MetaSIPSim: A tool for simulating metagenomic-SIP read libraries

The manual

Samuel Barnett
August 8th, 2019
version 0.1.0

Introduction

MetaSIPSim is a tool for simulating metagenomic stable isotope probing (metagenomic-SIP) datasets, including next-generation read libraries. Metagenomic-SIP allows us to link microbial genomes to function within an environmental sample. Metagenomic-SIP improves our ability to recover, assemble, and bin genomes from target, isotopically labeled microbes, compared to conventional shotgun metagenomics.

MetaSIPSim can be used to simulate read libraries for use in testing experimental parameters and in the development of analytical tools for metagenomic-SIP methodologies. This can save on costs and time where preliminary experiments or in vitro mock communities would otherwise be used.

Requirements

MetaSIPSim can be run using linux or unix based operating systems. It was designed using Ubuntu version 16.04.4 and has been tested with macOSX version 10.12.6. MetaSIPSim currently is incompatible with windows.

MetaSIPSim runs with python 2.7 and is currently incompatible with python 3.6. For those familiar with it, we highly recommend using anaconda environments for version control and installing the dependencies.

Installation

Dependencies

- numpy (≥ 1.16.3)
- pandas (≥ 0.24.2)
- Bioptyhon (≥ 1.73)
- scipy (≥ 1.2.1)
- pyfasta (≥ 0.5.2)
- InSilicoSeq (≥ 1.3.6) *Optional for converting reads from fasta to fastq format*

Installing MetaSIPSim

MetaSIPSim can be installed using git. To install, clone the repo (https://github.com/seb369/MetaSIPSim).

```
git clone https://github.com/seb369/MetaSIPSim
```

Running MetaSIPSim

Input files

Before running MetaSIPSim, you need to generate number of input files. Some input files can be generated using functions from SIPSim [1], generated based on empirical evidence, or generated using custom code.

Reference sequences: Reference sequences for MetaSIPSim can be whole genomes, scaffolds, or even contings. Each reference must be in a separate fasta formatted file. In cases where a single reference is actually made up of multiple sequences (*e.g.* genomes with multiple chromosomes or plasmids, draft genomes with multiple scaffolds, or MAGs with multiple contigs) all subs-sequences can be within the same reference file in multifasta format. It is best to keep references in a designated directory as you will input this path to the reference sequences in the final configuration file.

Reference index: A tab delimited file with the first column being the reference ID and the second column being the reference sequence filename.

Chlamydia_suis_SWA-2	GCF_900169085.1_SWA-2_genomic.fna
Xylella_fastidiosa_9a5c	GCF_000006725.1_ASM672v1_genomic.fna
Streptomyces_albireticuli_MDJK11	GCF_002192455.1_ASM219245v1_genomic.fna
Streptococcus_pantholopis_TA_26	GCF_001642085.1_ASM164208v1_genomic.fna

Community composition table: A tab delimited table indicating the relative abundance of each reference in each simulated community/sample. This file can be generated using the `communities` function from SIPSim. This table must include:

- **library:** Sample/community identification number.
- **taxon_name:** Reference identification (same as in reference index).
- **rel_abund_perc:** Relative abundance of the reference in the sample/community. This column should add up to 100 for each library.
- **Rank:** Abundance rank of the reference in the community.

library	taxon_name	rel_abund_perc	rank
1	Chlamydia_suis_SWA-2	75	1
1	Xylella_fastidiosa_9a5c	15	2
1	Streptomyces_albireticuli_MDJK11	7.5	3
1	Streptococcus_pantholopis_TA_26	2.5	4
2	Streptomyces_albireticuli_MDJK11	67	1
2	Xylella_fastidiosa_9a5c	20	2
2	Chlamydia_suis_SWA-2	8	3
2	Streptococcus_pantholopis_TA_26	5	4

Incorporator table: A tab delimited table indicating which references are labeled in which samples/communities. An incorporator is an organism that becomes isotopically labeled. This table must include:

- **taxon_name:** Reference identification (same as in reference index).
- **library:** Sample/community identification number.
- **percent_incorporation:** Mean atom % excess of isotope that this reference is labed with.
- **sd_incorporation:** Standard deviation of the atom % excess of isotope that this reference is labed with.

taxon_name	library	percent_incorporation	sd_incorporation
Xylella_fastidiosa_9a5c	1	95	5
Streptomyces_albireticuli_MDJK11	1	45	5
Xylella_fastidiosa_9a5c	2	80	5
Chlamydia_suis_SWA-2	2	65	5

Fraction BD table: A tab delimited table indicating the buoyant density range of each fraction from simulated gradients. This file can be generated with `gradient_fractions` function from SIPSim or can be generated using empirical data from real gradient fractionations. This table must include:

- **library:** Sample/community identification number.
- **fraction:** Fraction number.
- **BD_min:** Minimum buoyant density of the fraction (g/ml).
- **BD_max:** Maximum buoyant density of the fraction (g/ml).
- **fraction_size:** Size of the fraction (BD_max - BD_min).

library	fraction	BD_min	BD_max	fraction_size
1	1	1.675	1.677	0.002
1	2	1.677	1.680	0.003
1	3	1.680	1.682	0.002
...				
2	1	1.675	1.676	0.001
2	2	1.676	1.679	0.003
2	3	1.679	1.684	0.005

Configuration file

MetaSIPSim has many parameters that must be set that are experiment specific and can vary widely between labs and projects. Due to this complexity we set up the MetaSIPSim scripts to take a configuration file containing all settings and paths to input and output files, rather than arguments. Using a configuration file also allows for one to save the configuration parameters for later reference or to be used again in a later simulation. The same configuration file can be used for both scripts to simulation either metagenomic-SIP or conventional shotgun metagenomic datasets. The configuration file can be generated using the python module ConfigParser. An example configuration file is provided in https://github.com/seb369/MetaSIPSim/example. How to make this file is provided as well. These configuration files can be manipulated by hand in addition to using ConfigParser.

For a metagenomic-SIP simulation, the following parameters must be set. They are grouped based on the ConfigParser section within which they should reside:

Library

library_list: comma separated list of samples/communities to be sequenced (e.g. *1,2,3,4,5,6*).

window_or_fraction: Should the simulation sequence either a single heavy window (*window*) or each fraction (*fraction*).

min_buoyant_density_sequenced: The minimum buoyant density of the heavy window (used only when *window_or_fraction* is set to "*window*").

max_buoyant_density_sequenced: The maximum buoyant density of the heavy window (used only when *window_or_fraction* is set to "*window*").

Fragment

genomeDir: Path to the directory where the fasta files for all reference sequences can be found.

frag_length_distribution: Distribution used to generate random fragment lengths during reference fragmentation. This setting should be comma separated and follows the same scheme as in the SIPSim function *fragments*. Distributions can be:

uniform: *uniform,[low],[scale]*
normal: *normal,[mean],[SD]*
skewed-normal: *skewed-normal,[mean],[SD],[skew]*
truncated-normal: *truncated-normal,[mean],[SD],[low],[high]*

For more information on these distributions check out their documentation at SciPy.org. An example entry for each of these distributions is:

uniform,5000,10000
normal,10000,2000
skewed-normal,10000,1000,-1000
truncated-normal,10000,2000,8000,15000

coverage_of_fragments: Coverage that fragmentation should be repeated to. Each reference will be fragmented repeatedly until this given coverage is reached (*i.e.* number of repeated fragmentations).

temp_fragment_file: Path to a temporary file directory to store the fragments.

genome_index_file: Path to the reference index file (see above).

number_of_iterations: The number of iterations to break simulation into. This parameter is only required when the endpoint is '*fragment_list'*. In most cases iterations should not be used and is only necessary when the amount of memory needed exceeds the amount available. In most cases set this to *1*.

Gradient

temperature: Temperature at which centrifugation will occur (kelvin).

avg_density: Average density of the gradient (g/ml).

angular_velocity: Square of the angular velocity of centrifugation ($(rad/s)^2$).

min_rotation_radius: Minumum distance from axis of rotation to ultracentrifuge tube (cm).

max_rotation_radius: Minumum distance from axis of rotation to ultracentrifuge tube (cm).

tube_angle: Angle of the tube during centrifugation (degrees).

tube_radius: Radius of the ultracentrifuge tube (cm).

tube_height: Height of the ultracentrifuge tube (cm).

fraction_frag_in_DBL: Proportion of DNA that will be found in the diffusive boundary layer rather in the general solution.

isotope: element used for isotopic labeling. Can be either carbon (*C*) or nitrogen (*N*).

Model

min_bouyant_density: minimum buoyant density for the model gradient (g/ml). This should be lesser than the minimum buoyant density of the desired window or the lightest fraction to be sequenced

max_bouyant_density: maximum buoyant density for the model gradient (g/ml). This should be greater than the maximum buoyant density of the desired window or the heaviest fraction to be sequenced

buoyant_density_step: Increase in buoyant density for each model step (g/ml).

fraction_table_file: Path to the fraction BD table (see above).

Community

community_file: Path to the community composition table (see above).

incorporator_file: Path to the incorporator table (see above).

Sequencing

max_read_length: Maximum length of each next-gen sequencing read (base pairs). A common read length for Illumina metagenomic reads is 151.

avg_insert_size: Mean insert size between reads (base pairs). This can be based on the tagmentation step of standard shotgun sequencing protocols.

stddev_instert_size: Standard deviation of the insert size (base pairs).

final_number_of_sequences: Final read depth for sequencing (*i.e.* the number of paired end reads to simulate).

number_of_sequences_per_iteration: Number of sequences to generate per iteration. In most cases iterations are not necessary and are only for saving memory if the amount needed exceeds what is available. In most cases simply set this to *final_number_of_sequences*.

Other

temp_directory: Path to temporary directory to store intermediate files. This can be the same as *temp_fragment_file*.

threads: Number of processors to use for multiprocessing.

logfile: Path to the logfile to store log output.

endpoint: The desired simulation output. This can be:

fragment_list: List of all fragments and their abundances in each simulated window/fraction (see more info below).

Read_list: List of reads recovered in each simulated window/fraction including the identity of their parent reference (see more info below).

Read_sequences: Simulated paired end reads in fasta format. This will generate two fasta files, one for forward and one for reverse reads.

Running scripts

MetaSIPSim contains three scripts that can be run with python.

SIPSim_metagenome.py: Simulates metagenomic-SIP datasets based on input experimental parameters and a simulated community. Output from this script can be a table of DNA fragments with their abundances in a given gradient window, a table of next-gen sequencing reads recovered in a gradient window, or these recovered reads in fasta format.

nonSIP_metagenome.py: Simulates conventional shotgun metagenomic datasets. This can be used to compare metagenomic-SIP methodologies to conventional metagenomics. Input can the identical to that of SIPSim_metagenome.py. Output from this script can be a table of DNA fragments with their abundances in a given gradient window, a table of next-gen sequencing reads recovered in a gradient window, or these recovered reads in fasta format.

fasta2fastq.py: This script converts fasta formatted read libraries generated by SIPSim_metagenome.py and nonSIP_metagenome.py into fastq format with sequencing errors and quality scores. This script relies heavily on InSilicoSeq functions [2].

To run *SIPSim_metagenome.py* or *nonSIP_metagenome.py* you simply run:

```
python PATH_TO_MetaSIPSim/bin/SIPSim_metagenome.py CONFIG_FILE

python PATH_TO_MetaSIPSim/bin/nonSIP_metagenome.py CONFIG_FILE
```

To run *fasta2fastq.py* you run:

```
python PATH_TO_MetaSIPSim/bin/fasta2fastq.py i d e l t p
```

with the following inputs:

- **i:** Input fasta file
- **d:** Indicates whether the input file contains [*forward*] or [*reverse*] reads
- **e:** The error model file. This can be generated using InSilicoSeq or can be one of the ones provided with that toolkit.
- **l:** Length of the read in bp.
- **t:** Path to a temporary directory to store intermediate files in.
- **p:** The number of processors to use.

Output

The output for these scripts depends on the parameter *endpoint.*

fragment_list: Generates a table of fragments, their parent reference and their abundance in the sequenced heavy window/fraction. Each entry in the table is a separate fragment. This output is generated and saved regardless of what *endpoint* might be. The fragment table file is output as a compressed file with the naming scheme:

Metagenomic-SIP (SIPSim_metagenome.py)
Heavy window simulation: **library_*X*_window_*minBD-maxBD*_fragments.txt.gz**
Fractions simulation: **library_*X*_fraction_*Y*_fragments.txt.gz**

Shotgun metagenomics (nonSIP_metagenome.py)
nonSIP_library_*X*_fragments.txt.gz

Output table columns are:

- **taxon_name:** Reference identification (same as in reference index).
- **scaffoldID:** Name of sequence within reference fasta file. This is important when reference is a multi-fasta file with multiple scaffolds.
- **library:** Sample/community identification number.
- **OriBD:** Original estimated mean buoyant density of the fragment (g/ml).
- **percent_incorp:** Atom % excess of isotope for the fragment.
- **abundance:** Abundance of the fragment in the sequenced window or fraction. To get relative abundance of the fragment, divide this value by the sum of this column.
- **fragment_start:** Starting point of the fragment on the scaffold of the reference sequence.
- **fragment_length:** Length of the fragment (base pairs)

Read_list: Generates a table of reads and their parent reference and fragment recovered in the sequenced window/fraction. Each entry in the table is a separate read. The read table file is output as a compressed file with the naming scheme:

Metagenomic-SIP (SIPSim_metagenome.py)
Heavy window simulation: **library_*X*_window_*minBD-maxBD*_reads.txt.gz**
Fractions simulation: **library_*X*_fraction_*Y*_reads.txt.gz**

Shotgun metagenomics (nonSIP_metagenome.py)
nonSIP_library_*X*_reads.txt.gz

Output table columns are:

- **taxon_name:** Reference identification (same as in reference index).
- **scaffoldID:** Name of sequence within reference fasta file. This is important when reference is a multi-fasta file with multiple scaffolds.
- **library:** Sample/community identification number.
- **OriBD:** Original estimated mean buoyant density of the parent fragment (g/ml).
- **percent_incorp:** Atom % excess of isotope for the parent fragment.
- **fragment_start:** Starting point of the parent fragment on the scaffold of the reference sequence.
- **fragment_length:** Length of the fragment (base pairs).
- **read:** Read identification number.
- **forward_start:** Starting point of the forward read on the scaffold of the reference sequence.
- **reverse_start:** Starting point of the reverse read on the scaffold of the reference sequence.
- **read_length:** Length of the read (base pairs)

Read_sequences: Generates paired end sequencing reads in fasta format. Forward and reverse reads are in separate files. Forward and reverse read names follow standard Illumina naming scheme, matching between forward and reverse. The read sequence files are compressed and named with the following scheme:

Metagenomic-SIP (SIPSim_metagenome.py)
Forward
Heavy window simulation: **library_*X*_window_*minBD-maxBD*_reads_f.fasta.gz**
Fractions simulation: **library_*X*_fraction_*Y*_reads_f.fasta.gz**
Reverse
Heavy window simulation: **library_*X*_window_*minBD-maxBD*_reads_r.fasta.gz**
Fractions simulation: **library_*X*_fraction_*Y*_reads_r.fasta.gz**

Shotgun metagenomics (nonSIP_metagenome.py)
Forward: **nonSIP_library_*X*_reads_f_fasta.gz**
Reverse: **nonSIP_library_*X*_reads_r_fasta.gz**

References

1. Youngblut ND, Barnett SE, Buckley DH. SIPSim: A modeling toolkit to predict accuracy and aid design of DNA-SIP experiments. Front Microbiol 2018;9:570. doi:10.3389/fmicb.2018.00570

2. Gourlé H, Karlsson-Lindsjö O, Hayer J, Bongcam-Rudloff E. Simulating Illumina metagenomic data with InSilicoSeq. Bioinformatics 2019;35(3):521–2. doi:10.1093/bioinformatics/bty630