Version 1.4.0

SynTracker Manual

Table of Contents

Overview

SynTracker is a python and R based software to determine the relatedness of conspecific strains (microbial strains of the same species) across genomic and metagenomic samples. SynTrakcer tracks strains using genome microsynteny (i.e., the conservation of the order of K-mers along two chromosomes).

SynTracker is highly sensitive to structural differences between homologous genomic regions (insertions, deletions, recombinations, etc.) while having low sensitivity to Single Nucleotide Variants, and could be used to complement SNV based strain-tracking tools (for example, inStrain *(Olm et-al., Nature Biotechnology, 2021)*).

SynTracker consists of the following main stages, which are performed on each reference genome independently:

a) Fragmentation of the reference genome to "central regions" (of length 5000 bp by default).
b) For each "central region", perform the following processes:
    a. Execution of BLASTn search, with the "central region" as query, against a target dataset, consisting of the metagenomic samples (or genomes) that are to be compared.
    b. Calculation of *synteny score* for each pair of homologous genomic regions (i.e., two DNA sequences, from different metagenomic samples (or genomes), retrieved from a single BLASTn search, with the same "central region").

c) Finally, n regions are selected randomly, per pair of metagenomic samples (or genomes, if those are studied), and the Average Pairwise Synteny Score (APSS) is calculated.

Citation

If you use SynTracker please cite:

Strain tracking in complex microbiomes using synteny analysis reveals per-species modes of evolution.
Enav H, Paz I and Ley RE.
Nature Biotechnology (2024). DOI: https://doi.org/10.1038/s41587-024-02276-2.

Installation

Requirements:

NCBI BLAST+
All the other required packages are contained in the attached conda environment (.yml file).

Installing from source:

1. Download SynTrakcer latest release from: https://github.com/leylabmpi/SynTracker/releases.
2. Extract the tar.gz file into the desired working-directory.
3. Create a new conda environment using the 'SynTracker_env.yml' file (located in the root directory of SynTrakcer) using the following command: *conda env create -f SynTracker_env.yml*
4. Activate the newly created environment: *conda activate SynTracker_1_4*

Input files

SynTracker requires two types of data as input:

a) **Reference genomes:**
Reference genomes can be provided complete or as a collection of contigs. If using a number of contigs belonging to the same reference genome, all sequences should be placed in a single .fasta file.
If using more than one reference genome (i.e., analyzing more than one species per run) all reference genome files should be located in the same directory.

b) **Metagenomic assemblies/genomes:**

These are the metagenomic assemblies (or assembled genomes, if those are studied) that would be compared. These data should be organized in per-sample assembly files - i.e., all the contigs assembled from sample X would be kept in a single .fasta file.
If genomes are to be compared, each genome will be stored in a single .fasta file. All files should be stored in the same directory (referred below as the "target directory").

Sample input:

The directory ‘Sample_Data/Sample_Input/’, which is included in the SynTracker package, contains one reference genome and a collection of target genomes. It can be used to clarify the structure of the required input files and to test the installation.

Usage

SynTracker is executed from the command line. It has two main modes of execution:

1. '**New**' mode: a new run. In this case the user must provide the path to the reference genomes and to the target genomes. All the other parameters are optional (including the output directory, which is created by default under the working directory with the name 'Syntracker_output').

2. '**Continue**' mode: continue a previous run that has been terminated for some reason without having to start the process from the beginning. In this case, the user must provide only the path to the output folder of the run that he would like to continue. There are two options to continue a previous run:

    a. Using mode = ‘**continue**’: the run will continue from the point it stopped within the last reference genome that has been processed without finishing successfully.
    b. Using mode = ‘**continue_all_genomes**’: in this case all the reference genomes will be processed again, but the stage of building a blast DB from the target genomes will not be repeated. It makes sense to use this mode only for datasets which contain more than one reference genome.

SynTracker’s command-line arguments description:

```
python syntracker.py [-h/--help]
[-target target_directory_path] [-ref ref_directory_path]
[-out output_directory_path]
[-mode new/continue/continue_all_genomes]
[-blastDB blastDB_directory_path]
[-cores number_of_cores] [-length region_length]
[--identity blast_identity] [--coverage blast_coverage]
[--no_seed]
```

- **-target <target directory path>:**
  The path of the directory which contains the metagenome assemblies or genomes to be compared.

- **-ref <reference directory path>:**
  The path of the directory which contains the reference genome(s).

- **-out <output directory path>:**
  The path to the output directory.
  When running in 'new' mode (the default), this argument is optional. By default a folder named 'Syntracker_output/' will be created under the current directory (if the given path already exists, it will be written over).
  When running in 'continue' mode, it is mandatory to provide the path to the output directory of the run that is requested to be continued.

- **-mode <'new' / 'continue' / 'continue_all_genomes'>:**
  The running mode: 'new' or 'continue' or 'continue_all_genomes' (default='new').
  Mode 'new': start a new SynTracker run.
  Mode 'continue': continue a previous job that was terminated for some reason from the point it stopped.
  Mode 'continue_all_genomes': process all the reference genomes from the beginning, without repeating the stage of building a blast DB from the target genomes (the DB serves all the reference genomes). This mode is only relevant when running more than one reference genome.

- **-blastDB <blastDB_directory_path>:**
  The path to the directory which was previously created by syntracker_makeDB.py and contains the uniquely renamed target genomes and the blastDB.
  This is an advanced optional argument to be used when the blastDB has already been created by syntracker_makeDB.py. When using it, there is no need to provide the '-target' argument.

- **-cores <number of cores>:**
  The number of cores to use for the multi-processing stages of the calculation (optional). By default, SynTracker uses the maximal number of available cores.

- **-length <region length>:**
  The length of the compared region (in bp. Optional parameter, default=5000 bp).

  The SynTrakcer pipeline is based on pairwise alignments of homologous sequences in pairs of genomes or metagenomic sequences. By default, the length of the regions that are being aligned is 5000 bp, consisting of the "central region" (see figure 1) which is 1000 bp long and the "flanking regions" - two 2000 bp long regions, located upstream and downstream to the central region. The overall length of the compared regions can be adjusted using the –length command line argument. The length of the "central region" does not change and is always 1000 bp.

  A change in the length of the compared regions will likely result in a different distribution of APSS (Average Pairwise Synteny Score). This happens as the per-region synteny score is

based on the number of synteny breaks identified per region. When longer regions are analyzed, the probability of identifying more breaks per region increases.

Moreover, longer region lengths will probably yield a lower number of regions that can be compared for each pair of genomes or metagenomic samples. This happens as:

a) The entire reference genome is divided into a smaller number of regions.
b) The probability of identifying homologous regions of suitable length decreases.

In general, we recommend using the default region length and if possible, maximize the number of regions subsampled per pairwise comparison. However, if poor assemblies are being compared, it is possible to use smaller region length values to increase the number of compared regions per pair of samples.

Please note: If region length values that are different than the default are being used, the same-strain cutoff should be determined again, using an appropriate training set.

- **--identity <BLASTn identity>:**
  The minimal identity value for the BLASTn search (optional, default=97).

  The second step in the SynTracker pipeline consists of a BLASTn search using the "central regions" against a database that is composed of the metagenomes/genomes to be compared. In this search, only hits with a minimal identity of 97% are retrieved. This is done to reflect standard definitions for nucleotide identity at the species level, with the goal of only comparing strains that belong to the same species.
  While SynTracker was not designed to compare strains across species, it is possible in principle to use lower minimal identity values (for example, 95%), to perform analysis of genomes classified to different species. BLAST minimal identity can be modified using the –identity command-line argument.

- **--coverage <BLASTn coverage>:**
  The minimal coverage value for the BLASTn search (optional, default=70).

- **--no_seed:**
  Set no seed for the subsampling of n regions per pairwise (optional). This means that the average synteny scores may change between SynTracker runs due to the subsampling. By default, a seed=1 is set to enable reproducibility between different runs.

Usage examples using the provided sample data:

(With the minimal required mandatory input parameters)

- **A new run**:

```
python syntracker.py
-target Sample_Data/Input_example/Target_genomes/
-ref Sample_Data/Input_example/Reference_genomes/
-out SynTracker_output/
```

- **Continue a previous run that has been terminated**:

  1. Continue from the last reference genome that has been processed without finishing successfully:

     ```
     python syntracker.py -out SynTracker_output/ -mode continue
     ```

  2. Process all the reference genomes again without repeating the blastDB building stage (relevant only for datasets containing more than one reference genome):

     ```
     python syntracker.py -out SynTracker_output/
     -mode continue_all_genomes
     ```

- **Advanced use-case: create blastDB and use it in distributed batches:**

  This usage is recommended when the input dataset contains many reference genomes that can be divided into batches and be executed in a distributed way (as opposed to the normal SynTracker run, which runs the reference genomes one by one). In this case, SynTracker should be executed in two separated stages:

  1. Run the script '**syntracker_makeDB.py**' to create a directory containing the uniquely renamed target genomes and the blast database created from them:

     ```
     python syntracker_makeDB.py
     -target Sample_Data/Input_example/Target_genomes/
     -out blastDB_output/
     ```

  2. Run SynTracker using -blastDB argument, providing the previously created blastDB directory:

     ```
     python syntracker.py -blastDB blastDB_output/
     -out SynTracker_output/
     ```

  The first stage should be executed once (for the required dataset).
  The second stage of running SynTracker using the newly created blastDB_output_directory, can be executed in parallel for all the batches of reference genomes.

Output

Output files organization

1. **Reference genomes directories**: For each reference genome given by the user, SynTracker creates a subdirectory, under the main output directory.
   Within this directory, there are several subdirectories for intermediate calculation stages and one directory, named '**final_output**', for the final output tables of this reference genome.
   In case the user applied the –save_intermediate option, a directory named '**R_intermediate_objects**' is also created and contains the R data structures (a file per each compared region).

2. **Summary directory**:
   The directory '**summary_output**' is created under the main output folder and contains SynTracker's final output tables combining the results for all the reference genomes together.

Final output tables

SynTracker provides two types of output tables:

1. **Raw results per compared region**: The table '**synteny_scores_per_region.csv'** contains the raw results obtained by the comparison of each two homologous genomic regions in each two metagenomes in which they were detected (or genomes, if those are being compared).

   This type of tables is provided per each reference genome, under the '<ref_genome>/final_output/' folder and also under the 'summary_output' folder, containing the results for all the reference genomes concatenated.

   An example of the 'synteny_scores_per_region' output table:

   "Ref_genome","Sample1","Sample2","Region","Length1","Length2","Overlap","Blocks","Synteny_score"
   "E_coli_K12_MG1655","ASM15512v1","ASM17195v1","NC000913.2_0_1000",5000,5000,4902,2,0.6902
   "E_coli_K-12_MG1655","ASM15512v1","ASM17195v1","NC000913.2_10000_11000",5000,5000,5000,1,1
   "E_coli_K-12_MG1655","ASM15512v1","ASM17195v1","NC000913.2_30000_31000",4998,4998,4998,1,1

   Output columns description:

   - **Region**: the 'central region' in the reference genome that was used to find the pair of homologous regions in the two samples, using the BLASTn search.
   - **Length1** and **Length2** are the lengths of the compared regions of sample1 and sample2 correspondingly.
   - **Overlap:** the length of the overlap between the regions taken from the two samples.
   - **Blocks:** the number of synteny blocks that were found comparing the two regions.
   - **Synteny_score:** the calculated synteny score for the comparison between the homologous regions of sample1 and sample2.

2. **Average Pairwise Synteny Scores (APSS)**: The collection of tables, named '**avg_synteny_scores_[subsampling length]_regions.csv'**, gives the APSS (Average Pairwise Synteny Score) that was calculated by subsampling N regions per pair of samples from the overall regions that appear in the raw table (detailed above).
   For each pair of samples, SynTracker tries to subsample the following number of regions: 40, 60, 80, 100, 200. It then creates an output table for each number of subsampled regions (5 tables). Each table contains the APSS for the pairs, for which there were enough available regions in order to perform the subsampling. Pairs, for which there were not enough compared regions for a specific subsampling value, are excluded from the related tables.

In case there are no pairs at all, for which the higher subsampling number of regions are available (for example: no pairs with number of regions > 100), the output tables for these missing subsampling values will not be created.

An additional table, named `**avg_synteny_scores_all_regions.csv**`, outputs the APSS calculated using all the available regions per each pair of samples, without subsampling.

The collection of APSS output tables is provided per each reference genome, under the '<ref_genome>/final_output/' folder and also under the 'summary_output' folder, containing the results for all the reference genomes concatenated.

An example of an APSS output table for a subsampling value of 200:

"Ref_genome","Sample1","Sample2","APSS","Compared_regions"
"E_coli_K12_MG1655","ASM15512v1","ASM17195v1",0.974409823863967,200
"E_coli_K12_MG1655","ASM15512v1","ASM2222v1",0.981137402165184,200
"E_coli_K12_MG1655","ASM15512v1","ASM2516v1",0.967012514024994,200
"E_coli_K12_MG1655","ASM15512v1","ASM666v1",0.972798012230779,200

Sample output:

The directory 'Sample_Data/Output_example/', which is included in the SynTracker package, contains the output of a SynTracker's run, using the sample input data with default parameters. It can facilitate the user in better understanding the structure of the output directories and files.