hlmm Documentation

Release 1.2.0a1

Alexander Thomas Ian Strudwick Young

May 10, 2019

Contents:

1 Guide 1

2 Documentation for ‘sibreg’ module 3

3 Documentation for sGWAS.py script 7

4 Indices and tables 9

Python Module Index 11

CHAPTER 1

Guide

Introduction

sGWAS is a python library for performing regression with correlated observations within-class.

In the sGWAS/bin there is a script: 'sGWAS.py' (*Documentation for sGWAS.py script*). This script performs GWAS that fits both within and between family effects.

The core model is the sibreg model (`sibreg.model`), which consists of a linear model for the mean along with a vector of class labels that allows for correlations within-class. (The correlations within-class result from modelling mean differences between classes as independent, normally distributed random effects.) For the application in the script 'sGWAS.py', the classes are the families, so correlations are modelled between all siblings in a family.

The documentation for the sibreg module (*Documentation for 'sibreg' module*) contains information on how to define a `sibreg.model`, how to optimise a `sibreg.model`, how to predict from a `sibreg.model`, and how to simulate a `sibreg.model`.

Running tests

To check that the code is working properly and computing likelihoods and gradients accurately, you can run tests. In the sGWAS/tests subdirectory, type

```
python tests.py
```

The output should say

```
Ran 4 tests in...

OK
```

CHAPTER 2

Documentation for 'sibreg' module

Documentation for the sibreg model class.

class sibreg.**model**(*y*, *X*, *labels*)

Define a linear model with within-class correlations.

Parameters

y [array] 1D array of phenotype observations

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

labels [array] 1D array of sample labels

Returns

model [*sibreg.model*]

Methods

alpha_mle(self, tau[, sigma2, compute_cov])	Compute the MLE of alpha given variance parameters
likelihood_and_gradient(self, sigma2, tau)	Compute the loss function, which is -2 times the likelihood along with its gradient
optimize_model(self, init_params)	Find the parameters that minimise the loss function for a given regularisation parameter
predict(self, X)	Predict new observations based on model regression coefficients

set_alpha	

alpha_mle(*self*, *tau*, *sigma2=nan*, *compute_cov=False*)

Compute the MLE of alpha given variance parameters

Parameters

sigma2 [float] variance of model residuals

tau [float] ratio of variance of model residuals to variance explained by mean differences between classes

Returns

alpha [array] MLE of alpha

likelihood_and_gradient (*self*, *sigma2*, *tau*)

Compute the loss function, which is -2 times the likelihood along with its gradient

Parameters

sigma2 [float] variance of model residuals

tau [float] ratio of variance of model residuals to variance explained by mean differences between classes

Returns

L, grad [float] loss function and gradient, divided by sample size

optimize_model (*self*, *init_params*)

Find the parameters that minimise the loss function for a given regularisation parameter

Parameters

init_param [array] initial values for residual variance (sigma^2_epsilon) followed by ratio of residual variance to within-class variance (tau)

Returns

optim [dict] dictionary with keys: 'success', whether optimisation was successful (bool); 'warnflag', output of L-BFGS-B algorithm giving warnings; 'sigma2', MLE of residual variance; 'tau', MLE of ratio of residual variance to within-class variance; 'likelihood', maximum of likelihood.

predict (*self*, *X*)

Predict new observations based on model regression coefficients

Parameters

X [array] matrix of covariates to predict from

Returns

y [array] predicted values

sibreg.**simulate** (*n*, *alpha*, *sigma2*, *tau*)

Simulate from a linear model with correlated observations within-class. The mean for each class is drawn from a normal distribution.

Parameters

n [int] sample size

alpha [array] value of regression coefficeints

sigma2 [float] variance of residuals

tau [float] ratio of variance of residuals to variance of distribution of between individual means

Returns

model [`regrnd.model`] linear model with repeated observations

CHAPTER 3

Documentation for sGWAS.py script

This script uses genotypes of siblings to estimate 'within family' and 'between family' effects of SNPs.

The script fits models for all SNPs in a .bed file passing MAF and missingness thresholds.

The phenotype file should be a tab separate text file with columns FID, IID, Y1, Y2, ...

Siblings should have the same family id (FID), and non-siblings should have different FIDs.

The covariate file formats is the same. The first column is family ID, and the second column is individual ID; subsequent columns are phenotype or covariate observations.

Minimally, the script will output a file outprefix.models.gz, which contains a table of within family and between family effect estimates along with their standard errors and correlation.

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

Arguments

Required positional arguments:

genofile Path to genotypes in BED format

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)
--fit_covariates	Fit covariates for each locus. Default is to fit covariates for the null model and project out the covariates'
--tau_init	Initial value for the ratio of within family variance to residual variance. Default 1.0.
--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)

--no_covariate_estimates Suppress output of covariate effect estimates

--fit_VC fit variance components for each SNP. Default is to fix variance components at the MLE from the null model (much faster).

--sex_index provide an index (counting from 1) in the covariate file that gives sex coded as 0 for females and 1 for males. Providing an index causes the script to fit separate within and between family effects for men and women

Example Usage

Minimal usage:

```
python sGWAS.py genotypes.bed phenotype.fam phenotype
```

This will estimate between family and within family effects for all the SNPs in genotypes.bed passing MAF and missingness thresholds, using the first phenotype in phenotype.fam. It will output the results of fitting the models to phenotype.models.gz.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

sibreg, 3

Index

A

alpha_mle() (*sibreg.model method*), 3

L

likelihood_and_gradient() (*sibreg.model method*), 4

M

model (*class in sibreg*), 3

O

optimize_model() (*sibreg.model method*), 4

P

predict() (*sibreg.model method*), 4

S

sibreg (*module*), 3
simulate() (*in module sibreg*), 4