PSL-GWAS

Release 0.2.0

Jul 18, 2020

Contents:

PSL-GWAS performs genome-wide association studies (GWAS) on microbes using Probabilistic Soft Logic (PSL), a statistical relational programming framework.

A microbial genome-wide association study is a method for finding genetic elements associated with one or more phenotypes of interest. Genetic elements include chromosomal and plasmid genes, SNPs, indels, and copy number variants. Phenotypes of interest could include antibiotic resistance or virulence.

Given some microbial samples, including their sequenced genomes and information on which samples display which phenotypes, PSL-GWAS finds genetic elements likely to cause those phenotypes.

Unlike other microbial GWAS programs, PSL-GWAS implicitly models bacterial population structure and other domain-specific attributes, and can efficiently jointly infer over multiple phenotypes. PSL-GWAS is also easy to customize, allowing users to incorporate domain-specific knowledge in their models.

PSL-GWAS is open source and licensed under the MIT license. The source code is available on Github.

CHAPTER 1

Overview

PSL-GWAS uses variable-length *k*-mers as genetic elements. It first enumerates all *k*-mers present in all samples, and performs an initial filtering step to remove *k*-mer, phenotype pairs not strongly correlated with each other (with a customizable correlation threshold). It also computes a similarity score for each pair of samples.

It then creates a Probabilistic Soft Logic (PSL) model, which outputs a score between 0 and 1 (inclusive) for each *k*-mer, phenotype pair. The PSL model implements a kmowledge graph, where entities (samples, *k*-mers, and phenotypes) are nodes and associations between those entities (samples are similar, sample contains *k*-mer, sample displays phenotype, phenotypes are similar, *k*-mer causes phenotype) are edges. The PSL model efficiently performs convex optimization to find the best satisfying assignment for the '*k*-mer causes phenotype' associations.

The PSL model is composed of first-order logical rules, which are customizable. The base model, which generalizes across phenotypes and species, uses similarity and dissimilarity between samples to model the bacterial population structure without performing PCR, or MDS, and without making any assumptions of independence among entities.

After running the association test, PSL-GWAS ranks the *k*-mer, phenotype pairs by their confidence scores. The highest-scoring *k*-mers can be mapped to an annotated reference genome and/or subjected to further analysis.

CHAPTER 2

Installation

Required dependencies:

- Bash
- Python3
- Java

Recommended dependencies:

- Postgres

We recommend using a python3 virtual environment (or some equivalent like pyenv).

Installation:

```
git clone https://github.com/elip12/psl-gwas.git
cd psl-gwas
./bin/startproject.sh <project>
```

This will create a project directory with the given project name, copy all necessary files into it, and install all necessary python packages.

The project directory has the following structure:

```
<project>/
....gwas.psl            defines PSL model
....gwas.data           defines PSL data files
....parameters.yaml     optional, replaces some command-line params
....data/               holds data files
........raw/            holds raw data files
........preprocessed/   holds preprocessed data files
........postprocessed/  holds postprocessed data files
```

CHAPTER 3

Usage

3.1 Quick Start

```
./bin/run.sh --project <project> --sample <samples.tsv> --pheno <phenos.tsv>
```

3.2 Detailed Usage

The global run script (`bin/run.sh`) is idempotent, meaning the program fails, you can rerun it with the same arguments and it will skip any preprocessing files that already exist. Note that it is possible for the program to fail having written some, but not all data to a file. In this case, you will need to delete the file (in `<project>/data/preprocessed`) so that PSL-GWAS does not skip that file.

PSL-GWAS automatically uses the max number of available threads and 95% of the max memory of the system it runs on.

PSL-GWAS takes in *de novo* assembled contigs or fully sequenced genomes in FASTA (.fa, .fsa) format.

1. **Create a file holding the string IDs and paths to your sample genomes.** This is a tab-separated file with two columns. It must include a header but the column names in the header can be anything. The first column holds unique string IDs for each sample, and the second column holds paths to that sample's FASTA file. The paths are should either be absolute or relative to the root of the repo. So, if you store your sample genomes in a directory called `contigs/` which you put in the `raw/` directory, your sample file would look like this:

ID	Path
s1	<project>/data/raw/contigs/s1_ctgs.fa
s2	<project>/data/raw/contigs/s2_ctgs.fa
s3	<project>/data/raw/contigs/s3_ctgs.fa
...	...

You can call this file anything. `samples.tsv` is simple and easy. This file goes in `<project>/data/raw/`

2. **Create a file holding the string IDs and phenotype values for your samples.** This is a tab-separated file with at least two. It must include a header. The first column of the header can be anything (we recommend "ID"), and the remaining columns of the header should be the names of the phenotypes you are testing. The first column holds unique string IDs for each sample (that must match the ids in the samples file), and the remaining columns hold data on whether each sample displays the ID. When phenotype information is not known, use "NA". Phenotype information can be binary {0, 1} or continuous [0, 1]. 0 means the sample does not display the phenotype, 1 means it displays it with high strength, and intermediate values mean it displays it with weaker strength.

   If you are testing antibiotic resistance, your phenotypes file could look like this:

ID	Penicillin	Ampicillin	...
s1	0	0.75	...
s2	1	NA	...
s3	1	1	...
...	...	...	...

   You can call this file anything. `phenos.tsv` is simple and easy. This file goes in `<project>/data/raw/`

3. Run the gwas from the root of the repo

```
./bin/run.sh --project <project> --sample <samples.tsv> --pheno <phenos.tsv> ...
```

3.3 Options Reference for run.sh

Required flags:

--project PROJECT Name of project, defined with startproject.sh <project>.

--sample SAMPLE Basename of samples file. Ex: `samples.tsv`

--pheno PHENO Basename of phenos file. Ex: `phenos.tsv`

Optional flags:

-d, --debug More verbose logging.

-k K, --k K Kmer length in nucleotide bases. Default: 31

--minkf MINKF Minimum kmer frequency used during preprocessing filtering. Default: 0.01

--maxkf MAXKF Maximum kmer frequency, used during preprocessing filtering. Default: 0.99

--correlation-thresh CORRELATION_THRESH Correlation threshold for filtering in preprocessing. Default: 0.5

-p, --param Ignore k, minkf, maxkf, and correlation-thresh options and use param file in project directory.

--truth TRUTH Fasta file holding truths data for benchmarking or weight learning. Labels correspond to phenos, sequences hold genes or unitigs that cause the phenotype.

--postgres DATABASE Postgres database the default user (usually your username) can access.

--baseline BASELINE Optional baseline data (if you have some prior information on the strength of specific kmer, pheno associations)

--separate-phenos N Separate PSL output into one file per phenotype, taking the best N kmers for each phenotype

--no-consolidate Do not consolidate best N kmers for each phenotype using custom assembler. Defaults to false.

CHAPTER 4

PSL-GWAS Example

We provide example data to allow you to replicate the results of Pandolfo et al. 2020.

1. Ensure postgres is installed and running, and create a new database your default user can access.
2. Clone the repo and cd into it.
3. Download the example data using the `fetch_example` script.
4. Run the GWAS.

```
git clone https://github.com/elip12/psl-gwas.git
cd psl-gwas
./bin/fetch_example.sh
./bin/run.sh --project example --sample samples.tsv --pheno phenos.tsv --truth truths.
↪fa -p --postgres <DATABASE>
```

In `example/data/postprocessed/` you will see one file for each phenotype, holding the highest-ranking *k*-mers.

To evaluate the results, you can use the included evaluation script. The first argument is the path to the truth data, the second is the path to output *k*-mers, and the third is the number of *k*-mers to use when evaluating.

```
python3 psl-gwas/extra/evaluate.py example/data/raw/truths.fa example/data/
↪postprocessed/psl.tobramycin 10
```

This invocation will compute and print the precision, hits, and recirpocal rank of the first hit, as well as printing out the annotations of the specific truth sequences identified.

CHAPTER 5

Customizing Your Model

The base PSL model (defined in `<project>/gwas.psl`) has the following rules:

1. **Contains(S, K) & SamplePheno(S, P) >> KmerPheno(K, P)** If a sample contains a *k*-mer and displays a pheno, there is evidence that *k*-mer causes that pheno.
2. **Contains(S, K) & !SamplePheno(S, P) >> !KmerPheno(K, P)** If a sample contains a *k*-mer and does not display a pheno, there is evidence that *k*-mer does not cause that pheno.
3. **SimilarPheno(P1, P2) & KmerPheno(K, P1) & (P1 != P2) >> KmerPheno(K, P2)** If two phenos are similar, and a *k*-mer causes one of them, there is evidence it causes the other. Note that PSL-GWAS by default only considers two phenos 'similar' if they are very strongly correlated.
4. **!SimilarPheno(P1, P2) & KmerPheno(K, P1) & (P1 != P2) >> !KmerPheno(K, P2) ^2** Two dissimilar phenos are unlikely to be caused by the same kmer.
5. **SimilarSample(S1, S2) & SamplePheno(S1, P) & !SamplePheno(S2, P) & Contains(S1, K) & !Contains(S2, K) & (S1 != S2** If two samples are similar, and one contains a *k*-mer and displays a pheno and the other does neither, there is evidence that *k*-mer causes that pheno. This is more tailored toward finding causal SNPs. An extreme example of this could be two sister bacteria, one of which developed a mutation during mitosis that confers a phenotype.
6. **DissimilarSample(S1, S2) & SamplePheno(S1, P) & SamplePheno(S2, P) & Contains(S1, K) & Contains(S2, K) & (S1 != S** If two samples are dissimilar, and both display a pheno and both contain the same *k*-mer, there is evidence that *k*-mer causes the pheno. This is more tailored toward finding causal genes. An extreme example of this could be two bacteria of two different species that share almost no chromosomal DNA but share a phenotype-causing plasmid.
7. **!DissimilarSample(S1, S2) & SamplePheno(S1, P) & SamplePheno(S2, P) & Contains(S1, K) & Contains(S2, K) & (S1 !=** This is the opposite of the above rule; it serves to lower the confidence scores of kmers that appear in highly related samples.
8. **!KmerPheno(K, P)** Most *k*-mers do not cause phenos.

You can add, delete, or reweight any rules you like. For example, if you suspect phenotypic variation in your data is primarily driven by SNPs, you could increase the weight of rule 5. If you know two phenotypes usually do not occur in the same sample, you could add a rule formalizing that logic. Further, you can replace the data files PSL-GWAS

generates with your own; you could use something like ClonalFrameML to generate a recombination-aware model of the population structure, and use that as the data for the SimilarSample and DissimilarSample rules.

For more information, see PSL rule specification.