hlmm Documentation

Release 1.2.0a1

Alexander Thomas Ian Strudwick Young

Sep 13, 2018

Contents:

CHAPTER 1

Introduction

hlmm is a python library for fitting heteroskedastic linear mixed models to genetic data.

A heteroskedastic linear model (`hetlm.model`) can model the effect of a set of variables on the mean of a response (such as a continuous phenotype) and the effect of a (potentially different) set of variables of the variability of the response

A `hetlm.model` models all effects on both the mean and variance of a response as fixed effects. A heteroskedastic linear mixed model (`hetlmm.model`) adds modelling of random effects of a set of variables on the mean of the response.

Modelling random effects can make fitting a `hetlmm.model` very computationally demanding, with the number of operations scaling with the cube of sample size.

The package hlmm provides the ability to fit a `hetlmm.model` much more quickly when the number of variables with random effects is small compared to the sample size. We use an algorithm whose operations scale in proportion to the sample size multiplied by the number of random effects squared.

Main features

`hetlm.model`: define heteroskedastic linear models and find maximum likelihood estimates of parameters

`hetlmm.model`: define heteroskedastic linear mixed models and find maximum likelihood estimates of parameters

hlmm_chr.py (*Documentation for hlmm_chr.py script*): command line script that fits heteroskedastic linear models or heteroskedastic linear mixed models to a contiguous segment of the genome. The script takes bed formatted genotypes as input. and can incorporate covariates for the fixed effects on the mean and/or variance.

fit_hlmm_model.py (*Documentation for fit_hlmm_model.py script*): command line script that fits a heteroskedastic linear model or a heteroskedastic linear mixed model to a given response (phenotype), mean covariates, variance covariates, and variables to model random effects for.

Quick install

We recommend installing using pip (https://pip.pypa.io/en/stable/). At the command line, type

```
sudo pip install hlmm
```

Detailed Package Install Instructions

hlmm has the following dependencies:

python 2.7

Packages:

- numpy
- scipy
- pysnptools

We highly recommend using a python distribution such as Anaconda (https://store.continuum.io/cshop/anaconda/). This will come with both numpy and scipy installed and can include an MKL-compiled distribution for optimal speed.

To install from source, clone the git repository, and in the directory containing the HLMM source code, at the shell type

```
sudo python setupy.py install
```

or, on the windows command prompt, type

```
python setup.py install
```

Running tests

The tests directory contains scripts for testing the computation of the likelihoods, gradients, and maximum likelihood solutions for both heteroskedastic linear models (test_hetlm.py) and for heteroskedastic linear mixed models (test_hetlmm.py). To run these tests, a further dependency is required: numdifftools.

To run the tests, first install hlmm. Change to the tests/ directory and at the shell type

```
python test_hetlm.py
```

```
python test_hetlmm.py
```

Both tests should run without any failures.

CHAPTER 2

Tutorial

Tutorial on fitting heteroskedastic linear models (*hetlm.model*) to genetic data

To run this tutorial, you need plink installed and in your system path. (See http://zzz.bwh.harvard.edu/plink/download.shtml).

You also need R installed and in your system path. (See https://www.r-project.org/). Furthermore, you need to install the BEDMatrix package for R. To install this, type

```
install.packages('BEDMatrix')
```

at the command line prompt of an active R session.

In the installation folder for hlmm, there is an examples/ folder. Change to the examples/ folder.

In this folder, there is a bash script 'simulate_genotypes.sh'. At the command line, type

```
bash simulate_genotypes.sh
```

This will produce two .bed files, 'test.bed' and 'random.bed', each containing genotypes for 100,000 individuals at 500 SNPs. We will simulate phenotypes from these genotypes and then fit models to the genotypes in test.bed, while modelling random effects for the genotypes in random.bed.

To simulate the test phenotype, at the command line, type

```
Rscript phenotype_simulation.R
```

The phenotype simulation corresponds to the Gamma model described in the paper (link). Briefly, we simulate a Gamma distributed phenotype where every variant from both test.bed and random.bed has an effect on the mean of the trait and also an effect on the variance of the trait through the mean-variance relation of the Gamma distribution. The first variant in test.bed is also given a dispersion effect, an effect on the variance of the trait that cannot be explained by the general mean-variance relation of the Gamma distribution.

The phenotype is simulated so the heritability of the untransformed phenotype is 10%, i.e. the mean effects of the genetic variants in test.bed and random.bed explain 10% of the phenotypic variance.

The script performs an inverse-normal transformation of the phenotype before writing the phenotype to file as 'test_phenotype.fam'.

To fit HLMs (*hetlm.model*) to the SNPs in test.bed, type

```
python ../bin/hlmm_chr.py test.bed 0 500 test_phenotype.fam test
```

This produces a file 'test.models.gz' containing the results of fitting HLMs (*hetlm.model*) to all the SNPs in test.bed. This is a tab separated file with a header that can be read by R by typing the command

```
results=read.table('test.models.gz',header=T,stringsAsFactors=F)
```

The columns in this output file are:

SNP the SNP id from the .bim file

n the sample size used for that SNP

frequency the minor allele frequency of the SNP

likelihood the maximum of the log-likelihood of the data given the model

add the estimated additive effect of the SNP on the mean of the phenotype

add_se the standard error of the estimated additive effect

add_t the t-statistic for an additive effect of the SNP

add_pval the -log10(p-value) for an additive effect of the SNP

var the estimated log-linear variance effect of the SNP

var_se the standard error of the estimated log-linear variance effect

var_t the t-statistic for a log-linear variance effect of the SNP

var_pval the -log10(p-value) for a log-linear variance effect of the SNP

av_pval the -log10(p-value) from a combined (2 degree of freedom) test for additive and log-linear variance effects

You should find evidence for inflation of both additive and log-linear variance test statistics. Under the null, the squared t-statistic is asymptotically distribution as a Chi-Square distribution on 1 degree of freedom. The expected median of the squared statistics is therefore approximately 0.456.

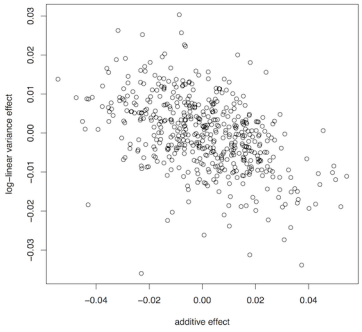
In our simulation, the median of the squared additive t-statistics was 5.63, clearly showing a strong signal for additive effects of SNPs.

In our simulation, the median of the squared log-linear variance t-statistics was 0.6748393, also showing evidence for log-linear variance effects. (Statistical significance for a deviation from the null can be tested by the Kolmogorov-Smirnov test. In our simulation, this gave a p-value of 3x10-4, showing significant inflation of log-linear variance test statistics).

There is evidence for inflation of log-linear variance effects because a relationship between the additive and log-linear variance effects exists. This exists because of the mean-variance relation of the Gamma distribution combined with the effects of inverse-normal transformation. To visualise it, type

```
plot(results$add,results$var,xlab='additive effect',ylab='log-linear
variance effect')
```

This should produce a plot that looks similar to this:



This shows a clear relationship between the additive effect of a SNP and its log-linear variance effect remains after inverse-normal transformation.

By inferring the relationship between additive effects and log-linear variance effects, one can estimate dispersion effects, which are effects on phenotypic variance that cannot be explained by a general mean-variance relation.

We have prepared an R script that estimates dispersion effects and adds them to the results table. To estimate dispersion effects, type

```
source('estimate_dispersion_effects.R')
```

The first SNP should have a dispersion effect. To see if there is evidence for this, example 'results[1,]', in particular, whether 'dispersion_pval' is large for the first SNP.

The other SNPs should not have dispersion effects. To test this, type

```
ks.test(results$dispersion_t[-1]^2,'pchisq',1)
```

The p-value should not be significant. This is in contrast to the log-linear variance effect p-value, which should be significant due to the general mean-variance relation.

We have shown how to infer additive, log-linear variance, and dispersion effects using HLMs (*hetlm.model*). We now show how to do the same while taking advantage of the favourable properties of linear mixed models for genetic association testing.

We model random effects for the genotypes in random.bed. All of these SNPs have (relatively weak) additive effects on the trait, so modelling random effects should increase power.

To fit HLMMs (*hetlmm.model*)to all loci in test.bed, at the UNIX terminal, type

```
python ../bin/hlmm_chr.py test.bed 0 500 test_phenotype.fam
test_random --random_gts random.bed
```

This will output 'test_random.models.gz' with the results of fitting the heteroskedastic linear mixed model to the SNPs in test.bed

It is much more computationally demanding to fit the mixed model, so this may take some time depending on your computer. Alternatively, one can fit the models for the first 10 SNPs:

```
python ../bin/hlmm_chr.py test.bed 0 10 test_phenotype.fam
test_random --random_gts random.bed
```

However, to estimate dispersion effects, one needs to have estimated additive and log-linear variance effects for a large number of SNPs. If one has fit models to all 500 SNPs, then one can analyse the results with the same process used for the non-mixed model analysis outlined above. The association signal for both additive and dispersion effects should be increased relative to the non-mixed model version.

CHAPTER 3

Documentation for 'hetlm' module

Documentation for the heteroskedastic linear model class.

class hetlm.**model**(*y*, *X*, *V*)

Define a heteroskedastic linear model and calculate likelihood, gradients, and maximum likelihood estimates of parameters.

Parameters

y [array] 1D array of phenotype observations

X [array] Design matrix for the fixed mean effects.

V [array] Design matrix for the fixed variance effects.

Returns

model [*hetlm.model*] heteroskedastic linear model class with input data

Methods

alpha_mle(beta)	Compute the maximum likelihood estimate of the fixed mean effect parameters, given particular fixed variance effect parameters and variance of random effects
alpha_ols()	Compute the ordinary least squares (OLS) estimate of the fixed mean effect parameters
approx_beta_mle()	Analytical approximation to the maximum likelihood estimate of the fixed variance effects
grad_beta(beta, alpha)	Compute the gradient with respect to the fixed variance effects of -2*L/n-log(2*pi), where L is the log-likelihood, the function that is minimized to find the MLE

Continued on next page

Table 1 – continued from previous page

likelihood(beta, alpha[, negative])	Compute the log of the likelihood, the likelihood at the maximum likelihood for the fixed mean effects
optimize_model()	Find the maximum likelihood estimate (MLE) of the parameters and their sampling distribution.

alpha_cov	
beta_cov	

alpha_mle(*beta*)

Compute the maximum likelihood estimate of the fixed mean effect parameters, given particular fixed variance effect parameters and variance of random effects

Parameters

beta [array] value of fixed variance effects

Returns

alpha [array] maximum likelihood estimate of alpha given beta

alpha_ols()

Compute the ordinary least squares (OLS) estimate of the fixed mean effect parameters

Returns

alpha [array] ordinary least-squares estimate of alpha

approx_beta_mle()

Analytical approximation to the maximum likelihood estimate of the fixed variance effects

Returns

beta [array] approximate MLE of beta

grad_beta(*beta*, *alpha*)

Compute the gradient with respect to the fixed variance effects of -2*L/n-log(2*pi), where L is the log-likelihood, the function that is minimized to find the MLE

Parameters

beta [array] value of fixed variance effects

alpha [array] value of fixed mean effects to gradient for

Returns

grad_beta [array]

likelihood(*beta*, *alpha*, *negative=False*)

Compute the log of the likelihood, the likelihood at the maximum likelihood for the fixed mean effects

Parameters

alpha [array] value of fixed mean effects to compute likelihood for

beta [array] value of fixed variance effects to compute likelihood for

negative [bool] compute -2*L/n-log(2*pi), where L is the log-likelihood, the function that is minimized to find the MLE. Default is False.

Returns

L [float] log-likelihood of data given parameters.

optimize_model()

Find the maximum likelihood estimate (MLE) of the parameters and their sampling distribution.

Returns

optim [dict] keys: MLEs ('alpha', fixed mean effects; 'beta', fixed variance effects), their standard errors ('alpha_se', 'beta_se'), covariance matrix for sampling distribution of parameter vectors ('beta_cov' and 'alpha_cov'), maximum likelihood ('likelihood'), whether optimisation was successful ('success'),

CHAPTER 4

Documentation for 'hetlmm' module

Documentation for the heteroskedastic linear mixed model class.

class hetlmm.**model**(*y*, *X*, *V*, *G*)

Define a heteroskedastic linear mixed model and calculate likelihood, gradients, and maximum likelihood estimates of parameters.

Parameters

y [array] 1D array of phenotype observations

X [array] Design matrix for the fixed mean effects.

V [array] Design matrix for the fixed variance effects.

G [array] Design matrix for the random effects.

Returns

model [*hetlmm.model*] heteroskedastic linear mixed model class with input data

Methods

alpha_mle(beta, h2)	Compute the maximum likelihood estimate of the fixed mean effect parameters, given particular fixed variance effect parameters and variance of random effects
alpha_ols()	Compute the ordinary least squares (OLS) estimate of the fixed mean effect parameters
likelihood(beta, h2[, negative])	Compute the log of the profile likelihood, the likelihood at the maximum likelihood for the fixed mean effects

Continued on next page

Table 1 – continued from previous page

likelihood_and_gradient(beta, h2[, return_grad])	Compute the function that is minimized to find the MLE, LL=-2*L/n-log(2*pi), where L is the log of the profile likelihood, the likelihood at the maximum for the fixed mean effects.
optimize_model(h2[, SEs, dx])	Find the maximum likelihood estimate (MLE) of the parameters and their sampling distribution.
parameter_covariance(alpha, beta, h2[, dx])	

alpha_mle(*beta*, *h2*)

Compute the maximum likelihood estimate of the fixed mean effect parameters, given particular fixed variance effect parameters and variance of random effects

Parameters

beta [array] value of fixed variance effects

h2: :class:'float' value of variance explained by random effects to compute likelihood for

Returns

alpha [array] maximum likelihood estimate of alpha given beta and h2

alpha_ols()

Compute the ordinary least squares (OLS) estimate of the fixed mean effect parameters

Returns

alpha [array] ordinary least-squares estimate of alpha

likelihood(*beta*, *h2*, *negative=False*)

Compute the log of the profile likelihood, the likelihood at the maximum likelihood for the fixed mean effects

Parameters

beta [array] value of fixed variance effects to compute likelihood for

h2: :class:'float' value of variance explained by random effects to compute likelihood for

negative [bool] compute -2*L/n-log(2*pi), where L is the log-likelihood, the function that is minimized to find the MLE. Default is False.

Returns

L [float] log-likelihood of data given parameters.

likelihood_and_gradient(*beta*, *h2*, *return_grad=True*)

Compute the function that is minimized to find the MLE, LL=-2*L/n-log(2*pi), where L is the log of the profile likelihood, the likelihood at the maximum for the fixed mean effects. Further, compute the gradient with respect to the fixed variance effects and the variance of the random effects. This forms the basis of the function passed to L-BFGS-B in order to find the maximum likelihood parameter estimates.

Parameters

beta [array] value of fixed variance effects to compute likelihood for

h2: :class:'float' value of variance explained by random effects to compute likelihood for

Returns

[LL,gradient] [list] the value of the function to be minimized, LL, and its gradient. The gradient is a 1d array that has the gradient with respect to beta first followed by the gradient with respect to h2.

optimize_model(*h2*, *SEs=True*, *dx=1e-06*)

Find the maximum likelihood estimate (MLE) of the parameters and their sampling distribution.

Parameters

h2 [float] initial value of variance explained by random effects

SEs [bool] whether to compute sampling distribution of parameter estimates. Default is True.

dx [float] the step size used to compute the Hessian for the computing the parameter sampling distribution

Returns

optim [dict] keys: MLEs ('alpha', fixed mean effects; 'beta', fixed variance effects; 'h2', variance explained by random effects), their standard errors ('alpha_se', 'beta_se', 'h2_se'), covariance matrix for sampling distribution of parameter vector ('par_cov', in order: alpha, beta, h2), maximum likelihood ('likelihood'), whether optimisation was successful ('success'), warnings from L-BFGS-B optimisation ('warnflag').

hetlmm.**simulate**(*n*, *l*, *alpha*, *beta*, *h2*)

Simulate from a heteroskedastic linear mixed model given a set of parameters. This uses a singular value decomposition to do the simulation quickly when l<<n.

The function simulates fixed and random effects design matrices of specified dimensions with independent Gaussian entries.

Parameters

n [int] sample size

l [int] number of random effects

alpha [array] value of fixed mean effects

beta [array] value of fixed variance effects

h2 [float] value of variance explained by random effects

Returns

model [*hetlmm.model*] heteroskedastic linear mixed model with data simulated from given parameters

CHAPTER 5

Documentation for hlmm_chr.py script

This script fits heteroskedastic linear models (HLMs) (*hetlm.model*) or heteroskedastic linear mixed models (HLMMs) (*hetlmm.model*) to a sequence of genetic variants contained in a .bed file. You need to specify the genotypes.bed file, which also has genotypes.bim and genotypes.fam in the same directory, along with the start and end indices of segment you want the script to fit models to.

The script runs from start to end-1 inclusive, and the first SNP has index 0. The script is designed to run on a chromosome segment to facilitate parallel computing on a cluster.

The phenotype file and covariate file formats are the same: plain text files with at least three columns. The first column is family ID, and the second column is individual ID; subsequent columns are phenotype or covariate observations. This is the same format used by GCTA and FaSTLMM.

If you specify a random_gts.bed file with the option –random_gts, the script will fit HLMMs (*hetlmm.model*), modelling random effects for the SNPs in random_gts.bed. If no –random_gts are specified, then HLMs (*hetlm.model*) are used, without random effects.

Minimally, the script will output a file outprefix.models.gz, which contains a table of the additive and log-linear variance effects estimated for each variant specified.

If –random_gts are specified, the script will output an estimate of the variance of the random effects in the null model in outprefix.null_h2.txt. –no_h2_estimate suppresses this output.

If covariates are also specified, it will output estimates of the covariate effects from the null model as outprefix.null_mean_effects.txt and outprefix.null_variance_effects.txt. –no_covariate_estimates suppresses this output.

Arguments

Required positional arguments:

genofile Path to genotypes in BED format

start Index of SNP in genofile from which to start computing test stats

end Index of SNP in genofile at which to finish computing test stats

phenofile Location of the phenotype (response) file with format: FID, IID, y1, y2, ...

outprefix Location to output csv file with association statistics

Options:

--mean_covar	Location of mean covariate file (default no mean covariates)
--var_covar	Locaiton of variance covariate file (default no variance covariates)
--fit_covariates	Fit covariates for each locus. Default is to fit covariates for the null model and project out (mean) and rescale (variance)
--random_gts	Location of the BED file with the genotypes of the SNPs that random effects should be modelled for. If random_gts are provided, HLMMs (*hetlmm.model*) are fit, rather than HLMs (*hetlm.model*).
--h2_init	Initial value for variance explained by random effects (default 0.05)
--phen_index	If the phenotype file contains multiple phenotypes, specify the phenotype to analyse. Default is first phenotype in file. Index counts starting from 1, the first phenotype in the phenotye file.
--min_maf	Ignore SNPs with minor allele frequency below min_maf (default 5%)
--missing_char	Missing value string in phenotype file (default NA)
--max_missing	Ignore SNPs with greater % missing calls than max_missing (default 5%)
--append	Append results to existing output file with given outprefix (default to open new file and overwrite existing file with same name)
--whole_chr	Fit models to all variants in .bed genofile. Overrides default to model SNPs with indices from start to end-1.
--no_covariate_estimates	Suppress output of covariate effect estimates
--no_h2_estimate	Suppress output of h2 estimate

Example Usage

We recommend working through the tutorial (*Tutorial*) to learn how to use hlmm_chr.py. We provide some additional examples of usage of the script here.

Minimal usage for fitting HLMs (*hetlm.model*):

```
python hlmm_chr.py genotypes.bed 0 500 phenotype.fam phenotype
```

This will fit heteroskedastic linear models to SNPs 0 to 499 in genotypes.bed using the first phenotype in phenotype.fam. It will output the results of fitting the models to phenotype.models.gz. See *Tutorial* for a description of the columns of phenotypes.models.gz.

Minimal usage for HLMMs (*hetlmm.model*):

```
python hlmm_chr.py genotypes.bed 0 500 phenotype.fam phenotype
--random_gts random.bed
```

This will fit heteroskedastic linear mixed models to SNPs 0 to 499 in genotypes.bed using the first phenotype in phenotype.fam. It will output the results of fitting the models to phenotype.models.gz. It will also output the estimate of h2, the variance of the random effects, to phenotype.null_h2.txt, unless –no_h2_estimate is added to the command.

Fitting covariates:

```
python hlmm_chr.py genotypes.bed 0 500 phenotype.fam phenotype
--mean_covar m_covariates.fam --var_covar v_covariates.fam
```

Before fitting locus specific models, the script will first fit a null model including the mean covariates in m_covariates.fam and the variance covariates in v_covariates.fam. The script will output the null model estimates of the mean covariates in phenotype.null_mean_effects.txt and null model estimates of the variance covariates in phenotypes.null_variance_effects.txt, unless –no_covariate_estimates is added to the command. Unless –fit_covariates is

added to the command, the phenotype is adjusted based on the null model estimates of the mean covariate effects and variance covariate effects. The adjusted phenotype is used to fit locus specific models without fitting the mean and variance covariates for each locus.

CHAPTER 6

Documentation for fit_hlmm_model.py script

This script fits a heteroskedastic linear model (HLMs) (*hetlm.model*) or heteroskedastic linear mixed model (HLMMs) (*hetlmm.model*) to a given response (phenotype), mean covariates, variance covariates, and variables to model random effects for.

The phenotype (response) file and covariate file formats are the same: plain text files with at least three columns. The first column is family ID, and the second column is individual ID; subsequent columns are phenotype or covariate observations. This is the same format used by GCTA and FaSTLMM.

If you specify a random_gts.bed file with the option –random_gts, the script will fit a HLMM (*hetlmm.model*), modelling random effects for the SNPs in random_gts.bed. If no –random_gts are specified, then a HLM (*hetlm.model*) is used, without random effects. If you add the flag –random_gts_txt, the program assumes that the file specified for –random_gts is a text file formatted as: FID, IID, x1, x2, ...

If mean and/or variance covariates are specified, the script will output two files: outprefix.mean_effects.txt, containing the estimated mean effects and their standard errors; and outprefix.variance_effects.txt, containing the estimated log-linear variance effects and their standard errors.

If –random_gts are specified, the script will output an estimate of the variance of the random effects in outprefix.h2.txt. –no_h2_estimate suppresses this output.

Arguments

Required positional arguments:

phenofile Location of the phenotype (response) file with format: FID, IID, y1, y2, ...

outprefix Location to output csv file with association statistics

Options:

--mean_covar Location of mean covariate file (default no mean covariates)

--var_covar Locaiton of variance covariate file (default no variance covariates)

--random_gts Location of the BED file with the genotypes of the SNPs that random effects should be modelled for. If random_gts are provided, HLMMs (*hetlmm.model*) are fit, rather than HLMs (*hetlm.model*).

--random_gts_txt	If this flag is specified, the program assumes the file specified in for –random_gts is formatted as a text file with at least three columns: FID, IID, x1, x2, ...
--h2_init	Initial value for variance explained by random effects (default 0.05)
--phen_index	If the phenotype file contains multiple phenotypes, specify the phenotype to analyse. Default is first phenotype in file. Index counts starting from 1, the first phenotype in the phenotye file.
--missing_char	Missing value string in phenotype file (default NA)
--no_h2_estimate	Suppress output of h2 estimate

Example Usage

We recommend working through the tutorial (*Tutorial*) to learn how to use hlmm_chr.py. The usage of this script is similar, if a little simpler.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index