KinshipCorrelationGenerator

Release 1.2.1

Matthijs D. van der Zee

Mar 25, 2021

CONTENTS

Welcome to the KinshipCorrelationGenerator's documentation!

First time here? Best start at *Get started*

CHAPTER

ONE

CONTENTS

1.1 Get started

1.1.1 Installing Python

The script has been built using Python 3.7.7. Since it doesn't rely on very extensive Python features I would expect it to run just fine on any version of Python3. If you do not already have Python installed I recommend you download Anaconda.

1.1.2 Installing dependencies

The script uses several dependencies, once Python is installed you can install these by running

```
pip install --user --upgrade numpy pandas matplotlib statsmodels scipy openpyxl
```

1.1.3 Downloading script

Download the latest version of the script from GitHub.

1.1.4 Reformatting pedigree

Start by generating the reformatted pedigree file, if this has not already been done for your sample. Note you will only have to do this once! See *Generating a reformatted pedigree file*.

1.1.5 Generating correlations

Finally, make sure the reformatted pedigree, script, and your data file are in the same directory (not all of which is strictly necessary, but makes it easier). Then generate correlations, for example:

```
python KinshipCorrelationGenerator.py --data my_input.csv --outprefix my_output
```

See *Computing correlations*.

1.2 Generating a reformatted pedigree file

1.2.1 Creating the input pedigree file

Generating a reformatted pedigree file requires your existing pedigree file to have a specific format. First, the file must be uncompressed, and comma-separated. The header-line must be as follows:

```
!FamID,ID,Father,Mother,Gender,Twincode,DZtwincode,TwinHousehold3,SibHousehold2,
↪SpouseHousehold3
```

Each of these columns explained in more detail below:

- `!FamID`: Family identifier, unique for each family (definition of *family* is up to the user)
- `ID`: Personal identifier, unique for each individual
- `Father`: Personal identifier (`ID`) of the father.
- `Mother`: Personal identifier (`ID`) of the mother.
- `Gender`: Gender (M=Male, F=Female)
- `TwinCode`: MZ Twin code, unique identifier of each MZ twin pair. (First MZ twin pair in the data is TwinCode 1, second pair is TwinCode 2, etc)
- `DZtwincode`: DZ Twin code, unique identifier of each DZ twin pair.
- `TwinHousehold3`: Twin pair code, unique identifier of each MZ DZ pair within a family.
- `SibHousehold2`: Sibling household pair, unique identifier for all sets of siblings.
- `SpouseHousehold3`: Spouse household pair, unique identifier for each spousal pair.

1.2.2 Running the code

Running the following code:

```
python KinshipCorrelationGenerator.py --pedigree mypedigree.csv
```

Will generate a reformatted pedigree file `reformatted_pedigree.pickle`. This reformatted pedigree file will allow you to generate twin- sibling- spouse- and parent-offspring correlations.

If you would like to generate an extended pedigree, use the following:

```
python KinshipCorrelationGenerator.py --pedigree mypedigree.csv --extended
```

Will generate a reformatted extended pedigree file `reformatted_extended_pedigree.pickle`. This file also contains twin- sib- spouse- and parent-offspring pairs. Therefore if you have a reformatted extended pedigree, you do not need a regular reformatted pedigree.

Note: It is recommended to generate a reformatted pedigree file for the entire pedigree. The script can handle missing data, and reformatting the pedigree takes considerably longer than generating correlations. Therefore this reformatting only needs to be done once for an entire pedigree, and can subsequently be used for any number of correlations.

Warning: Generating a pedigree file will always overwrite any existing file named `reformatted_pedigree.pickle` or `reformatted_extended_pedigree.pickle`

If you have existing reformatted pedigrees in your directory that you would like to continue using, please rename this before generating a new one!

1.3 Computing correlations

1.3.1 Creating the input data file

First you will need to create an input data file. This file should be a comma-seperated file with a header and the following columns:

- `FISNumber`: Personal identifier
- `age`: Age

As well as any additional phenotype columns. Any column other than FISNumber and age will be treated as a phenotype, unless otherwise specified using `--exclude`.

The following names for phenotype columns are not allowed: `FISNumber`, `sex`, `age`, `Source`, `index`. As these names are used internally.

1.3.2 Generating weighted correlations

By default this script will assign weights to each pair of observations (within kinship within phenotype) depending on the number of occurrences of the personal identifier of both members of that pair to prevent bias from larger families. Pairs where both phenotypic values are missing are excluded from weight calculations. Other options for dealing with larger families are available, see *Script arguments*.

To generate the kinsip correlations run the following:

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output
```

This will generate 2 files: `my_output_Fam_correlations.csv`, which contains 1 column per phenotype containing all the phenotypic correlations, and `my_output_Fam_N.csv` which contains the sum of weights by default.

To generate the extended kinsip correlations run the following:

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended
```

1.3.3 Generating cross-trait correlations

To generate cross-trait (ie. bivariate) correlations of the combinations of your phenotypes (i.e. twin1 trait1 - twin1 trait2, mother trait1, son trait2, etc.) you can use the bivar option:

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --bivar
```

By adding this argument the script will now save two excel (.xlsx) files, one for the correlations, one for the sum of weights (or N depeding on your other *Script arguments*) named `my_output_bivar_Fam_correlations.xlsx` and `my_output_bivar_Fam_N.xlsx`. These tables will have 1 sheet per kinship (1 for MZM, 1 for MZF, etc), and these sheets contain the full correlation matrix of all phenotypes included in your input file.

The diagonal of these matrices are the standard within-phenotype kinship correlations (same as the standard output), the off-diagonal are the cross-trait cross-kinship correlations. Note these tables are not symmetrical, as the values on either side of the diagonal represent different correlations. Columns are phenotypes in ID_0, and rows are phenotypes in ID_1. As a specific example: in the MotherSon correlation table, column x, row y represents mother trait x and son trait y. Conversiley, column y row x represents the correlation of son's trait x with mother's trait y.

Note: The calculation for values on the diagonal in each sheet of the bivariate table is identical to that of the output without the `--bivar` option. Therefore if this option is enabled no csv files are stored, only the excel files.

1.3.4 Other common options

By default the script calculates weighted Pearson correlation, you can change this to a weighted Spearman correlation by specifying method.

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended --method spearman
```

To linearly correct your phenotypes for age before calculating the correlations, you can specify correct.

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended --correct age
```

Correct accepts many combinations of input, for example if you want to correct for age and sex and their interaction use:

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended --correct age+sex+age*sex
```

You can also use custom covariates as long as they are present with the same name in your input file. If you do so make sure to also add custom covariates to `--exclude`!

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended --correct age+bmi --exclude bmi
```

By default, no correlation is computed (and NA returned) when there are less than 30 complete pairs. You can change this to 15 (for example) as follows:

```
python KinshipCorrelationGenerator.py --data mydata.csv --outprefix my_output --
↪extended --min_n 15
```

1.4 Script arguments

1.4.1 -h, –help

Prints a short help summary with overview of arguments

1.4.2 –morehelp

Print more help on a specific function, or specific functions. Pass the argument name, or argument names (comma seperated), or `all` for a more detailed description of the whole script and its arguments.

1.4.3 –data

Path to the datafile. This datafile should at least contain the columns `FISNumber` and `age`. See *Creating the input data file*.

1.4.4 –outprefix

Prefix to use for output files.

1.4.5 –extended

Add calculation of extended-pedigree correlations (UncleAunt, AuntNephew, NephewNiece, etc) when used to compute correlations. Generates an extended reformatted pedigree when together with *–pedigree*

1.4.6 –pedigree

Path to the raw pedigree file, only used when generating a new reformatted pedigree file. See *Generating a reformatted pedigree file*.

1.4.7 –method

Method for computing correlation, should be Pearson, or Spearman, default is Pearson.

1.4.8 –bivar

Compute bivariate correlations of all combinations of phenotypes. See *Generating cross-trait correlations*.

1.4.9 –correct

Formula to correct phenotypes before computing correlations. Defaults to no correction

1.4.10 –exclude

Phenotypes for which no correlations should be calculated. For example, when custom covariates are used. Identifier column, age and sex columns are always excluded.

1.4.11 –raw_n

Return an additional .csv file containing the raw number of samples in each kinship in addition to the weighted N file.

1.4.12 –randomsample

Use only 1 pair per family, instead of calculating weighted correlations.

1.4.13 –use_repeated_families

Include all participants from larger families, i.e. don't drop or weigh for duplicate samples or larger families.

1.4.14 –longitudinal

When the input data is of longitudinal nature, this script can perform family-based selection, such that the difference in the year of survey completion among family-members is minimal. If you use this option the input data should contain the columns `index` (within-subject number of the survey, starting at 1 for fist survey), and `Source` (string label of the survey). Additionally, you should also add the argument *–surveycompletion*.

1.4.15 –surveycompletion

Dat file with survey completion years. Should contain the columns `FISNumber` (identical to datafile), `Source` (identical to datafile), and `invjr` (year of survey completion).

1.4.16 Additonal non-argument settings

The top of the Python file contains some additional settings you can tweak. * `upper_boundary`: Upper age boundary for inclusion in `--longitudinal`. default = 110 * `lower_boundary`: Lower age boundary for inclusion in `--longitudinal`. default = 0 * `check_cutoff_drops`: save an Age-cutoff_drops.txt file detailing subjects dropped by the cutoffs. default = False * `seed`: seed for random selection of subjects with `--randomsample`. default = 1415926536 * `explore_plot`: Generate scatterplots of each correlation. default = False: * `save_separate_data`: Store the raw data used for each correlation in separate files. default = False * `parallel`: Generate reformatted pedigree using multiple processing threads (Currently not working). default = False

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex