HAYSTAC

Release beta

Evangelos A. Dimopoulos, Evan K. Irving-Pease

Mar 02, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

HAYSTAC is a comprehensive computational tool for identifying species from DNA sequence data. It can pre-process sequencing data (adapter trimming), build a database, analyse sequencing data and perform a deamination profile analysis. It can work in two different modes. One mode performs metagenomic identifications from samples containing multiple organisms, and outputs mean posterior species abundances. The second mode can perform species identifications from single organism samples and it outputs species assignment posterior probabilities.

CHAPTER

TWO

INSTALLATION

HAYSTAC can be run on either macOS or Linux based systems, and the source code is available on `github`.

The recommended way to install `haystac`, and all if its dependencies, is via the [mamba](https://mamba.readthedocs.io/en/latest/installation.html) package manager (a fast replacement for [conda](https://docs.conda.io/projects/conda/en/latest/index.html)).

You can install `haystac` using `conda`, however, it will take significantly longer to install and analyses will run slower.

2.1 Install mamba

If you do not have either `mamba` or `conda` already installed, please refer to the [install instructions](https://mamba.readthedocs.io/en/latest/installation.html) for `mambaforge`.

If you have `conda` installed, but not `mamba`, then install `mamba` into the base environment:

```
conda install -n base -c conda-forge mamba
```

2.2 Install haystac

Then use `mamba` to install `haystac` into a new environment:

```
mamba create -c conda-forge -c bioconda -n haystac haystac
```

And activate the environment:

```
conda activate haystac
```

We recommend that you install `haystac` into a new environment to avoid dependency conflicts with other software.

2.3 Git

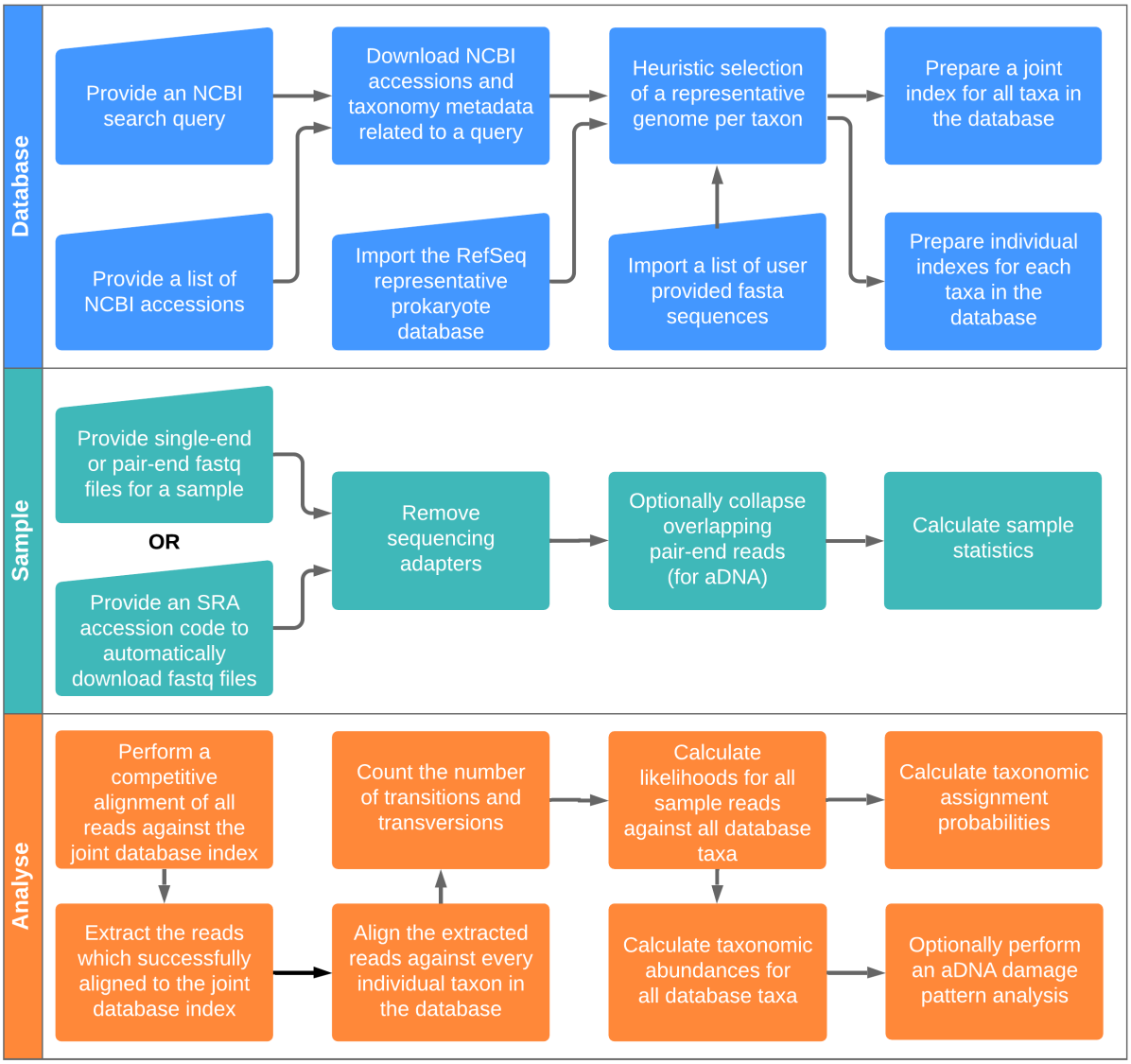
Clone from `github`:

```
git clone https://github.com/antonisdim/haystac.git
```

CHAPTER

THREE

WORKFLOW



HAYSTAC was designed to be used as a species identifier for either single organism or metagenomic samples. The full execution of the pipeline includes the construction of a database, the processing of a sample file and lastly the analysis of a sample against a specific database. For that purpose three modules have been designed, each of which has its own

outputs.

First the `haystac database` module can be used to construct a database, based on the user's needs and preferences (custom NCBI query, and/or custom NCBI accessions for each species, and/or prokarytic representative RefSeq, and/or custom sequences). After the user input is collected, the user must provide a path where the outputs of the database can be stored.

The `haystac sample` module prepares a sample to be analysed. It deals with trimming adapters and collapsing PE reads if needed and it counts the number of reads that are included in the sample file provided by the user.

The `haystac analyse` module performs the analysis of a sample against a specific database. All its outputs are created under the path that the user specifies with the `--output` path.

CHAPTER

FOUR

OUTPUTS

This is a list of all the outputs produced after a successful run of each of `haystac`'s modules. All these outputs will be found under the `--output` path specified by the user when running any of `haystac`'s modules. We advise to create separate output directories for each one of the modules (`database`, `sample` and `analyse`).

4.1 Expected outputs for `haystac database`

db_taxa_accessions.tsv: includes all the taxon/accession pairs that are included in a database.

idx_database.done: file indicating that all the individual bowtie2 indices for each taxon have been prepared

entrez: directory containing all the results from querying the NCBI, including the nucleotide and taxonomy databases

bowtie: directory containing the bowtie2 index files for all out database, that will be used for the filtering alignment

database_inputs: directory containing the representative RefSeq table that is downloaded from NCBI.

4.2 Expected outputs for `haystac sample`

fastq_inputs: folder containing the outputs of the sample module.

fastq_inputs/meta: directory that includes the read count file.

fastq_inputs/(SE | PE | COLLAPSED): directory containing the trimmed reads produced by AdapteRremoval

4.3 Expected outputs for `haystac analyse`

bam: directory containing the bam file of the filtering alignment

fastq: directory containing the filtered reads in fastq format along with their average read length

alignments: directory where all the individual alignment bam files for each taxon in a database of the filtered reads are outputted

ts_tv_counts: directory where all the transition and transversion counts are stored per taxon

probabilities: directory where the likelihood matrix and final posterior abundances/probabilities are stored. The final output for abundance calculation has the suffix posterior_abundance.tsv,

mapdamage: directory that includes all the mapdamage profiles for every taxon in our Database

reads: directory including all the Dirichlet reads for each taxon in out database.

CHAPTER

FIVE

TUTORIAL

5.1 Configuring HAYSTAC

If a user is running haystac for the first time on a machine, they might want to run the `haystac config` module first. The user can use this module to provide an API key for querying NCBI and their preferred path for the cache genomes folder, among other things. All of the options have default values that can be used. If a user later on wishes to change any of these parameters specifically they can either run `haystac config` to pass a value to a specific argument.

Here is an example command that allows the configuration of using conda as a package manager for running the other haystac modules.:

```
haystac config --use-conda True
```

5.2 Building the database

In order to build the database we will be using the `database` module of HAYSTAC. First we need to know what organisms we would like to include in our database. Do we only need the complete genomes of a specific genus or do we want more genera?

5.3 Constructing the Query

After deciding what taxa we would like to include in our database, we need to construct an NCBI query that will return all the accessions that belong to the taxa that we are interested in. One of the best ways to construct such a query is to go on the website of NCBI's Nucleotide database (https://www.ncbi.nlm.nih.gov/nucleotide/), type in our query and get the correctly formatted query string from the "Search details" box. We can then use that string for the construction of our database.

For example if we would like to build a database of all the complete genomes of the species in the Yersinia genus we can use the following command::

```
haystac database --mode build \
    --query '"Yersinia"[Organism] AND "complete genome"[All Fields]' \
    --output yersinia_example
```

For each species (or any other user defined taxonomic rank), the longest sequence per taxon will be used to populate our database.

5.4 Representative RefSeq species

When constructing a database there is always the option to include the species of the prokaryotic representative RefSeq database as well. All you need to do is include the corresponding flag in your command.:

```
haystac database --mode build \
    --query '"Yersinia"[Organism] AND "complete genome"[All Fields]' \
    --output yersinia_example \
    --refseq-rep prokaryote_rep
```

5.5 Important note on RefSeq databases

`haystac database` currently can build databases from three RefSeq tables, the prokaryotic representative RefSeq table, the eukaryotes RefSeq table and the viruses RefSeq table. When the prokaryotic representative the database is built, only species of microorganisms are included (strains are excluded), whereas in the eukaryotes and viruses databases subspecies and strains are included respectively. To build any of the above databases, specify the desired RefSeq table to be used by the `--refseq-rep` flag (`prokaryote_rep` for the prokaryotic representative, `eukaryotes` for the eukaryotes and `viruses` for the viruses table).

5.6 Providing custom accessions

It is also possible to provide your own accessions for a selected species/taxon. For that you will need to prepare a tab delimited file with two columns. The first column is the name of the taxon, that cannot contain any special characters, other than an underscore ('_'), and the second column is a valid NCBI accession code.

Here is an example of the contents of such a file::

```
Yersinia_ruckeri    NZ_CP025800.1
```

The we can simply run the following command::

```
haystac database --mode build \
    --query '"Yersinia"[Organism] AND "complete genome"[All Fields]' \
    --output yersinia_example \
    --accessions-file acc_example.txt
```

5.7 Providing custom sequences

It is also possible to provide your own sequences for a taxon. To do that you will need a a tab delimited file containing the the name of the taxon with no special characters, and an underscore ('_') instead of spaces, a user defined accession code and the path of the fasta file. The fasta file that the path point to can be either uncompressed or compressed with gzip/bgzip.

Here is an example of such a file::

```
Yersinia_ruckeri    user_seq_1  ~/example_sequence/user_seq_1.fasta
```

The we can simply run the following command::

```
haystac database --mode build \
    --query '"Yersinia"[Organism] AND "complete genome"[All Fields]' \
    --db-output yersinia_example \
    --sequences-file seq_example.txt
```

5.8 Combinations

All of the previous options can be combined into one command. It is important to note that only one sequence file per taxon is allowed in our database, and the priority goes user defined accessions or sequences, representative RefSeq and then user specified query. It should be noted that for custom NCBI queries, plasmids can be fetched if they are part of a genome assembly. The only exception to the one sequence file per taxon rule is plasmid sequences of the RefSeq representative that are not part of complete genome assemblies and for that reason they are downloaded separately.

5.9 Index building

For the first part of the analysis an index out of all the genomes that are included in our database needs to be build. This is a process that can take big amounts of memory depending on the number and the complexity of sequences that our database includes. For that reason the user can specify the desired amount of memory resources available to `haystac` and the program will try to build the required index. This can be specified through the `--mem` flag, that can be appended to the any of the commands shown above. Memory resources need to be specified in MB. If the memory resources provided are less than the size of the files that need to be indexed an error will be raised. We also advise caution when changing the bowtie2 file size scaling factor.

5.10 Database building modes

For the complete construction of a database, sequences need to be downloaded and subsequently indexed. By specifying `--mode build` to `haystac database`, the program downloads and indexes all the sequences that have been requested by the user in one step. If a user would like to only download sequence data and index them later it is possible to do so, by specifying `haystac database --mode fetch`, to download the sequences first and then execute `haystac database --mode index` in order to perform the indexing. If mode `fetch` is run first then mode `index` should be run subsequently, and not mode `build`, otherwise an error will be raised.

Here is an example of building a database in two steps instead of one::

```
haystac database --mode fetch \
    --query '"Yersinia"[Organism] AND "complete genome"[All Fields]' \
    --output yersinia_example
haystac database --mode index \
    --output yersinia_example
```

5.11 Building a mitochondrial DNA database

When a user is providing a query about eukaryotes it is also possible to build a database with only mitochondrial genomes (by default whole genome assemblies will be fetched for a given query). In order to do that a user can specify the `--mtDNA` flag when running `haystac database`. We strongly advise against having a mixed database of full eukaryotic genome assemblies for certain taxa and only mtDNA sequences for other taxa, as this will bias the identifications towards the taxa with full genome assemblies.

5.12 Preparing a sample for analysis

After our database is built we need to prepare our samples for analysis. For that purpose, we are using the `sample` module of haystac. The input files can be SE, PE or collapsed reads. If the reads are collapsed they are going to be treated as SE reads.

It is possible to trim sequencing adapters and collapse PE reads by specifying the relative flags. Samples (specific sequencing runs) can be also downloaded from the SRA if an sra run accession is provided.

If you have SE or already collapsed reads you only need to specify a file path for the `--fastq` flag. If your input is PE reads then you will need to specify file paths for both the `--fastq-r1` and `--fastq-r2`. If you want to download files from the SRA all you need to do is provide an SRA accession for the `--sra` flag.

Here is an example of downloading reads from the SRA, trimming sequencing adapters and collapsing reads.:

```
haystac sample --sra ERR1018966 \
    --output sample_example
```

5.13 Sample analysis

In order to analyse any sample we will need to use the `analyse` module of haystac.

5.14 Filtering Alignment

The first step for the sample analysis is to filter in all the reads that align to any of the genomes in our database. For that we will need to use the `haystac analyse --mode filter`.

Here is an example command::

```
haystac analyse --mode filter \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

5.15 Database Alignments

After we have filtered our libraries we can align the filtered reads against all the genomes that are included in our database. This can be done by using mode `align` of `haystac analyse`.

For example::

```
haystac analyse --mode align \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

Unless the user has a deep understanding of their dataset we advise to be cautious when changing the base mismatch probability that is used later on in the method's probabilistic model.

5.16 Likelihood calculation

After all the individual alignments have been competed, the number of transitions and transversions will be counted for every read that has aligned against any of the reference genomes in our database. Then the likelihoods and posterior probabilities for each read being sampled from a given reference genome will be calculated. For this step we can use the `likelihoods` mode of `haystac analyse`.:

```
haystac analyse --mode likelihoods \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

5.17 Important Note on the Dirichlet Assignment process during Likelihood calculation

It is important to be aware of the individual read posterior probability threshold, for a read to be assigned to a taxon. As a default HAYSTAC uses the conservative 0.75 probability threshold for the Dirichlet assignment. The higher value you pick the more conservative the assignments become. It is useful to sometimes pick a value depending on what taxa are being identified. If there is a need to distinguish between closely related taxa then a more conservative threshold would increase the specificity of the analysis therefore being more appropriate, whereas when you're trying to generally characterise a metagenome a less conservative value could increase the sensitivity of the analysis be more helpful.

5.18 Single organism sample or metagenome ?

Depending on whether we would like to identify the species a sample is belongs to, or perform a metagenomic analysis, we can use the `probabilities` or `abundances` mode of `haystac analyse` respectively.

5.19 Assignment Probability Calculation

In order to calculate posterior assignment probabilities we can run the following command::

```
haystac analyse --mode probabilities \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

5.20 Mean Posterior Abundances

In order to calculate mean posterior abundances we can run the following command::

```
haystac analyse --mode abundances \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

Along with the abundance calculation, we also perform a chi2 test to assess if the reads that have been assigned to a taxon are clustering around specific genomic areas or if they represent a random sample of the organism's genome. The results of this test should be trusted for low depth sequencing data (equal or less than 1X). The null hypothesis is that there is no read clustering.

5.21 Reads

After the mean posterior abundances have been calculated for a sample, all the reads that have been assigned to a taxon through the Dirichlet process can be outputted in separate bam files ready for further downstream analyses (like assembling or variant calling for instance) via the `reads` module. Reads that have been assigned to the Grey and Dark Matter are outputted in fastq files as they have not been uniquely assigned to a taxon.

Here is an example command::

```
haystac analyse --mode reads \
    --database yersinia_example \
    --sample sample_example \
    --output analysis_output
```

5.22 Mapdamage analysis

If our samples are ancient we can use mapDamage to estimate the level of deamination in the reads that have aligned to any taxon in our database. For that we can use the `mapdamage` module of haystac. The mapDamage analysis will be performed on the subset of reads that have been uniquely assigned to a taxon through the dirichlet process. This module can be either run independently or after the `reads` module.

Here is an example command::

```
haystac analyse --mode mapdamage \
    --database yersinia_example \
```

(continues on next page)

(continued from previous page)

```
    --sample sample_example \
    --output analysis_output
```

5.23 Important note on sample analysis

The first 3 steps (modes: `filter`, `align`, `likelihoods`) can be executed automatically when you call the `probabilities` or `abundances` mode of haystac.

CHAPTER

SIX

COMMAND LINE INTERFACE

6.1 haystac config

Optional arguments:

```
-h, --help            Show this help message and exit
--cache <path>        Cache folder for storing genomes downloaded from NCBI
                      and other shared data (default:
                      /home/antony/haystac/cache)
--clear-cache         Clear the contents of the cache folder, and delete the
                      folder itself (default: False)
--api-key <code>      Personal NCBI API key (increases max concurrent requests
                      from 3 to 10,
                      https://www.ncbi.nlm.nih.gov/account/register/)
--use-conda <bool>    Use conda as a package manger (default: False)
```

6.2 haystac database

Required arguments:

```
--mode <mode>         Database creation mode for haystac [fetch, index,
                      build]
--output <path>       Path to the database output directory
```

Required choice:

```
--query <query>       Database query in the NCBI query language. Please
                      refer to the documentation for assistance with
                      constructing a valid query.
--query-file <path>   File containing a database query in the NCBI query
                      language.
--accessions-file <path>
                      Tab delimited file containing one record per row: the
                      name of the taxon, and a valid NCBI accession code
                      from the nucleotide, assembly or WGS databases.
--sequences-file <path>
                      Tab delimited file containing one record per row: the
                      name of the taxon, a user defined accession code, and
                      the path to the fasta file (optionally compressed).
```

(continues on next page)

(continued from previous page)

```
--refseq-rep <table>  Use one of the RefSeq curated tables to construct a
                      DB. Includes all prokaryotic species (excluding
                      strains) from the representative RefSeq DB, or all the
                      species and strains from the viruses DB, or all the
                      species and subspecies from the eukaryotes DB. If
                      multiple accessions exist for a given species/strain,
                      the first pair of species/accession is kept. Available
                      RefSeq tables to use [prokaryote_rep, viruses,
                      eukaryotes].
```

Optional arguments:

```
--force-accessions    Disable validation checks for 'anomalous' assembly
                      flags in NCBI (default: False)
--exclude-accessions <accession> [<accession> ...]
                      List of NCBI accessions to exclude. (default: [])
--resolve-accessions  Pick the first accession when two accessions for a
                      taxon can be found in user provided input files
                      (default: False)
--bowtie2-scaling <float>
                      Rescaling factor to keep the bowtie2 mutlifasta index
                      below the maximum memory limit (default: 25.0)
--rank <rank>         Taxonomic rank to perform the identifications on
                      [genus, species, subspecies, serotype] (default:
                      species)
--genera <genus> [<genus> ...]
                      List of genera to restrict the abundance calculations.
--mtDNA               For eukaryotes, download mitochondrial genomes only.
                      Not to be used with --refseq-rep or queries containing
                      prokaryotes (default: False)
--seed <int>          Random seed for database indexing
```

Common arguments:

```
-h, --help            Show this help message and exit
--cores <int>         Maximum number of CPU cores to use (default: MAX_CPUs)
--mem <int>           Maximum memory (MB) to use (default: MAX_MEM)
--unlock              Unlock the output directory following a crash or hard
                      restart (default: False)
--debug               Enable debugging mode (default: False)
--snakemake '<json>'  Pass additional flags to the `snakemake` scheduler..
```

6.3 haystac sample

Required arguments:

```
--output <path>       Path to the sample output directory
```

Required choice:

```
--fastq <path>          Single-end fastq input file (optionally compressed).
--fastq-r1 <path>       Paired-end forward strand (R1) fastq input file.
--fastq-r2 <path>       Paired-end reverse strand (R2) fastq input file.
--sra <accession>       Download fastq input from the SRA database
```

Optional arguments:

```
--collapse <bool>       Collapse overlapping paired-end reads, e.g. for aDNA
                        (default: False)
--trim-adapters <bool>
                        Automatically trim sequencing adapters from fastq
                        input (default: True)
```

Common arguments:

```
-h, --help              Show this help message and exit
--cores <int>           Maximum number of CPU cores to use (default: MAX_CPUs)
--mem <int>             Maximum memory (MB) to use (default: MAX_MEM)
--unlock                Unlock the output directory following a crash or hard
                        restart (default: False)
--debug                 Enable debugging mode (default: False)
--snakemake '<json>'    Pass additional flags to the ``snakemake`` scheduler.
```

6.4 haystac analyse

Required arguments:

```
--mode <mode>           Analysis mode for the selected sample [filter, align,
                        likelihoods, probabilities, abundances, reads,
                        mapdamage]
--database <path>       Path to the database output directory
--sample <path>         Path to the sample output directory
--output <path>         Path to the analysis output directory
```

Optional arguments:

```
--genera <genus> [<genus> ...]
                        List of genera to restrict the abundance calculations
                        (default: [])
--min-prob <float>      Minimum posterior probability to assign an aligned
                        read to a given species (default: 0.75)
--mismatch-probability <float>
                        Base mismatch probability (default: 0.05)
```

Common arguments:

```
-h, --help              Show this help message and exit
--cores <int>           Maximum number of CPU cores to use (default: MAX_CPUs)
--mem <int>             Maximum memory (MB) to use (default: MAX_MEM)
--unlock                Unlock the output directory following a crash or hard
                        restart (default: False)
```

(continues on next page)

(continued from previous page)

```
--debug                Enable debugging mode (default: False)
--snakemake '<json>'   Pass additional flags to the `snakemake` scheduler.
```

CHAPTER

SEVEN

DEVELOPER DOCUMENTATION

Todo: Write developer documentation.

CHAPTER

EIGHT

FAQS

Todo: write FAQs

CHAPTER

NINE

TRACKING ISSUES AND BUGS

haystac is under active development and we encourage you to report any issues you encounter via the GitHub issue tracker (https://github.com/antonisdim/haystac/issues).

CHAPTER

TEN

CITATIONS

A preprint describing `haystac` is available on *bioRxiv*:

Dimopoulos, E.A.*, Carmagnini, A.*, Velsko, I.M., Warinner, C., Larson, G., Frantz, L.A.F., Irving-Pease, E.K., 2020. HAYSTAC: A Bayesian framework for robust and rapid species identification in high-throughput sequencing data. *bioRxiv* 2020.12.16.419085. https://www.biorxiv.org/content/10.1101/2020.12.16.419085v1

CHAPTER

ELEVEN

CONTRIBUTING

Evangelos Antonios Dimopoulos, Evan K. Irving-Pease, Alberto Carmagnini

CHAPTER

TWELVE

LICENSE

MIT