InSilicoSeq Documentation

Release 1.5.4

Hadrien Gourlé

Sep 29, 2021

Contents

InSilicoSeq is a sequencing simulator producing realistic Illumina reads. Primarily intended for simulating metagenomic samples, it can also be used to produce sequencing data from a single genome.

InSilicoSeq is written in python, and use kernel density estimators to model the read quality of real sequencing data.

InSilicoSeq supports substitution, insertion, deletion errors, and models gc bias and insert size distribution.

CHAPTER 1

Details

- **Authors:** Hadrien Gourlé, Juliette Hayer, Oskar E. Karlsson and Erik Bongcam-Rudloff
- **Contact:** hadrien.gourle@slu.se

- **GitHub:** HadrienG/InSilicoSeq
- **License:** MIT

- **Article:** 10.1093/bioinformatics/bty630

CHAPTER 2

Contents

2.1 Installing InSilicoSeq

2.1.1 Using pip

InSilicoSeq requires python>=3.5. To install InSilicoSeq, type the following in your terminal:

```
pip install InSilicoSeq
```

It will install InSilicoSeq as well as the following dependencies:

- biopython
- joblib
- numpy
- pysam
- scipy

Other installation options

- With conda:

```
conda install -c bioconda insilicoseq
```

- Upgrading InSilicoSeq to the latest version:

```
pip install --upgrade InSilicoSeq
```

- If you don't have administration rights on your machine:

```
pip3 install --user InSilicoSeq
```

- if you wish to install InSilicoSeq at a custom location (i.e with a module system):

```
prefix="/path/to/install/prefix"
pip install --install-option="--prefix=/$prefix" InSilicoSeq
```

then add `$prefix/bin` to your `PATH`, and `$prefix/lib/python3.X/site-packages` to your `PYTHONPATH` (replacing python3.X with your python version)

2.1.2 Using docker

If you wish to use InSilicoSeq using docker

```
docker pull hadrieng/insilicoseq:latest
```

To use InSilicoSeq with docker, you need to provide a *volume* to the `docker run` command. Given with the `-v` option, the volume is your way of exchanging data (in this case, your input and output files) with the docker container.

```
docker run -v /Users/hadrien/data:/mnt/data -it --rm \
    hadrieng/insilicoseq iss generate \
    --genomes /mnt/data/genomes.fasta -m miseq \
    -o /mnt/data/reads
```

The above command will mount the local folder `/Users/hadrien/data` onto `/mnt/data` on the docker side. The output reads will be located in `/Users/hadrien/data` when InSilicoSeq has finished running.

2.2 Generating reads

InSilicoSeq comes with a set of pre-computed error models to allow the user to easily generate reads from the most popular Illumina instruments:

- HiSeq
- MiSeq
- NovaSeq

Per example generate 1 million MiSeq reads from a set of input genomes:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate --genomes SRS121011.fasta --model miseq --output miseq_reads
```

This will create 2 fastq files, *miseq_reads_R1.fastq* and *miseq_reads_R2.fastq* in your current directory, as well as *miseq_reads_abundance.txt*, a tab-delimited file containing the abundance of each genomes.

InSilicoSeq will use 2 cpus by default. For multithreading, use `--cpus`:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate --cpus 8 --genomes SRS121011.fasta --model hiseq --output hiseq_reads
```

If you have created your custom model, give to `--model` the path of your custom model file:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate --genomes SRS121011.fasta --model model.npz --output model_reads
```

If your multi-fasta file contain more genomes than the number of organisms for which you wish to simulate reads, you can use the `--n_genomes`/`-u` parameter:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate --genomes SRS121011.fasta --n_genomes 5 --model novaseq --output novaseq_
↪reads
```

The above command will pick 5 random genomes in your multi-fasta and generate reads from them.

You can also provide multiple input files:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
curl -O -J -L https://osf.io/37kg8/download  # download another example file
iss generate --genomes SRS121011.fasta minigut.fasta --n_genomes 5 --model novaseq --
↪output novaseq_reads
```

2.2.1 Draft genomes

InSilicoseq's `--genomes` option assumes complete genomes in multifasta format. That is, each record in fasta files passed to the `--genomes` option is treated as a different genome If you have draft genome files containing contigs, you can give them to the `--draft` option:

```
# input file not provided in this example
iss generate --draft my_draft_genome.fasta --model novaseq --output novaseq_reads
```

Or if you have more than one draft:

```
# input file not provided in this example
iss generate --draft draft1.fasta draft2.fasta draft3.fasta --model novaseq --output␣
↪novaseq_reads
```

You can also combine your drafts with complete genomes:

```
# input file not provided in this example
iss generate -g complete_genomes.fasta --draft draft.fasta --model novaseq --output␣
↪novaseq_reads
```

2.2.2 Required input files

By default, InSilicoSeq only requires 1 file in order to start generating reads: 1 (multi-)fasta files containing your input genome(s).

If you don't want to use a multi-fasta file or don't have one at hand but are equipped with an Internet connection, you can download random genomes from the ncbi with the `--ncbi`/`-k` parameter:

```
iss generate --ncbi bacteria -u 10 --model miseq --output miseq_ncbi
```

The above command will generate reads from 10 random bacterial genomes from the NCBI

Additionally, you can supply tab separated kingdoms if you wish to have mixed datasets:

```
iss generate -k bacteria viruses -u 10 4 --model miseq --output miseq_ncbi
```

The above command will generate reads from 10 random bacteria and 4 random viruses. `--ncbi`/`-k` accepts the following values: `bacteria`, `viruses` and `archaea`.

In addition the the 2 fastq files and the abundance file, the downloaded genomes will be saved in *miseq_ncbi_genomes.fasta* in your current directory.

The `--ncbi` is compatible with `--draft` and `--genomes` so you can combine the 3 options.

2.2.3 Abundance distribution

With default settings, the abundance of the input genomes is drawn from a log-normal distribution.

Alternatively, you can use other distributions with the `--abundance` parameter: *uniform*, *halfnormal*, *exponential* or *zero-inflated-lognormal*

If you wish to fine-tune the distribution of your genomes, InSilicoSeq also accepts an abundance file:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate -g SRS121011.fasta --abundance_file abundance.txt -m HiSeq -o HiSeq_reads
```

Example abundance file for a multi-fasta containing 2 genomes: genome_A and genome_B.

```
genome_A    0.2
genome_B    0.8
```

For the abundance to make sense, the total abundance in your abundance file must equal 1.

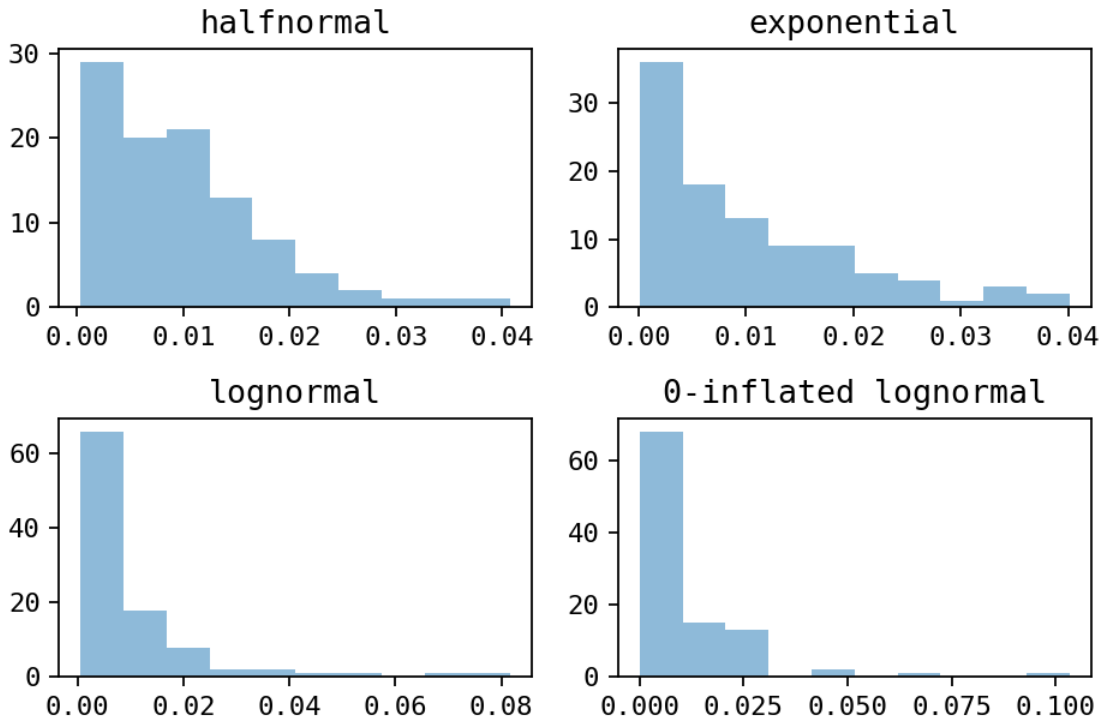

Fig. 1: Histograms of the different distribution (drawn with 100 samples)

2.2.4 Coverage distribution

In the context of InSilicoSeq, the *abundance* is the proportion of reads in a sample, which since it does not acount for the length of the genome, does not necessarily reflect the number of organisms present in a sample.

The `coverage` and `coverage_file` options allow for simulating reads according to a coverage distribution instead of abundance.

```
iss generate --ncbi bacteria -U 50 --coverage lognormal -n 25M \
    --model novaseq --output reads
```

The `coverage_file` option works similarly to the `abundance_file` option. For two genomes A and B:

```
iss generate --genomes genomes.fasta --coverage_file coverage.txt \
    --model novaseq --output reads
```

with, for a coverage of 20x for genome_A and 100x for genome_B, the coverage file *coverage.txt* will be:

```
genome_A    20
genome_B    100
```

2.2.5 GC bias

InSilicoSeq can also model gc bias:

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate -g SRS121011.fasta --model miseq --gc_bias --output reads
```

2.2.6 Basic error model

By default InSilicoSeq uses Kernel Density Estimators for generating reads. Both the pre-built models (miseq, hiseq and novaseq), as well as the model files you build yourselves are that way.

If you wish to use a much simpler model (because you don't have the need for insertions and deletion errors per example), you can use `--mode basic`

```
curl -O -J -L https://osf.io/thser/download  # download the example data
iss generate -g SRS121011.fasta --mode basic --output basic_reads
```

2.2.7 Full list of options

–genomes

Input genome(s) from where the reads will originate

–draft

Input draft genome(s) from where the reads will originate

–ncbi

Download input genomes from RefSeq instead of using –genomes. Requires –n_genomes option. Can be bacteria, viruses, archaea or a combination of the three (space-separated)

–n_genomes

How many genomes will be downloaded from the ncbi. Required if –ncbi is set. If more than one kingdom is set with –ncbi, multiple values are necessary (space-separated).

–abundance

Abundance distribution (default: lognormal). Can be uniform, halfnormal, exponential, lognormal or zero_inflated_lognormal.

–abundance_file

Abundance file for coverage calculations (default: None).

–coverage

coverage distribution. Can be uniform, halfnormal, exponential, lognormal or zero-inflated-lognormal.

–coverage_file

file containing coverage information (default: None).

–n_reads

Number of reads to generate (default: 1000000). Allows suffixes k, K, m, M, g and G (ex 0.5M for 500000).

–mode

Error model. If not specified, using kernel density estimation (default: kde). Can be 'kde' or 'basic'

–model

Error model file. (default: None). Use HiSeq, NovaSeq or MiSeq for a pre-computed error model provided with the software, or a file generated with iss model. If you do not wish to use a model, use –mode basic. The name of the built-in models is case insensitive.

–gc_bias

If set, may fail to sequence reads with abnormal GC content. Does not guarantee –n_reads (default: False)

–cpus

Number of cpus to use. (default: 2).

–seed

Seed all the random number generators

–quiet

Disable info logging

–debug

Enable debug logging

–output

Output file path and prefix (Required)

–compress

Compress the output in gzip format (default: False).

2.3 Error models

2.3.1 Pre-built models

Available models

Model name	Read length
MiSeq	300 bp
HiSeq	125 bp
NovaSeq	150 bp

Qualities

The prebuilt models were built with the following commands:

```
megahit -1 reads_R1.fastq -2 reads_R2.fastq -o asm
bowtie2-build asm/final.contigs.fa miseq_asm/final.contigs
bowtie2 -x asm/final.contigs -1 reads_R1.fastq \
    -2 reads_R2.fastq | samtools view -bS | samtools sort -o mapping.bam
samtools index mapping.bam
iss model -b mapping.bam -o name_of_model
```

The MiSeq model was built from an ocean metagenomics project in Kenya (sample accession number ERR1912174).

The HiSeq and NovaSeq models were built from human run obtained via basespace.

2.3.2 Creating an Error Model

If you do not wish to use the pre-computed error models provided with InSilicoSeq, it is possible to create your own.

InSilicoSeq creates error models from .bam files. The input bam file should be a set of reads aligned against a reference genome or metagenome.

Given you have two read files, *reads_R1.fastq.gz* and‘reads_R2.fastq.gz‘, and a reference metagenome *ref.fasta*:

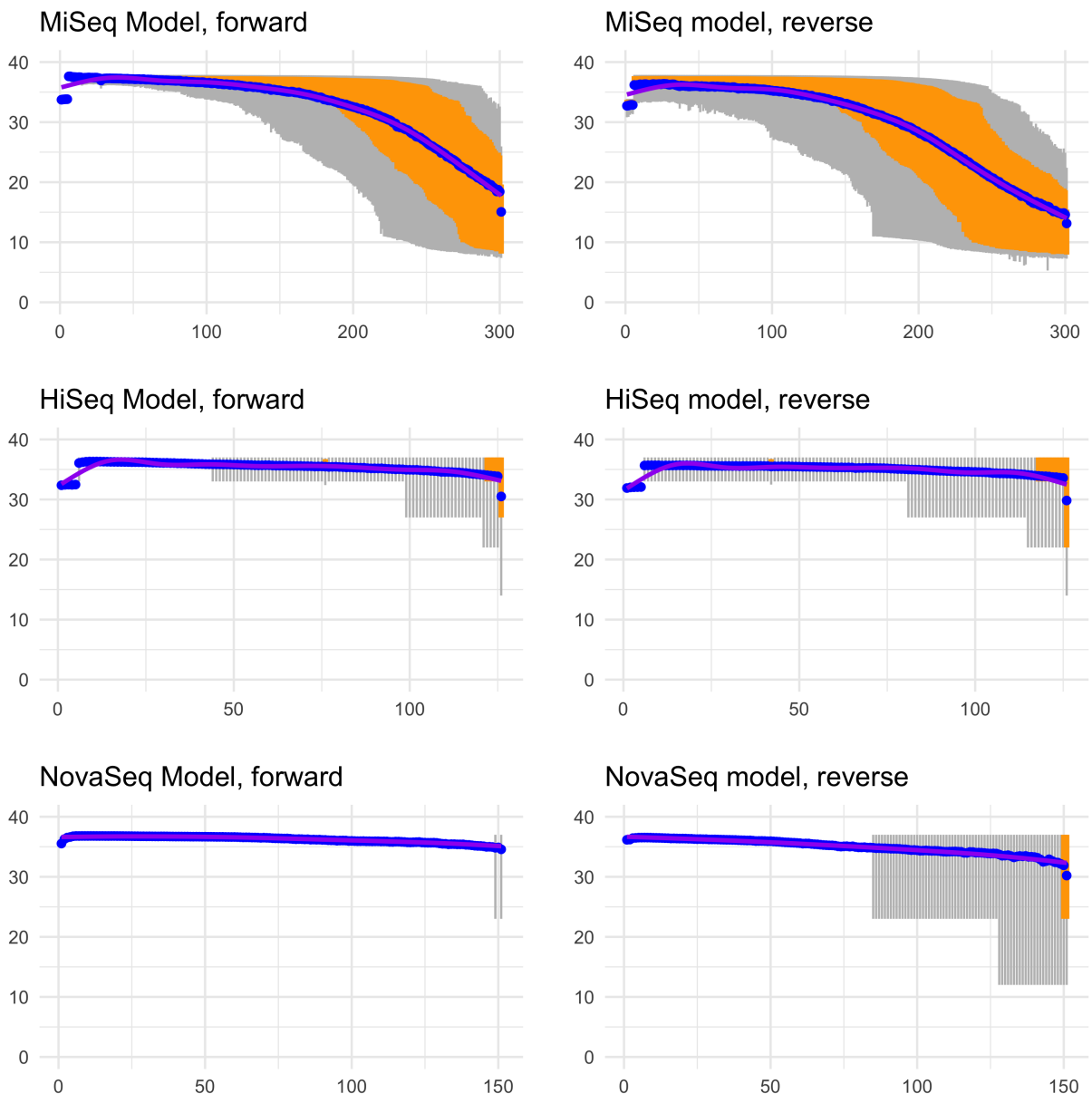

Fig. 2: Average per base quality profiles for the prebuilt error models

Align you reads against the reference

```
bowtie2-build ref.fasta ref
bowtie2 -x ref -1 reads_R1.fastq.gz \
    -2 reads_R2.fastq.gz | samtools view -bS | samtools sort -o ref.bam
samtools index ref.bam
```

Build the model

```
iss model -b ref.bam -o my_model
```

which will create a *my_model.npz* file containing your newly built model

2.3.3 Full list of options

–bam

aligned reads from which the model will be inferred (Required)

–model

Error model to build. If not specified, using kernel density estimation (default: kde). Can be 'kde' or 'cdf'

–output

Output file path and prefix (Required)

–quiet

Disable info logging

–debug

Enable debug logging

2.4 iss package

2.4.1 Subpackages

iss.error_models package

Submodules

iss.error_models.basic module

class iss.error_models.basic.**BasicErrorModel**
Bases: *iss.error_models.ErrorModel*

Basic Error Model class

Basic error model. The phred scores are based on a normal distribution. Only substitutions errors occur. The substitution rate is assumed equal between all nucleotides.

gen_phred_scores(*mean_quality*, *orientation*)
Generate a normal distribution, transform to phred scores

Generate a list of phred score according to a normal distribution centered around the ErrorModel quality

Parameters **mean_quality** (*int*) – mean phred score

Returns list of phred scores following a normal distribution

Return type list

random_insert_size()
Fake random function returning the default insert size of the basic arror model

Returns insert size

Return type int

iss.error_models.cdf module

iss.error_models.kde module

class iss.error_models.kde.**KDErrorModel**(*npz_path*)
Bases: *iss.error_models.ErrorModel*

KDErrorModel class.

Error model based on an .npz files derived from read alignments. the npz file must contain:

- the length of the reads
- the mean insert size
- the size of mean sequence quality bins (for R1 and R2)
- **a cumulative distribution function of quality scores for each position** (for R1 and R2)
- the substitution for each nucleotide at each position (for R1 and R2)
- the insertion and deletion rates for each position (for R1 and R2)

gen_phred_scores(*cdfs*, *orientation*)
Generate a list of phred scores based on cdfs and mean bins

For each position, draw a phred score from the cdf and append to the phred score list

Parameters

- **cdfs** (*ndarray*) – array containing the cdfs
- **orientation** (*string*) – orientation of the read. Can be 'forward' or 'reverse'

Returns a list of phred scores

Return type list

random_insert_size()
Draw a random insert size from the insert size cdf

Parameters **i_size_cdf** – cumulative distribution function of the insert size

Returns an insert size

Return type int

Module contents

class iss.error_models.**ErrorModel**
Bases: object

Main ErrorModel Class

This class is used to create inheriting classes and contains all the functions that are shared by all ErrorModel classes

adjust_seq_length(*mut_seq*, *orientation*, *full_sequence*, *bounds*)
Truncate or Extend reads to make them fit the read length

When insertions or deletions are introduced to the reads, their length will change. This function takes a (mutable) read and a reference sequence, and extend or truncate the read if it has had an insertion or a deletion

Parameters

- **mut_seq** (*MutableSeq*) – a mutable sequence
- **orientation** (*string*) – orientation of the read. Can be 'forward' or 'reverse'
- **full_sequence** (*Seq*) – the reference sequence from which mut_seq comes from
- **bounds** (*tuple*) – the position of the read in the full_sequence

Returns a sequence fitting the ErrorModel

Return type Seq

introduce_error_scores(*record*, *orientation*)
Add phred scores to a SeqRecord according to the error_model

Parameters

- **record** (*SeqRecord*) – a read record
- **orientation** (*string*) – orientation of the read. Can be 'forward' or 'reverse'

Returns a read record with error scores

Return type SeqRecord

introduce_indels(*record*, *orientation*, *full_seq*, *bounds*)
Introduce insertions or deletions in a sequence

Introduce insertion and deletion errors according to the probabilities present in the indel choices list

Parameters

- **record** (*SeqRecord*) – a sequence record
- **orientation** (*string*) – orientation of the read. Can be 'forward' or 'reverse'
- **full_seq** (*Seq*) – the reference sequence from which mut_seq comes from

- **bounds** (*tuple*) – the position of the read in the full_sequence

Returns a sequence with (eventually) indels

Return type Seq

load_npz(*npz_path*, *model*)
load the error profile .npz file

Parameters

- **npz_path** (*string*) – path to the npz file
- **model** (*string*) – type of model. Could be 'cdf' or 'kde'. 'cdf' has been deprecated and is no longer available

Returns

numpy object containg variables necessary for error model construction

Return type ndarray

logger

mut_sequence(*record*, *orientation*)
Introduce substitution errors to a sequence

If a random probability is higher than the probability of the basecall being correct, introduce a substitution error

Parameters

- **record** (*SeqRecord*) – a read record with error scores
- **orientation** (*string*) – orientation of the read. Can be 'forward' or 'reverse'

Returns a sequence

Return type Seq

2.4.2 Submodules

2.4.3 iss.abundance module

iss.abundance.**coverage_scaling**(*total_n_reads*, *abundance_dic*, *genome_file*, *read_length*)
Scale coverage distribution according to the n_reads parameter

Parameters

- **total_n_reads** (*int*) – total amount of reads to simulate
- **abundance_dic** (*dict*) – a dictionary with records as keys, coverage as values
- **genome_file** (*str*) – path to input fasta file containing genomes
- **read_length** (*int*) – length of the reads in the dataset

Returns scaled coverage dictionary

Return type dict

iss.abundance.**draft**(*genomes*, *draft*, *distribution*, *output*, *mode='abundance'*)
Computes the abundance dictionary for a mix of complete and draft genomes

Parameters

- **genomes** (*list*) – list of all input records
- **draft** (*list*) – draft genome files
- **distribution** (*function*) – distribution function to use
- **output** (*str*) – output file

Returns the abundance dictionary

Return type dict

iss.abundance.**expand_draft_abundance**(*abundance_dic*, *draft*, *mode='abundance'*)
Calculate abundance for each contig of a draft genome The function takes the abundance dictionary and automatically detects draft genomes. In coverage mode the function simply assign the coverage value to each contig

Parameters

- **abundance_dic** (*dict*) – dict with genome (paths or id) as key and abundance as value
- **draft** (*list*) – draft genome files
- **mode** (*str*) – abundance or coverage

Returns abundance dictionary with abundance value for each contig

Return type dict

iss.abundance.**exponential**(*record_list*)

Generate scaled exponential abundance distribution from a number of records

Parameters **record_list** (*list*) – a list of record.id

Returns a dictionary with records as keys, abundance as values

Return type dict

iss.abundance.**halfnormal**(*record_list*)

Generate scaled halfnormal abundance distribution from a number of records

Parameters **record_list** (*list*) – a list of record.id

Returns a dictionary with records as keys, abundance as values

Return type dict

iss.abundance.**lognormal**(*record_list*)

Generate scaled lognormal abundance distribution from a number of records

Parameters **record_list** (*list*) – a list of record.id

Returns a dictionary with records as keys, abundance as values

Return type dict

iss.abundance.**parse_abundance_file**(*abundance_file*)
Parse an abundance or coverage file

The abundance/coverage file is a flat file of the format "genome_id<TAB>abundance" or "genome_id<TAB>coverage"

Parameters **abundance_file** (*string*) – the path to the abundance file

Returns genome_id as keys, abundance as values

Return type dict

iss.abundance.**to_coverage**(*total_n_reads*, *species_abundance*, *read_length*, *genome_size*)
Calculate the coverage of a genome in a metagenome given its size and abundance

Parameters

- **total_n_reads** (*int*) – total amount of reads in the dataset
- **species_abundance** (*float*) – abundance of the species, between 0 and 1
- **read_length** (*int*) – length of the reads in the dataset
- **genome_size** (*int*) – size of the genome

Returns coverage of the genome

Return type float

iss.abundance.**to_file**(*abundance_dic*, *output*, *mode='abundance'*)
Write the abundance dictionary to a file

Parameters

- **abundance_dic** (*dict*) – the abundance dictionary
- **output** (*str*) – the output file name

iss.abundance.**uniform**(*record_list*)
Generate uniform abundance distribution from a number of records

Parameters **record_list** (*list*) – a list of record.id

Returns a dictionary with records as keys, abundance as values

Return type dict

iss.abundance.**zero_inflated_lognormal**(*record_list*)

Generate scaled zero-inflated lognormal abundance distribution from a number of records

Parameters **record_list** (*list*) – a list of record.id

Returns a dictionary with records as keys, abundance as values

Return type dict

2.4.4 iss.app module

iss.app.**generate_reads**(*args*)
Main function for the *iss generate* submodule

This submodule generates reads from an ErrorModel and write them to args.output + _R(1|2).fastq

Parameters **args** (*object*) – the command-line arguments from argparse

iss.app.**main**()

iss.app.**model_from_bam**(*args*)
Main function for the *iss model* submodule

This submodule write all variables necessary for building an ErrorModel to args.output + .npz

Parameters **args** (*object*) – the command-line arguments from argparse

2.4.5 iss.bam module

iss.bam.**random**() → x in the interval [0, 1).

iss.bam.**read_bam**(*bam_file*, *n_reads=1000000*)
Bam file reader. Select random mapped reads from a bam file

Parameters **bam_file** (*string*) – path to a bam file

Yields *read* – a pysam read object

iss.bam.**to_model**(*bam_path*, *output*)
from a bam file, write all variables needed for modelling reads in a .npz model file

For a brief description of the variables that will be written to the output file, see the bam.write_to_file function

Parameters

- **bam_path** (*string*) – path to a bam file
- **output** (*string*) – prefix of the output file

iss.bam.**write_to_file**(*model*, *read_length*, *mean_f*, *mean_r*, *hist_f*, *hist_r*, *sub_f*, *sub_r*, *ins_f*, *ins_r*, *del_f*, *del_r*, *i_size*, *output*)
Write variables to a .npz file

Parameters

- **model** (*string*) – the type of error model
- **read_length** (*int*) – read length of the dataset
- **mean_f** (*list*) – list of mean bin sizes
- **mean_r** (*list*) – list of mean bin sizes
- **hist_f** (*list*) – list of cumulative distribution functions for the forward read quality
- **hist_r** (*list*) – list of cumulative distribution functions for the reverse read quality
- **sub_f** (*list*) – list of dictionaries representing the substitution probabilities for the forward reads
- **sub_r** (*list*) – list of dictionaries representing the substitution probabilities for the reverse reads
- **ins_f** (*list*) – list of dictionaries representing the insertion probabilities for the forward reads
- **ins_r** (*list*) – list of dictionaries representing the insertion probabilities for the reverse reads
- **del_f** (*list*) – list of dictionaries representing the deletion probabilities for the forward reads
- **del_r** (*list*) – list of dictionaries representing the deletion probabilities for the reverse reads
- **i_size** (*int*) – distribution of insert size for the aligned reads
- **output** (*string*) – prefix of the output file

2.4.6 iss.generator module

iss.generator.**reads**(*record*, *ErrorModel*, *n_pairs*, *cpu_number*, *output*, *seed*, *gc_bias=False*, *mode='default'*)

Simulate reads from one genome (or sequence) according to an ErrorModel

This function makes use of the *simulate_read* function to simulate reads and save them in a fastq file

Parameters

- **record** (*SeqRecord*) – sequence or genome of reference
- **ErrorModel** (ErrorModel) – an ErrorModel
- **n_pairs** (*int*) – the number of reads to generate
- **cpu_number** (*int*) – an int indentifying the cpu that is used by the function. Is used for naming the output file
- **output** (*str*) – the output file prefix
- **seed** (*int*) – random seed to use
- **gc_bias** (*bool*) – if set, the function may skip a read due to abnormal GC content

Returns the name of the output file

Return type str

iss.generator.**simulate_read**(*record*, *ErrorModel*, *i*, *cpu_number*)

From a read pair from one genome (or sequence) according to an ErrorModel

Each read is a SeqRecord object returns a tuple containing the forward and reverse read.

Parameters

- **record** (*SeqRecord*) – sequence or genome of reference
- **ErrorModel** (ErrorModel) – an ErrorModel class
- **i** (*int*) – a number identifying the read
- **cpu_number** (*int*) – cpu number. Is added to the read id.

Returns tuple containg a forward read and a reverse read

Return type tuple

iss.generator.**to_fastq**(*generator*, *output*)

Write reads to a fastq file

Take a generator or a list containing read pairs (tuples) and write them in two fastq files: output_R1.fastq and output_R2.fastq

Parameters

- **generator** (*generator*) – a read generator (or list)
- **output** (*string*) – the output files prefix

2.4.7 iss.modeller module

iss.modeller.**dispatch_indels**(*read*)

Return the x and y position of a insertion or deletion to be inserted in the indel matrix.

The substitution matrix is a 2D array of size 301 * 9 The x axis (301) corresponds to the position in the read, while the y axis (9) represents the match or indel (see the dispatch dict in the function). Positions 0 is match or substitution, other positions in 'N1' are insertions, 'N2 are deletions'

The size of x axis is 301 because we haven't calculated the read length yet

Parameters **read** (*read*) – an aligned read object

Yields *tuple* – a tuple with the x, y position for dispatching the indel in the indel matrix

iss.modeller.**dispatch_subst**(*base*, *read*, *read_has_indels*)
Return the x and y position of a substitution to be inserted in the substitution matrix.

The substitution matrix is a 2D array of size 301 * 16 The x axis (301) corresponds to the position in the read, while the y axis (16) represents the match or substitution (see the dispatch dict in the function). Positions 0, 4, 8 and 12 are matches, other positions are substitutions

The size of x axis is 301 because we haven't calculated the read length yet

Parameters

- **base** (*tuple*) – one base from an aligmnent object. According to the pysam documentation: an alignment is a list of tuples: aligned read (query) and reference positions. the parameter with_seq adds the ref sequence as the 3rd element of the tuples. substitutions are lower-case.
- **read** (*read*) – a read object, from which the alignment comes from
- **read_has_indels** (*bool*) – a boolean flag to keep track if the read has an indel or not

Returns x and y position for incrementing the substitution matrix and a third element: True if an indel has been detected, False otherwise

Return type tuple

iss.modeller.**divide_qualities_into_bins**(*qualities*, *n_bins=4*)
Divides the raw quality scores in bins according to the mean phred quality of the sequence they come from

Parameters

- **qualities** (*list*) – raw count of all the phred scores and mean sequence quality
- **n_bins** (*int*) – number of bins to create (default: 4)

Returns a list of lists containing the binned quality scores

Return type list

iss.modeller.**indel_matrix_to_choices**(*indel_matrix*, *read_length*)
Transform an indel matrix into probabilties of indels for at every position

From the raw indel count at one position, returns a dictionary with probabilties of indel

Parameters

- **indel_matrix** (*np.array*) – the substitution matrix is a 2D array of size read_length * 16. fhe x axis (read_length) corresponds to the position in the read, while the y axis (9) represents the match or indel. Positions 0 is match or substitution, other positions in 'N1' are insertions, 'N2 are deletions'
- **read_length** (*int*) – read length

Returns tuple containing two lists of dictionaries representing the insertion or deletion probabilities for a collection of reads

Return type tuple

iss.modeller.**insert_size**(*insert_size_distribution*)
Calculate cumulative distribution function from the raw insert size distributin. Uses 1D kernel density estimation.

Parameters

- **insert_size_distribution** (*list*) – list of insert sizes from aligned
- **pairs** (*read*) –

Returns a cumulative density function

Return type 1darray

iss.modeller.**quality_bins_to_histogram**(*bin_lists*)
Wrapper function to generate cdfs for each quality bins

Generate cumulative distribution functions for a number of mean quality bins

Parameters

- **bins_lists** (*list*) – list of list containing raw count of all phred
- **scores** –

Returns a list of lists containg cumulative density functions

Return type list

iss.modeller.**raw_qualities_to_histogram**(*qualities*)
Approximate the distribution of base quality at each position in a read using a pseudo 2d kernel density estimation

Generate cumulative distribution functions

Parameters **qualities** (*list*) – raw count of all phred scores

Returns

list of cumulative distribution functions. One cdf per base. The list has the size of the read length

Return type list

iss.modeller.**subst_matrix_to_choices**(*substitution_matrix*, *read_length*)
Transform a substitution matrix into probabilties of substitutions for each base and at every position

From the raw mismatches at one position, returns a dictionary with probabilties of substitutions

Parameters

- **substitution_matrix** (*np.array*) – the substitution matrix is a 2D array of size read_length * 16. fhe x axis (read_length) corresponds to the position in the read, while the y axis (16) represents the match or substitution. Positions 0, 4, 8 and 12 are matches, other positions are substitutions
- **read_length** (*int*) – read length

Returns

list of dictionaries representing the substitution probabilities for a collection of reads

Return type list

2.4.8 iss.util module

iss.util.**cleanup**(*file_list*)
remove temporary files

Parameters **file_list** (*list*) – a list of files to be removed

iss.util.**compress**(*filename*, *remove=True*)
gzip a file

Parameters **filename** (*string*) – name of file to be compressed

iss.util.**concatenate**(*file_list*, *output*)
Concatenate files together

Parameters

- **file_list** (*list*) – the list of input files (can be a generator)
- **output** (*string*) – the output file name

iss.util.**convert_n_reads**(*unit*)
For strings representing a number of bases and ending with k, K, m, M, g, and G converts to a plain old number

Parameters **n** (*str*) – a string representing a number ending with a suffix

Returns a number of reads

Return type float

iss.util.**count_records**(*fasta_file*)
Count the number of records in a fasta file and return a list of recods id

Parameters **fasta_file** (*string*) – the path to a fasta file

Returns a list of record ids

Return type list

iss.util.**dump**(*object*, *output*)
dump an object, like pickle.dump. This function uses pickle.dumps to dump large objects

Parameters **object** (*object*) – a python object

iss.util.**genome_file_exists**(*filename*)
Checks if the output file from the –ncbi option already exists

Parameters **filename** (*str*) – a file name

iss.util.**load**(*filename*)
load a pickle from disk This function uses pickle.loads to load large objects

Parameters **filename** (*string*) – the path of the pickle to load

iss.util.**nplog**(*type*, *flag*)

iss.util.**phred_to_prob**(*q*)
Convert a phred score (Sanger or modern Illumina) in probabilty

Given a phred score q, return the probabilty p of the call being right

Parameters **q** (*int*) – phred score

Returns probabilty of basecall being right

Return type float

iss.util.**prob_to_phred**(*p*)
Convert a probabilty into a phred score (Sanger or modern Illumina)

Given a probabilty p of the basecall being right, return the phred score q

Parameters **p** (*int*) – probabilty of basecall being right

Returns phred score

Return type int

iss.util.**reservoir**(*records*, *record_list*, *n=None*)
yield a number of records from a fasta file using reservoir sampling

Parameters **records** (*obj*) – fasta records from SeqIO.parse

Yields *record (obj)* – a fasta record

iss.util.**rev_comp**(*s*)
A simple reverse complement implementation working on strings

Parameters **s** (*string*) – a DNA sequence (IUPAC, can be ambiguous)

Returns reverse complement of the input sequence

Return type list

iss.util.**split_list**(*l*, *n_parts=1*)
Split a list in a number of parts

Parameters

- **l** (*list*) – a list
- **n_parts** (*in*) – the number of parts to split the list in

Returns a list of n_parts lists

Return type list

2.4.9 Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

This documentation was generated on 2021-09-29 *at* 08:55.

Python Module Index

i

Index

P

Q

R

S

T

U

W

Z