# InSilicoSeq Documentation

***Release 0.8.1***

**Hadrien Gourlé**

**Nov 20, 2017**

# Contents

InSilicoSeq (or iss) is a sequencing simulator producing (relatively) realistic Illumina reads primarily intended for simulating metagenomic samples, although it can be used to produce sequencing data from a single genome.

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

InSilicoSeq support substitution, insertion and deletion errors. If you don't have the use for insertion and deletion error a basic error model is provided.

CHAPTER 1

# Contents

## 1.1 Installing InSilicoSeq

### 1.1.1 Using pip

To install InSilicoSeq, type the following in your terminal:

```
pip install InSilicoSeq
```

It will install InSilicoSeq as well as the following dependencies:

- biopython
- numpy
- pysam
- scipy

#### Advanced options

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

```
pip install --user InSilicoSeq
```

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

mdoule system):

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

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

### 1.1.2 Using docker

If you wish to use InSilicoSeq using docker

```
docker pull hadrieng/insilicoseq:0.8.1
```

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 to 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:0.8.0 iss generate \
    --genomes /mnt/data/ecoli.fasta -f MiSeq \
    -o /mnt/data/reads_ecoli_miseq
```

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.

## 1.2 Generating reads

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

- HiSeq 2500
- MiSeq

Per example generate 1 million MiSeq 150bp reads from a set of input genomes:

```
iss generate --genomes genomes.fasta --model_file MiSeq \
--output MiSeq_reads
```

This will create 2 files, *MiSeq_reads_R1.fastq* and *MiSeq_reads_R2.fastq* in your current directory

If you have created your custom model, change `--model_file MiSeq` to your custom model file:

```
iss generate --genomes genomes.fasta --model_file my_model.npz \
--output my_model_reads
```

### 1.2.1 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:

```
iss generate --ncbi bacteria --n_genomes 10 --model_file MiSeq \
--output MiSeq_ncbi
```

In addition the the 2 fastq files, the downloaded genomes will be in the file *MiSeq_ncbi_genomes.fasta* in your current directory.

*Note: If possible, I recommend using InSilicoSeq with a fasta file as input. The eutils utilities from the ncbi can be slow and quirky.*

### 1.2.2 Abundance distribution

The abundance of the input genomes is determined (by default) by 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:

```
iss generate --genomes genomes.fasta --abundance_file abundance.txt \
--model_file HiSeq2500 --output 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.

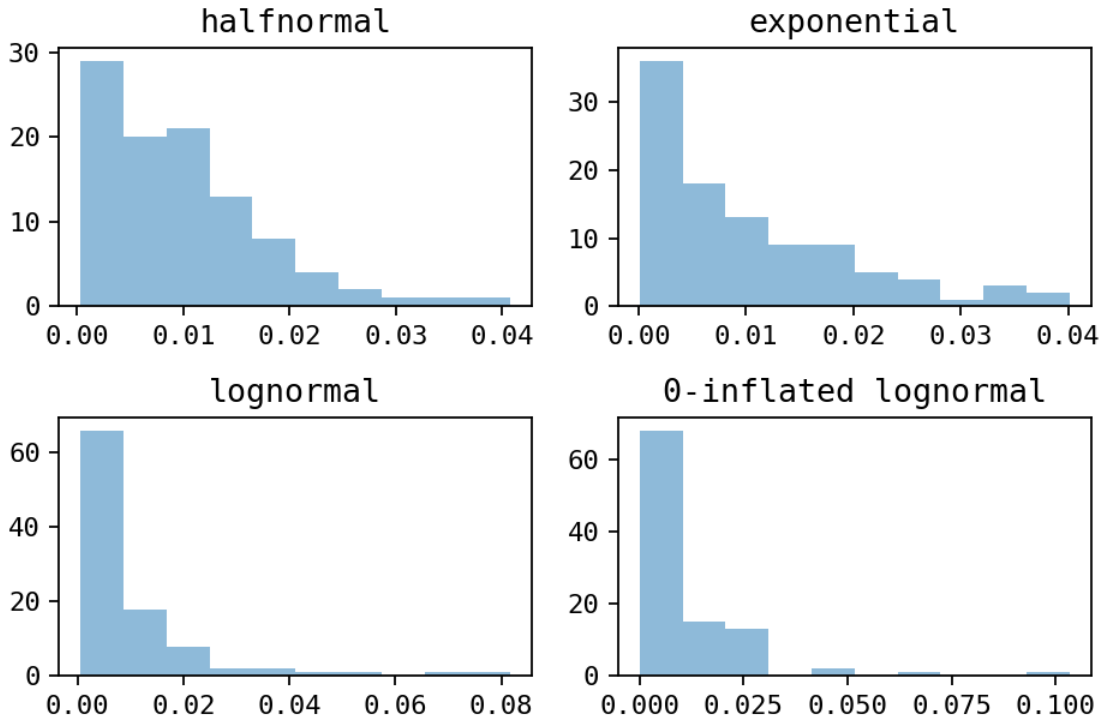


Fig. 1.1: Histograms of the different distribution (drawn with 100 samples)

### 1.2.3 Full list of options

#### –genomes

Input 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 or archaea.

### –n_genomes

How many genomes will be downloaded from the ncbi. Required if –ncbi is set.

### –abundance

Abundance distribution (default: lognormal). Can be uniform, halfnormal, exponential, lognormal or zero-inflated-lognormal.

### –abundance_file

Abundance file for coverage calculations (default: None).

### –n_reads

Number of reads to generate (default: 1000000)

### –model

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

### –model_file

Error model file. If not specified, using a basic error model instead (default: None). Use 'HiSeq2500' or 'MiSeq' for a pre-computed error model provided with the software.

### –cpus

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

### –quiet

Disable info logging

### –debug

Enable debug logging

### –output

Output file prefix (Required)

## 1.3 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.

Given you have two read files, *reads_R1.fastq.gz* and‘reads_R2.fastq.gz‘, and a referene genome *genome.fasta*:

### 1.3.1 Align you reads against the reference

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

### 1.3.2 Build the model

```
iss model -b genomes.bam -o genomes
```

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

### 1.3.3 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 prefix (Required)

#### –quiet

Disable info logging

#### –debug

Enable debug logging

## 1.4 iss package

### 1.4.1 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

**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** read_length

**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. Can be 'cdf' or 'kde'

**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

### 1.4.2 Submodules

### 1.4.3 iss.abundance module

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 list of floats

**Return type** list

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 list of floats

**Return type** list

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 list of floats

**Return type** list

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

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

**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** genome coverage

**Return type** float

iss.abundance.**to_file**(*abundance_dic*, *output*)
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 list of floats

**Return type** list

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 list of floats

**Return type** list

### 1.4.4 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

### 1.4.5 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 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
- **model** (*string*) – model to be used. Can be 'cdf' or 'kde'
- **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
- **insert_size** (*int*) – mean insert size of the aligned reads
- **mean_count_forward** (*list*) – list of mean bin sizes
- **mean_count_reverse** (*list*) – list of mean bin sizes
- **quality_hist_forward** (*list*) – list of weights, indices if model is cdf, list of cumulative distribution functions if model is kde
- **quality_hist_reverse** (*list*) – list of weights, indices if model is cdf, list of cumulative distribution functions if model is kde
- **subst_choices_forward** (*list*) – list of dictionaries representing the substitution probabilities for the forward reads
- **subst_choices_reverse** (*list*) – list of dictionaries representing the substitution probabilities for the reverse reads
- **ins_forward** (*list*) – list of dictionaries representing the insertion probabilities for the forward reads
- **ins_reverse** (*list*) – list of dictionaries representing the insertion probabilities for the reverse reads
- **del_forward** (*list*) – list of dictionaries representing the deletion probabilities for the forward reads
- **del_reverse** (*list*) – list of dictionaries representing the deletion probabilities for the reverse reads
- **output** (*string*) – prefix of the output file

### 1.4.6 iss.generator module

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

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

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

Outputs two files: output_R1.fastq and output_R2.fastq

**Parameters**

- **file_list** (*list*) – the list of input files prefix
- **output** (*string*) – the output files prefix

iss.generator.**reads**(*record*, *ErrorModel*, *n_pairs*, *cpu_number*, *gc_bias=False*)
Simulate reads from one genome (or sequence) according to an ErrorModel

Each read is a SeqRecord object Return a generator of tuples containing the forward and reverse read.

**Parameters**

- **record** (*SeqRecord*) – sequence or genome of reference
- **coverage** (*float*) – desired coverage of the genome
- **ErrorModel** (ErrorModel) – an ErrorModel class

- **gc_bias** (*bool*) – if set, the function may skip a read due to abnormal GC content

**Yields** *tuple* –

**tuple containg a forward read and a reverse** read

iss.generator.**simulate_read**(*record*, *ErrorModel*, *i*)
From a sequence record and an ErrorModel, generate a read pair

EXPERIMENTAL. SHOULD BE MULTI-THREADABLE

iss.generator.**to_fastq**(*generator*, *output*)
Write reads to fastq

Take the read generator and write read pairs in two fastq files: output_R1.fastq and output_R2.fastq

**Parameters**

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

### 1.4.7 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)

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
- **reads** –

**Returns** TODO

iss.modeller.**quality_bins_to_histogram**(*bin_lists*)
Test function that calculates pseudo 2d cumulative density functions for each position

Generate cumulative distribution functions for a number of mean quality bins

EXPERIMENTAL

**Parameters**

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

**Returns** list

iss.modeller.**raw_qualities_to_histogram**(*qualities*)
Calculate probabilities of each phred score at each position of the read

Generate cumulative distribution functions

contains the distribution/probabilities of the phred scores for one position in all the reads. Returns a list of numpy arrays for each position

**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

### 1.4.8 iss.util module

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.**genome_file_exists**(*filename*)

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.**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

### 1.4.9 Module contents

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

*This documentation was generated on* 2017-11-20 *at* 09:59.

# Python Module Index

## i

# Index

## S



## T



## U



## W



## Z