.InputData(verbose=False)
>>> indata.addRelatedness(relatedness = relatedness)
>>> print indata.relatedness.shape
(100, 100)
>>> print indata.relatedness_samples.shape
(100,)
```

addVarianceComponents(*Cg*, *Cn*)

Add [*P* x *P*] matrices of [*P*] trait covariance estimates of the genetic trait variance component (Cg) and the non-genetic (noise) variance component (Cn) to InputData instance

Parameters

- **Cg** (*array-like*) – [*P x 'P*] matrix of *P* trait covariance estimates of the genetic trait covaraince component
- **Cn** (*array-like*) – [*P x 'P*] matrix of *P* trait covariance estimates of the non-genetic (noise) trait covaraince component

Returns

updated the following attributes of the InputData instance:

- **self.Cg** (np.array): [*P x 'P*] matrix of *P* trait covariance estimates of the genetic trait covariance component
- **self.Cn** (np.array): [*P x 'P*] matrix of *P* trait covariance estimates of the non-genetic trait covaraince component

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io import input
>>> import numpy as np
>>> from numpy.random import RandomState
>>> from numpy.linalg import cholesky as chol
>>> data = reader.ReadData(verbose=False)
>>> file_pheno = resource_filename('limmbo',
...                     'io/test/data/pheno.csv')
>>> data.getPhenotypes(file_pheno=file_pheno)
>>> file_Cg = resource_filename('limmbo',
...                     'io/test/data/Cg.csv')
>>> file_Cn = resource_filename('limmbo',
...                     'io/test/data/Cn.csv')
>>> data.getVarianceComponents(file_Cg=file_Cg,
...                            file_Cn=file_Cn)
>>> indata = input.InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = data.phenotypes)
>>> indata.addVarianceComponents(Cg = data.Cg, Cn=data.Cn)
>>> print indata.Cg.shape
(10, 10)
>>> print indata.Cg.shape
(10, 10)
```

commonSamples (*samplelist=None*)

Get [*M*] common samples out of phenotype, relatedness and optional covariates with [*N*] samples (if all samples present in all datasets [*M*] = [*N*]) and ensure that samples are in same order.

Parameters **samplelist** (*array-like*) – array of sample IDs to select from data

Returns

updated the following attributes of the InputData instance:

- **self.phenotypes** (pd.DataFrame): [*M* x *P*] phenotype matrix
- **self.pheno_samples** (np.array): [*M*] sample IDs
- **self.relatedness** (pd.DataFrame): [*M x M*] relatedness matrix
- **self.relatedness_samples** (np.array): [*M*] sample IDs of relatedness matrix
- **self.covariates** (pd.DataFrame): [*M* x *K*] covariates matrix
- **self.covs_samples** (np.array): [*M*] sample IDs
- **self.genotypes** (pd.DataFrame): [*M* x *NrSNPs*] genotypes matrix
- **self.geno_samples** (np.array): [*M*] sample IDs
- **self.pcs** (pd.DataFrame): [*M* x *PCs*] principal component matrix
- **self.pc_samples** (np.array): [*M*] sample IDs

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy as np
>>> import pandas as pd
>>> from numpy.random import RandomState
>>> from numpy.linalg import cholesky as chol
>>> random = RandomState(5)
>>> P = 2
>>> K = 4
>>> N = 10
>>> SNP = 1000
>>> pheno = random.normal(0,1, (N, P))
>>> pheno_samples = np.array(['S{}'.format(x+4)
...     for x in range(N)])
>>> phenotype_ID = np.array(['ID{}'.format(x+1)
...     for x in range(P)])
>>> phenotypes = pd.DataFrame(pheno, index=pheno_samples,
...     columns=phenotype_ID)
>>> X = (random.rand(N, SNP) < 0.3).astype(float)
>>> relatedness = np.dot(X, X.T)/float(SNP)
>>> relatedness_samples = np.array(['S{}'.format(x+1)
...     for x in range(N)])
>>> covariates = random.normal(0,1, (N-2, K))
>>> covs_samples = np.array(['S{}'.format(x+1)
...     for x in range(N-2)])
>>> indata = input.InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = pheno,
...                      pheno_samples = pheno_samples,
...                      phenotype_ID = phenotype_ID)
>>> indata.addRelatedness(relatedness = relatedness,
...                   relatedness_samples = relatedness_samples)
>>> indata.addCovariates(covariates = covariates,
...                      covs_samples = covs_samples)
>>> indata.covariates.shape
(8, 4)
>>> indata.phenotypes.shape
(10, 2)
>>> indata.relatedness.shape
(10, 10)
>>> indata.commonSamples(samplelist=["S4", "S6", "S5"])
>>> indata.covariates.shape
(3, 4)
>>> indata.phenotypes.shape
(3, 2)
>>> indata.relatedness.shape
(3, 3)
```

getAlleleFrequencies()

Compute allele frequencies of genotypes.

Returns

updated the following attributes of the InputData instance:

- **self.freqs** (pandas DataFrame): [*NrSNP* x *2*] matrix of alt and ref allele frequencies; index: snp IDs

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io import input
>>> from limmbo.utils.utils import makeHardCalledGenotypes
>>> from limmbo.utils.utils import AlleleFrequencies
>>> data = reader.ReadData(verbose=False)
>>> file_geno = resource_filename('limmbo',
...                               'io/test/data/genotypes.csv')
>>> data.getGenotypes(file_genotypes=file_geno)
>>> indata = input.InputData(verbose=False)
>>> indata.addGenotypes(genotypes=data.genotypes,
...                     genotypes_info=data.genotypes_info,
...                     geno_samples=data.geno_samples)
>>> freqs = indata.getAlleleFrequencies()
>>> freqs.iloc[:5,:]
                  p         q
rs1601111   0.292186  0.707814
rs13270638  0.303581  0.696419
rs75132935  0.024295  0.975705
rs72668606  0.119091  0.880909
rs55770986  0.169338  0.830662
```

regress()

Regress out covariates (optional).

Returns

updated the following attributes of the InputData instance:

- **self.phenotypes** (np.array): [*M* x *P*] phenotype matrix of residuals of linear model
- **self.covariates**: None

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy as np
>>> from numpy.random import RandomState
>>> from numpy.linalg import cholesky as chol
>>> random = RandomState(5)
>>> P = 5
>>> K = 4
>>> N = 100
>>> pheno = random.normal(0,1, (N, P))
>>> pheno_samples = np.array(['S{}'.format(x+1)
...     for x in range(N)])
>>> phenotype_ID = np.array(['ID{}'.format(x+1)
...     for x in range(P)])
>>> covariates = random.normal(0,1, (N, K))
>>> covs_samples = np.array(['S{}'.format(x+1)
...     for x in range(N)])
>>> indata = input.InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = pheno,
```

(continues on next page)

(continued from previous page)

```
...                       pheno_samples = pheno_samples,
...                       phenotype_ID = phenotype_ID)
>>> indata.addCovariates(covariates = covariates,
...                       covs_samples = covs_samples)
>>> indata.phenotypes.values[:3, :3]
array([[ 0.44122749, -0.33087015,  2.43077119],
       [ 1.58248112, -0.9092324 , -0.59163666],
       [-1.19276461, -0.20487651, -0.35882895]])
>>> indata.regress()
>>> indata.phenotypes.values[:3, :3]
array([[ 0.34421705, -0.01470998,  2.25710966],
       [ 1.69886647, -1.41756814, -0.55614649],
       [-1.10700674, -0.66017713, -0.22201814]])
```

standardiseGenotypes()

Standardise genotypes:

$$w_{ij} = \frac{x_{ij} - 2p_i}{\sqrt{2p_i(1-p_i)}}$$

where x_{ij} is the number of copies of the reference allele for the i th SNP of the j th individual and p_i is the frequency of the reference allele (as described in (Yang et al 2011)).

Returns

updated the following attributes of the InputData instance:

- **self.genotypes_sd** (numpy array): [*N* x *NrSNP*] matrix of *NrSNP* standardised genotypes for *N* samples.

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io import reader
>>> from limmbo.io import input
>>> from limmbo.utils.utils import makeHardCalledGenotypes
>>> from limmbo.utils.utils import AlleleFrequencies
>>> data = reader.ReadData(verbose=False)
>>> file_geno = resource_filename('limmbo',
...                               'io/test/data/genotypes.csv')
>>> data.getGenotypes(file_genotypes=file_geno)
>>> indata = input.InputData(verbose=False)
>>> indata.addGenotypes(genotypes=data.genotypes,
...                     genotypes_info=data.genotypes_info)
>>> geno_sd = indata.standardiseGenotypes()
>>> geno_sd.iloc[:5,:3]
             0         1       2
ID_1 -2.201123 -2.141970 -8.9622
ID_2 -2.201123 -2.141970 -8.9622
ID_3 -2.201123 -2.141970 -8.9622
ID_4  0.908627 -0.604125 -8.9622
ID_5 -0.646248 -2.141970 -8.9622
```

subsetTraits(*traitlist=None*)

Limit analysis to specific subset of traits

Parameters **traitlist** (*array-like*) – array of trait numbers to select from phenotypes

Returns

updated the following attributes of the InputData instance:

- **self.traitlist** (list): of [t] trait numbers (int) to choose for analysis
- **self.phenotypes** (pd.DataFrame): reduced set of [N x t] phenotypes
- **self.phenotype.ID** (np.array): reduced set of [t] phenotype IDs

Return type None

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.input import InputData
>>> from limmbo.io.utils import file_type
>>> data = ReadData(verbose=False)
>>> file_pheno = resource_filename('limmbo',
...                                'io/test/data/pheno.csv')
>>> data.getPhenotypes(file_pheno=file_pheno)
>>> traitlist = data.getTraitSubset(traitstring="1-3,5")
>>> indata = InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = data.phenotypes)
>>> print indata.phenotypes.shape
(1000, 10)
>>> print indata.phenotype_ID.shape
(10,)
>>> indata.subsetTraits(traitlist=traitlist)
>>> print indata.phenotypes.shape
(1000, 4)
>>> print indata.phenotype_ID.shape
(4,)
```

transform(*transform*)

Transform phenotypes

Parameters **transform** (*string*) – transformation method for phenotype data:

- scale: mean center, divide by sd
- gaussian: inverse normalisation

Returns

updated the following attributes of the InputData instance:

- **self.phenotypes** (np.array): [N x P] (transformed) phenotype matrix

Return type None

Examples

```
>>> from limmbo.io import input
>>> import numpy as np
>>> from numpy.random import RandomState
```

(continues on next page)

(continued from previous page)

```
>>> from numpy.linalg import cholesky as chol
>>> random = RandomState(5)
>>> P = 5
>>> K = 4
>>> N = 100
>>> pheno = random.normal(0,1, (N, P))
>>> pheno_samples = np.array(['S{}'.format(x+1)
...     for x in range(N)])
>>> phenotype_ID = np.array(['ID{}'.format(x+1)
...     for x in range(P)])
>>> SNP = 1000
>>> X = (random.rand(N, SNP) < 0.3).astype(float)
>>> relatedness = np.dot(X, X.T)/float(SNP)
>>> relatedness_samples = np.array(['S{}'.format(x+1)
...     for x in range(N)])
>>> indata = input.InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = pheno,
...                      pheno_samples = pheno_samples,
...                      phenotype_ID = phenotype_ID)
>>> indata.addRelatedness(relatedness = relatedness,
...                   relatedness_samples = relatedness_samples)
>>> indata.phenotypes.values[:3, :3]
array([[ 0.44122749, -0.33087015,  2.43077119],
       [ 1.58248112, -0.9092324 , -0.59163666],
       [-1.19276461, -0.20487651, -0.35882895]])
>>> indata.transform(transform='gaussian')
>>> indata.phenotypes.values[:3, :3]
array([[ 0.23799988, -0.11191464,  2.05785598],
       [ 1.41041953, -0.81365681, -0.92217818],
       [-1.55977999,  0.01240937, -0.62091817]])
```

CHAPTER 3

Variance decomposition

3.1 LiMMBo

class limmbo.core.vdbootstrap.**LiMMBo**(*datainput*, *S*, *timing=False*, *iterations=10*, *verbose=False*)

combineBootstrap(*results*)

Combine the [*S* x *S*] subset covariance matrices to find the overall [*P* x *P*] covariance matrices Cg and Cn.

Parameters **results** (*list*) – results of runBootstrapVarianceDecomposition()

Returns

dictionary containing:

- **Cg_fit** (numpy.array): [*P* x *P*] genetic covariance matrix via fitting
- **Cn_fit** (numpy.array): [*P* x *P*] noise covariance matrix via fitting
- **Cg_average** (numpy.array): [*P* x *P*] genetic covariance matrix via simple average
- **Cn_average** (numpy.array): [*P* x *P*] noise covariance matrix via simple average
- **Cg_all_bs** (numpy.array): [*runs* x *S* x *S*] genetic covariance matrices of *runs* phenotype subsets of size *S*
- **Cn_all_bs** (numpy.array): [*runs* x *S* x *S*] noise covariance matrices of *runs* phenotype subsets of size *S*
- **proc_time_ind_bs** (list): individual run times for all variance decomposition runs of [*S* x *S*] Cg and Cn
- **proc_time_sum_ind_bs** (list): sum of individual run times for all variance decomposition runs of [*S* x *S*] Cg and Cn
- **proc_time_combine_bs** (list): run time for finding [*P* x *P*] trait covariance estimates from fitting [*S* x *S*] bootstrap covariance estimates
- **nr_of_bs** (int): number of bootstrap runs

- **nr_of_successful_bs** (int): total number of successful bootstrapping runs i.e. variance decomposition converged
- **results_fit_Cg** (): results parameters of the bfgs-fit of the genetic covariance matrices (via scipy.optimize.fmin_l_bfgs_g)
- **results_fit_Cn** (): results parameters of the bfgs-fit of the noise covariance matrices (via scipy.optimize.fmin_l_bfgs_g)

Return type (dictionary)

runBootstrapCovarianceEstimation(*seed*, *cpus*, *minCooccurrence=3*, *n=None*)

Distribute variance decomposition of subset matrices via pp

Parameters

- **seed** (*int*) – seed to initialise random number generator for bootstrapping
- **minCooccurrence** (*int*) – minimum number a trait pair should be sampled; once reached for all trait pairs, sampling is stopped if n is None; default=3
- **n** (*int*) – if not None, sets the total number of permutations, otherwise n determined by minCooccurrence; default: None
- **cpus** (*int*) – number of cpus available for covariance estimation

Returns list containing variance components for all bootstrap runs

Return type (list)

saveVarianceComponents(*resultsCombineBootstrap*, *output*, *intermediate=True*)

Save variance components as comma-separated files or python objects (via Cpickle).

Parameters

- **resultsCombineBootstrap** (*dictionary*) –
  - **Cg_fit** (numpy.array): [P x P] genetic covariance matrix via fitting
  - **Cn_fit** (numpy.array): [P x P] noise covariance matrix via fitting
  - **Cg_average** (numpy.array): [P x P] genetic covariance matrix via simple average
  - **Cn_average** (numpy.array): [P x P] noise covariance matrix via simple average
  - **Cg_all_bs** (numpy.array): [*runs* x S x S] genetic covariance matrices of *runs* phenotype subsets of size S
  - **Cn_all_bs** (numpy.array): [*runs* x S x S] noise covariance matrices of *runs* phenotype subsets of size S
  - **proc_time_ind_bs** (list): individual run times for all variance decomposition runs of [S x S] Cg and Cn
  - **proc_time_sum_ind_bs** (list): sum of individual run times for all variance decomposition runs of [S x S] Cg and Cn
  - **proc_time_combine_bs** (list): run time for finding [P x P] trait covariance estimates from fitting [S x S] bootstrap covariance estimates
  - **nr_of_bs** (int): number of bootstrap runs
  - **nr_of_successful_bs** (int): total number of successful bootstrapping runs i.e. variance decomposition converged
  - **results_fit_Cg** (): results parameters of the bfgs-fit of the genetic covariance matrices (via scipy.optimize.fmin_l_bfgs_g)

- **results_fit_Cn** (): results parameters of the bfgs-fit of the noise covariance matrices (via scipy.optimize.fmin_l_bfgs_g)

- **output** (*string*) – path/to/directory where variance components will be saved; needs writing permission
- **intermediate** (*bool*) – if set to True, intermediate variance components (average covariance matrices, bootstrap matrices and results parameter of BFGS fit) are saved

Returns None

3.2 Standard REML

limmbo.core.vdsimple.**vd_reml**(*datainput*, *iterations=10*, *verbose=True*)

Compute variance decomposition of phenotypes into genetic and noise covariance via standard REML: approach implemented in LIMIX

Parameters

- **datainput** (InputData) – object with ID-matched [*N* x *P*] phenotypes with [*N*] individuals and [*P*] phenotypes and [*N* x *N*] relatedness estimates
- **output** (*string, optional*) – output directory with user-writing permissions; needed if caching is True
- **cache** (*bool, optional*) – should results be cached
- **verbose** (*bool, optional*) – should messages be printed to stdout

Returns

tuple containing:

- **Cg** (numpy.array): [*P* x *P*] genetic variance component
- **Cn** (numpy.array): [*P* x *P*] noise variance component
- **process_time** (double): cpu time of variance decomposition

Return type (tuple)

Examples:

```
>>> import numpy
>>> from numpy.random import RandomState
>>> from numpy.linalg import cholesky as chol
>>> from limmbo.core.vdsimple import vd_reml
>>> from limmbo.io.input import InputData
>>> random = RandomState(10)
>>> N = 100
>>> S = 1000
>>> P = 5
>>> snps = (random.rand(N, S) < 0.2).astype(float)
>>> kinship = numpy.dot(snps, snps.T) / float(10)
>>> y  = random.randn(N, P)
>>> pheno = numpy.dot(chol(kinship), y)
>>> pheno_ID = [ 'PID{}'.format(x+1) for x in range(P)]
>>> samples = [ 'SID{}'.format(x+1) for x in range(N)]
    >>> datainput = InputData()
>>> datainput.addPhenotypes(phenotypes = pheno,
...                         phenotype_ID = pheno_ID,
```

(continues on next page)

(continued from previous page)

```
...                           pheno_samples = samples)
>>> datainput.addRelatedness(relatedness = kinship,
...                           relatedness_samples = samples)
>>> Cg, Cn, ptime = vd_reml(datainput, verbose=False)
>>> Cg.shape
(5, 5)
```

CHAPTER 4

Association analysis

class limmbo.core.gwas.**GWAS**(*datainput*, *seed=10*, *verbose=True*, *searchDelta=False*)

computeEmpiricalP(*pvalues*, *nrpermutations=1000*)

Compute empirical p-values: permute the genotypes, do the association test, record if permuted p-value of SNP is smaller than original p-value. Sum these occurrences and divide by total number of permutation.

Parameters

- **pvalues** (*array-like*) – [*P* x *NrSNP*] (single-trait) or [1 x *NrSNP*] (multi-trait) array of p-values
- **nrpermutations** (*int*) – number of permutations; 1/nrpermutations is the maximum level of significance (alpha)to test for, e.g. nrpermuations=100 -> alpha=0.01

Returns [*P* x *NrSNP*] (single-trait) or [1 x *NrSNP*] (multi-trait) array of empirical p-values

Return type (numpy array)

computeFDR(*fdr*)

Create an empirical p-values distribution by permuting the genotypes and running the association test on these (10 random permuations). Recored the observed pvalues, order by rank and find the rank of the desired FDR to determine the empirical FDR.

Parameters **fdr** (*float*) – desired fdr threshold

Returns

tuple containing:

- **empirical fdr** (float):
- **empirical_pvalue_distribution** (numpy array): array of empirical p-values for 10 permutations tests

Return type (tuple)

manhattanQQ(*results*, *colourS='DarkBLue'*, *colourNS='Orange'*, *alphaS=0.5*, *alphaNS=0.1*, *thr_plotting=None*, *saveTo=None*)

Plot manhattan and quantile-quantile plot of association results.

Parameters

- **results** (*dictionary*) – dictionary generated via runAssociation analysis
- **colourS** (*string, optional*) – colour of significant points
- **colourNS** (*string, optional*) – colour of non-significant points
- **alphaS** (*float, optional*) – plotting transparency of significant points
- **alphaNS** (*float, optional*) – plotting transparency of non-significant points
- **thr_plotting** (*float, optional*) – y-intercept for horizontal line as a marker for significance
- **saveTo** (*string, optional*) – /path/to/output/directory to automatically save plot as pdf; needs user writing permission.

Returns (None)

runAssociationAnalysis(*mode*, *setup='lmm'*, *adjustSingleTrait=None*)

Analysing the association between phenotypes, genotypes, optional covariates and random genetic effects.

Parameters

- **mode** (*string*) – pecifies the type of linear model: either 'multitrait' for multivariate analysis or 'singletrait' for univariate analysis.
- **setup** (*string*) – specifies the linear model setup: either 'lmm' for linear mixed model or 'lm' for a simple linear model.
- **adjustSingleTrait** (*string*) – Method to adjust single-trait association p-values for testing multiple traits; If None (default) no adjusting. Options are 'bonferroni' (for bonferroni correction') or 'effective' (for correcting for the effective number of tests as described in (Galwey,2009).

Returns

dictionary containing:

- **lm** (`limix.qtl.LMM`): LIMIX LMM object
- **pvalues** (numpy array): [*NrSNP* x *P*] (when mode is singletrait) or [1 x'NrSNP'] array of p-values.
- **betas** (numpy array): [*NrSNP* x *P*] array of effect size estimates per SNP across all traits.
- **pvalues_adjust** (numpy array): only returned if mode is 'singletrait' and 'adjustSingleTrait' is not None; contains single-trait p-values adjusted for the number of single-trait analyses conducted.

Return type (dictionary)

Examples

```
>>> from pkg_resources import resource_filename
>>> from limmbo.io.reader import ReadData
>>> from limmbo.io.input import InputData
>>> from limmbo.core.gwas import GWAS
>>> data = ReadData(verbose=False)
>>> file_pheno = resource_filename('limmbo',
...                                'io/test/data/pheno.csv')
>>> file_geno = resource_filename('limmbo',
...                                'io/test/data/genotypes.csv')
>>> file_relatedness = resource_filename('limmbo',
...                     'io/test/data/relatedness.csv')
>>> file_covs = resource_filename('limmbo',
...                               'io/test/data/covs.csv')
>>> file_Cg = resource_filename('limmbo',
...                     'io/test/data/Cg.csv')
>>> file_Cn = resource_filename('limmbo',
...                     'io/test/data/Cn.csv')
>>> data.getPhenotypes(file_pheno=file_pheno)
>>> data.getCovariates(file_covariates=file_covs)
>>> data.getRelatedness(file_relatedness=file_relatedness)
>>> data.getGenotypes(file_genotypes=file_geno)
>>> data.getVarianceComponents(file_Cg=file_Cg,
...                            file_Cn=file_Cn)
>>> indata = InputData(verbose=False)
>>> indata.addPhenotypes(phenotypes = data.phenotypes)
>>> indata.addRelatedness(relatedness = data.relatedness)
>>> indata.addCovariates(covariates = data.covariates)
>>> indata.addGenotypes(genotypes=data.genotypes,
...                     genotypes_info=data.genotypes_info)
>>> indata.addVarianceComponents(Cg = data.Cg, Cn=data.Cn)
>>> indata.commonSamples()
>>> indata.regress()
>>> indata.transform(transform="scale")
>>> gwas = GWAS(datainput=indata, seed=10, verbose=False)
>>>
>>>
>>> # Example of multi-trait single-variant association testing
>>> # using a linear mixed model.
>>> resultsAssociation = gwas.runAssociationAnalysis(
...     setup="lmm", mode="multitrait")
>>> resultsAssociation.keys()
['lm', 'betas', 'pvalues']
>>> resultsAssociation['pvalues'].shape
(1, 20)
>>> '{:0.3e}'.format(resultsAssociation['pvalues'].min())
'4.584e-09'
>>> resultsAssociation['betas'].shape
(10, 20)
>>>
>>>
>>> # Example of single-trait single-variant association testing
>>> # using a linear mixed model.
>>> resultsAssociation = gwas.runAssociationAnalysis(
...     setup="lmm", mode="singletrait",
...     adjustSingleTrait = "effective")
>>> resultsAssociation.keys()
['pvalues_adjust', 'lm', 'betas', 'pvalues']
```

(continues on next page)

(continued from previous page)

```
>>> resultsAssociation['pvalues'].shape
(10, 20)
>>> resultsAssociation['betas'].shape
(10, 20)
>>> '{:0.3e}'.format(resultsAssociation['pvalues_adjust'].min())
'2.262e-03'
```

saveAssociationResults (*results*, *outdir*, *name=''*, *pvalues_empirical=None*)

Saves results of association analyses.

Parameters

- **results** (*dictionary*) – dictionary generated via runAssociation analysis, containing:
  - **lm** (`limix.qtl.LMM`): LIMIX LMM object
  - **pvalues** (numpy array): [*P* x *NrSNP*] array of p-values
  - **betas** (numpy array): [*P* x *NrSNP*] array of effect size estimates per SNP across all traits
  - **pvalues_adjust** (numpy array): only returned mode == singletrait and if 'adjustSingleTrait' is not None; contains single-trait p-values adjusted for the number of single-trait analyses conducted
- **outdir** (*string*) – '/path/to/output/directory'; needs user writing permission
- **name** (*string*) – input name specific name, such as chromosome used in analysis
- **pvalues_empirical** (*pandas dataframe, optional*) – empirical pvalues via adjustSingleTrait

Returns (None)

CHAPTER 5

Command-line interface

5.1 Association analysis

Models the association between phenotypes and genotypes, accepting additional covariates and parameters to account for population structure and relatedness between samples. Users can choose between single-trait and multi-trait models, simple linear or linear mixed model set-ups.

```
usage: runAssociation [-h] [-p FILE_PHENO] [--pheno_delim PHENO_DELIM]
                      [-g FILE_GENOTYPES] [--geno_delim GENOTYPES_DELIM] -o
                      OUTDIR (-st | -mt) (-lm | -lmm) [-n NAME] [-v]
                      [-k FILE_RELATEDNESS]
                      [--kinship_delim RELATEDNESS_DELIM] [-cg FILE_CG]
                      [--cg_delim CG_DELIM] [-cn FILE_CN]
                      [--cn_delim CN_DELIM] [-pcs FILE_PCS]
                      [--pcs_delim PCS_DELIM] [-c FILE_COVARIATES]
                      [--covariate_delim COVARIATE_DELIM]
                      [-adjustP {bonferroni,effective,None}]
                      [-nrpermutations NRPERMUTATIONS] [-fdr FDR] [-seed SEED]
                      [-tr {scale,gaussian}] [-reg] [-traitset TRAITSTRING]
                      [-nrpcs NRPCS]
                      [--file_samplelist FILE_SAMPLELIST | --samplelist SAMPLELIST]
                      [--plot] [-colourS COLOURS] [-colourNS COLOURNS]
                      [-alphaS ALPHAS] [-alphaNS ALPHANS] [-thr THR]
                      [--version]
```

5.1.1 Basic required arguments

-p, --file_pheno Path [string] to [(N+1) x (P+1)] .csv file of [P] phenotypes with [N] samples (first column: sample IDs, first row: phenotype IDs). Default: None

--pheno_delim Delimiter of phenotype file. g Default: ","

-g, --file_geno Genotype file: either [S x N].csv file (first column: SNP id, first row: sample IDs) or plink formated genotypes (.bim/.fam/.bed). Default: None

--geno_delim	Delimiter of genotype file (if not in plink format). Default: “,”
-o, --outdir	Path [string] of output directory; user needs writing permission. Default: None
-st, --singletrait	Set flag to conduct a single-trait association analysesDefault: False
-mt, --multitrait	Set flag to conduct a multi-trait association analysesDefault: False
-lm, --lm	Set flag to use a simple linear model for the associationanalysis
-lmm, --lmm	Set flag to use a linear mixed model for the associationanalysis

5.1.2 Output arguments

-n, --name	Name (used for output file naming). Default:
-v, --verbose	[bool]: should analysis progress be displayed. Default: False

5.1.3 Optional files

-k, --file_kinship	Path [string] to [N x (N+1)] file of kinship/relatedness matrix with [N] samples (first row: sample IDs). Required when –lmm/-lm. Default: None
--kinship_delim	Delimiter of kinship file. g Default: “,”
-cg, --file_cg	Required for large phenotype sizes when –lmm/-lm; computed via runLiMMBo; specifies file name for genetic trait covariance matrix (rows: traits, columns: traits). Default: None
--cg_delim	Delimiter of Cg file. g Default: “,”
-cn, --file_cn	Required for large phenotype sizeswhen –lmm/-lm; computed via runLiMMBo; specifies file name for non-genetic trait covariance matrix (rows: traits, columns: traits). Default: None
--cn_delim	Delimiter of Cn file. g Default: “,”
-pcs, --file_pcs	Path to [N x PCs] file of principal components from genotypes to be included as covariates (first column: sample IDs, first row: PC IDs); Default: None
--pcs_delim	Delimiter of PCs file. g Default: “,”
-c, --file_cov	Path [string] to [(N+1) x C] file of covariates matrix with [N] samples and [K] covariates (first column: sample IDs, first row: phenotype IDs). Default: None
--covariate_delim	Delimiter of covariates file. g Default: “,”

5.1.4 Optional association parameters

-adjustP, --adjustP	Possible choices: bonferroni, effective, None Method to adjust single-trait p-values formultiple hypothesis testing when runningmultiple single-trait GWAS: bonferroni/effective number of tests ‘(Galwey,2009) <http://onlinelibrary.wiley.com/doi/10.1002/gepi.20408/abstract>‘_Default: None
-nrpermutations, --nrpermutations	Number of permutations for computing empirical p-values; 1/nr-permutations is maximum level of testing for significance. Default: None
-fdr, --fdr	FDR threshold for computing empirical FDR. Default: None

-seed, --seed Seed [int] to inittiate random number generation for permutations. Default: 256

5.1.5 Optional data processing parameters

-tr, --transform_method Possible choices: scale, gaussian

Choose type [string] of data preprocessing: scale (mean center, divide by sd) or gaussian (inverse normalise). Default: "scale"

-reg, --reg_covariates [bool]: should covariates be regressed out? Default: False

5.1.6 Optional subsetting options

-traitset, --traitset Comma- (for list of traits) or hyphen- (for trait range) or comma and hyphen-separated list [string] of traits (trait columns) to choose; default: None (=all traits). Default: None

-nrpcs, --nrpcs First PCs to chose. Default: 10

--file_samplelist Path [string] to file with samplelist for sample selection, with one sample ID per line. Default: None

--samplelist Comma-separated list [string] of samples IDs to restrict analysis to, e.g. ID1,ID2,ID5,ID9,ID10. Default: None

5.1.7 Plot arguments

Arguments for depicting GWAS results as manhattan plot

--plot Set flag if results of association analysis should be depicted as manhattan and quantile-quantile plot

-colourS, --colourS Colour of significant points in manhattan plot

-colourNS, --colourNS Colour of non-significant points in manhattan plot

-alphaS, --alphaS Transparency of significant points in manhattan plot

-alphaNS, --alphaNS Transparency of non-significant points in manhattan plot

-thr, --threshold Significance threshold; when –fdr specified, empirical fdr used as threshold

5.1.8 Version

--version show program's version number and exit

5.2 Variance decomposition

Estimates the genetic and non-genetic traitcovariance matrix parameters of a linear mixed model with random genetic and non-genetic effect via a bootstrapping-based approach.

```
usage: runVarianceEstimation [-h] [-p FILE_PHENO] [--pheno_delim PHENO_DELIM]
                             [-k FILE_RELATEDNESS]
                             [--kinship_delim RELATEDNESS_DELIM] -o OUTDIR
                             [-c FILE_COVARIATES]
                             [--covariate_delim COVARIATE_DELIM] [-seed SEED]
                             -sp S [-r RUNS] [-t]
                             [--minCooccurrence MINCOOCCURRENCE]
                             [-i ITERATIONS] [-cpus CPUS]
                             [-tr {scale,gaussian}] [-reg]
                             [-traitset TRAITSTRING]
                             [--file_samplelist FILE_SAMPLELIST | --samplelist␣
↪SAMPLELIST]
                             [-dontSaveIntermediate] [-v] [--version]
```

5.2.1 Basic required arguments

-p, --file_pheno Path [string] to [(N+1) x (P+1)] .csv file of [P] phenotypes with [N] samples (first column: sample IDs, first row: phenotype IDs). Default: None

--pheno_delim Delimiter of phenotype file. g Default: “,”

-k, --file_kinship Path [string] to [N x (N+1)] file of kinship/relatedness matrix with [N] samples (first row: sample IDs). Required when –lmm/-lm. Default: None

--kinship_delim Delimiter of kinship file. g Default: “,”

-o, --outdir Path [string] of output directory; user needs writing permission. Default: None

5.2.2 Optional files

-c, --file_cov Path [string] to [(N+1) x C] file of covariates matrix with [N] samples and [K] covariates (first column: sample IDs, first row: phenotype IDs). Default: None

--covariate_delim Delimiter of covariates file. g Default: “,”

5.2.3 Bootstrapping parameters

-seed, --seed seed [int] used to generate bootstrap matrix. Default: 234

-sp, --smallp Size [int] of phenotype subsamples used for variance decomposition. Default: None

-r, --runs Total number [int] of bootstrap runs. Default: None

-t, --timing [bool]: should variance decomposition be timed. Default: False

--minCooccurrence Minimum count [int] of the pairwise sampling of any given trait pair. Default: 3

-i, --iterations Number [int] of iterations for variance decomposition attempts. Default: 10

-cpus, --cpus Number [int] of available CPUs for parallelisation of variance decomposition steps. Default: None

5.2.4 Optional data processing parameters

-tr, --transform_method Possible choices: scale, gaussian

Choose type [string] of data preprocessing: scale (mean center, divide by sd) or gaussian (inverse normalise). Default: "scale"

-reg, --reg_covariates [bool]: should covariates be regressed out? Default: False

5.2.5 Optional subsetting options

-traitset, --traitset Comma- (for list of traits) or hyphen- (for trait range) or comma and hyphen-separated list [string] of traits (trait columns) to choose; default: None (=all traits). Default: None

--file_samplelist Path [string] to file with samplelist for sample selection, with one sample ID per line.Default: None

--samplelist Comma-separated list [string] of samples IDs to restrict analysis to, e.g. ID1,ID2,ID5,ID9,ID10. Default: None

5.2.6 Output arguments

-dontSaveIntermediate, --dontSaveIntermediate Set to suppress saving intermediate variance components. Default: True

-v, --verbose [bool]: should analysis step description be printed. Default: False

5.2.7 Version

--version show program's version number and exit

CHAPTER 6

Additional functions

limmbo.utils.utils.**boolanize**(*string*)
Convert command line parameter "True"/"False" into boolean

Parameters

- **string** (*string*) –
- **or "True"** (*"False"*) –

Returns False/True

Return type (bool)

limmbo.utils.utils.**generate_permutation**(*P*, *S*, *n*, *seed=12321*, *exclude_zero=False*)
Generate permutation.

Parameters

- **seed** (*int*) – used as seed for pseudo-random numbers generation; default: 12321
- **n** (*int*) – number of permutations to generated
- **P** (*int*) – total number of traits
- **S** (*int*) – subsampling size
- **exclude_zero** (*bool*) – should zero be in set to draw from

Returns Returns list of length n containing [np.arrays] of length [*S*] with subsets/permutations of numbers range(P)

Return type (list)

limmbo.utils.utils.**getEigen**(*covMatrix*, *reverse=True*)
Get eigenvectors and values of hermitian matrix:

Parameters

- **covMatrix** (*array-like*) – hermitian matrix
- **reverse** (*bool*) – if True (default): order eigenvalues (and vectors) in decreasing order

Returns

tuple containing:

- eigenvectors
- eigenvalues

Return type (tuple)

limmbo.utils.utils.**getVariance**(*eigenvalue*)

Based on input eigenvalue computes cumulative sum and normalizes to overall sum to obtain variance explained

Parameters **eigenvalue** (*array-like*) – eigenvalues

Returns variance explained

Return type (float)

limmbo.utils.utils.**inflate_matrix**(*bootstrap_traits*, *bootstrap*, *P*, *zeros=True*)

Project small matrix into large matrix using indeces provided:

Parameters

- **bootstrap_traits** (*array-like*) – [S x S] covariance matrix estimates
- **bootstrap** (*array-like*) – [S x 1] array with indices to project [S x S] matrix values into [P x P] matrix
- **P** (*int*) – total number of dimensions
- **zeros** (*bool*) – fill void spaces in large matrix with zeros (True, default) or nans (False)

Returns Returns [P x P] matrix containing [S x S] matrix values at bootstrap indeces and zeros/nans elswhere

Return type (numpy array)

limmbo.utils.utils.**match**(*samples_ref*, *data_compare*, *samples_compare*, *squarematrix=False*)

Match the order of data and ID matrices to a reference sample order,

Parameters

- **samples_ref** (*array-like*) – [M] sammple Ids used as reference
- **data_compare** (*array-like*) – [N x L] data matrix with [N] samples and [L] columns
- **samples_compare** (*array-like*) – [N] sample IDs to be matched to samples_ref
- **squarematrix** (*bool*) – is data_compare a square matrix i.e. samples in cols and rows

Returns

tuple containing:

- data_compare (numpy array): [M x L] data matrix of input data_compare
- samples_compare (numpy array): [M] sample IDs of input samples_compare
- samples_before (int): number of samples in data_compare/samples_compare before matching to samples_ref
- samples_after (int): number of samples in data_compare/samples_compare after matching to samples_ref

Return type (tuple)

limmbo.utils.utils.**nans**(*shape*)
Create numpy array of NaNs

Parameters **shape** (*tuple*) – shape of the empty array

Returns numpy array of NaNs

Return type (numpy array)

limmbo.utils.utils.**regularize**(*m*, *verbose=True*)
Make matrix positive-semi definite by ensuring minimum eigenvalue >= 0: add absolute value of minimum eigenvalue and 1e-4 (for numerical stability of abs(min(eigenvalue) < 1e-4 to diagonal of matrix

Parameters **m** (*array-like*) – symmetric matrix

Returns

Returns tuple containing:

- positive, semi-definite matrix from input m (numpy array)
- minimum eigenvalue of input m

Return type (tuple)

limmbo.utils.utils.**scale**(*x*)
Mean center and unit variance input array

Parameters **x** (*array-like*) – array to be scaled by column

Returns mean-centered, unit-variance array of x

Return type (numpy array)

limmbo.utils.utils.**verboseprint**(*message*, *verbose=True*)
Print message if verbose option is True.

Parameters

- **message** (*string*) – text to print
- **verbose** (*bool*) – flag whether to print message (True) or not (False)

Python Module Index

l

Index

T

V