sibreg Documentation

Alexander Young

Apr 21, 2022

CONTENTS:

CHAPTER

ONE

TUTORIAL

Tutorial on inferring IBD between siblings, imputing missing parental genotypes, and performing family based GWAS and polygenic score analyses. Before working through the tutorial, please first install the package and run the tests (see github).

1.1 Test data

In the example/ directory, there is some example data. The file h2_quad_0.8.txt is a simulated phenotype with direct, paternal, and maternal effects, where 80% of the phenotypic variance is explained by the combined direct, paternal and maternal effects of the SNPs; and the pairwise correlations between the direct, paternal, and maternal effects are 0.5.

The genotype data has been simulated so that there are 3000 independent families, where 1000 have two siblings but no parents genotyped, 1000 have one parent genotyped and a 50% chance of having a genotyped sibling, and the final 1000 have both parents genotyped and a 50% chance of having a genotyped sibling. The example data includes genotype data formatted in both PLINK .bed format (example/sample.bed) and phased genotype data in .bgen format (example/sample.bgen with associated sample file example/sample.sample).

1.2 Inferring IBD between siblings

The first step is to infer the identity-by-descent segments shared between siblings. SNIPar contains a script, ibd.py, that employs a Hidden Markov Model (HMM) to infer the IBD segments for the sibling pairs. To infer the IBD segments from the genotype data in example/sample.bed, use the following command

```
python ibd.py example/sample --king example/sample.king.kin0 --agesex example/
sample.agesex --map example/sample.genetic_map.txt --outprefix example/
--threads 4
```

This will output the IBD segments to a gzipped text file example/chr_1.ibd.segments.gz. The *–king* argument requires the address of the relations (parent-offspring, sibling) inferred by KING by using the –related command, and the *–agesex* argument requires the address of a white-space separated text file with column 'FID' (family ID), 'IID' (individual ID), 'age', 'sex' (coded as 'M' for male and 'F' for female). It is necessary to provide the genetic map positions in centiMorgans (cM) of the SNPs: the *–map* argument allows the user to specify a genetic map (in the same format as used by SHAPEIT: https://mathgen.stats.ox.ac.uk/genetics_software/shapeit/shapeit.html#formats); otherwise, the script can use the genetic map positions (in cM) provided by the bim file. The *–threads* argument controls the number of threads used for computation.

If the user has a pedigree file with columns FID (family ID), IID (individual ID), FATHER_ID (father ID), MOTHER_ID (mother ID), they can input that instead of the *–king* and *–agesex* arguments. Missing IDs in the pedigree are denoted by 0. Siblings are inferred as individuals in the pedigree that share both parents. Using the example pedigree in example/sample.ped, you can infer IBD using this command:

```
python ibd.py example/sample --pedigree example/sample.ped --map example/
sample.genetic_map.txt --outprefix example/ --threads 4
```

While the script can be run for a .bed file containing a single chromosome, it can also be run for multiple bed files each containing a single chromosome. If the bed files are chr_1.bed, chr_2.bed, ..., chr_22.bed, then you can specify these files to the script as 'chr_~', where '~' is interpreted as a numerical wildcard character. The script first infers a genotyping error probability from Mendelian Errors between parents and offspring across all the input chromosomes, then it will infer the IBD segments shared between siblings for each chromosome.

The IBD inference can be performed on a smaller set of SNPs than will be imputed to save computation time. For example, IBD inference could be performed using SNPs from a genotyping array, and the imputation performed using all SNPs that have been imputed from a reference panel. For imputation from siblings, SNPs that fall outside of regions covered by the IBD segments will be imputed as missing values.

1.3 Imputing missing parental genotypes

To impute the missing parental genotypes without using phase information, type:

```
python impute_runner.py example/chr_1.ibd --bed example/sample --king example/
sample.king.kin0 --agesex example/sample.agesex --output_address example/sample
--threads 4 --snipar_ibd
```

The script constructs a pedigree from the output of KING's relatedness inference (example/sample.king), and age and sex information (example/sample.agesex). The pedigree along with the IBD segments shared between siblings recorded in example/chr_1.ibd.segments.gz are used to impute missing parental genotypes from the observed sibling and parental genotypes in example/sample1.bed. The imputed parental genotypes are in a HDF5 file example/sample.hdf5. The *–snipar_ibd* flag indicates the IBD segments are formatted as output by SNIPar.

If phased haplotypes are available in .bgen format, the imputation can use these as input, which improves the information gained by imputation in certain situations. To perform imputation from the phased .bgen file in example/, use the following command:

```
python impute_runner.py example/chr_1.ibd --bgen example/sample --king example/
sample.king.kin0 --agesex example/sample.agesex --output_address example/sample
--threads 4 --from_chr 1 --to_chr 2 --snipar_ibd
```

It is necessary to provide the *–from_chr* and *–to_chr* arguments when imputing from .bgen files since they often do not contain information on which chromosome the SNPs are located on, and we need to match up the IBD segments to the SNPs on the same chromosome.

To use IBD segments output by KING (example/sample.king.segments.gz), use the following command:

```
python impute_runner.py example/sample.king --bgen example/sample --king
example/sample.king.kin0 --agesex example/sample.agesex --output_address
example/sample --threads 4 --from_chr 1 --to_chr 2
```

As with the ibd.py script, the impute_runner.py script can use a user input pedigree (with the *–pedigree* argument) rather than the *–king* and *–agesex* arguments.

1.4 Family based GWAS

To compute summary statistics for direct, paternal, and maternal effects for all SNPs in the .bed file, type:

```
python fGWAS.py example/sample example/h2_quad_0.8.txt --outprefix example/
h2_quad_0.8 --bed example/sample
```

This takes the observed genotypes in example/sample.bed and the imputed parental genotypes in example/sample.hdf5 and uses them to perform, for each SNP, a joint regression onto the proband's genotype, the father's (imputed) genotype, and the mother's (imputed) genotype. This is done using a random effects model that models phenotypic correlations between siblings, where sibling relations in the pedigree are stored in the output of the imputation script: example/sample.hdf5. The 'family variance estimate' output is the phenotypic variance explained by mean differences between sibships, and the residual variance is the remaining phenotypic variance.

To use the .bgen file instead, type:

```
python fGWAS.py example/sample example/h2_quad_0.8.txt --outprefix example/
h2_quad_0.8 --bgen example/sample
```

The script outputs summary statistics in a gzipped text file: h2_quad_0.8.sumstats.gz. This file gives the chromosome, SNP id, position, alleles (A1, the allele that effects are given with respect to; and A2, the alternative allele), the frequency of the A1 allele, then summary statistics for each type of effect. For each effect, we give the effective N for each SNP; this differs from the actual N due to the fact that there are differing amounts of information for each type of effect, and due to relatedness in the sample. We give the effect estimate in the first column for each effect, the column 'effect_Beta', where 'effect' can be direct, paternal, etc; this is followed by the standard error, the Z-score, and the negative log10 P-value for a non-zero effect. In addition to effects directly estimated by the script, we also output the average parental effect estimate (estimate of the average of maternal and paternal effects), and the population effect estimate, which is equivalent to what is estimated by standard GWAS methods that regress phenotype onto genotype without control for parental genotypes. The final columns give the sampling correlations between the different effect estimates at that SNP.

In addition to the plain text output, the effects and their sampling variance-covariance matrices are output in example/h2_quad_0.8.sumstats.hdf5. The contents of the HDF5 file can be read into Python (using h5py) and R (using rhdf5) easily. The output contains different datasets:

1. *estimate*, the estimated SNP effect, where each row gives a SNP, and each column gives an effect
2. *bim*, equivalent to the bim file for plink, recording the information on each SNP
3. *estimate_cols*, gives the names of the effects estimate for each SNP: direct, paternal, maternal, etc.
4. *estimate_ses*, the standard errors for the effect estimates in *estimate*
5. *estimate_covariance*, 3 dimensional array with sampling variance-covariance matrices for each SNP's estimated effects, with SNPs indexed by the first axis
6. *freqs*, frequencies of the effect alleles
7. *sigma2*, maximum likelihood estimate of the residual variance in the null model
8. *tau*, maximum likelihood estimate of the ratio between the residual variance and family variance
9. *N*, the sample size
10. *NAs*, the number of missing values for each of SNPs, given for each relative in the regression (individual, father, mother, etc.)

Now we have estimated SNP specific summary statistics. To compare to the true effects, run

```
python example/estimate_sim_effects.py example/h2_quad_0.8.sumstats.hdf5
example/h2_quad_0.8.effects.txt
```

This should print estimates of the bias of the effect estimates.

The bias estimates for direct, paternal, maternal, and average parental effects should not be statistically significantly different from zero (with high probability). Population effects (which are estimated by univariate regression of individuals' phenotypes onto their genotypes – as in standard GWAS) here are biased estimates of direct effects, since population effects include both direct and indirect parental effects.

If the imputation has been performed from siblings alone, then the regression onto proband (focal, phenotyped individual), imputed paternal, and imputed maternal becomes collinear. This is because the imputation is the same for paternal and maternal genotypes. In this case, the regression should be performed onto proband and sum of imputed paternal and maternal genotypes. This can be achieved by providing the *–parsum* option to the script. The script can also estimate indirect sibling effects for each SNP by providing the *–fit_sib* option; however, this will reduce power for estimating other effects.

1.5 Polygenic score analyses

In addition to family based GWAS, SNIPar provides a script (fPGS.py) for computing polygenic scores (PGS) based on observed/imputed genotypes, and for performing family based polygenic score analyses. Here, we give some examples of how to use this script. The script computes a PGS from weights provided in LD-pred format . The true direct genetic effects for the simulated trait are given as PGS weights in this format in example/h2_quad_0.8.direct_weights.txt. This is a tab-delimited text file with a header and columns 'chrom' (chromosome), 'pos' (position), 'sid' (SNP ID), 'nt1' (allele 1), 'nt2' (allele 2), 'raw_beta' (raw effect estimates), 'ldpred_beta' (LD-pred adjusted weight). The script uses as weights the 'ldpred_beta' column.

To compute the PGS from the true direct effects, use the following command:

```
python fPGS.py example/direct --bedfiles example/sample --impfiles example/
sample --weights example/h2_quad_0.8.direct_weights.txt
```

This uses the weights in the weights file to compute the polygenic scores for each genotyped individual for whom observed or imputed parental genotypes are available. It outputs the PGS to example/direct.pgs.txt, which is a white-space delimited text file with columns FID (family ID, shared between siblings), IID (individual ID), proband (PGS of individual with given IID), maternal (observed or imputed PGS of that individual's mother), paternal (observed or imputed PGS of that individual's father). The script also supports bed files and imputed files split by chromosome. If you had bed files as chr_1.bed, chr_2.bed, ..., chr_22.bed; and imputed parental genotype files as chr_1.hdf5, chr_2.hdf5, ..., chr_22.hdf5, then you can specify this in a command as:

```
--bedfiles chr_~ --impfiles chr_~
```

The script looks for all files that match the path given with '~' replaced by 1,2,...,22: chr_1.bed & chr_1.hdf5, chr_2.bed & chr_2.hdf5, etc. To use .bgen input, replace the *–bedfiles* argument with *–bgenfiles*.

To estimate direct, paternal, and maternal effects of the PGS, use the following command:

```
python fPGS.py example/direct --pgs example/direct.pgs.txt --phenofile example/
h2_quad_0.8.txt
```

This uses a linear mixed model that has a random effect for mean differences between families (defined as sibships here) and fixed effects for the direct, paternal, and maternal effects of the PGS. It also estimates the 'population' effect of the PGS: the effect from regression of individuals' phenotypes onto their PGS values. The estimated effects and their standard errors are output to example/direct.pgs_effects.txt, with the effect names (direct, paternal, maternal, population) in the first column, their estimates in the second column, and their standard errors in the final column. The sampling variance-covariance matrix of direct, paternal, and maternal effects is output in example/direct.pgs_vcov.txt.

Estimates of the direct effect of the PGS should be equal to 1 in expectation since we are using the true direct effects as the weights, so the PGS corresponds to the true direct effect component of the trait. The parental effect estimates capture the correlation between the direct and indirect parental effects. The population effect estimate should be greater

than 1, since this captures both the direct effect of the PGS, and the correlation between direct and indirect parental effects.

If parental genotypes have been imputed from sibling data alone, then imputed paternal and maternal PGS are perfectly correlated, and the above regression on proband, paternal, and maternal PGS becomes co-linear. To deal with this, add the –parsum option to the above command, which will estimate the average parental effect rather than separate maternal and paternal effects of the PGS.

It is also possible to estimate indirect effects from siblings. We can compute the PGS for genotyped individuals with genotyped siblings and estimate direct, indirect sibling, paternal and maternal effects in one command with the addition of the –fit_sib option:

```
python fPGS.py example/direct_sib --bedfiles example/sample --impfiles example/
sample --weights example/h2_quad_0.8.direct_weights.txt --phenofile example/
h2_quad_0.8.txt --fit_sib
```

This outputs the PGS values for each individual along with the PGS value of their sibling, and imputed/observed paternal and maternal PGS to example/direct_sib.pgs.txt. (If an individual has multiple genotyped siblings, the average of the siblings' PGS is used for the PGS of the sibling.) It outputs estimates of direct, indirect sibling, paternal, and maternal effects of the PGS to example/direct_sib.pgs_effects.txt and their sampling variance-covariance matrix to example/direct_sib.pgs_vcov.txt. Since indirect effects from siblings were zero in this simulation, the estimated sibling effect should be close to zero.

Note that the standard error for the direct effect estimate increases: this is due both to a drop in sample size since only those probands with genotyped siblings are included, and due to the fact that adding the sibling effect to the regression decreases the independent information on the direct effect.

CHAPTER

TWO

GUIDE

Introduction

SNIPar (single nucleotide imputation of parents) is a python library for imputing missing parental genotypes from observed genotypes in a nuclear family, and for performing family based genome-wide association and polygenic score analyses using the resulting imputed parental genotypes.

In the main SNIPar directory, there is a script for imputing missing parental genotypes (impute_runner.py).

The script outputs expected genotypes of missing parents, which are used as input for the fGWAS.py script that perform family based GWAS using observed and imputed parental genotypes.

The impute_runner.py script takes un-phased genotypes in .bed format, and phased haplotypes in .bgen format. The imputation becomes more efficient when using phased haplotypes, at the cost of slower runtime. The script requires IBD segments in the KING (https://people.virginia.edu/~wc9c/KING/manual.html) format as input, where IBD segments shared between first-degree relatives are used. This can be computed by using *–ibdseg –degree 1* options in KING.

The script will construct a pedigree for you if you provide it with the KING relatedness inference (output using the –related –degree 1 options) and age & sex information. Alternatively, a user-input pedigree can be provided. Providing the script with KING output is recommended since this ensures the pedigree is constructed in the correct way.

The pedigree file is a plain text file with header and columns: FID (family ID), IID (individual ID), FATHER_ID (ID of father), MOTHER_ID (ID of mother).

Note that individuals are assumed to have unique individual IDS (IID).

Siblings are identified through individuals that have the same FID.

We recommend working through the tutorial (https://github.com/AlexTISYoung/SNIPar/edit/master/docs/tutorial.rst) to get an idea of the workflow required for a full analysis.

Family based GWAS is performed using the fGWAS.py script, which uses a linear mixed model with a random-effect that models correlations between individuals with the same family ID to estimate direct and indirect effects for genome-wide SNPs.

Polygenic score analyses are performed using the fPGS.py script, which computes polygenic scores for individuals and their first degree relatives based on observed/imputed genotypes. It also estimates direct and indirect effects of the polygenic score in the linear mixed model.

***Package Install Instructions**

SNIPar has the following dependencies:

python 3.7

Packages:

- h5py
- bgen-reader

- numpy
- scipy
- pysnptools
- pandas
- networkx
- Cython

We highly recommend using a python distribution such as Anaconda 3 (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 SNIPar source code, at the shell type

'python setupy.py install'

'python setup.py build_ext –inplace'

Running tests

To check that the code is working properly and that the C modules have compiled, you should run tests. To run the tests, in the main SNIPar directory enter the command:

```
python setup.py pytest
```

CHAPTER

THREE

SIBREG.BIN PACKAGE

3.1 Submodules

3.2 sibreg.bin.impute_from_sibs module

3.3 sibreg.bin.impute_from_sibs_hdf5 module

3.4 sibreg.bin.impute_from_sibs_setup module

3.5 sibreg.bin.impute_po module

3.6 sibreg.bin.impute_runner module

3.7 sibreg.bin.make_rdr_grms module

3.8 sibreg.bin.pGWAS module

3.9 sibreg.bin.poGWAS module

3.10 sibreg.bin.preprocess_data module

3.11 sibreg.bin.sGWAS module

3.12 sibreg.bin.triGWAS module

3.13 Module contents

Contains the main code for the imputations.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

M

S