snipar

Alexander Young

Jul 18, 2023

CONTENTS:

1 Guide **1**
- 1.1 Introduction . . . 1
- 1.2 Installation . . . 1
- 1.3 Workflow . . . 2

2 Tutorial **7**
- 2.1 Test data . . . 7
- 2.2 Inferring IBD between siblings . . . 7
- 2.3 Imputing missing parental genotypes . . . 8
- 2.4 Family based GWAS . . . 9
- 2.5 Correlations between effects . . . 9
- 2.6 Polygenic score analyses . . . 10

3 Input files **11**
- 3.1 IDs . . . 11
- 3.2 Observed genotypes . . . 11
- 3.3 Pedigree . . . 11
- 3.4 kinship file . . . 12
- 3.5 agesex file . . . 12
- 3.6 phenotype file . . . 12
- 3.7 covariate file . . . 12
- 3.8 weights file . . . 12

4 Output files **15**
- 4.1 IBD segments file . . . 15
- 4.2 imputed parental genotypes file . . . 16
- 4.3 text summary statistics . . . 17
- 4.4 HDF5 summary statistics . . . 19
- 4.5 PGS file . . . 19
- 4.6 PGS effects . . . 20
- 4.7 PGS effects sampling covariance . . . 20

5 Command Line Scripts **21**
- 5.1 ibd.py . . . 21
- 5.2 impute.py . . . 22
- 5.3 gwas.py . . . 25
- 5.4 pgs.py . . . 26
- 5.5 correlate.py . . . 28
- 5.6 simulate.py . . . 29

6 Simulation Exercise **31**

6.1 Simulating data . . . 31
6.2 Inferring IBD between siblings . . . 32
6.3 Imputing missing parental genotypes . . . 32
6.4 Polygenic score analyses . . . 32
6.5 Adjusting for assortative mating . . . 33

7 Indices and tables **35**

Index **37**

CHAPTER

ONE

GUIDE

1.1 Introduction

snipar (single nucleotide imputation of parents) is a Python package for inferring identity-by-descent (IBD) segments shared between siblings, imputing missing parental genotypes, and for performing family based genome-wide association and polygenic score analyses using observed and/or imputed parental genotypes.

snipar can use any genotyped samples who have at least one genotyped full-sibling or parent.

The imputation method and the family-based GWAS and polygenic score models are described in Young et al. 2022.

1.2 Installation

snipar currently supports Python 3.7-3.9 on Linux, Windows, and Mac OSX. We recommend using Anaconda 3 (https://store.continuum.io/cshop/anaconda/).

1.2.1 Installing Using pip

The easiest way to install is using pip:

```
pip install snipar
```

Sometimes this may not work because the pip in the system is outdated. You can upgrade your pip using:

```
pip install –upgrade pip
```

You may encounter problems with the installation due to Python version incompatability or package conflicts with your existing Python environment. To overcome this, you can try installing in a virtual environment. In a bash shell, this could be done by using the following commands in your directory of choice:

```
python -m venv path-to-where-you-want-the-virtual-environment-to-be
```

You can activate and use the environment using

```
source path-to-where-you-want-the-virtual-environment-to-be/bin/activate
```

snipar can also be installed within a conda environment using pip.

1.2.2 Installing From Source

To install from source, clone the git repository (https://github.com/AlexTISYoung/snipar), and in the directory containing the *snipar* source code, at the shell type:

pip install .

1.2.3 Python version incompatibility

snipar does not currently support Python 3.10 or higher due to version incompatibilities of dependencies. To overcome this, create a Python3.9 environment using conda and install using pip in the conda environment:

conda create -n myenv python=3.9

conda activate myenv

pip install snipar

1.2.4 Running tests

To check that the code is working properly and that the C modules have been compiled, you can run the tests using this command:

python -m unittest snipar.tests

1.3 Workflow

A typical *snipar* workflow for performing family-based GWAS (see flowchart below) is:

1. Inferring identity-by-descent (IBD) segments shared between siblings (*ibd.py*)
2. Imputing missing parental genotypes (*impute.py*)
3. Estimating direct effects and non-transmitted coefficients (NTCs) of genome-wide SNPs (*gwas.py*)

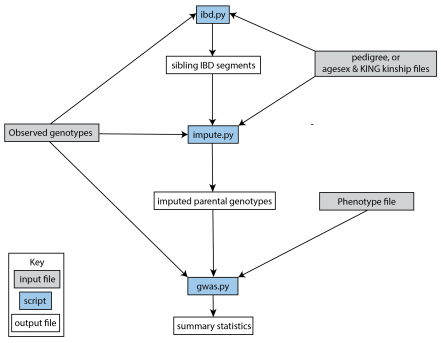


Fig. 1: Illustration of a typical workflow for performing family-based GWAS

A *snipar* workflow requires input files in certain formats. See *input files*. Output files are documented *here*.

The *tutorial* allows you to work through an example workflow before trying real data.

1.3.1 Inputting multiple chromosomes

We recommend splitting up *observed genotype files* by chromosome since certain scripts in *snipar* cannot handle observed genotype files with SNPs from multiple chromosomes.

To run scripts for all chromosomes simultaneously (recommended), the @ character can be used as a numerical wildcard. For example, if you had observed genotype files chr_1.bed, chr_2.bed, ..., chr_22.bed, then you could specify these as inputs to the command line scripts as "–bed chr_@". If you only want to analyse a subset of the chromosomes, you can use the "–chr_range" argument; for example, '–bed chr_@ –chr_range 1-9' would specify analysing observed genotype files chr_1.bed, chr_2.bed, ..., chr_9.bed.

This will result in *output files* that are also split by chromosome. The names of the output files can also be specified using the numerical wildcard character, @, e.g. '–out /path/to/output/dir/chr_@'.

1.3.2 Inferring identity-by-descent segments

If your sample contains full-sibling pairs (without both parents genotyped), it is necessary to first infer the identity-by-descent (IBD) segments shared between the siblings before imputing the missing parental genotypes. If your sample does not contain any full-sibling pairs, but has genotyped parent-offspring pairs (i.e. one parent's genotype is missing), imputation can proceed without inferring IBD.

snipar contains a Hidden Markov Model (HMM) algorithm for inferring IBD shared between siblings, which can be accessed through the command line script *ibd.py*.

The ibd.py script requires the *observed genotypes* of the siblings and information on the sibling and parent-offspring relations in the genotyped sample.

To infer IBD, one can use a smaller set of genetic variants than one intends to use in downstream analyses (imputation, gwas, etc.). For example, one could use the variants on a genotyping array to infer IBD segments, and these IBD segments could be used to impute missing parental genotypes for a larger set of variants imputed from a reference panel. This can be useful since the accuracy of IBD inference plateaus as the density of variants increases, so inputting millions of variants imputed from a reference panel to *ibd.py* will result in a long computation time for little gain in accuracy over using variants from a genotyping array.

The information on the relations present in the genotyped sample can be provided through a *pedigree file* or through the *output of KING relationship inference* (as output using the –related –degree 1 options: see https://www.kingrelatedness.com/manual.shtml#RELATED) along with a *file giving the age and sex information* on the genotyped sample. (The age and sex information along with the parent-offspring and sibling relations inferred by KING are used to construct a pedigree if a pedigree is not provided.)

The algorithm requires a genetic map to compute the probabilities of transitioning between different IBD states. If the genetic map positions (in cM) are provided in the .bim file (if using .bed formatted genotypes), the script will use these. Alternatively, 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). If no genetic map is provided, then the deCODE sex-averaged map on GRCh38 coordinates (Halldorsson, Bjarni V., et al. "Characterizing mutagenic effects of recombination through a sequence-level genetic map." Science 363.6425 (2019).), which is distributed as part of *snipar*, will be used.

The HMM employs a genotyping error model that requires a genotyping error probability parameter. By default, the algorithm will estimate the per-SNP genotyping error probability from Mendelian errors observed in parent-offspring pairs. However, if your data does not contain any genotyped parent-offspring pairs, then you will need to supply a genotyping error probability. If you have no external information on the genotyping error rate in your data, using a value of 1e-4 has worked well when applied to typical genotyping array data.

The HMM will output the IBD segments to a *gzipped text file with suffix ibd.segments.gz*. As part of the algorithm, LD scores are calculated for each SNP. These can also be output in LDSC format using the –ld_out option.

1.3.3 Imputing missing parental genotypes

impute.py is responsible for imputing the missing parental genotypes. This is done for individuals with at least one sibling and/or parent genotyped but without both parents genotyped.

You should provide the script with identity-by-descent (IBD) segments shared between the siblings if there are genotyped sibling pairs in the sample. Although we strongly recommend using IBD segments inferred by *ibd.py*, we also support IBD segments in the format that KING outputs (see https://www.kingrelatedness.com/manual.shtml#IBDSEG). If IBD segments in KING format are used, it is necessary to add the –ibd_is_king flag.

The script needs information about family structure of the sample. You can either supply it with a *pedigree file* or let it build the pedigree from *kinship* and *agesex* files.

If you are imputing for a chromosome with a large number of SNPs, you may encounter memory issues. If this is the case, you can use the –chunks argument to perform the imputation in chunks. When the script is run with '-chunks x', it will split the imputation into 'x' batches. Alternatively, you can do the imputation for only on a subset of SNPS by using -start and -end options.

For each chromosome, imputed parental genotypes and other information about the imputation will be written to a file in HDF5 format. The contents of the HDF5 output, which a typical user does not need to interact with directly, are documented *here*.

The expected proportion of variants that have been imputed from a sibling pair in IBD0 (i.e. the parental alleles are fully observed) can be computed from the pedigree. At the end of the imputation, the script will output the expected IBD0 proportion and the observed IBD0 proportion. If there have been issues with the imputation (such as failure to match IBD segments to observed genotypes), this will often should up as a large discrepancy between expected and observed IBD0 proportions.

1.3.4 Family-based genome-wide association analysis

Family-based GWAS is performed by the *gwas.py* script. This script estimates direct effects, non-transmitted coefficients, and population effects of input genetic variants on the phenotype specified in the phenotype file. (If multiple phenotypes are present in the phenotype file, the phenotype to analyse can be specified using the '–phen_index' argument, where '–phen_index 1' corresponds to the first phenotype.)

The script will use both *observed* and *imputed parental genotypes* to estimate these effects. Note that if no imputed parental genotypes are input, effects will be estimated using individuals with both parents genotyped only, provided that a *pedigree file* is also input. (A pedigree input is not needed when inputting *imputed parental genotypes*.)

By default, for each variant, the script performs a regression of an individual's phenotype onto their genotype, their (imputed/observed) father's genotype, and their (imputed/observed) mother's genotype. This estimates the direct effect of the variant, and the paternal and maternal non-transmitted coefficients (NTCs). See Young et al. 2022 for more details.

If no parental genotypes are observed, then the imputed maternal & paternal genotypes become perfectly correlated. In this case, to overcome collinearity, gwas.py will perform a regression of an individual's phenotype onto their genotype, and the imputed sum of their parents' genotypes. This will estimate the direct effect of the SNP, and the average NTC.

If one wishes to model indirect genetic effects from siblings, one can use the '–fit_sib' option to add the genotype(s) of the individual's sibling(s) to the regression.

The gwas.py script first estimates a variance component model that models the phenotypic correlation between siblings, then does a transformation that allows the SNP effects to be estimated by simple linear regression while accounting for correlations between siblings.

The script outputs summary statistics in both gzipped *text format* and *HDF5 format*.

1.3.5 Estimating correlations between effects

As part of Young et al. 2022, we estimated the genome-wide correlations between direct and population effects and between direct effects and average non-transmitted coefficients (NTCs). The correlation between direct effects and population effects is a measure of how different direct effects and effects estimated by standard GWAS (population effects) are.

We provide a script, *correlate.py*, that estimates these correlations. It takes as input the *summary statistics* files output by *gwas.py* and LD-scores for the SNPs (as output by *ibd.py* or by LDSC). It applies a method-of-moments based estimator that accouts for the known sampling variance-covariance of the effect estimates, and for the correlations between effect estimates of nearby SNPs due to LD.

Note that this is different to genetic correlation as estimated by LDSC. LDSC attempts to use LD-scores to estimate heritability and to separate out this from bias due to population stratification. The *correlate.py* estimator only uses LD-scores to account for correlations between nearby SNPs, not to separate out population stratification. This is because we are (potentially) interested in the contribution of population stratification to population effects, and whether population stratification makes population effects different from direct effects. The approach used by LDSC would remove some of the contribution of population stratification to differences between direct and population effects.

1.3.6 Family-based polygenic score analyses

As in previous work (e.g. Kong et al. 2018: https://www.science.org/doi/abs/10.1126/science.aan6877), parental polygenic scores can be used as 'controls' to separate out the component of the association between phenotype and polygenic score (PGS) that is due to direct genetic effects. In Young et al. 2022, we showed how this can be done using parental PGSs computed from imputed parental genotypes. *snipar* provides a script, *pgs.py*, that can be used for computing and analysing PGSs using observed/imputed parental genotypes.

The *pgs.py* script takes similar inputs to the *gwas.py* script. The main addition is that in order to compute a PGS, a *weights file* must be provided.

By default, if no phenotype file is provided, the *pgs.py* script will compute the PGS values of all the genotyped individuals for whom *observed* or *imputed parental genotypes* are available. The script will output a *PGS file*, including the imputed/observed PGS values for each individual's parents, facilitating family-based polygenic score analyses.

If the '–fit_sib' argument is provided, the *PGS file* will include a column corresponding to the average PGS value of the individual's sibling(s).

To estimate the direct and population effects as well as the non-transmitted coefficients (NTCs) of the PGS on a phenotype, input a phenotype file to *pgs.py*. One can first compute the PGS and write it to *file*, and then use this as input to *pgs.py* along with a phenotype file.

The direct effect and NTCs of the PGS are estimated as fixed effects in a linear mixed model that includes a random effect that models (residual) phenotypic correlations between siblings. The population effect is estimated from a separate linear mixed regression model that includes only the proband PGS as a fixed effect. The estimates and their standard errors are output to *file* along with a separate *file* giving the sampling variance-covariance matrix of the direct effect and NTCs.

CHAPTER

TWO

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 read the *guide*.

2.1 Test data

If *snipar* has been installed succesfully, the *command line scripts* should be accessible as executables in your terminal. A script that should be accessible loads the tutorial example data into a specified directory. To create a directory called 'example_data/' in the current directory and load the example data into it, use the command:

```
snipar_example_data.py --dest example_data
```

You can create the example data directory elsewhere by changing the –dest argument. Please change your working directory to example_data/:

```
cd example_data
```

In this directory, there is some example data. The file phenotype.txt is a phenotype file containing 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 *observed genotype data* formatted in both PLINK .bed format (chr_1.bed) and phased genotype data in .bgen format (chr_1.bgen with associated sample file chr_1.sample).

2.2 Inferring IBD between siblings

The first step is to infer the identity-by-descent (IBD) 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. The per-SNP genotyping error probability will be inferred from parent-offspring pairs when available; alternatively, a genotyping error probability can be provided using the –p_error option. By default, SNPs with genotyping error rates greater than 0.01 will be filtered out, but this threshold can be changed with the –max_error argument. To infer the IBD segments from the genotype data in chr_1.bed,use the following command

```
ibd.py --bed chr_@ --king king.kin0 --agesex agesex.txt --out chr_@ --threads 4
--ld_out
```

This will output the IBD segments to a *gzipped text file* chr_1.ibd.segments.gz. Genotype files split over multiple chromosomes can be specified using '@' as a numerical wildcard character: see *here*. In this example, –bed chr_@

instructs ibd.py to search for .bed files chr_1.bed, chr_2.bed, . . . , chr_22.bed, where each bed file contains SNPs from the numbered chromosome. In this case, only one bed file is in example_data/, chr_1.bed. If bed files for multiple chromosomes are found, IBD will be inferred separately for each chromosome, with one output file per chromosome, with the chromosome number filling in the numerical wildcard in the –out argument.

The *–king* argument requires the address of the *KING kinship file*, and the *–agesex* argument requires the address of the *agesex file*.

The algorithm requires a genetic map to compute the probabilities of transitioning between different IBD states. If the genetic map positions (in cM) are provided in .bim file, the script will use these. Alternatively, 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) an example of which is provided in genetic_map.txt.

If no genetic map is provided, then the deCODE sex-averaged map on GRCh38 coordinates (Halldorsson, Bjarni V., et al. "Characterizing mutagenic effects of recombination through a sequence-level genetic map." Science 363.6425 (2019).), which is distributed as part of *snipar*, will be used.

The algorithm computes LD scores of SNPs in order to account for correlations between SNPs. The '–ld_out' argument writes the LD scores to file in the same format as LDSC (https://github.com/bulik/ldsc).

The user can also input a phased .bgen file. For example, to infer IBD from chr_1.bgen using the genetic map in genetic_map.txt, use this command:

```
ibd.py --bgen chr_@ --king king.kin0 --agesex agesex.txt --out chr_@ --threads
4 --ld_out --map genetic_map.txt
```

If the user has a *pedigree file*, they can input that instead of the *–king* and *–agesex* arguments. Siblings are inferred as individuals in the pedigree that share both parents. Using the example pedigree in pedigree.txt, you can infer IBD using this command:

```
ibd.py --bed chr_@ --pedigree pedigree.txt --map genetic_map.txt --out chr_@
--threads 4 --ld_out
```

2.3 Imputing missing parental genotypes

This is performed using the *impute.py* script. To impute the missing parental genotypes without using phase information, use this command:

```
impute.py --ibd chr_@.ibd --bed chr_@ --king king.kin0 --agesex agesex.txt
--out chr_@ --threads 4
```

The script constructs a pedigree from the output of KING's relatedness inference (king.kin0), and age and sex information (agesex.txt). The pedigree along with the IBD segments shared between siblings recorded in chr_1.ibd.segments.gz are used to impute missing parental genotypes from the observed sibling and parental genotypes in chr_1.bed. The imputed parental genotypes are output to a *HDF5 file*, chr_1.hdf5.

If phased haplotypes are available in .bgen format, the imputation can use these as input, which improves the accuracy of the imputation. To perform imputation from the phased .bgen file in example_data/, use the following command:

```
impute.py --ibd chr_@.ibd --bgen chr_@ --king king.kin0 --agesex agesex.txt
--out chr_@ --threads 4
```

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

2.4 Family based GWAS

This is performed using the *gwas.py* script. To compute summary statistics for direct effects, non-transmitted coefficients (NTCs), and population effects for the SNPs in the .bed file, use this command:

```
gwas.py phenotype.txt --bed chr_@ --imp chr_@ --threads 4
```

This takes the observed genotypes in chr_1.bed and the imputed parental genotypes in chr_1.hdf5 and uses them to perform, for each SNP, a joint regression onto the proband's genotype, the father's (imputed/observed) genotype, and the mother's (imputed/observed) genotype. This is done using a linear mixed model that models phenotypic correlations between siblings, where sibling relations are stored in the *output of the imputation script*. 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, use this command:

```
gwas.py phenotype.txt --bgen chr_@ --imp chr_@ --threads 4
```

The script outputs summary statistics in a *gzipped text file*: chr_1.sumstats.gz. In addition to the text summary statistics, *HDF5 format summary statistics* are also output to chr_1.sumstats.hdf5

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

```
python estimate_sim_effects.py chr_1.sumstats.hdf5 phenotype.effects.txt
```

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

The bias estimates for direct, paternal NTCs, maternal NTCs, and average NTCs should not be statistically significantly different from zero (with high probability). Population effects (as estimated by standard GWAS) are biased estimates of direct effects for this simulated phenotype because they also include indirect genetic effects.

GWAS can also be performed without imputed parental genotypes. In this case, only probands with genotypes for both parents available will be used. In order to do this, one must provide a pedigree to gwas.py, as in:

```
gwas.py phenotype.txt --out trios_ --bgen chr_@ --pedigree pedigree.txt
--threads 4
```

2.5 Correlations between effects

snipar provides a script (*correlate.py*) to compute correlations between direct and population effects and between direct effects and average NTCs. To compute these correlations from the effects estimated in this tutorial (output by gwas.py to chr_1.sumstats.gz) using the LD scores computed by ibd.py (and output to chr_1.l2.ldscore.gz), use the following command:

```
correlate.py chr_@ effect --ldscores chr_@
```

This should give a correlation between direct effects and average NTCs of close to 0.5. The estimated correlations and their standard errors, estimated by block-jacknife, are output to effect_corrs.txt.

The method is similar to LDSC, but correlates the marginal effects (not joint-fit effects adjusted for population stratification, as LDSC attempts to use), adjusting for the known sampling variance-covariance matrix of the effects. The LD scores are used for weighting. LD scores output by LDSC can be input. If LD scores are not available, they can be computed from .bed files by providing them through the –bed argument to *correlate.py*.

2.6 Polygenic score analyses

For an exercise involving polygenic score analysis, please see the *Simulation Exercse*.

CHAPTER

THREE

INPUT FILES

We describe the input files here. Examples are available in the *tutorial* data.

3.1 IDs

We have followed a typical convention that input files giving individual level information (phenotype, pedigree, etc.) tend to have both family ID (FID) and individual ID (IID) columns. Note, however, that *snipar* ignores the entries in the FID column, using only individual IDs. Therefore, entries in the IID column need to be unique.

3.2 Observed genotypes

Observed genotypes can be provided either in PLINK .bed format or in phased .bgen format. (Unphased .bgen format is not currently supported). Phased .bgen files are the recommended input since the imputation is more accurate when phase information can be used (see Young et al. 2022 [ref]). If one has phased genotypes in another format (for example VCF), then these can be converted to phased .bgen format using QCTOOL (https://www.well.ox.ac.uk/~gav/qctool_v2/documentation/examples/converting.html).

The *snipar* *workflow* has only been tested with high-quality genotype information on bi-allelic variants. We recommend first filtering your observed genotypes to keep only bi-allelic variants with INFO/R-square>0.99 (if using genotypes imputed from a reference panel) and Hardy-Weinberg Equilibrium P-value>1e-6. Using low-quality SNPs may result in bias in the imputed parental genotypes and direct effect estimates and is not recommended.

Observed genotypes should be split into separate files for each chromosome in the autosome (*snipar* does not support sex-chromosomes). For example, if you have phased genotypes in .bgen format in a directory /path/to/haplotypes/ so that the genotypes for each chromosome are in /path/to/haplotypes/chr_1.bgen, /path/to/haplotypes/chr_2.bgen, ..., /path/to/haplotypes/chr_22.bgen, then this can be specified to *snipar* scripts with '–bgen /path/to/haplotypes/chr_@', where @ is interpreted as a numerical wildcard character.

3.3 Pedigree

This is a white-space delimited text file with header "FID", "IID", "FATHER_ID", "MOTHER_ID", corresponding to columns for family-ID (FID), individual ID (IID), father's ID, and mother's ID. A '0' in 'father's ID' or 'mother's ID' implies the parent is unknown. If the missing value is indicated by something other than '0', be sure to specify it with –pedigree_nan option.

WARNING: monozygotic (identical) twins, when coded in the pedigree the same way as full-siblings, will cause errors in IBD inference and imputation. We recommend filtering out one individual from each identical twin pair to prevent downstream issues.

Instead of providing a pedigree file, one can provide the results of KING relationship inference and age and sex information (see below).

3.4 kinship file

The kinship file is as output by KING: https://www.kingrelatedness.com/manual.shtml#RELATED. WARNING: KING relationship inference behaves differently if the .fam file of the input .bed file has meaningful family IDs (FIDs) and parental IDs; i.e., if your .fam file already contains pedigree/family information, KING will produce a .kin file. We have found the output of KING in this case (the .kin file) to be unpredictable in ways that causes issues with *snipar* analyses. We therefore recommend setting the family IDs (FIDs) to the individual IDs (IIDs) in .fam file input to KING, and removing information on parents (if present), before running KING with the –related command. This will output a .kin0 file containing the sibling and parent-offspring relations needed by *snipar*.

3.5 agesex file

This is a white-space delimited text file with header "FID", "IID", "sex", "age". Each row contains the family-ID, individual-ID, age, and sex of one individual. Male and Female sex should be represented with 'M' and 'F' respectively. The age column is used for distinguishing between parent and child in a parent-offspring relationship inferred from the *kinship file*. ID1 is a parent of ID2 if there is a parent-offspring (PO) relationship between them and 'ID1' is at least 12 years older than ID2.

3.6 phenotype file

The phenotype file is a white-space delimited text file with a header. It has columns (in order) for family-ID, individual-ID, and phenotype values for the different phenotypes. To specify the k'th phenotype for analysis (relevant for *gwas.py* and *pgs.py*), add '–phen_index k' to your command; by default, the first phenotype will be used.

3.7 covariate file

The covariate file has the same format as the phenotype file (above). It is a white-space delimited text file with a header. It has columns (in order) for family-ID, individual-ID, and covariate values for the different covariates. (Note, covariates must be numerical).

3.8 weights file

This file is used to input the SNP weights to the *pgs.py* script for computation of the PGS. The weights file is a plain-text file with columns giving (minimally) the SNP ID, the SNP weight, the effect allele, and the alternative allele. The script is setup to process weights files as output by LD-pred by default. If your weights file has different column names, these can be specified through the command line arguments of the *pgs.py* script:

'–SNP'
: the column name for the column containing the SNP IDs

'–beta_col'
: the column name for the column with the SNP weights

‘–A1’
the column name for the column with the effect allele

‘–A2’
the column name for the column with the alternative allele

CHAPTER

FOUR

OUTPUT FILES

We describe the output files here. Examples are produced by working through the *tutorial*.

4.1 IBD segments file

The *ibd.py* script outputs one gzipped text file per chromosome containing the IBD segments for each sibling pair in the input. The IBD segments file is a tab delimited text file where each row contains information on a particular IBD segment. It has columns:

'ID1'
: Individual ID (IID) of the first sibling in pair

'ID2'
: Inidividual ID (IID) of the second sibling in the pair

'IBDType'
: The IBD type (0, 1, or 2) of the segment

'Chr'
: The chromosome of the segment

'start_coordinate'
: The base-pair (bp) position of the start of the segment (inclusive)

'stop_coordinate'
: The base-pair (bp) position of the end of the segment (inclusive)

'start_SNP'
: The ID of the variant at the start of the segment

'stop_SNP'
: The ID of the variant at the end of the segment

'length'
: The length of the segment in centi Morgans (cM)

4.2 imputed parental genotypes file

For each chromosome, the *impute.py* script outputs a HDF5 file containing the imputed parental genotypes. (Users typically do not need to look in this file, but if they want to, HDF5 files can be read in *R* using *rhdf5* and in Python using *h5py*.) Consider imputing missing parental genotypes for n families on a chromosome with L genotyped variants. The resulting HDF5 file will contain the following datasets:

'imputed_par_gts'
: [n x L] floating point array of imputed parental genotypes. It's the imputed missing parent if only one parent is missing and the imputed average of the both parents if both are missing.

'pos'
: [L] vector of base pair (bp) position of variants (in the order of appearance in genotypes)

'families'
: [n] vector of family (sibship) ids of the imputed parents (in the order of appearance in genotypes); these are used internally by *snipar* and are distinct from FIDs in input files

'parental_status'
: [n*3] Array where each row shows the family status [ref] of the family of the corresponding row in families. Columns are has_father, has_mother and, single_parent.

'sib_ratio_backup'
: [L] vector giving the ratio of backup imputation (not using IBD information) among families with 2 or more genotyped siblings for each variant.

'parent_ratio_backup'
: [L] vector giving the ratio of backup imputation among parent-offspring imputations for each variant.

'mendelian_error_ratio'
: [L] vector giving ratio of mendelian errors among parent-offspring pairs for each variant

'estimated_genotyping_error'
: [L] estimated genotyping error rate for each variant

'ratio_ibd0'
: [L] vector giving the fraction of sibships with an observed IBD0 pair

'bim_columns'
: [k] vector giving the column names of the 'bim' file (table containing information similar to a PLINK .bim file)

'bim_values'
: [L x k] matrix of variant-level information (see 'bim_columns')

'pedigree'
: pedigree with columns family ID, individual ID, father ID, mother ID, has_father, has_mother. the family ID here corresponds to the family ID in the 'families' dataset, which indexes the rows of 'imputed_par_gts'. 'has_father' and 'has_mother' denotes whether a genotyped father or genotyped mother (respectively) was used in the imputation

'non_duplicates'
: [L] vector of indexes of the unique snps; imputation is restricted to these SNPs

'standard_f'
: Whether the allele frequencies are just population average instead of MAFs estimated using PCs

'MAF_*'
: info about the MAF estimator if MAF estimator is used.

4.3 text summary statistics

The *gwas.py* script outputs one gzipped text file per chromosome containing the summary statistics for variants in the input. Variants that have been filtered out (by having an MAF below the threshold, too much missingness, or missing IBD/genetic map information) will appear in the output file but the summary statistics will be 'nan'. The sumstats file is a white-space delimited text file. Exactly which columns are present depends on the model used. If using a model with proband and maternal and paternal genotypes, the sumstats file will have the following columns:

'chromosome'
The chromosome of the variant

'SNP'
The ID of the variant

'pos'
The base-pair (bp) position of the variant

'A1'
The effect allele

'A2'
The alternative allele

'freq'
The frequency of the 'A1' effect allele

'direct_N'
The effective sample size for estimation of the direct effect

'direct_Beta'
The estimated direct effect

'direct_SE'
The standard error of the direct effect estimate

'direct_Z'
The Z-score of the direct effect estimate

'direct_log10_P'
The negative log10 P-value for a non-zero direct effect

'paternal_N'
The effective sample size for estimation of the paternal non-transmitted coefficient (NTC)

'paternal_Beta'
The estimated paternal NTC

'paternal_SE'
The standard error of the paternal NTC estimate

'paternal_Z'
The Z-score of the paternal NTC estimate

'paternal_log10_P'
The negative log10 P-value for a non-zero paternal NTC

'maternal_N'
The effective sample size for estimation of the maternal non-transmitted coefficient (NTC)

'maternal_Beta'
The estimated maternal NTC

'maternal_SE'
The standard error of the maternal NTC estimate

'maternal_Z'
The Z-score of the maternal NTC estimate

'maternal_log10_P'
The negative log10 P-value for a non-zero maternal NTC

'avg_NTC_N'
The effective sample size for estimation of the average non-transmitted coefficient (NTC): average of maternal and paternal NTCs

'avg_NTC_Beta'
The estimated average NTC

'avg_NTC_SE'
The standard error of the average NTC estimate

'avg_NTC_Z'
The Z-score of the average NTC estimate

'avg_NTC_log10_P'
The negative log10 P-value for a non-zero average NTC

'population_N'
The effective sample size for estimation of the population effect: sum of direct effect and average NTC

'population_Beta'
The estimated population effect

'population_SE'
The standard error of the population effect estimate

'population_Z'
The Z-score of the population effect estimate

'population_log10_P'
The negative log10 P-value for a non-zero population effect

'r_direct_avg_NTC'
The sampling correlation between the direct effect and average NTC estimates

'r_direct_population'
The sampling correlation between the direct effect and population effect estimates

'r_paternal_maternal'
The sampling correlation between paternal and maternal NTC estimates

Note that, if using parental genotypes imputed from siblings (without any observed parents), then separate maternal and paternal NTCs cannot be estimated, so only the average NTC will appear in the summary statistics output. Also, if '–fit_sib' is used to include an indirect effect from siblings, this will be included in the output.

4.4 HDF5 summary statistics

The *gwas.py* script outputs one HDF5 file per chromosome containing the summary statistics for variants in the input. This is to allow for easier access, compared to the text file, of the parameter vector estimate along with its sampling variance-covariance matrix for each SNP. Variants that have been filtered out (by having an MAF below the threshold, too much missingness, or missing IBD/genetic map information) will appear but the summary statistics will be 'nan'. For a chromosome with L variants, the HDF5 file contains the following datasets:

'bim'
: [L x 5] matrix of variant level information with columns: chromosome, SNP ID, base-pair (bp) position, Allele 1, Allele 2

'estimate'
: [L x k] matrix of effect estimates for each SNP. The effects estimated depend on the model.

'estimate_cols'
: [k] vector giving the names of the effects estimated for each SNP (corresponding to columns of 'estimate' dataset)

'estimate_covariance'
: [L x k x k] array giving the [k x k] sampling variance-covariance matrix for the effect estimates for each SNP

'estimate_ses'
: [L x k] matrix giving the standard errors for each effect estimate for each SNP

'freqs'
: [L] vector giving the estimated allele frequencies (of allele 1) for each SNP

'sigma2'
: scalar giving the MLE of the residual variance from the linear-mixed model

'tau'
: scalar giving the ratio between the residual variance and variance explained by differences in means between sibships

The variance components in the HDF5 file can be used to reconstruct the phenotypic variance by sigma2*(1+1/tau), and the phenotypic correlation between siblings by 1/(1+tau).

4.5 PGS file

The *pgs.py* script can be used to compute polygenic scores for genotyped individuals along with the parental polygenic scores based on observed and/or imputed parental genotypes. The *pgs.py* script will output a PGS file, which is a white-space delimited text file with columns: FID (family ID), IID (individual ID), FATHER_ID, MOTHER_ID, proband PGS, paternal PGS, maternal PGS. (If the PGS has been computed from parental genotypes imputed from siblings alone, it will output the imputed parental PGS, the sum of paternal and maternal PGS values, instead of the paternal and maternal PGS values separately.)

Here, the family ID is the same as used internally by *snipar*, so that individuals who share a family ID are full-siblings. The FATHER_ID and MOTHER_ID columns are set to NA when the parents are not genotyped. The proband PGS column gives the PGS value for the individual given by the IID in that row. The paternal and maternal PGS columns give the PGS values of the individual's father and mother respectively, and these values can be computed from either imputed or observed parental genotypes.

If the PGS was computed with the '–fit_sib' option, the output will include a column for the average of the proband's siblings' PGS values.

4.6 PGS effects

The *pgs.py* script can be used to compute the direct and population effects of the PGS along with the non-transmitted coefficients (NTCs). By default, the script will fit both a one-generation model (regression of phenotype onto proband PGS, which estimates the population effect of the PGS) and a two-generation model (regression of phenotype onto proband, paternal, and maternal PGS; or proband and combined parental PGS).

When this is done, the script will write a file with suffix 1.effects.txt for the results of one-generation analysis, and 2.effects.txt for the results of two-generation analysis. The output includes the estimated intercept and covariate coefficients for the regression as well as the PGS effect estimates.

This file has three columns: the first gives the name of the regression coefficient (i.e. proband, for proband PGS; parental, for parental PGS; etc.), the second gives the corresponding regression coefficient, and the third gives the standard error. For example, in the two-generation analysis, if your PGS file has columns proband, paternal, maternal, then the effects file contains the estimate of the direct effect (regression coefficient on proband PGS when controlling for paternal and maternal PGS), the maternal NTC, and the paternal NTC.

4.7 PGS effects sampling covariance

The *pgs.py* script will write a file with suffix vcov.txt in addition to the file with suffix effects.txt (described above). This file contains the sampling variance-covariance matrix of the regression coefficients.

CHAPTER

FIVE

COMMAND LINE SCRIPTS

5.1 ibd.py

Infers identity-by-descent (IBD) segments shared between full-siblings.

Minimally: the script requires observed sibling genotypes in either .bed or .bgen format, along with information on the relations present in the dataset, which can be provided using a pedigree file or the results of KING kinship inference along with age and sex information (from which a pedigree can be constructed).

Args:

'-h', '–help'
[, default===SUPPRESS==] show this help message and exit

'–bgen'
[str] Address of the phased genotypes in .bgen format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–bed'
[str] Address of the unphased genotypes in .bed format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–chr_range'
[] number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

'–king'
[str] Address of the king file

'–agesex'
[str] Address of file with age and sex information

'–pedigree'
[str] Address of pedigree file

'–map'
[str] None

'–out'
[str, default=ibd] The IBD segments will output to this path, one file for each chromosome. If the path contains '#', the '#' will be replaced with the chromosome number. Otherwise, the segments will be output to the given path with file names chr_1.ibd.segments.gz, chr_2.segments.gz, etc.

'–p_error'
[float] Probability of genotyping error. By default, this is estimated from genotyped parent-offspring pairs.

'–min_length'
[float, default=0.01] Smooth segments with length less than min_length (cM)

'–threads'
[int] Number of threads to use for IBD inference. Uses all available by default.

'–min_maf'
[float, default=0.01] Minimum minor allele frequency

'–max_missing'
[float, default=5] Ignore SNPs with greater percent missing calls than max_missing (default 5)

'–max_error'
[float, default=0.01] Maximum per-SNP genotyping error probability

'–ibdmatrix'
[] Output a matrix of SNP IBD states (in addition to segments file)

'–ld_out'
[] Output LD scores of SNPs (used internally for weighting).

'–chrom'
[int] The chromosome of the input .bgen file. Helpful if inputting a single .bgen file without chromosome information.

'–batches'
[int, default=1] Number of batches to split the data (by sibpair) into for IBD inference. Useful for large datasets.

Results:

IBD segments
For each chromosome, a gzipped text file containing the IBD segments for the siblings is output.

5.2 impute.py

This script performs imputation of missing parental genotypes from observed genotypes in a family. It can impute missing parents from families with no genotyped parents but at least two genotyped siblings, or one genotyped parent and one or more genotyped offspring. To specify the siblings, one can either provide a pedigree file (–pedigree option) or

the relatedness inference output from KING with the –related –degree 1 options along with age and sex information.

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 and the same FATHER_ID and MOTHER_ID.

Use the –king option to provide the KING relatedness inference output (usually has suffix .kin0) and the –agesex option to provide the age & sex information. The script constructs a pedigree from this information and outputs it in the HDF5 output.

Args:

'-h', '–help'
[, default===SUPPRESS==] show this help message and exit

'-c'
[] Duplicates offsprings of families with more than one offspring and both parents and add '_' to the start

of their FIDs. These can be used for testing the imputation. The tests.test_imputation.imputation_test uses these.

'-silent_progress'
[] Hides the percentage of progress from logging

'-use_backup'
[] Whether it should use backup imputation where there is no ibd infomation available

'–ibd'
[str] Address of the IBD file without suffix. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome(chr_range is an optional parameters for this script).

'–ibd_is_king'
[] If not provided the ibd input is assumed to be in snipar. Otherwise its in king format with an allsegs file

'–bgen'
[str] Address of the phased genotypes in .bgen format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome(chr_range is an optional parameters for this script).

'–bed'
[str] Address of the unphased genotypes in .bed format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome(chr_range is an optional parameters for this script).

'–chr_range'
[] number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

'–bim'
[str] Address of a bim file containing positions of SNPs if the address is different from Bim file of genotypes

'–fam'
[str] Address of a fam file containing positions of SNPs if the address is different from fam file of genotypes

'–out'
[str, default=parent_imputed] Writes the result of imputation for chromosome i to outprefix{i}

'–start'
[int] The script can do the imputation on a slice of each chromosome. This is the start of that slice(it is inclusive)

'–end'
[int] The script can do the imputation on a slice of each chromosome. This is the end of that slice(it is inclusive).

'–pedigree'
[str] Address of the pedigree file. Pedigree file is a ' ' seperated csv with columns 'FID', 'IID', 'FATHER_ID', 'MOTHER_ID'. Default NaN value of Pedigree file is '0'. If your NaN value is something else be sure to specify it with –pedigree_nan option.

'–king'
[str]

Address of a kinship file in KING format. kinship file is a ' ' seperated csv with columns "FID1", "ID1", "FID2", "ID2, "InfType".

Each row represents a relationship between two individuals. InfType column states the relationship between two individuals. The only relationships that matter for this script are full sibling and parent-offspring which are shown by 'FS' and 'PO' respectively. This file is used in creating a pedigree file and can be generated using KING.

'–agesex'
[str]

Address of the agesex file. This is a " " seperated CSV with columns "FID", "IID", "FATHER_ID", "MOTHER_ID", "sex", "age".

Each row contains the age and sex of one individual. Male and Female sex should be represented with 'M' and 'F'. Age column is used for distinguishing between parent and child in a parent-offsring relationship inferred from the kinship file. ID1 is a parent of ID2 if there is a 'PO' relationship between them and 'ID1' is at least 12 years older than ID2.

'–pcs'
[str] Address of the PCs file with header "FID IID PC1 PC2 ...". If provided MAFs will be estimated from PCs

'–pc_num'
[int] Number of PCs to consider

'-find_optimal_pc'
[] It will use Akaike information criterion to find the optimal number of PCs to use for MAF estimation.

'–threads'
[int, default=1] Number of the threads to be used. This should not exceed number of the available cores. The default number of the threads is one.

'–processes'
[int, default=1] Number of processes for imputation chromosomes. Each chromosome is done on one process.

'–chunks'
[int, default=1] Number of chunks load data in(each process).

'–output_compression'
[str] Optional compression algorithm used in writing the output as an hdf5 file. It can be either gzip or lzf

'–output_compression_opts'
[int] Additional settings for the optional compression algorithm. Take a look at the create_dataset function of h5py library for more information.

'–pedigree_nan'
[str, default=0] The value representing NaN in the pedigreee.

Results:

HDF5 files
For each chromosome i, an HDF5 file is created at outprefix{i}. This file contains imputed genotypes, the position of SNPs, columns of resulting bim file, contents of resulting bim file, pedigree table and, family ids of the imputed parents, under the keys 'imputed_par_gts', 'pos', 'bim_columns', 'bim_values', 'pedigree' and, 'families', 'parental_status' respectively. There are also other details of the imputation in the resulting file. For more explanation see the documentation of snipar.imputation.impute_from_sibs.impute

5.3 gwas.py

Infers direct effects, non-transmitted coefficients (NTCs), and population effects of genome-wide SNPs on a phenotype.

Minimally: the script requires observed genotypes on phenotyped individuals and their parents, and/or parental genotypes imputed by snipar's impute.py script, along with a phenotype file.

Args:

'-h', '–help'
[, default===SUPPRESS==]

show this help message and exit

: str
Location of the phenotype file

'–bgen'
[str] Address of the phased genotypes in .bgen format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–bed'
[str] Address of the unphased genotypes in .bed format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–imp'
[str] Address of hdf5 files with imputed parental genotypes (without .hdf5 suffix). If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range (chr_range is an optional parameters for this script).

'–chr_range'
[] number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

'–out'
[str, default=./] The summary statistics will output to this path, one file for each chromosome. If the path contains '@', the '@' will be replaced with the chromosome number. Otherwise, the summary statistics will be output to the given path with file names chr_1.sumstats.gz, chr_2.sumstats.gz, etc. for the text sumstats, and chr_1.sumstats.hdf5, etc. for the HDF5 sumstats

'–pedigree'
[str] Address of pedigree file. Must be provided if not providing imputed parental genotypes.

'–parsum'
[] Regress onto proband and sum of (imputed/observed) maternal and paternal genotypes. Default uses separate paternal and maternal genotypes when available.

'–fit_sib'
[] Fit indirect effect from sibling

'–covar'
[str] Path to file with covariates: plain text file with columns FID, IID, covar1, covar2, ..

'–phen_index'
[int, default=1] If the phenotype file contains multiple phenotypes, which phenotype should be analysed (default 1, first)

'–min_maf'
[float, default=0.01] Ignore SNPs with minor allele frequency below min_maf (default 0.01)

'–threads'
[int] Number of threads to use for IBD inference. Uses all available by default.

'–max_missing'
[float, default=5] Ignore SNPs with greater percent missing calls than max_missing (default 5)

'–batch_size'
[int, default=100000] Batch size of SNPs to load at a time (reduce to reduce memory requirements)

'–no_hdf5_out'
[] Suppress HDF5 output of summary statistics

'–no_txt_out'
[] Suppress text output of summary statistics

'–missing_char'
[str, default=NA] Missing value string in phenotype file (default NA)

'–tau_init'
[float, default=1] Initial value for ratio between shared family environmental variance and residual variance

Results:

sumstats.gz
For each chromosome, a gzipped text file containing the SNP level summary statistics.

5.4 pgs.py

Infers direct effects, non-transmitted coefficients (NTCs), and population effects of a PGS on a phenotype.

Minimally: the script requires observed genotypes on individuals and their parents, and/or parental genotypes imputed by snipar's impute.py script, along with a SNP weights file.

Args:

'-h', '–help'
[, default===SUPPRESS==]

show this help message and exit

: str
Prefix for computed PGS file and/or regression results files

'–bgen'
[str] Address of the phased genotypes in .bgen format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–bed'
[str] Address of the unphased genotypes in .bed format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–imp'
[str] Address of hdf5 files with imputed parental genotypes (without .hdf5 suffix). If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range (chr_range is an optional parameters for this script).

'–chr_range'
[] number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

'–pedigree'
[str] Address of pedigree file. Must be provided if not providing imputed parental genotypes.

'–weights'
[str] Location of the PGS allele weights

'–SNP'
[str, default=SNP] Name of column in weights file with SNP IDs

'–beta_col'
[str, default=b] Name of column with betas/weights for each SNP

'–A1'
[str, default=A1] Name of column with allele beta/weights are given with respect to

'–A2'
[str, default=A2] Name of column with alternative allele

'–sep'
[str] Column separator in weights file. If not provided, an attempt to determine this will be made.

'–phenofile'
[str] Location of the phenotype file

'–pgs'
[str] Location of the pre-computed PGS file

'–covar'
[str] Path to file with covariates: plain text file with columns FID, IID, covar1, covar2, ..

'–fit_sib'
[] Fit indirect effects from siblings

'–parsum'
[] Use the sum of maternal and paternal PGS in the regression (useful when imputed from sibling data alone)

'–grandpar'
[] Calculate imputed/observed grandparental PGS for individuals with both parents genotyped

'–gparsum'
[] Use the sum of maternal grandparents and the sum of paternal grandparents in the regression (useful when no grandparents genotyped)

'–gen_models'
[, default=1-2] Which multi-generational models should be fit. Default fits 1 and 2 generation models. Specify a range by, for example, 1-3, where 3 fits a model with parental and grandparental scores

'–h2f'
[str] Provide heritability estimate in form h2f,h2f_SE (e.g. 0.5,0.01) from MZ-DZ comparison, RDR, or sibling realized relatedness. If provided when also fitting 2 generation model, will adjust results for assortative mating assuming equilibrium.

'–rk'
[str] Provide estimate of the correlation between parents PGIs in the form rk,rk_SE (e.g 0.1,0.01). If provided with h2f, will use for adjusting estimates for assortative mating.

'–bpg'
[] Restrict sample to those with both parents genotyped

'–phen_index'
[int, default=1] If the phenotype file contains multiple phenotypes, which phenotype should be analysed (default 1, first)

'–ibdrel_path'
[str] Path to KING IBD segment inference output (without .seg prefix).

'–sparse_thresh'
[float, default=0.05] Threshold of GRM/IBD sparsity

'–scale_phen'
[] Scale the phenotype to have variance 1

'–scale_pgs'
[] Scale the PGS to have variance 1 among the phenotyped individuals

'–compute_controls'
[] Compute PGS for control families (default False)

'–missing_char'
[str, default=NA] Missing value string in phenotype file (default NA)

'–no_am_adj'
[] Do not adjust imputed parental PGSs for assortative mating

'–force_am_adj'
[] Force assortative mating adjustment even when estimated correlation is noisy/not significant

'–threads'
[int, default=1] Number of threads to use

'–batch_size'
[int, default=10000] Batch size for reading in SNPs (default 10000)

Results:

PGS file
Output when inputting observed and imputed genotype files and a weights file. A file with PGS values for each individual and their parents, with suffix .pgs.txt. Also includes sibling PGS if –fit_sib is specified, and grandparental PGS if –grandpar is specified.

PGS effect estimates
Output when inputting a phenotype file. A file with suffix effects.txt containing estimates of the PGS effects and their standard errors, and a file with suffix vcov.txt containing the sampling variance-covariance matrix of the effect estimates

5.5 correlate.py

Infers correlations between direct effects and population effects, and between direct effects and average non-transmitted coefficients (NTCs). Minimally: the script requires summary statistics as output by snipar's gwas.py script, and either LD-scores (as output by snipar's ibd.py script or LDSC) or .bed files from which LD-scores can be computed Args:

'-h', '–help'
[, default===SUPPRESS==]

show this help message and exit

: str
Address of sumstats files in SNIPar sumstats.gz text format (without .sumstats.gz suffix). If there is a @ in the address, @ is replaced by the chromosome numbers in chr_range (optional argument)

'–chr_range'
[]

number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

: str
Prefix for output file(s)

'–ldscores'
[str] Address of ldscores as output by LDSC

'–bed'
[str] Address of observed genotype files in .bed format (without .bed suffix). If there is a # in the address, # is replaced by the chromosome numbers in the range of 1-22.

'–threads'
[int] Number of threads to use for IBD inference. Uses all available by default.

'–min_maf'
[float, default=0.05] Ignore SNPs with minor allele frequency below min_maf (default 0.05)

'–corr_filter'
[float, default=6.0] Filter out SNPs with outlying sampling correlations more than corr_filter SDs from mean (default 6)

'–n_blocks'
[int, default=200] Number of blocks to use for block-jacknife variance estimate (default 200)

'–save_delete'
[] Save jacknife delete values

'–ld_wind'
[float, default=1.0] The window, in cM, within which LD scores are computed (default 1cM)

'–ld_out'
[str] Output LD scores in LDSC format to this address

Results:

correlations
A text file containing the estimated correlations and their standard errors.

5.6 simulate.py

Simulates genotype-phenotype data using forward simulation. Phenotypes can be affected by direct genetic effects, indirect genetic effects (vertical transmission), and assortative mating.

Args:

'-h', '–help'
[, default===SUPPRESS==]

show this help message and exit

: int
Number of causal loci

: float
Heritability due to direct effects in first generation

: str
Prefix for simulation output files

'–bgen'
[str] Address of the phased genotypes in .bgen format. If there is a @ in the address, @ is replaced by the chromosome numbers in the range of chr_range for each chromosome (chr_range is an optional parameters for this script).

'–chr_range'
[] number of the chromosomes to be imputed. Should be a series of ranges with x-y format or integers.

'–nfam'
[int] Number of families to simulate. If inputting bgen and not given, will be one half of samples in bgen

'–min_maf'
[float, default=0.05] Minimum minor allele frequency for simulated genotyped, which will be simulted from density proportional to 1/x

'–maf'
[float] Minor allele frequency for simulated genotypes (not needed when providing bgen files)

'–n_random'
[int] Number of generations of random mating

'–n_am'
[int] Number of generations of assortative mating

'–r_par'
[float] Phenotypic correlation of parents (for assortative mating)

'–v_indir'
[float] Variance explained by parental indirect genetic effects as a fraction of the heritability, e.g 0.5

'–r_dir_indir'
[float] Correlation between direct and indirect genetic effects

'–beta_vert'
[float] Vertical transmission coefficient

'–save_par_gts'
[] Save the genotypes of the parents of the final generation

'–impute'
[] Impute parental genotypes from phased sibling genotypes & IBD

'–unphased_impute'
[] Impute parental genotypes from unphased sibling genotypes & IBD

Results:
genotype data in .bed format; full pedigree including phenotype and genetic components for all generations

CHAPTER

SIX

SIMULATION EXERCISE

Exercise on simulating data using the simulate.py script and performing family-based polygenic score analysis.

6.1 Simulating data

If *snipar* has been installed succesfully, the *command line scripts* should be accessible as executables in your terminal. *snipar* includes a script for simulating genotype-phenotype data according to different scenarios: direct and indirect genetic effects, with and without assortative mating. To simulate data, please first create a directory to store the data:

```
mkdir sim
```

Now, we are going to simulate data for 3000 families, each with two full-siblings, genotyped at 1000 independent SNPs. We simulate a phenotype affected by direct genetic effects and assortative mating. We are going to simulate 20 generations of assortative mating with parental phenotype correlation 0.5, reaching an approximate equilibrium. The command for this is:

```
simulate.py 1000 0.5 sim/ --nfam 3000 --impute --n_am 20 --r_par 0.5
--save_par_gts
```

where the first argument gives the number of causal SNPs, the second argument gives the random mating heritability, the third gives the output directory, –nfam gives the number of families, –impute tells the script to impute missing parental genotypes, –n_am gives the number of generations of assortative mating, the parental phenotype correlation is given by –r_par, and –save_par_gts tells the script to output the genotypes of the parents of the final generation in addition to the genotypes of the final generation.

Please change your working directory to sim/:

```
cd sim
```

In this directory, the file phenotype.txt is a phenotype file containing the simulated phenotype.

The genotype data (chr_1.bed) has been simulated so that there are 3000 independent families, each with two siblings genotyped.

6.2 Inferring IBD between siblings

The first step is to infer the identity-by-descent (IBD) segments shared between siblings. However, for the purpose of this simulation exercise (where SNPs are independent, so IBD inference doesn't work) we have provided the true IBD states in the file chr_1.segments.gz.

6.3 Imputing missing parental genotypes

This is performed using the *impute.py* script. To impute the missing parental genotypes without using phase information, use this command:

```
impute.py --ibd chr_@ --bed chr_@ --pedigree pedigree.txt --out chr_@ --threads
4
```

The pedigree along with the IBD segments shared between siblings recorded in chr_1.segments.gz are used to impute missing parental genotypes from the observed sibling and parental genotypes in chr_1.bed. The imputed parental genotypes are output to a *HDF5 file*, chr_1.hdf5.

6.4 Polygenic score analyses

snipar provides a script (*pgs.py*) for computing polygenic scores (PGS) based on observed/imputed genotypes, and for performing family based polygenic score analyses. The script computes a PGS from a *weights file*.

To compute the PGS using the true direct genetic effects as weights, use the following command:

```
pgs.py direct --bed chr_@ --imp chr_@ --weights causal_effects.txt --beta_col
direct
```

It outputs the PGS to a *PGS file*: direct.pgs.txt. The pgs computation script automatically estimates the correlation between parents PGS values (also using full-sibling offspring PGS values to do this) and performs an adjustment for assortative mating when using the imputed parental genotypes to compute the PGS.

To estimate direct effect and average NTC of the PGS, use the following command:

```
pgs.py direct --pgs direct.pgs.txt --phenofile phenotype.txt
```

This will output a population effect estimate (1 generation model) to direct.1.effects.txt, and direct effect and average NTC estimates to (2 generation model) to direct.2.effects.txt. The population and direct effect estimates are the coefficients on the proband PGS in the 1 and 2 generation models, so are indicated by the 'proband' row. The average NTC estimate is the coefficient on the parental PGS in the two-generation model. The first column gives the name of the covariate/PGS column, the second column gives the estimated regression coefficient, and the third column gives the standard error of the estimate. The sampling variance-covariance matrix of the estimates is output to direct.1.vcov.txt (for the 1 generation model) and direct.2.vcov.txt (for the 2 generation model).

As we are using the true direct effects as weights, the PGS captures all of the heritability, and the direct and population effects should both be the same (1 in expectation), and the average parental NTC should be zero (in expectation). To check this, read in the effect estimate output files in *R* or look at them using a text viewer (e.g. less -S on a unix system).

To compute the PGS from the true direct genetic effects+estimation error (such as would be obtained from a family-based GWAS), use the following command:

```
pgs.py direct_v1 --bed chr_@ --imp chr_@ --weights causal_effects.txt
--beta_col direct_v1
```

It outputs the PGS to a *PGS file*: direct_v1.pgs.txt. (Notice also that the inferred correlation between parents' PGSs is lower than when using the true direct genetic effects as weights due to estimation error in the weights.)

To estimate direct effect and average NTC of the PGS, use the following command:

```
pgs.py direct_v1 --pgs direct_v1.pgs.txt --phenofile phenotype.txt
```

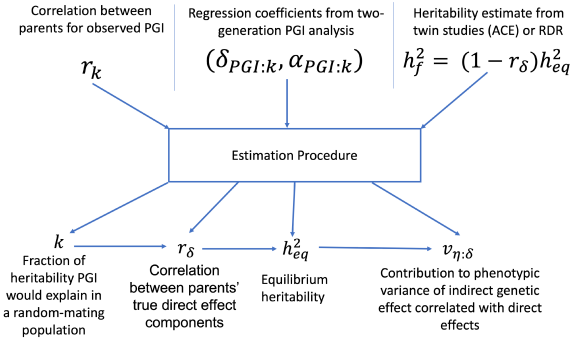
This will output a population effect estimate (1 generation model) to direct_v1.1.effects.txt, and direct effect and average NTC estimates to (2 generation model) to direct_v2.2.effects.txt.

Unlike when using the true direct genetic effects as weights, the direct effect of the PGS estimated from noisy weights (in direct_v1.1.effects.txt) will be smaller than the population effect (direct_v1.2.effects.txt). This is because the PGS does not capture all of the heritability due to estimation error in the weights. The PGS has its population effect inflated (relative to its direct effect) by assortative mating, which induces a correlation of the PGS with the component of the heritability not directly captured by the PGS due to estimation error. This inflation is not captured by the direct effect of the PGS because of the within-family variation used to estimate the direct effect is due to the random segregation of genetic material during meiosis. Here, the ratio between direct and population effects of the PGS should be around 0.86.

One should also observe a statistically significant average parental NTC (in direct_v1.2.effects.txt) of the PGS from the two-generation model despite the absence of parental indirect genetic effects in this simulation. Here, the ratio between the average NTC and the direct effect should be around 0.15. This demonstrates that statistically significant average NTC estimates cannot be interpreted as demonstrating parental indirect genetic effects, especially for phenotypes affected by assortative mating.

6.5 Adjusting for assortative mating

We now show how to adjust two-generation PGI results for assortative mating using the procedure outlined in Estimation of indirect genetic effects and heritability under assortative mating. The estimation procedure is summarized in this diagram:



The estimation requires as inputs: an estimate of the correlation between parents' scores, r_k; the regression coefficients from two-generation PGI analysis, $(\delta_{\text{PGI:}k}, \alpha_{\text{PGI:}k})$; and a heritability estimate, h_f^2,from MZ-DZ twin comparisons, RDR, or sib-regression.

The estimation procedure outputs estimates of: k, the fraction of heritability the PGI would explain in a random mating population; r_δ, the correlation between parents' true direct genetic effect components; h_{eq}^2, the equilibrium heritability, adjusting for the downward bias in heritability estimates from MZ-DZ comparisons, RDR, and sib-regression; α_δ, the indirect genetic effect of true direct genetic effect PGI; and $v_{\eta:\delta}$, the fraction of phenotypic variance contribued by the indirect genetic effect component that is correlated with the direct genetic effect component.

We can use *snipar* to compute the two-generation PGI estimates and the correlation between parents' scores, and we can input a heritability estimate into *pgs.py* script to complete the inputs, so that *snipar* will perform the two-generation analysis adjusting for assortative mating.

To perform the estimation, we will combine the offspring and parental genotype files. This enables us to estimate the correlation between parents' scores using the observed parental genotypes. (This is better than using the sibling

genotypes because the correlation estimate from observed parental genotypes is uncorrelated with the PGS regression coefficients.)

```
plink --bfile chr_1 --bmerge chr_1_par --out chr_1_combined
```

We now compute the noisy PGI using the observed offspring and parental genotypes:

```
pgs.py direct_v1_obs --bed chr_@_combined --weights causal_effects.txt
--beta_col direct_v1 --pedigree pedigree.txt
```

To complete the inputs to two-generation PGI analysis, we need an estimate of heritability, as one would obtain from sib-regression, RDR, or MZ-DZ twin comparisons. This estimate is a downard biased estimate of the equilibrium heritability, h^2_{eq}, by a factor of $(1-r_\delta)$, where r_δ is the correlation between the parents' direct genetic effect components.

We can obtain this from the VCs.txt output of the simulation, which can be read into R/Python/etc as table. Each row gives, for each generation, the variance of the direct genetic effect component, the phenotypic variance, and the correlation between parents' direct genetic effect components. The equilibrium heritability is obtained by using the values in the last row: dividing the variance of the direct genetic effect component (first column) by the phenotypic variance (second column). To then obtain the heritability as estimated by sib-regression, RDR, and MZ-DZ twin comparisons, we multiply the equilibrium heritability by $(1 - r_\delta)$, where r_δ is obtained from the third column of the last row. The equilibrium heritability should be around 0.59, and r_δ should be around 0.29, so the heritability as estimated by sib-regression, RDR, MZ-DZ twin comparisons should be around $h^2_f \approx (1 - 0.29) \times 0.59 = 0.42$.

We can now perform two-generation PGI analysis accounting for assortative mating using the following command, with the h2f argument set to the number computed from your VCs.txt file as outlined above (here we use 0.42):

```
pgs.py direct_v1_obs --pgs direct_v1_obs.pgs.txt --phenofile phenotype.txt
--h2f 0.42,0
```

This script will take the input heritability estimate (0.42) and the standard error of the estimate (here 0 since we used the true value) to estimate the fraction of heritability the PGI would explain in a random mating population, k, which should be around 0.5; the correlation between parents' direct genetic effect components, r_δ, which should be around 0.29; the equilibrium heritability, h^2_{eq}, which should be around 0.59; the ratio between direct and population effects that would be expected based on assortative mating alone, ρ_k, which should be around 0.86; the indirect genetic effect of true direct genetic effect PGI, α_δ, which should not be statistically significantly different from zero (with high probability) because there are no parental indirect genetic effects in this simulation; and $v_{\eta:\delta}$, the contribution to the phenotypic variance from the indirect genetic effect component correlated with direct genetic effect component, which should also not be statistically indistinguishable from zero (with high probability). These estimates are output to direct_v1_obs.am_adj_pars.txt.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

M

module
    snipar.scripts.correlate, 28
    snipar.scripts.gwas, 25
    snipar.scripts.ibd, 21
    snipar.scripts.impute, 22
    snipar.scripts.pgs, 26
    snipar.scripts.simulate, 29

S

snipar.scripts.correlate
    module, 28
snipar.scripts.gwas
    module, 25
snipar.scripts.ibd
    module, 21
snipar.scripts.impute
    module, 22
snipar.scripts.pgs
    module, 26
snipar.scripts.simulate
    module, 29