RetSynth 1 documentation

Navigation

- RetSynth 1 documentation »
- RetSynth 1 documentation

Welcome to RetSynth's documentation!¶

RetSynth is a tool, developed with python, which identfies enzyme/reaction pairs that are required in a user specified microbial organisms to produce a target chemical compound.

Note

RetSynth works on operating systems: Windows

Mac

Linux

Required Non-python Dependencies: GLPK (v4.64 preferred)

GraphViz (only if `--figures_graphviz` option is being used, See Visualization_graphviz option below)

Python Dependencies: PULP

glpk

CobraPy

libsbml

PyGraphViz (only if `--figures_graphviz` option is being used, See Visualization_graphviz option below)

bs4

pubchempy

openpyxl

scipy

tqdm

soupsieve

filelock

distlib

jsonschema

matplotlib

Purpose¶

In the process of bioengineering a microbial organism, identifying reactions/enzymes to transform into an organism for optimal production of a target compound is extremely difficult due to the vast number of reactions/enzymes in numerous organisms that maybe suitable for optimal production.

By assembling a database of genome-wide metabolic networks for a plethora of microbial organisms and framing this information into an integer linear program, we can identify the optimal number of enzyme/reaction pairs that need to be added into an organism for optimal synthesis of a compound. Specifically the goal is to identify the minimal number of reactions/enzymes as this is beneficial for metabolic engineering because less genetic manipulation would be required. Using flux balance analysis FBA RetSynth can predict production yields of a compound in a select microbial organism.

The overarching goal of RetSynth is to streamline an arduous and complex step of bioengineering a microbial organism, which will enable scientists to inexpensively expedite the production of important target compounds.

Installation¶

1. git clone https://github.com/sandialabs/RetSynth.git
2. Enter the master directory and run `python setup.py install`

RetSynth Workflow¶

1. Construction of Metabolic Database¶

In order for RetSynth to identify potential reactions/enzymes to engineer to an organism for synthesis of a target compound a database of genome-wide metabolic information for numerous microbial organisms must be available to the software. RetSynth builds an SQLite database. RetSynth can compile a metabolic database which can include metabolic information from the following Repositories:

1. PATRIC
2. MetaCyc
3. KEGG
4. Metabolic In Silico Network Expansion Database
5. ATLAS of BioChemistry
6. SPRESI

Note

The SPRESI database has to be purchased by the user and for RetSynth to integrate SPRESI data into its database it currently has to be in the Rxnfiles (.rdf) format.

To build a database the user must identify the metabolic repository with which the database is to be constructed (`--patric`, `--metacyc`, `--kegg`, `--atlas`, `--mine` and/or `--SPRESI`). Furthermore the directory or file which contains the raw metabolic repository (specifically needed for –kbase, –metacyc, –mine and –atlas) information must be specified:

`-a_dir or --atlas_dump_directory` for atlas data (csv files) with the `--atlas` option,

`-m_dir or --mine_dump_directory` for mine data (msp files) with the `--mine` option,

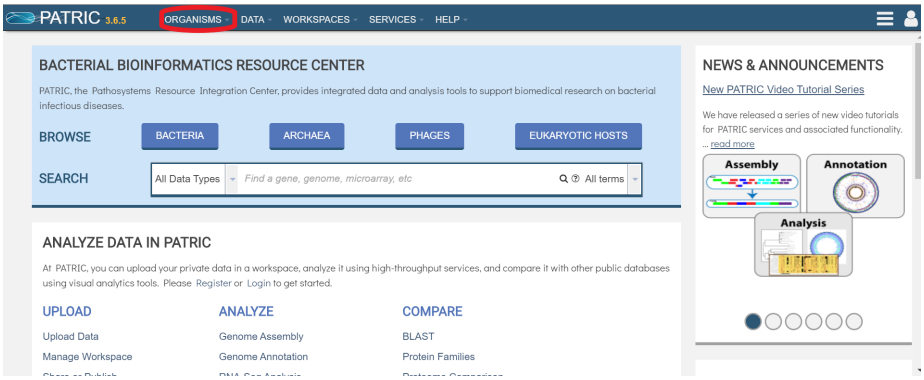
`s_dir or --spresi_dump_directory` for spresi data (rdf files) with the `--SPRESI` option and

`-mc or --metacyc_addition` (which is the xml (metabolic-reactions.xml) file for the metacyc reactions repository) with the –metacyc option.

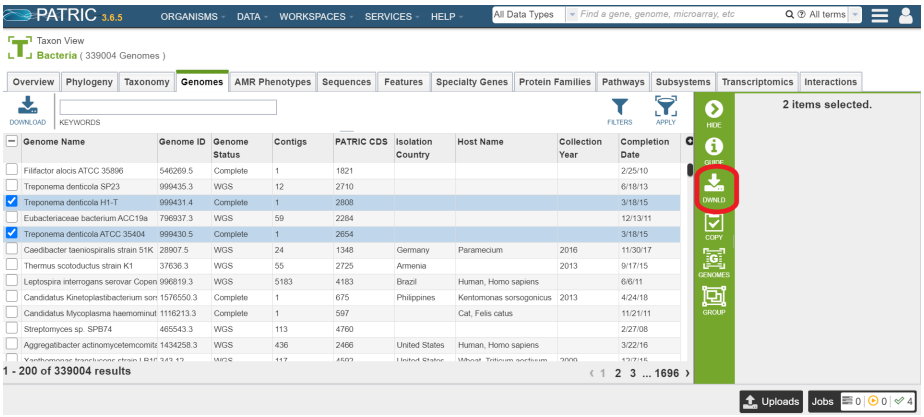
The `--kegg` option does not require a directory specification as RetSynth connects directly to the kegg database.

Additionally the `--patric_models` option connects directly to the patric server using the mackinac python library which was integrated into RetSynth. In order to use patric the user must input a patric username `--patric_username` and patric password `--patric_password`. Note the user can select which patric models are integrated into the RetSynth database (defualt is currently Escherichia coli DH1, Streptomyces venezuelae ATCC 10712 and Corynebacterium argentoratense DSM 44202) by utilizing the `--patricfile`. The `--patricfile` option is a csv file which tells which RetSynth which genomes from patric to download into the database. In order to generate your own list of genomes:

1. go to PATRIC and click on `organisms` then `All Bacteria`. Click on the `Genomes` tab and select the genomes you want to include in the database.



2. Click the download button selecting the csv option.



If the user is constructing a database using multiple repositories it is suggested that the `--inchidb` option is specified although this option drastically increases the time required to build the database. This converts all compound IDs to their inchi values and therefore elimiates duplicate entries of metabolites and reactions from the different metabolic repositories. If this option is not selected KEGG compound IDs will be used. Additionally, the user must designate a database name with the option `-gdb or --generate_database`. In the event that a database has already been built and the user wants to use that database instead of building a new one, the database can be accessed by utilizing the option `-db or --database` with the name of the database.

2. Generating Stoichiometric Matrix for Reactions and Compounds in the Metabolic Database¶

To find the minimal reactions needed to optimally produce a compound, a stoichiometric matrix for all compounds and reactions in a database is required. The stoichiometric matrix is a mathmatical representation of the reactions and the compounds they degrade and synthesize. If a stoichiomet-

ric has not been generated for a database the user must specify `-gdbc or --generate_database_constraints` with the file name which will generate .constraints file.

Because the .constraints file does not have to be recreated if the user wants to perform multiple searchs for multiple target compounds within the same database, the user can specify the option `-dbc or --database_constraints` with the .constraints file name. This option saves time as initial construction of constraints file is time-consuming.

Note

Each constructed database requires its own .constraints file. Additionally if a database has both chemical and biological reactions in it and the user only wants say biological reactions it is good to generate a new constraints file (i.e. constraintfile_bio.constraints) with the `-gdbc` option.

3. Target Compounds (input file)¶

RetSynth also requires a list of compounds and the organism they are desired to be synthesized

The target compound ID, pubchem ID, inchi value or name (required)

2. The desired organism ID or organism name (optional, however, if not specified the software proceeds to identify the number of reactions that need to be added to each organism in the database to produce the target)
3. Reaction IDs that the user does not want the software to identify as a potential method of producing the target (optional)

Example input file 1:

```
#compoundid   organismID    ignore reactions
cpdT organism_name  rxn10
```

Example input file 2:

```
#pubchem organism
263   organism_name
```

Example input file 3:

```
#name organismid
1-butanol   organism_name1, organism_name2
```

Example input file 4:

```
#name organismid
InChI=1S/C4H10O/c1-2-3-4-5/h5H,2-4H2,1H3  organism_name1, organism_name2
```

4. Examples running RetSynth¶

Examples for running RetSynth¶

Example 1: If database and constraint file have not already been generated:

```
from RetSynthGC import RetSynthGC
RetSynthGC(targets=targetfile.txt, generate_database=databasefile.db, generate_database_cons
or
rs_gc.py -t targetfile.txt -gdb databasefile.db -gdbc databasefile.constraints --patric_mode
```

Example 2: If database and constraint file have been generated:

```
from RetSynthGC import RetSynthGC
RetSynthGC(targets=targetfile.txt, database=databasefile.db, database_constraints=databasefi
or
rs_gc.py -t targetfile.txt -db databasefile.db -dbc databasefile.constraints
```

Example 3: If database has been generated but constraint file has not been generated:

```
from RetSynthGC import RetSynthGC
RetSynthGC(targets=targetfile.txt, database=databasefile.db, generate_database_constraints=c
or
rs_gc.py -t targetfile.txt -db databasefile.db -gdbc databasefile.constraints
```

The remainder of this documentation goes over other options that can be added for further results and analysis provided by RetSynth.

Results¶

An output file is generated named optimal_pathways.txt which gives the number pathways found which could synthesize the target compound with the mimimal number of reactions.

Example optimal_pathways.txt file:

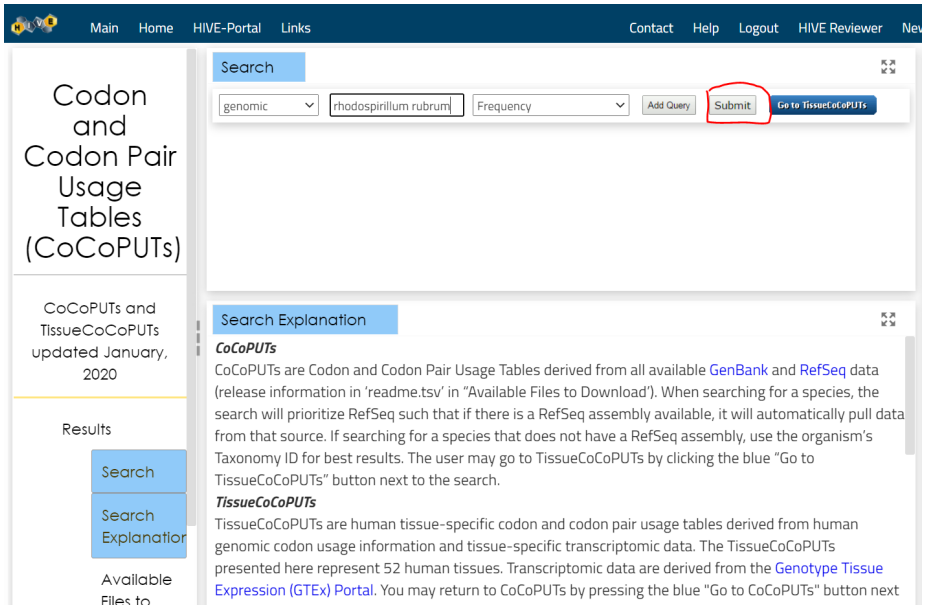
```
SHORTEST PATHS FOR cpdT_c0 cpdT_c0 in target organism test1.xml
Solution 1
rxn7_c0  rxn7_c0  forward  None  1 number of species that contain this reaction
   cpdZ_c0  cpdI_c0 reactant
   cpdE_c0  cpdB_c0 product
rxn8_c0  rxn8_c0  forward  None  1 number of species that contain this reaction
   cpdE_c0  cpdB_c0 reactant
   cpdX_c0  cpdX_c0 product
rxn11_c0 rxn11_c0 forward  None  1 number of species that contain this reaction
   cpdY_c0  cpdY_c0 reactant
   cpdT_c0  cpdT_c0 product
rxn9_c0  rxn9_c0  forward  None  1 number of species that contain this reaction
   cpdX_c0  cpdX_c0 reactant
   cpdW_c0  cpdW_c0 product
```

```
rxn10_c0 rxn10_c0 forward  None  1 number of species that contain this reaction
   cpdW_c0  cpdW_c0 reactant
   cpdY_c0  cpdY_c0 product
```

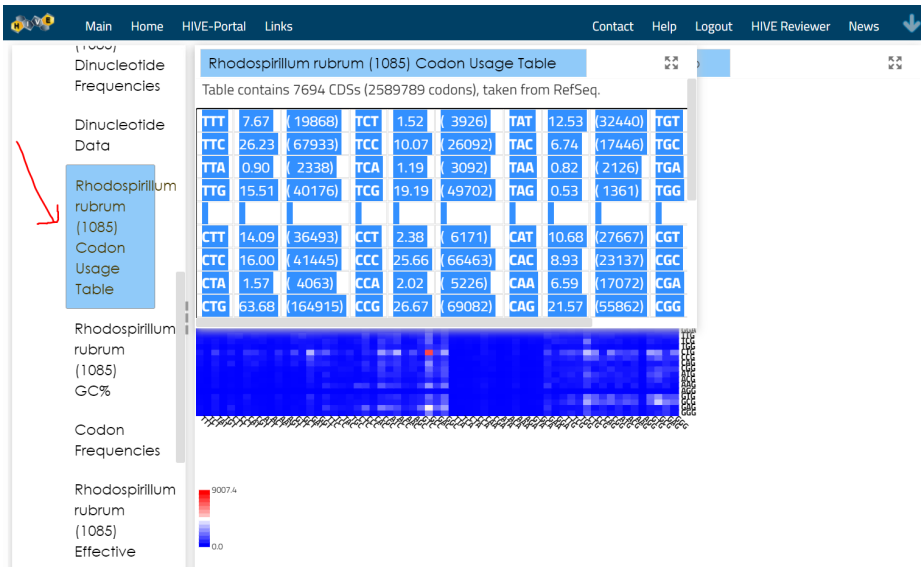
Gene Compatability (GC)¶

RetSynth has a built in module which after identifying a novel path to compound production will identify the optimal gene sequences for each gene/enzyme in the novel pathway. A host microbial organism will have a unique codon bias, meaning that the organism better translates particular codon sequences to amino acids. By optimizing the codon sequences to match the host organisms codon bias we can increase efficiency of gene experession of the genes/enzymes in the novel pathway and subsequently optimal production of the target compound. The software comes with 4 organism codon usage tables which include Escherichia coli DH1, Streptomyces venezuelae ATCC 10712, Corynebacterium argentoratense DSM 44202 and Rhodospirillum rubrum.

The user can generate there own codon bias usage table by: 1) going to the HIVE-CUTS database and inputing the name of the organism they wish to create a table for in the search box and clicking the submit button.



2. Click on codon usage bias table tab and copy and paste the table into a text file and then save that text file



3. Then specify the path to this file using the `user_cai_table` option when running the code

To implement the gene compatability feature:

```
from RetSynthGC import RetSynthGC
RetSynthGC(targets=targetfile.txt, database=databasefile.db, database_constraints=databasefi
or
rs_gc.py -t targetfile.txt -db databasefile.db -gdbc databasefile.constraints -gc -cai_thres
```

The codon adaptation index (CAI) is a measure of how close the gene sequence is to a given reference set (in our case the host organism)

Flux Balance Analysis (FBA)¶

RetSynth, using the FBA tool CobraPy, can simulate the metabolic activity of an organism and calculated the maximum theoretical yield of a target compound using the identified enzyme/reaction pairs.

Note

Currently FBA can only be run on organisms added from PATRIC.

To run FBA the `-fba or --flux_balance_analysis` option must be specified. Using the option `-media or --media_for_FBA` will specify that the organism's metabolism will be simulated on a user specified media. Currently these media files can also be obtained from KBase. Finally, the user can perform knockouts, `-ko or --knockouts` of each reaction in the organism, which will help in the understanding of which enzymes, reactions and metabolic pathways are important for synthesis of the target compound.

FBA Results¶

Several files are generated when the `-fba` option is run. flux_output.txt is the main output file which gives the objective function values before and after the reactions identified in the previous step are added to the organism. Additionally, reactions whose flux changes 2.5 fold from when metabolism is simulated prior to the reactions being add to after they are added are shown in this output document.

Example flux_output.txt:

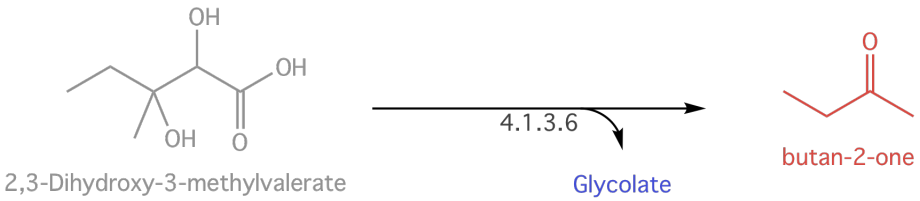
```
FBA Solutions for cpdT_c0
0.0   1000.0 objective function solutions for wild-type and mutant, respectively

Fluxes that differ by 1.5 fold for reactions between wildtype and mutant:
      wildtype flux  mutantflux
   Trans_Z_c0  0.0   1000.0
   EX_I_e0  0.0   -1000.0

Fluxes for added reactions in mutant:
   rxn8_c0  rxn8_c0  1000.0
   rxn7_c0  rxn7_c0  1000.0
   rxn10_c0 rxn10_c0 1000.0
   rxn11_c0 rxn11_c0 1000.0
   Sink_E2  None  1000.0
   rxn9_c0  rxn9_c0  1000.0

External pathway with most flux:
   Path 1   ['rxn7_c0', 'rxn8_c0', 'rxn11_c0', 'rxn9_c0', 'rxn10_c0']
   Total flux through path: 5000.0
```

Additionaly, theoretical_yield.txt file is generated which gives the percent theoretical yields of the target compounds produced in a select organism. (Note: this is only accurate if `-fba` is run on with `-media` option)

If the `-ko` parameter is used, essentialrxns_output.txt file is generated which lists all the reactions, which when knocked out prevent production of a target compound.

Visualization_chemdraw¶

This module generates figures of the pathways needed to synthesize the target compound in chemdraw (cdxml) format. To use this module the user must specify the optin `figures_chemdraw`. Additionally, if chemical reactions have been added to the database and the user wants all chemical reaction information to be shown in the figure (i.e solvent, catalyst, pressure...) then the user must specify `--show_chemical_rxn_info` option.



Gray compounds indicate the compound is native to the target organism

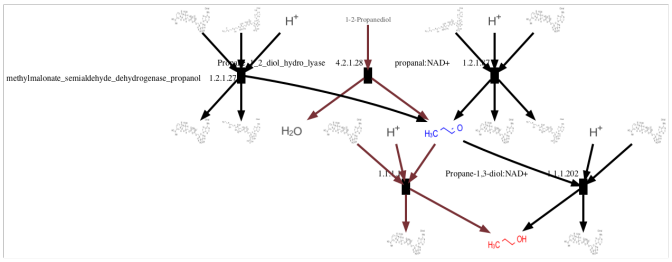
Black square nodes show reactions that need to be added to target organism

Blue compounds indicate that the compound is not native to the target organism

Red compound indicates the target compound

Visualization_graphviz¶

This module, which requires GraphViz , generates graphs of the reactions that are required to synthesize the compound in a desired organism. To use this module the user must specify the option `--figures_graphviz` . Additionaly if the user wants the chemical structures to be used in the figures instead of round nodes option `--images` must be specified. Example Figure:



This figure shows the reactions and compounds needed to produce a target compound (1-propa

Gray compounds indicate the compound is native to the target organism

Black square nodes show reactions that need to be added to target organism

Blue compounds indicate that the compound is not native to the target organism

Red compound indicates the target compound

In this case there are a multiple pathways (with the same minimal number of reactions) to get to 1-propanol in E. Coli. The red edges indicate that those are the reactions that produce the maximum theoretical yield of the target compound.

Visualize results in html¶

This option `output_html` will output all results into an html file (Results.html in the output_path folder) which is easy to view all pathways for each of the target compounds. Note that this can only integrate figures from the `figures_graphviz` option.

Example of how to get Results.html file:

```
RetSynthGC(targets=targetfile.txt, database=databasefile.db, database_constraints=databasefi
or
rs.py -t targetfile.txt -db databasefile.db -dbc databasefile.constraints --figures_graphviz
```

5. Other options in RetSynth¶

In RetSynth there are other options the user can specify to make their searches specific to their needs.

1. The option `--tan_thresh or --tanimoto_threshold` can be used if the inchi value is being used in the targetfile to specify the target compounds and if the `--inchidb` option is specified when generating a database(must be specifed when using option `-gdb` or when using an already developed database `-db`). This option tells RetSynth to find pathways for all compounds in the database that have a structure similarity of the specified tanimoto score or greater (default 1) to the target compound in the target file.
2. The option `-k or --k_number_of_paths` specifies the number of shortest pathways (pathways with mimal number of reaction/enzyme pairs) wanted by the user. If the user specified `-k 1` that would tell RetSynth to identify the shortest pathways and then next shortest pathways, giving the user pathways of 2 different lengths to the target compound where as `-k 2` would result in pathways of three different lengths. The default for -k is 0.
3. The options `-ms or --multiple_solutions` and `-cy or --cycles` indicate to RetSynth that all pathways having the minimal number of steps should be identified and that these pathways should not contain cycles. Both of these option defaults are set true.

Indices and tables¶

- Welcome to RetSynth's documentation!
- Parameter Documentation
- RetSynth Modules

Navigation

- RetSynth 1 documentation »

- RetSynth 1 documentation

© Copyright 2019, Leanne Whitmore, Bernard Nguyen, Corey Hudson. Created using Sphinx 3.1.2.