SOFTWARE TOOL ARTICLE

Seqfam: A python package for analysis of Next Generation Sequencing DNA data in families [version 1; referees: 1 approved with reservations, 1 not approved]

Matthew Frampton [1], Elena R. Schiff[1], Nikolas Pontikos [2], Anthony W. Segal[1], Adam P. Levine[1]

[1]Centre for Molecular Medicine, Division of Medicine, University College London, London, UK
[2]Institute of Ophthalmology, Moorfields Eye Hospital, University College London, London, UK

v1 **First published:** 06 Mar 2018, **7**:281 (doi: 10.12688/f1000research.13930.1)
Latest published: 06 Mar 2018, **7**:281 (doi: 10.12688/f1000research.13930.1)

Abstract

This article introduces *seqfam*, a python package which is primarily designed for analysing next generation sequencing (NGS) DNA data from families with known pedigree information in order to identify rare variants that are potentially causal of a disease/trait of interest. It uses the popular and versatile Pandas library, and can be straightforwardly integrated into existing analysis code/pipelines. *Seqfam* can be used to verify pedigree information, to perform Monte Carlo gene dropping, to undertake regression-based gene burden testing, and to identify variants which segregate by affection status in families via user-defined pattern of occurrence rules. Additionally, it can generate scripts for running analyses in a “MapReduce pattern” on a computer cluster, something which is usually desirable in NGS data analysis and indeed “big data” analysis in general.

This article summarises how *seqfam’s* main user functions work and motivates their use. It also provides explanatory context for example scripts and data included in the package which demonstrate use cases. With respect to verifying pedigree information, software exists for efficiently calculating kinship coefficients, so *seqfam* performs the necessary extra steps of mapping pedigrees and kinship coefficients to expected and observed degrees of relationship respectively. Gene dropping and the application of variant pattern of occurrence rules in families can provide evidence for a variant being causal. The authors are unaware of other software which performs these tasks in familial cohorts, so *seqfam* fulfils this need. Gene burden rather than single marker tests are often used to detect rare causal variants due to greater power. *Seqfam* may be an attractive alternative to existing gene burden testing software due to its flexibility, particularly in grouping and aggregating variants.

Keywords

python, bioinformatics, NGS, DNA, pedigree-information, gene-drop, gene-burden, kinship, mapreduce

This article is included in the Python Collection collection.

Open Peer Review

Referee Status: ? ✗

	Invited Referees 1	2
version 1 published 06 Mar 2018	? report	✗ report

1 **Brent S. Pedersen**, University of Utah, USA

2 **Alexandre Bureau**, Université Laval, Canada
CERVO Brain Research Centre, Canada
Ingo Ruczinski, Johns Hopkins Bloomberg School of Public Health, USA

Discuss this article

Comments (0)

Corresponding author: Matthew Frampton (mjeframpton@gmail.com)

Author roles: Frampton M: Conceptualization, Formal Analysis, Methodology, Resources, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Schiff ER**: Conceptualization, Data Curation, Resources, Writing – Original Draft Preparation, Writing – Review & Editing; **Pontikos N**: Data Curation, Resources, Software, Writing – Review & Editing; **Segal AW**: Conceptualization, Funding Acquisition, Project Administration, Resources, Supervision, Writing – Review & Editing; **Levine AP**: Conceptualization, Methodology, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Frampton M, Schiff ER, Pontikos N *et al.* ***Seqfam:*** **A python package for analysis of Next Generation Sequencing DNA data in families [version 1; referees: 1 approved with reservations, 1 not approved]** *F1000Research* 2018, **7**:281 (doi: 10.12688/f1000research.13930.1)

Grant information: This work was funded by the Charles Wolfson Charitable Trust.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

First published: 06 Mar 2018, **7**:281 (doi: 10.12688/f1000research.13930.1)

Introduction

Seqfam is a python package which is most useful in analysing next generation sequencing (NGS) DNA data from families with known pedigree information in order to identify rare variants that are potentially causal of a disease/trait. It was originally developed to analyse the whole exome sequencing data of a cohort of 200 families affected by a particular complex disease. Family-based study designs are often used to identify rare causal variants because such variants may be substantially more frequent in affected families than the general population (Auer & Lettre, 2015). *Seqfam* contains modules for Monte Carlo gene dropping (*gene_drop.py*) (MacCluer *et al.*, 1986), flagging variants based on their pattern of occurrence within families (*pof.py*), verification of ascertained pedigrees via kinship coefficients (*relatedness.py*), regression-based gene burden testing (*gene_burden.py*), and an additional module that facilitates the creation of job submission scripts on a computer cluster (*sge.py*).

For a rare variant to be considered potentially causal of a particular trait/disease based on *in silico* analysis, it must satisfy various criteria, such as being biologically plausible and predicted to be pathogenic. The user can run analyses with the *gene_drop.py* and *pof.py* modules to acquire additional evidence. Given the structure of the families, Monte Carlo gene dropping can assess whether a variant is enriched in the cohort relative to the general population, and assuming the trait/disease is more prevalent in the cohort, such enrichment supports causality. The user can use the *pof.py* module to identify variants which are carried by most or all affected members of a family, or even which segregate between affected and unaffected members. The authors are unaware of existing software packages for performing these analyses in familial cohorts, so *gene_drop.py* and *pof.py* fulfil this need. The *gene_drop.py* module can be considered complementary to the *RVsharing R* package (Bureau *et al.*, 2014), which calculates the probability of multiple affected relatives sharing a rare variant under the assumption of no disease association or linkage.

The *gene_burden.py* module implements the Combined Multivariate and Collapsing (CMC) burden test (Li & Leal, 2008) for detecting rare causal variants, where the multivariate test is a log-likelihood ratio test. The user can supply covariates to control for potential confounders such as divergent ancestry. This burden test should be applied to unrelated samples, and hence is of no use for cohorts containing few families. However, for cohorts containing a relatively large number of families, a sufficient number of unrelated cases can be extracted and combined with a separate set of unrelated controls. Burden tests aggregate rare variants in a gene or functional unit into a single score (Li & Leal, 2008; Madsen & Browning, 2009; Morris & Zeggini, 2010; Price *et al.*, 2010), and are one broad class of statistical methods which combine the effects of rare variants in order to increase power over single marker approaches. Sequence kernel association testing (SKAT) (Wu *et al.*, 2011) is another widely-used sub-category of such methods. In general, burden testing is more powerful than SKAT when a large proportion of variants are causal and are all deleterious/protective.

Stand-alone software exists for performing CMC testing with covariates e.g. RVTESTS (Zhan *et al.*, 2016) and PLINK/SEQ, but *gene_burden.py* is an attractive alternative due to its use of the versatile *Pandas* library. The CMC test function takes malleable *Pandas* data frames as input/output, which gives the user great flexibility in pre/post-processing the data, and this in turn may reduce I/O overhead. For example, let us consider variant grouping (e.g. by gene) and aggregation. This is specified via columns in the data frame, so the user can experiment widely by modifying them in python code e.g. group variants by functional unit instead of gene, aggregate within alternative sets of population allele frequency ranges, include/exclude unaggregated variants, and even aggregate by another factor such as consequence. As well as this flexibility, the function also produces rich output, including information about the independent variables.

The potential for genetic discovery in DNA sequencing data is reduced when samples are mislabelled. Hence, necessary quality control steps include identifying duplicates, and in the case of familial samples, verifying the ascertained familial relationships described in the pedigrees. The *relatedness.py* module facilitates these quality control steps and is used in conjunction with KING software (Manichaikul *et al.*, 2010). Given genotypes for relatively common variants, KING can efficiently calculate a kinship coefficient for each sample pair. The *relatedness.py* module can then map each kinship coefficient to a degree of relationship and check it corresponds with the pedigree. KING is often already part of NGS analysis pipelines, so incorporating *relatedness.py* is straightforward. *Peddy* (Pedersen & Quinlan, 2017) is an alternative which does not require KING.

The final module, *sge.py*, has general utility in running analyses of NGS data (and indeed any "big data") on computer clusters. Many NGS data analyses can be cast as "embarrassingly parallel problems" and hence executed more efficiently on a computer cluster using a "MapReduce pattern": the overall task is decomposed into independent sub-tasks ("map" tasks), then the map tasks run in parallel and after their completion, a "reduce" action merges/filters/summarises the results. For example, gene burden testing across the whole exome can be decomposed into independent sub-tasks by splitting the exome into sub-units e.g. chromosomes. Sun Grid Engine (SGE) is a widely used batch-queueing system, and analyses can be performed in a *MapReduce* pattern on SGE via so-called array jobs. The *sge.py* module can be used to automatically create the scripts required for submitting and running an array job.

Methods

Implementation

This section describes the functionality and methods employed by *seqfam's* 5 modules, which are:

1. *gene_drop.py*: Monte Carlo gene dropping;
2. *pof.py*: variant pattern of occurrence in families;
3. *gene_burden*.py: regression-based gene burden testing;

4. *relatedness.py*: identification of duplicates and verification of ascertained pedigree information via kinship coefficients;

5. *sge.py*: Sun Grid Engine (SGE) array job creation.

Figure 1 provides a visual representation of modules 1–4.

Gene dropping. By default, for each variant of interest, the *gene_drop.py* module performs 10,000 iterations of gene dropping in the familial cohort. In each iteration it gene drops in each family once, seeding the founder genotypes based on the population allele frequency. It then calculates the resulting simulated cohort allele frequency from samples specified by the user i.e. those which were sequenced. After completing all iterations

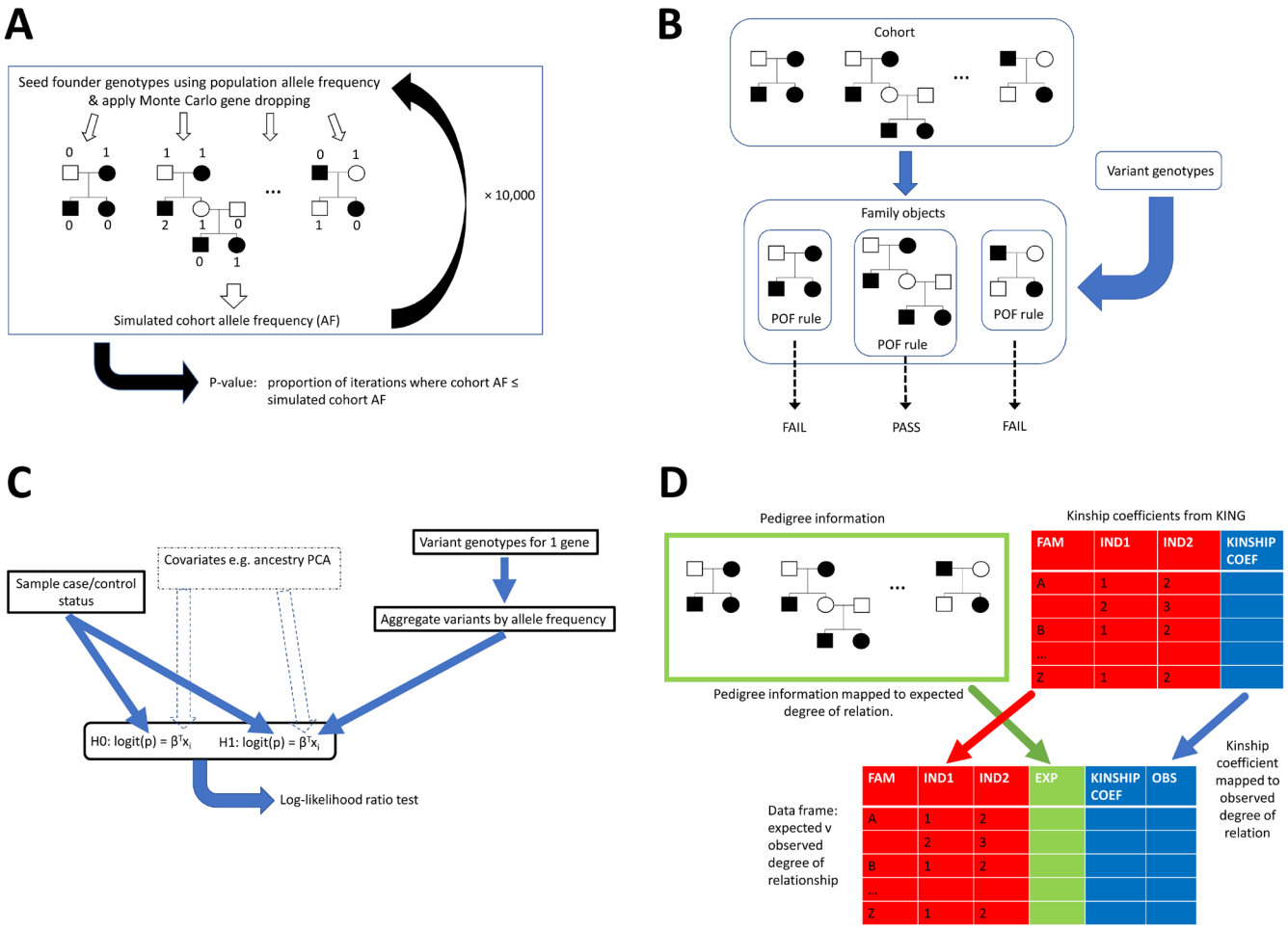

Figure 1. Flow charts representing functionality of the 4 main seqfam modules. Panel **A** represents the *Cohort.gene_drop* method in the *gene_drop.py* module which performs Monte Carlo gene dropping. On a single iteration, for each family the algorithm seeds founder genotypes based on the variant population allele frequency and then gene drops via depth-first traversals. Having done this for all families, a simulated cohort allele frequency (AF) is calculated and following many iterations (e.g. 10,000), a p-value, the proportion of iterations where cohort AF < simulated cohort AF, is outputted. Panel **B** represents the *Pof.get_family_pass_name_l* method in the pof.py module. Prior to calling the method, each family is assigned a variant pattern of occurrence in family (POF) rule. The method then takes a variant's genotypes and returns families whose POF rule is passed. Panel **C** represents the *CMC.do_multivariate_tests* method in the gene_burden.py module. This method takes sample affection status and variant genotypes across multiple genes, plus optionally covariates such as ancestry PCA coordinates. For each gene, the method aggregates the variants by allele frequency, constructs null and alternative hypothesis logit models which may include the covariates, and then performs a log-likelihood ratio test. Panel **D** represents the *Relatedness.get_exp_obs_df* method in the relatedness.py module. For input, this takes pedigree information and kinship coefficients from KING for each within-family sample pair. It maps these data to expected and observed degrees of relationship respectively, returning a data frame.

of gene dropping, *gene_drop.py* outputs the proportion of iterations in which the true cohort allele frequency is less than or equal to the simulated cohort allele frequency: a low proportion, e.g. < 5%, is evidence of enrichment.

The module gene drops in a family in the following way. First, it assigns a genotype (number of copies of the mutant allele) to each founder using a Binomial distribution where the number of trials is 2 and the probability of success in each trial is the population allele frequency. Hence the founders are assumed to be unrelated. It then performs a depth-first traversal starting from each founder (1 per spousal pair), and for heterozygotes, uses a random number generator to determine which parental allele to pass onto the child. Thus, every individual in the family is assigned a genotype.

Variant pattern of occurrence in families. For each family, the user can use the *pof.py* module to define a variant pattern of occurrence rule and check whether any supplied variants pass. The rule can specify a minimum value for the proportion of affected members (As) who are carriers (*A.carrier.p*), and/or a minimum difference between the proportion of As and unaffected members (Ns) who are carriers (*AN.carrier.diff*). Constraints for the number of genotyped As and Ns can also be added.

As an illustrative example, consider a cohort in which families can be categorised as follows based on their number of As and Ns:

1. "A4N1": ≥ 4 As and ≤ 1 N
2. "A3N2": ≥ 3 As and ≥ 2 Ns

For the A4N1 families, the user may be interested in variants carried by all As and so require *A.carrier.p* = 1, while for the A3N2 families, they may be interested in variants which are more prevalent in As than Ns and so require *AN.carrier.diff* ≥ 0.5.

Gene burden. To use the *gene_burden.py* module, the user must first read the various required data into *Pandas* data frames. These data include variant annotations by which to group (e.g. gene/functional unit) and aggregate (e.g. population allele frequency), and the genotypes, affection status and covariates for the unrelated samples. The user can specify multiple categories in which to aggregate variants (e.g. into population allele frequency ranges of 0–1% and 1–5%), and variants outside these categories (e.g. more common variants) remain unaggregated. An aggregated variant category takes the value 0 or 1. For each variant group, having aggregated the variants, *gene_burden.py* will perform a multivariate test, which is a log-likelihood ratio test based on Wilk's theorem:

$$X^2 = 2(ll_{h1} - ll_{h0});\ df = df_{h1} - df_{h0}$$

where *ll* is log-likelihood, *h1* is the alternative hypothesis, *h0* is the null hypothesis and *df* is degrees of freedom. Specifically, it is a log-likelihood ratio test on null and alternative hypothesis logit models where the dependent variable is derived from affection status, the variant variables (aggregated and/or unaggregated) are independent variables in the alternative model and the covariates are independent variables in both. The logit models are fitted using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm**.**

Since aggregation by population allele frequency is common, the module provides a function to assign variants to allele frequency ranges based on multiple columns (derived from different population databases such as *gnomAD* (Lek *et al.*, 2016)). The user specifies a preference order in case the variant is absent from the most preferred database(s).

Relatedness. As input, the *relatedness.py* module requires pedigree information and a file containing kinship coefficients outputted by KING. For each sample pair, the module will map the pedigree information and kinship coefficient to an expected and observed degree of relationship respectively. The mapping from kinship coefficient to relationship is as specified in KING documentation: > 0.354 for duplicate samples/monozygotic twins, 0.177–0.354 for 1st degree relatives, 0.0884–0.177 for 2nd degree relatives, 0.0442–0.0884 for 3rd degree relatives, and < 0.0442 for unrelated. The user can change this mapping if they wish.

Computer cluster array job creation. To use the *sge.py* module, the user must first create lists of map tasks, map tasks requiring execution and reduce tasks. The map tasks requiring execution are map tasks which have not previously completed successfully and hence need to run. Given these lists, the *sge.py* module can create all necessary scripts/files for submitting and running an array job. This includes scripts for all map tasks, a text file specifying that only the map tasks requiring execution should run, and a master executable submit script for submitting the array job to the job scheduler.

Operation

Seqfam is compatible with Windows, Mac OS X and Linux operating systems. It is coded using Python 3.6, but can also be run by Python 2.7. It requires various data analysis libraries/packages, almost all of which can be acquired by downloading and installing the Anaconda python distribution. The *StatsModels* module, which is not in the Anaconda distribution, is also required. Having cloned the repository, the user should add the repository *src* directory to their environmental python path.

Use cases

The repository contains additional scripts in *src/examples* which demonstrate the functionality of the modules on example data, including files in the *data* directory. The scripts are *1_test_gene_drop.py*, *2_test_pof.py*, *3_test_gene_burden.py*, *4_test_relatedness.py*, and *5_test_sge.py*. The reader can also refer to Table 1 for a summary of the main user functions of the

Table 1. Summary of main user functions in *seqfam* modules. tsv = tab-separated values; AF = allele frequency; POF = pattern of occurrence in family.

Module	Method (Class)	Description	Input	Output
gene_drop	*gene_drop (Cohort)*	Monte Carlo gene dropping	Cohort tsv file (pedigree info), variant population AF, cohort AF, list of samples genotyped, number of iterations	p-value
pof	*get_family_pass_name_l (Pof)*	Variant POF with respect to affected (& unaffected) members	Variant POF rule & genotypes	List of families whose POF rule is passed by variant.
gene_burden	*do_multivariate_tests (CMC)*	Regression-based gene burden testing	Files containing samples, genotypes & covariates files; output path	Data frame and csv file containing burden test results
relatedness	*find_duplicates (Relatedness)*	Identify duplicates from kinship coefficient	King file	List of duplicates
	get_exp_obs_df (Relatedness)	Map pedigrees & kinship coefficients to expected & observed degrees of relationship.	Cohort tsv, KING within-family sample pair kinship coefficient file	Data frame of expected & observed degrees of relationship.
sge	*make_map_reduce_jobs (SGE)*	Make compute cluster array job scripts	Filename prefix, lists of map tasks, map tasks to execute and reduce tasks.	Scripts required to run array job including master submit script.

5 *seqfam* modules, which includes their input/output. Data in the example data files are derived from the whole exome sequencing of a large cohort of over 200 families with inflammatory bowel disease (unpublished study[1]).

Gene dropping

The only input file for *1_test_gene_drop.py* is *cohort.tsv*, which contains the pedigree information for an example familial cohort. This cohort has 3,608 samples from 251 families, and the complexity of the families, calculated as *2n-f* where *n* and *f* are the number of non-founders and founders respectively (Abecasis *et al.*, 2002), has median 9 and range 0–103. The *cohort.csv* file is in *fam file* format (Purcell *et al.*, 2007), meaning it has 1 row per individual and 6 columns for family ID, person ID, father, mother, sex and affection.

The *1_test_gene_drop.py* script first creates a *Cohort* object from *cohort.tsv,* which stores each family tree, then calls the *gene_drop* method with the following arguments: *pop_af* and *cohort_af* are the allele frequency of a particular variant in the general population and cohort respectively, *sample_genotyped_l* is the list of cohort samples with a genotype, and *gene_drop_n* is the number of iterations of gene dropping to perform. Hence the samples in *sample_genotyped_l* are used by the user to calculate *cohort_af*, and by the method to calculate the simulated cohort allele frequencies. The method returns a p-value. The script calls the *gene_drop* method with ascending values for *cohort_af*, and so descending p-values are returned.

In benchmarking with the Unix *time* command on an Intel Xeon CPU E7-4830 v2 processor (20M Cache, 2.20 GHz), creating a cohort object and then executing a single call of the *gene_drop* method (*gene_drop_n*=1,000) required approximately 25 seconds of Computer Processing Unit (CPU) time. CPU time scales linearly with *gene_drop_n* and cohort size.

Variant pattern of occurrence in families

There are no input files for *2_test_pof.py*. The example script first creates a *Pof* object which stores a couple of *Family* objects, each representing an example family and its variant pattern of occurrence rule. Next it calls the *Pof* object's *get_family_pass_name_l* method with the argument *genotypes_s,* which is a *Pandas Series* containing sample genotypes for a particular variant. The method returns a list of families whose pattern of occurrence rule is passed by this variant.

Gene burden

The *3_test_gene_burden.py* script uses the *gene_burden.py* module to perform CMC tests on example data: it performs 1 CMC test per gene where variants in the population allele frequency ranges of 0-1% and 1-5% are aggregated, and any variants ≥ 5% remain unaggregated. The input files are in the *data/gene_burden* directory: *samples.csv*, *genotypes.csv* and optionally *covariates.csv*. The *samples.csv* file contains the samples' ID and affection status where 2 indicates a case and 1 a control, and *genotypes.csv* contains 1 row per variant with columns for the sample genotypes, and for variant grouping and aggregation e.g. gene and population allele frequency.

A sample's genotype is the number of alternate alleles which it carries (0-2). The *covariates.csv* file contains the covariates to control for, which in this case are ancestry PCA coordinates.

The script first reads *samples.csv* into a *Pandas Series*, and *genotypes.csv* and *covariates.csv* into *Pandas DataFrames.* These data frames are indexed by variant ID and covariate name respectively. Having created a *CMC object*, the script calls its *assign_vars_to_pop_frq_cats* method in order to map the variants to the desired population allele frequency ranges. Multiple population allele frequency columns (databases) are used here, ordered by descending preference. The mapping is stored in a new column in the genotypes data frame. Finally, the script calls the *do_multivariate_tests* method to perform the CMC tests, specifying the gene column for grouping the variants, and the new allele frequency range column for aggregation. The results are written to a CSV (comma-separated values) file and returned in a data frame. They include the number of variants in each aggregation category, the number of unaggregated variants ("unagg" column), the log-likelihood ratio test p-value with/without covariates ("llr_p" and "llr_cov_p"), and the coefficient/p-value for each aggregated variant variable ("_c" and "_p").

Relatedness

The input files for *4_test_relatedness.py* are *cohort.tsv* (as used in gene dropping), and *data/relatedness/king.kinship.ibs* which was outputted by KING and contains kinship coefficients for within-family sample pairs. The example script first creates a *Relatedness* object which stores the paths to these files, then calls the object's *find_duplicates* and *get_exp_obs_df* methods. The former returns any within-family sample duplicates, and the latter returns a *Pandas DataFrame* containing the expected and observed degree of relationship for each within-family sample pair. Finally, the script prints the sample pairs which have a different expected and observed degree of relationship.

Computer cluster array job creation

There are no input files for *5_test_sge.py*. This script first makes lists of map tasks (*map_task_l*), map tasks to execute (*map_task_exec_l*), and reduce tasks (*reduce_task_l*). Here *map_tasks_exec_l* contains every other map task. Next, the script creates an *SGE* object which stores the directory where job scripts will be written (here *data/sge*). Finally, it calls the object's *make_map_reduce_jobs* method with the following arguments: a prefix for all job script names (here "test") and the above 3 lists. This writes the job scripts, and were they for a real array job (they are not), the user could then submit it to the job scheduler by running the master executable submit script *data/sge/submit_map_reduce.sh*. The *test.map_task_exec.txt* file specifies which map tasks to run i.e. the map tasks in the *map_tasks_exec_l* list.

Conclusions

This article has introduced *seqfam*, a python package, primarily designed for analysing NGS DNA data from families with known pedigree information in order to identify rare variants that are potentially causal of a disease/trait of interest. It currently includes modules for verification of pedigree information, gene dropping, applying variant pattern of occurrence rules in families, gene burden testing and job script generation on a computer cluster.

Data and software availability

Latest source code and example files are available at: https://github.com/mframpton/seqfam

Archived source code as at time of publication: http://doi.org/10.5281/zenodo.1173768 (Frampton, 2018)

License: GNU General Public License v3.0

Footnotes

[1]Authors involved in the unpublished study: E. R. Schiff[a], M. Frampton[a], F. Semplici[a], N. Pontikos[b], S. Bloom[c], S. McCartney[c], R. Vega[c], L. Lovat[d], E. Wood[e], A. Hart[f], D. Crespi[g], M. Furman[g], S. Mann[h], C. Murray[i], A. P. Levine[a] and A. W. Segal[a]

[a]Centre for Molecular Medicine, Division of Medicine, University College London

[b]Institute of Ophthalmology, Moorfields Eye Hospital, University College London

[c]Department of Gastroenterology. University College London Hospital

[d]Research Department of Tissue and Energy, Division of Surgery and Interventional Science, UCL, U.K.

[e]Gastroenterology Department, Homerton University Hospital, London, U.K.

[f]Gastroenterology Department, St Marks Hospital, U.K.

[g]Centre for Paediatric Gastroenterology, Royal Free Hospital, London, U.K.

[h]Gastroenterology Department, Barnet General Hospital, U.K.

[i]Centre for Gastroenterology, Royal Free Hospital, London, U.K.

Competing interests

The authors declare no competing interests.

Grant information

This work was funded by the Charles Wolfson Charitable Trust.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

We gratefully acknowledge all study participants whose sequencing data and pedigree information were used in developing and testing *seqfam*, plus their referring clinicians.

References

Abecasis GR, Cherny SS, Cookson WO, *et al.*: **Merlin--rapid analysis of dense genetic maps using sparse gene flow trees.** *Nat Genet.* 2002; **30**(1): 97–101.
PubMed Abstract | Publisher Full Text

Auer PL, Lettre G: **Rare variant association studies: Considerations, challenges and opportunities.** *Genome Med.* 2015; **7**(1): 16.
PubMed Abstract | Publisher Full Text | Free Full Text

Bureau A, Younkin SG, Parker MM, *et al.*: **Inferring rare disease risk variants based on exact probabilities of sharing by multiple affected relatives.** *Bioinformatics.* 2014; **30**(15): 2189–2196.
PubMed Abstract | Publisher Full Text | Free Full Text

Frampton M: **seqfam.** *Zenodo.* 2018.
Data Source

Lek M, Karczewski KJ, Minikel EV, *et al.*: **Analysis of protein-coding genetic variation in 60,706 humans.** *Nature.* 2016; **536**(7616): 285–91.
PubMed Abstract | Publisher Full Text | Free Full Text

Li B, Leal SM: **Methods for detecting associations with rare variants for common diseases: application to analysis of sequence data.** *Am J Hum Genet.* 2008; **83**(3): 311–321.
PubMed Abstract | Publisher Full Text | Free Full Text

MacCluer JW, VandeBerg JL, Read B, *et al.*: **Pedigree analysis by computer simulation.** *Zoo Biol.* 1986; **5**(2): 147–160.
Publisher Full Text

Madsen BE, Browning SR: **A groupwise association test for rare mutations using a weighted sum statistic.** *PLoS Genet.* 2009; **5**(2).
PubMed Abstract | Publisher Full Text | Free Full Text

Manichaikul A, Mychaleckyj JC, Rich SS, *et al.*: **Robust relationship inference in genome-wide association studies.** *Bioinformatics.* 2010; **26**(22): 2867–2873.
PubMed Abstract | Publisher Full Text | Free Full Text

Morris AP, Zeggini E: **An evaluation of statistical approaches to rare variant analysis in genetic association studies.** *Genet Epidemiol.* 2010; **34**(2): 188–193.
PubMed Abstract | Publisher Full Text | Free Full Text

Pedersen BS, Quinlan AR: **Who's Who? Detecting and Resolving Sample Anomalies in Human DNA Sequencing Studies with *Peddy*.** *Am J Hum Genet.* 2017; **100**(3): 406–413.
PubMed Abstract | Publisher Full Text | Free Full Text

Price AL, Kryukov GV, de Bakker PI, *et al.*: **Pooled Association Tests for Rare Variants in Exon-Resequencing Studies.** *Am J Hum Genet.* 2010; **86**(6): 832–838.
PubMed Abstract | Publisher Full Text | Free Full Text

Purcell S, Neale B, Todd-Brown K, *et al.*: **PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses.** *Am J Hum Genet.* 2007; **81**(3): 559–575.
PubMed Abstract | Publisher Full Text | Free Full Text

Wu MC, Lee S, Cai T, *et al.*: **Rare-variant association testing for sequencing data with the sequence kernel association test.** *Am J Hum Genet.* 2011; **89**(1): 82–93.
PubMed Abstract | Publisher Full Text | Free Full Text

Zhan X, Hu Y, Li B, *et al.*: **RVTESTS: An efficient and comprehensive tool for rare variant association analysis using sequence data.** *Bioinformatics.* 2016; **32**(9): 1423–1426.
PubMed Abstract | Publisher Full Text | Free Full Text

Open Peer Review

Current Referee Status: ? ✗

Version 1

Referee Report 22 March 2018

doi:10.5256/f1000research.15143.r31577

✗ **Alexandre Bureau** [1,2], **Ingo Ruczinski** [3]

[1] Department of Social and Preventive Medicine, Université Laval, Quebec, QC, Canada

[2] CERVO Brain Research Centre, Quebec, QC, Canada

[3] Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, Baltimore, MD, USA

The python package *seqfam* is a collection of genetic association analysis tools, some but not all tailored for familial data. The instructions provided by the authors allow for a successful installation of the *seqfam* package. Example scripts provided with the package perform the analyses described in the manuscript.

This manuscript, however, also raises concerns with these reviewers. Tools performing similar tasks already exist in other computing languages (PLINK/R) and are widely used. The gene dropping and filtering based on pattern of occurrence in families implemented in seqfam differ in some respects from approaches implemented in other packages, which have been thoroughly assessed for their statistical properties. As this is a software manuscript, an assessment of the statistical properties of the *seqfam* approaches is not required, but we note that it will be very hard to convince a practitioner to use this approach if no consideration of power for various settings is given. Moreover, we are also of the opinion that some of the *seqfam* tools should not be used as currently proposed.

Major concerns:

1. Referring to gene dropping and the application of variant pattern of occurrence rules in families, the authors state in the abstract that they are "unaware of other software which performs these tasks in familial cohorts", but unfortunately missed a somewhat large body of literature. A reasonably thorough review of the literature is imperative for any manuscript.

   Most applicable in our opinion for testing rare variants are RareIBD (Sul et al. 2016[1]) which uses the number of affected relatives carrying a rare allele similar to the allele frequency used in *seqfam*, and GESE (Qiao et al. 2017[2]), based on the probability of multiple affected relatives sharing a rare variant, which we proposed in a recent manuscript that the authors do cite (Bureau et al. 2014[3]). Of note, we have since expanded our approach to gene-based analyses, refined the rare variant definition based on haplotypes, and introduced a partial sharing test based on rare variant sharing probabilities for subsets of affected family members (Bureau et al. 2018[4], posted online after the *seqfam* manuscript was submitted, so the authors could not be aware of it). Another alternative is testing of co-segregation of variants with disease under a parametric linkage analysis as in the pedigree Variant Annotation, Analysis and Search Tool pVAAST (Hu et al. 2014[5]).

   There are also other implementations of variant filtering based on proportions of affected and

unaffected carriers, e.g. the R package Mendelian of Broeckx et al. (2015)[6].

2. The inference of the *seqfam* gene dropping test depends on knowledge of the variant allele frequencies in the study population. Methods that depend on such knowledge tend to be extremely sensitive to deviations from the truth (see for example the above mentioned Qiao et al 2017 and Bureau et al 2018 papers). In most instances we have only a very rough idea about exact allele frequencies (especially for rare variants) when big population-based studies of said population already have been carried out, or we have no knowledge at all if a new population is investigated. Thus, methods that do not depend on knowledge of allele frequencies are to be preferred, but at a minimum the authors must report the impact of allele frequency misspecification (as for example done in Bureau et al 2018), and the size of the control group from the appropriate population which is needed to obtain reliable allele frequencies for the proposed method (as for example done in the Qiao et al paper) to guide the user.

3. There is no notion of error control due to multiple comparisons in the manuscript. And with the number of variants present even in whole exome sequencing data, the recommended number of iterations of gene dropping (10,000) most likely will be inadequate to estimate the p-values with sufficient precision. For example, even if none of 10,000 iterations yields anything more extreme than observed in the actual data, the upper bound for the 95% confidence interval for the true p-value will be 0.00037 (for example type binom.test(0,10000)$conf in R). Thus, with a couple hundred variants assessed, the upper bound for the confidence interval would not beat the Bonferroni threshold to declare significance.

4. The inference based on so-called "patterns of occurrence in families" (POFs) does not quantify the total evidence for linkage/association per se (see for example the RareIBD paper by Sul et al) but investigates user-chosen patterns instead. This could be powerful if one knew exactly what patterns to look for a priori, but we cannot imagine this ever being the case in practice, nor is any guidance on this offered in the manuscript. The example script involves specifying the pattern of occurrence rule with example families, but in practice family samples contain various family structures, so it is not clear how it generalizes. Why not simply input values for A.carrier.p and AN.carrier.diff? As it stands, it is left to the user to tinker with the vast number of possibilities. In that sense, inference based on these specific patterns should at best be considered secondary analysis.

5. No functionality is provided in *seqfam* to apply the gene burden test to familial samples. As the authors rightly point out, the test is valid with unrelated subjects. As it stands, users must extract unrelated subjects from familial cohorts outside of *seqfam*. This function should offer the option to perform selection of unrelated cases from familial cohorts, or otherwise it should be removed from *seqfam*.

6. The authors should also mention that in many family-based sequencing studies genotypes of founders are unavailable (as for example DNA is not available from family members several generations ago), and quite frequently only distantly related subjects are sequenced. That means that for these studies only a single unrelated subject can be extracted from each family, limiting the scope of the gene burden test substantially.

7. When executing the test program, the p-value of the LR test is identical with or without covariates, so adjustment for covariates does not seem to do anything. The expressions “log-likelihood ratio test p-value with/without covariates” for *llr_p* and *llr_cov_p* are ambiguous. In the output, the *llr_p*

test is missing when one of the variant classes is empty, while *llr_cov_p* is always reported. Is *llr_cov_p* a test of the covariates?

There are also some more minor concerns and questions:

1. For the gene dropping test it is not clear in the manuscript whether the variant frequency is computed only for affected family members or all family members together.

2. The *gene_drop.py* module would be more useful if it took genotype data as input like the *gene_burden.py* and *pof.py* modules and computed the cohort allele frequency for every variant. The authors state that the *gene_drop.py* module performs gene dropping “for each variant of interest”, but this is misleading, as the *gene_drop* method takes as argument the frequency of a single variant. Users must write a script to loop over variants in their genotype file (pre-filtered by population allele frequency and possibly other annotations), compute variant frequency in their cohort, and pass it to *gene_drop*. By contrast, competing packages GESE, RareIBD, pVAAST and our new Bioconductor RVS package process genotype data. They also offer the option to perform analyses at the gene level, while the *gene_drop.py* module is restricted to single variant analyses.

3. Filtering of variants (e.g. those predicted to be pathogenic) is critical and relies heavily on available annotation. However, this information is largely available only for exomes. Filtering of variants by annotation should be discussed. Is *seqfam* foremost a package for targeted sequencing studies (i.e. exome sequencing)? If yes, this should be stated. Otherwise, some guidance on application to whole genome sequencing studies should be given.

4. What exactly is the additional value of *relatedness.py*? For example, the GENESIS Bioconductor package allows for the fast calculation of genetic relatedness matrices (GRMs), which can easily be subsetted using the cut-offs suggested by the authors. It is unclear whether *relatedness.py* also updates the pedigrees accordingly (which can be done using kinship2 in R). Also, what happens if founders are found to be related? This should be part of the quality control and considered in the inference.

5. The *relatedness.py* module could be generalized by allowing input from other kinship computation packages than King.

6. The program for computer cluster job creation can be a useful utility program complementing the *seqfam* analysis tools, but it has no specific connection to family sequencing studies. It does not deserve to be featured as a main user function in the manuscript, and instead should be mentioned as a utility program.

7. There is no mention of dominant and recessive inheritance patterns, which are often encountered with rare causal variants. The gene dropping statistics is likely better suited for dominant variants, while different POF in families could be defined for dominant and recessive variants. This should be discussed.

References

1. Sul JH, Cade BE, Cho MH, Qiao D, Silverman EK, Redline S, Sunyaev S: Increasing Generality and Power of Rare-Variant Tests by Utilizing Extended Pedigrees.*Am J Hum Genet*. 2016; **99** (4): 846-859 PubMed Abstract | Publisher Full Text
2. Qiao D, Lange C, Laird NM, Won S, Hersh CP, Morrow J, Hobbs BD, Lutz SM, Ruczinski I, Beaty TH,

Silverman EK, Cho MH: Gene-based segregation method for identifying rare variants in family-based sequencing studies.*Genet Epidemiol*. **41** (4): 309-319 PubMed Abstract | Publisher Full Text
3. Bureau A, Younkin SG, Parker MM, Bailey-Wilson JE, Marazita ML, Murray JC, Mangold E, Albacha-Hejazi H, Beaty TH, Ruczinski I: Inferring rare disease risk variants based on exact probabilities of sharing by multiple affected relatives.*Bioinformatics*. 2014; **30** (15): 2189-96 PubMed Abstract | Publisher Full Text
4. Bureau A, Begum F, Taub MA, Hetmanski J, Parker MM, Albacha-Hejakzi H, Scott AF, Murray JC, Marazita ML, Bailey-Wilson JE, Beaty TH, Ruczinski I: Inferring Disease Risk Genes from Sequencing Data in Multiplex Pedigrees Through Sharing of Rare Variants. *bioRxiv*. 2018.
5. Hu H, Roach JC, Coon H, Guthery SL, Voelkerding KV, Margraf RL, Durtschi JD, Tavtigian SV, Shankaracharya, Wu W, Scheet P, Wang S, Xing J, Glusman G, Hubley R, Li H, Garg V, Moore B, Hood L, Galas DJ, Srivastava D, Reese MG, Jorde LB, Yandell M, Huff CD: A unified test of linkage analysis and rare-variant association for analysis of pedigree sequence data.*Nat Biotechnol*. 2014; **32** (7): 663-9 PubMed Abstract | Publisher Full Text
6. Broeckx BJ, Coopman F, Verhoeven G, Bosmans T, Gielen I, Dingemanse W, Saunders JH, Deforce D, Van Nieuwerburgh F: An heuristic filtering tool to identify phenotype-associated genetic variants applied to human intellectual disability and canine coat colors.*BMC Bioinformatics*. 2015; **16**: 391 PubMed Abstract | Publisher Full Text

Is the rationale for developing the new software tool clearly explained?
No

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
No

Competing Interests: No competing interests were disclosed.

Referee Expertise: Statistical genetics, genetic epidemiology

We have read this submission. We believe that we have an appropriate level of expertise to state that we do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Referee Report 09 March 2018

doi:10.5256/f1000research.15143.r31578

Brent S. Pedersen
Department of Human Genetics, USTAR Center for Genetic Discovery, University of Utah, Salt Lake City, UT, USA

Frampton et al describe seqfam, a set of tools to perform family-based analysis on sequence data. It does gene-dropping, burden-testing, relatedness evaluation, "pattern of occurrence", and contains a script for running cluster analysis.

My main critique is that this seems like a set of unrelated modules, some of which have (potentially) better alternatives. For example, the SKAT family of burden tests have been widely used in place of CMC implemented in this package and several implementations of CMC exist. Therefore, it's not clear from reading why this package is needed. In addition, as the authors note, this is only useful if there are sufficient unrelated controls. Another example is the relatedness module which relies on KING output. The additional utility provided in seqfam is a subset of the utility provided in peddy which does not rely on plink or KING. Again, it's not clear when the use of seqfam in this context would be preferable.

The parts that are more novel are the gene dropping and the "POF" scripts. Running the script currently requires setting the PYTHONPATH. The package should have a requirements.txt and a proper setup.py so it can be used in a more standard manner. I was able to run the example script for gene dropping and see sensible outputs. Other than this, there are no tests in the repository and nothing in the manuscript indicating any of the methods were evaluated for correctness or accuracy performance.

There is also very little documentation. For example,the gene_burden_test has:

```
cmc_result_df = cmc.do_multivariate_tests(sample_s, geno_df, group_col=gene_col,
agg_col="pop_frq_cat", agg_val_l=["rare","mod_rare"], covar_df=covar_df, results_path=results_path)
```

A user must read the code to see that agg_val_l is: `(list of strs): names of the aggregated categories.` but it's still not clear what that means or how to use it. For each module intended for use, there should be clear text and API documentation. Figure 1 and some of the manuscript would be useful as documentation as well.

Figure 1B could be made clearer by adding a rule and using PASS/FAIL for how/if families meet that rule.

This is certainly just a matter of preference, but I found the manuscript hard to digest due to the layout. To get all the information on a particular tool, I have to keep 3 sections of the paper in my head, the intro, the methods, and then the use-cases. This would have been more readable to me with a short intro describing the use-case and more per-tool description in the methods.

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Referee Expertise: Genomics, bioinformatics, algorithms

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.