coralME Documentation

Release 1.0

Juan D. Tibocha Bonilla, Rodrigo Santibanez-Palominos

Aug 29, 2023

CONTENTS

CHAPTER ONE

DESCRIPTION

The **COmprehensive Reconstruction ALgorithm for ME-models (coralME)** is an automatic pipeline for the reconstruction of ME-models. coralME integrates existing ME-modeling packages COBRAme, ECOLIme, and solveME, generalizes their functions for implementation on any prokaryote, and processes readily available organism-specific inputs for the automatic generation of a working ME-model.

coralME has four main objectives:

1. **Synchronize** input files to remove contradictory entries.
2. **Complement** input files from homology with a template organism to complete the E-matrix.
3. **Build** a working ME-model
4. **Inform** the user about necessary steps to curate the ME-model.

This resource is intended to:

1. Describe basic inputs required for ME-model reconstruction.
2. Describe the architecture of coralME.
3. Demonstrate how to build a ME-model with coralME.
4. Describe how to perform manual curation guided by coralME's curation notes.

CHAPTER

TWO

CONTENT

2.1 Getting started

For details on inputs go to *Description of Inputs*.

For information about coralME architecture go to *Arquitecture of coralME*.

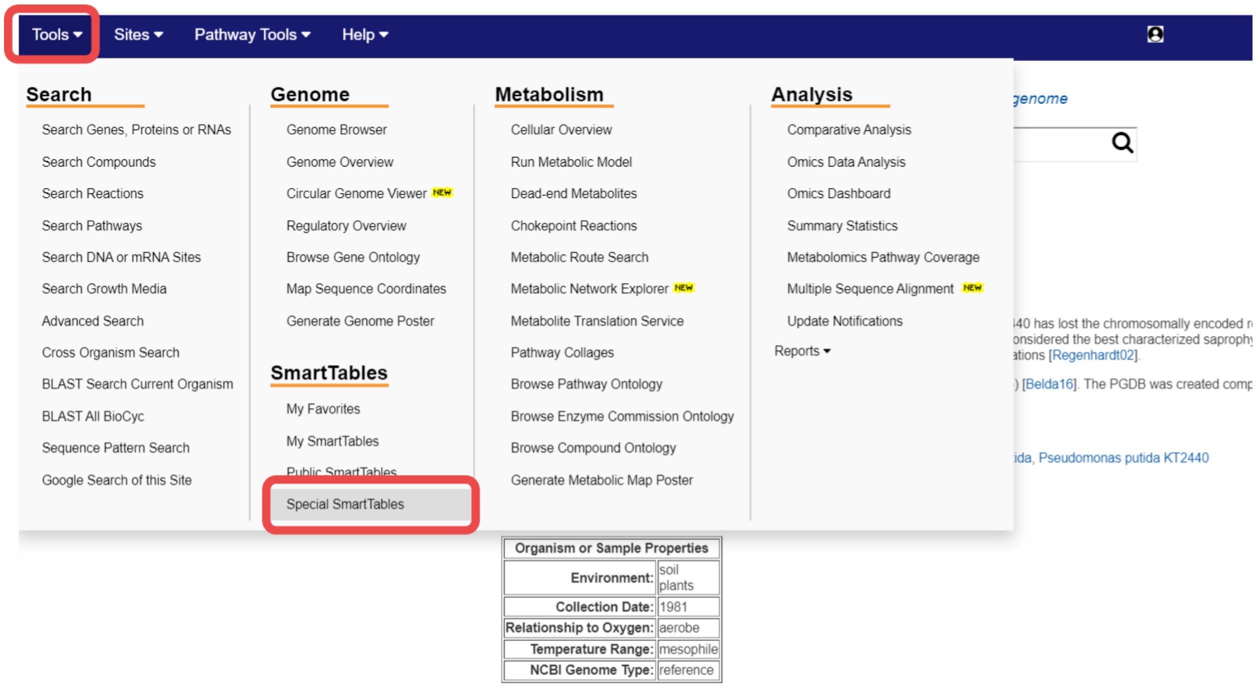
2.1.1 Download files from BioCyc

BioCyc files are optional but useful, you should download them after having your gene id consistent M-model and genbank files.

The quickest way to do this is to copy one of the genes from the genbank file into the BioCyc search bar. Your organism should appear in the list if it is available in BioCyc.

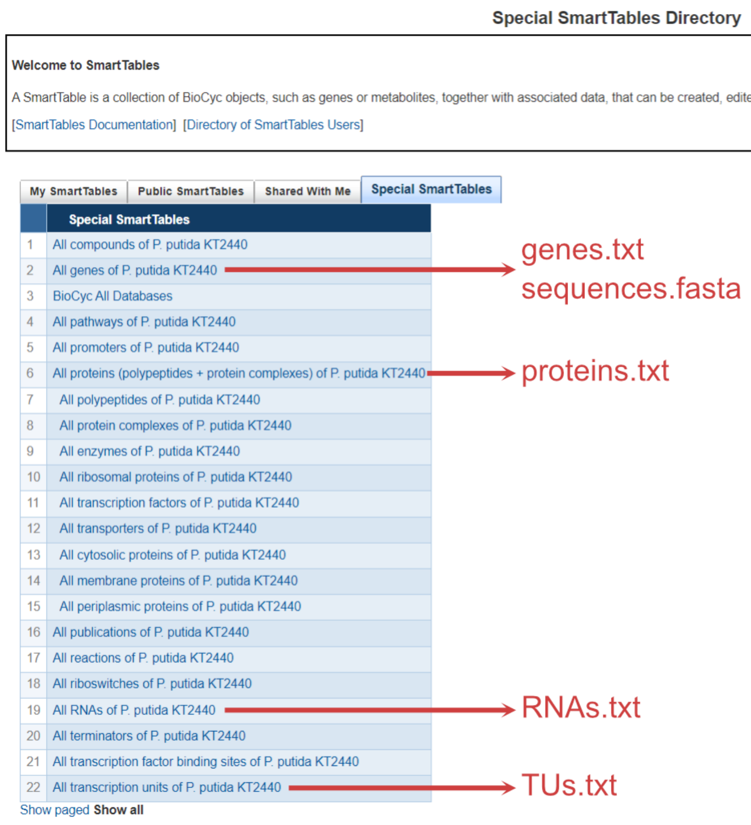
To download:

Go to Tools>Special SmartTables



Organism or Sample Properties	
Environment:	soil plants
Collection Date:	1981
Relationship to Oxygen:	aerobe
Temperature Range:	mesophile
NCBI Genome Type:	reference

From here you can download the 5 optional BioCyc files for your organism.



Special SmartTables Directory

Welcome to SmartTables

A SmartTable is a collection of BioCyc objects, such as genes or metabolites, together with associated data, that can be created, edited, manipulated, and shared on the web.

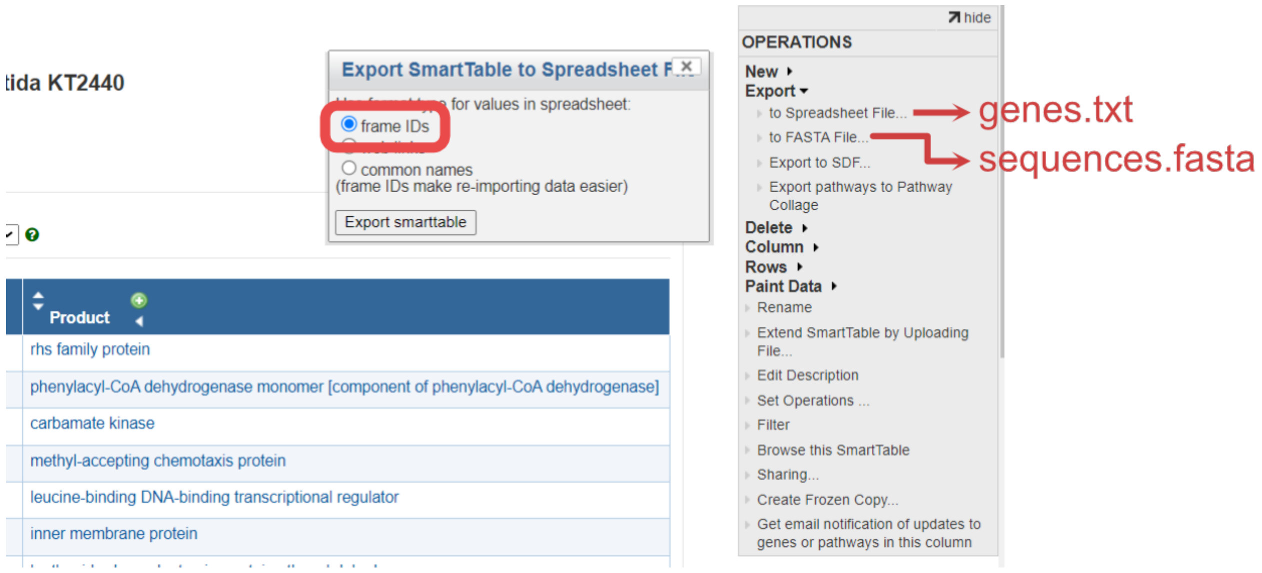
[SmartTables Documentation] [Directory of SmartTables Users]

My SmartTables | Public SmartTables | Shared With Me | Special SmartTables

	Special SmartTables
1	All compounds of P. putida KT2440
2	All genes of P. putida KT2440
3	BioCyc All Databases
4	All pathways of P. putida KT2440
5	All promoters of P. putida KT2440
6	All proteins (polypeptides + protein complexes) of P. putida KT2440
7	All polypeptides of P. putida KT2440
8	All protein complexes of P. putida KT2440
9	All enzymes of P. putida KT2440
10	All ribosomal proteins of P. putida KT2440
11	All transcription factors of P. putida KT2440
12	All transporters of P. putida KT2440
13	All cytosolic proteins of P. putida KT2440
14	All membrane proteins of P. putida KT2440
15	All periplasmic proteins of P. putida KT2440
16	All publications of P. putida KT2440
17	All reactions of P. putida KT2440
18	All riboswitches of P. putida KT2440
19	All RNAs of P. putida KT2440
20	All terminators of P. putida KT2440
21	All transcription factor binding sites of P. putida KT2440
22	All transcription units of P. putida KT2440

Show paged **Show all**

Download genes.txt and sequences.fasta



tida KT2440

Export SmartTable to Spreadsheet F...

[illegible] for values in spreadsheet:

[✓ frame IDs]

[illegible]

common names

(frame IDs make re-importing data easier)

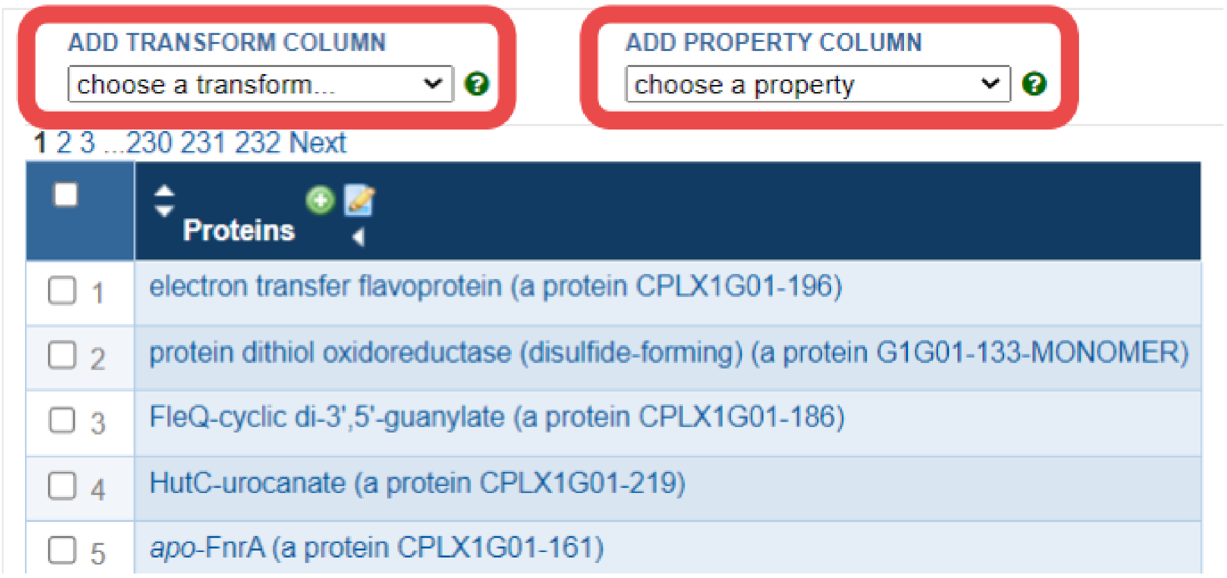
Export smarttable

Product
rhs family protein
phenylacyl-CoA dehydrogenase monomer [component of phenylacyl-CoA dehydrogenase]
carbamate kinase
methyl-accepting chemotaxis protein
leucine-binding DNA-binding transcriptional regulator
inner membrane protein

Download proteins.txt, RNAs.txt and TUs.txt

The same process of genes.txt applies to proteins.txt, RNAs.txt and TUs.txt.

Some columns must be added manually using BioCyc's dropdown lists **ADD PROPERTY COLUMN** and **ADD TRANSFORM COLUMN** within the SmartTable editing webpage.



	Proteins
1	electron transfer flavoprotein (a protein CPLX1G01-196)
2	protein dithiol oxidoreductase (disulfide-forming) (a protein G1G01-133-MONOMER)
3	FleQ-cyclic di-3',5'-guanylate (a protein CPLX1G01-186)
4	HutC-urocanate (a protein CPLX1G01-219)
5	*apo*-FnrA (a protein CPLX1G01-161)

Download proteins.txt

The index (Proteins Complexes) is in the SmartTable by default, but you need to add the columns Common-Name, Genes of polypeptide, complex, or RNA, and Locations.

- Common-Name is available in the dropdown list **ADD PROPERTY COLUMN**
- Genes of polypeptide, complex, or RNA is available in the dropdown list **ADD TRANSFORM COLUMN**
- Locations is available in the dropdown list **ADD PROPERTY COLUMN**.

Download RNAs.txt

The index (All-tRNAs Misc-RNAs rRNAs) is in the SmartTable by default, but you need to add the columns Common-Name, and Gene.

- Common-Name is available in the dropdown list **ADD PROPERTY COLUMN**
- Gene is available in the dropdown list **ADD PROPERTY COLUMN**.

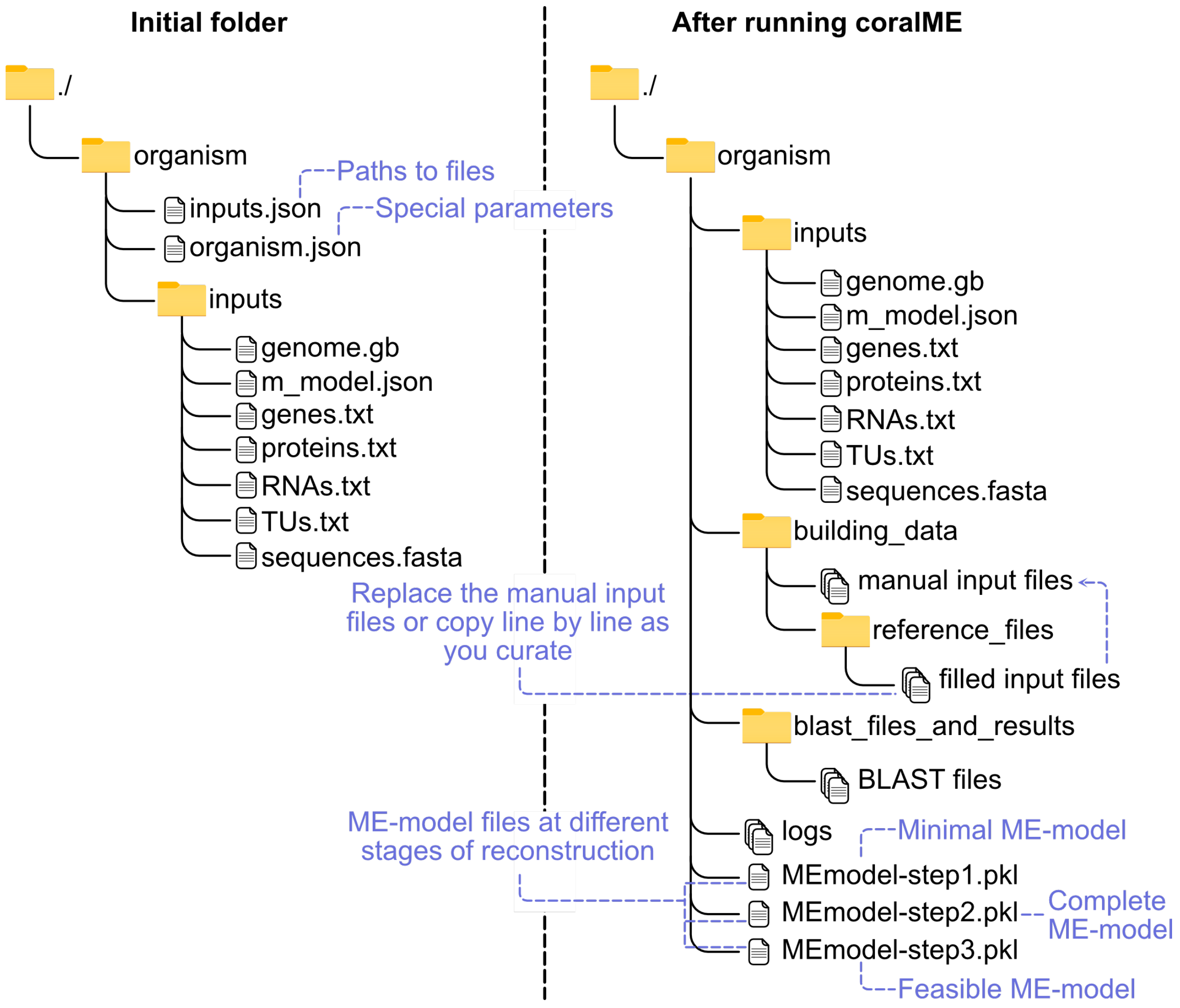
Download TUs.txt

The index Transcription-Units is in the SmartTable by default, but you need to add the columns Genes of transcription unit, and Direction.

- Genes of transcription unit is available in the dropdown list **ADD TRANSFORM COLUMN**
- Direction is available in the dropdown list **ADD PROPERTY COLUMN**.

2.1.2 Initialize the folder for your organism

Copy your files to create your initial folder



Define inputs in input.json.

See an example of input.json

Define parameters in organism.json.

See an example of organism.json

Note: You do not need to modify these parameters right away. But once you are at the curation stage you will have to ensure these parameters are applicable to your organism.

2.1.3 Reconstruct with coralME

Here we show an example to reconstruct a dME-model of *B. subtilis*

Import packages

```
[ ]: from coralme.builder.main import MEBuilder
```

Define organism and inputs

```
[ ]: org = "./helper_files/tutorial/"
```

Load configuration files

```
[ ]: import os
     os.chdir(org)
```

```
[ ]: organism = './organism.json'.format(org)
     inputs = './input.json'.format(org)
```

Create builder

`class coralme.builder.main.`**`MEBuilder`**`(`*`*args`*`, `*`**kwargs`*`)`

MEBuilder class to coordinate the reconstruction of ME-models.

Parameters

- **`*args`** – Positional arguments are passed as paths to JSON files that update the configuration of the parent class.
- **`**kwargs`** – Further keyword arguments are passed on as dictionaries to update the configuration of the parent class.

`generate_files``(`*`overwrite=True`*`)`

Performs the Synchronize and Complement steps of the reconstruction.

This function will read the Organism and the Reference. It will synchronize the input files, complement them, and finally build the OSM for the Organism.

Parameters
`overwrite` (*bool*) – If True, overwrite the OSM using the defined path in the configuration.

`get_homology`(*evalue=1e-10*)
Calculates homology between Organism and Reference.

Parameters
`evalue` (*float, default 1e-10*) – Sets the E-value cutoff for calling protein homologs using BLAST.

`get_trna_to_codon`()
Gets tRNA to codon association from the Genome.

`prepare_model`()
Performs initial preparation of the M-model.

This function will fix some known issues that M-models can

Parameters
`overwrite` (*bool*) – If True, overwrite the OSM using the defined path in the configuration.

`troubleshoot`(*growth_key_and_value=None*, *skip={}*, *platform=None*, *solver='gurobi'*)

growth_key_and_value:
dictionary of Sympy.Symbol and value to replace

skip:
set of ME-components to not evaluate during gapfilling

platform:
'win32' to use gurobi (default) or cplex as solver

solver:
'gurobi' (default, if platform is 'win32') or 'cplex'

```
[ ]: builder = MEBuilder(*[organism, inputs])
```

Generate files

```
[ ]: builder.generate_files(overwrite=True)
```

Build ME-model

```
[ ]: builder.build_me_model(overwrite=False)
```

Troubleshoot ME-model

```
[ ]: builder.troubleshoot(growth_key_and_value = { builder.me_model.mu : 0.001 })
```

Note: We set 0.001 as a standard value for feasibility checking, but feel free to modify it! Sometimes too high a value could put a significant strain on the model and give too many gaps to start with. Too low a value might not show you all the gaps needed.

2.1.4 Curate manually

For details on manual curation go to *How to manually curate a ME-model using coralME*.

2.2 Description of Inputs

coralME takes a total of 7 inputs, 2 required and 5 optional:

2.2.1 Types of inputs

Required

1. **Genome file** (**genome.gb**)
2. **M-model** (**m_model.json** or **m_model.xml**)

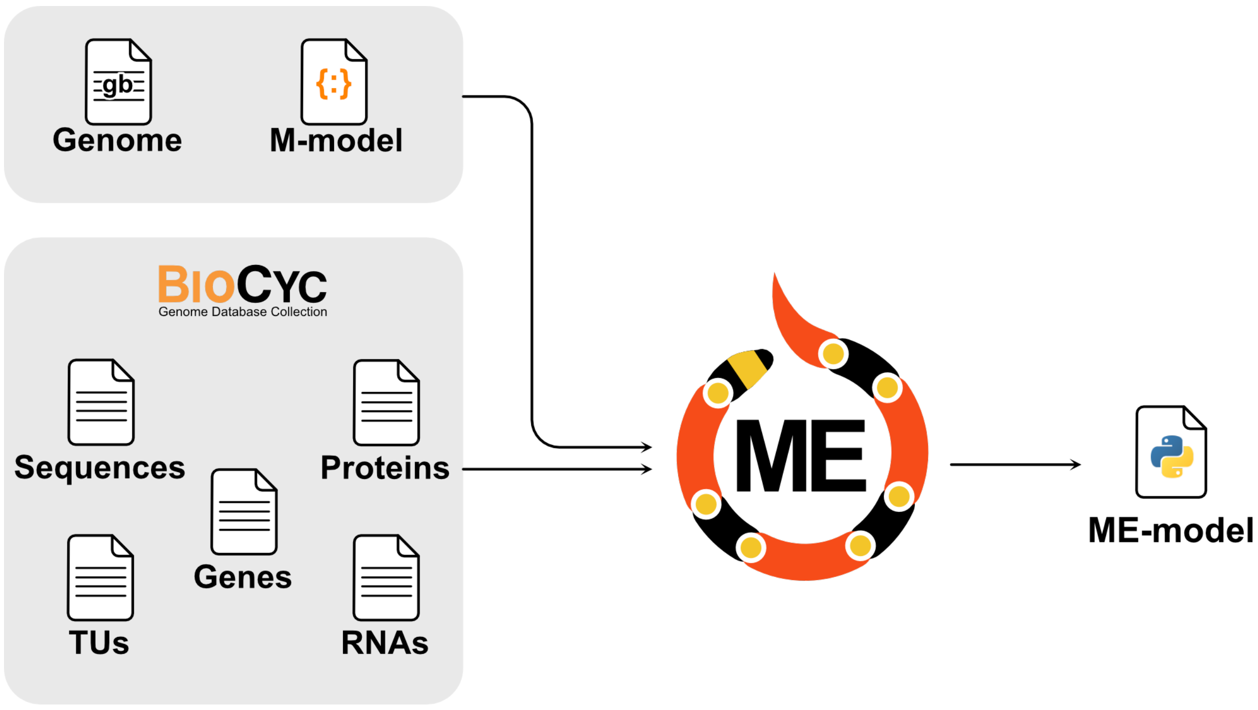
Optional

Downloadable from an existing **BioCyc** database under **Special SmartTables**. If no optional files are provided, coralME complements them with **genome.gb**

3. **Genes file**, by default: **genes.txt**
4. **RNAs file**, by default: **RNAs.txt**
5. **Proteins file**, by default: **proteins.txt**
6. **TUs file**, by default: **TUs.txt.**
7. **Sequences file**, by default: **sequences.fasta**

Configuration

8. **Paths file**, by default: **inputs.json**
9. **Parameters file**, by default: **organism.json**



2.2.2 Description

Genome (genome.gb)

Description

The genome file contains provides coralME with:

- Gene annotations.
- Gene sequences.

Requirements

1. Locus tags (locus_tag or old_locus_tag) MUST be consistent with **m_model.json**. Make sure you download the same genome file that was used to reconstruct the M-model.
2. Has name **genome.gb**.
3. Genbank-compliant file. Must be read by BioPython correctly.
4. It must contain the entire genome sequence. Make sure to enable **Customize View>Show Sequence** before downloading the genbank file from NCBI.

See an example of genome.gb and sequences.fasta

M-model (m_model.json)

Description

The M-model provides coralME with the metabolic model components:

- Metabolic network (M-matrix)
- Gene-protein-reaction associations
- Environmental and internal constraints
- Reaction subsystems
- Biomass composition

Requirements

1. Gene identifiers MUST be consistent with **genome.gb** locus_tag or old_locus_tag. Make sure you download the same genome file that was used to reconstruct the M-model.
2. Has name **m_model.json**.
3. COBRApy-compliant. Must be read by cobrapy-0.25.0.

See an example of m_model.json

Gene dictionary (genes.txt) [optional]

Description

genes.txt is a gene information table that can be downloaded from the **All genes of organism SmartTable** of the BioCyc database. Click **Export>to Spreadsheet File>frame IDs**. This file is optional and is meant to complement the information from **genome.gb** in case the latter is missing genes.

genes.txt provides coralME with:

- Gene locus tags
- Gene names
- Gene annotations
- Gene positions
- Gene products (protein, tRNA, etc.)

Requirements

1. Contains the index **Gene Name** and columns **Accession-1**, **Left-End-Position**, **Right-End-Position**, and **Product**.
2. **Accession-1** MUST be consistent with the gene IDs in the GPRs of **m_model.json** and with the locus_tag (or old_locus_tag) in **genome.gb**.
3. **Gene Name** is consistent with:
   - Column **Genes of polypeptide, complex, or RNA** of **proteins.txt**
   - Column **Gene** of **RNAs.txt**

  - Column **Genes of transcription unit** of **TUs.txt**
  - Gene identifiers in **sequences.fasta**

4. **Product** is consistent with:
  - Index of **proteins.txt**
  - Index of **RNAs.txt**
5. Must be tab-separated

See an example of genes.txt

Note: Requirement 2 regarding ID consistency should be directly met if the files are downloaded from the correct BioCyc database.

Note: Requirements 3, 4 and 5 regarding ID consistency should be directly met if the files are downloaded from the same BioCyc database.

Proteins (proteins.txt) [optional]

Description

proteins.txt is a protein complex information table that can be downloaded from the **All proteins of organism SmartTable** of the BioCyc database. Click **Export>to Spreadsheet File>frame IDs**. This file is optional and is meant to complement the information from **genome.gb**.

proteins.txt provides coralME with: * Protein complex compositions

Requirements

1. Contains the index **(Proteins Complexes)** and columns **Common-Name**, **Genes of polypeptide, complex, or RNA**, and **Locations**.
2. **(Proteins Complexes)** is consistent with:
  - Column **Product** of **genes.txt**
3. **Genes of polypeptide, complex, or RNA** is consistent with:
  - Index **Gene Name** of **genes.txt**
4. Must be tab-separated

See an example of proteins.txt

Note: Requirements 2, 3 and 4 regarding ID consistency should be directly met if the files are downloaded from the same BioCyc database.

RNAs (RNAs.txt) [optional]

Description

RNAs.txt is an RNA annotation table that can be downloaded from the **All RNAs of organism SmartTable** of the BioCyc database. Click **Export>to Spreadsheet File>frame IDs**. This file is optional and is meant to complement the information from **genome.gb**.

RNAs.txt provides coralME with:

- Genes annotated as RNA products (e.g. tRNA, rRNA, etc.)
- RNA gene annotations (e.g. amino acids - tRNA associations)

Requirements

1. Contains the index **(All-tRNAs Misc-RNAs rRNAs)** and columns **Common-Name**, and **Gene**
2. **(All-tRNAs Misc-RNAs rRNAs)** is consistent with:
    - Column **Product** of **genes.txt**
3. **Gene** is consistent with:
    - Index **Gene Name** of **genes.txt**
4. Must be tab-separated

See an example of RNAs.txt

Note: Requirements 2, 3 and 4 regarding ID consistency should be directly met if the files are downloaded from the same BioCyc database.

TUs (TUs.txt) [optional]

Description

TUs.txt is a transcription unit annotation table that can be downloaded from the **All TUs of organism SmartTable** of the BioCyc database. Click **Export>to Spreadsheet File>frame IDs**. This file is optional and is meant to complement the information from **genome.gb**.

TUs.txt provides coralME with:

- Co-transcribed genes (operons).
- Direction of transcription.
- TU IDs.

Requirements

1. Contains the index **Transcription-Units** and columns **Genes of transcription unit**, and **Direction**
2. **Genes of transcription unit** is consistent with:
    - Index **Gene Name** of **genes.txt**
3. Must be tab-separated

See an example of TUs.txt

Note: Requirements 2 and 3 regarding ID consistency should be directly met if the files are downloaded from the same BioCyc database.

Gene sequences (sequences.fasta) [optional]

Description

sequences.fasta is a nucleotide FASTA file that can be downloaded from the **All genes of organism SmartTable** of the BioCyc database. Click **Export>FASTA>Find sequences**. This file is optional and is meant to complement the information from **genome.gb** in case the latter is missing genes.

sequences.fasta provides coralME with:

- Gene sequences

Requirements

1. Gene identifiers are consistent with:
    - Index **Gene Name** of **genes.txt**
2. Must be tab-separated

See an example of sequences.fasta

Note: Requirements 1, 2 and 3 regarding ID consistency should be directly met if the files are downloaded from the same BioCyc database.

Configuration of paths to files (inputs.json)

Description

inputs.json is a JSON file containing paths to input files for coralME.

inputs.json provides coralME with:

- Paths to input files

Requirements

1. Must be JSON-compliant
2. Must contain paths to required files (M-model and Genome).
3. All defined files must exist.

See an example of input.json

Configuration of parameters (organism.json)

Description

organism.json is a JSON file containing paths to input files for coralME.

organism.json provides coralME with:

- ME-modeling parameters

Requirements

1. Must be JSON-compliant
2. Must contain the standard fields.

See an example of organism.json

2.3 Arquitecture of coralME

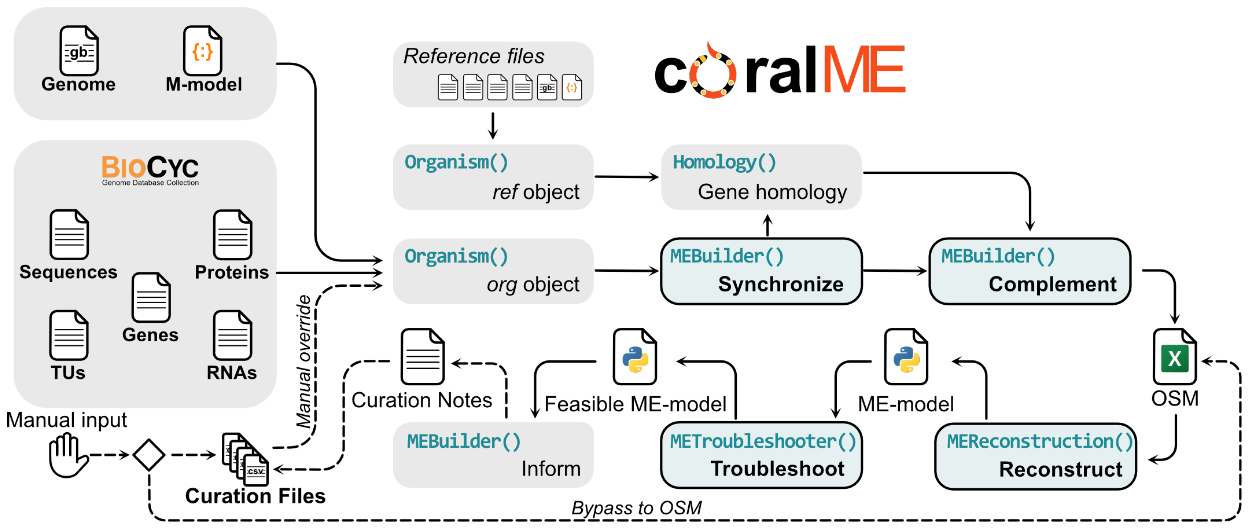
coralME is composed of 4 main classes that process and exchange organism-specific information for the reconstruction of a ME-model. The classes are:

```
class Organism()

class MEBuilder()

class MEReconstruction()

class Homology()
```



2.3.1 Organism()

`class` `coralme.builder.organism.`**`Organism`**(*config*, *is_reference*)

Organism class for storing information about an organism

This class acts as a database containing all necessary information to reconstruct a ME-model. It is used to retrieve and store information of the main (org) and the reference (ref) organisms. Information in Organism is read and manipulated by methods in the MEBuilder class.

Parameters

- **`config`** (*`dict`*) – Dictionary containing configuration and settings.
- **`is_reference`** (*`bool`*) – If True, process as reference organism.

Role: Store information about an organism

This class acts as a database containing all necessary information to reconstruct a ME-model. It is used to retrieve and store information of the main (**org**) and the reference (**ref**) organisms. Information in Organism() is read and manipulated by methods in the MEBuilder() class. The reference can be set as any of the provided organisms in coralME, available here, although we advise to choose *E. coli* and *B. subtilis* for gram-negative and gram-positive bacteria, respectively.

2.3.2 MEBuilder()

`class` `coralme.builder.main.`**`MEBuilder`**(**args*, ***kwargs*)

MEBuilder class to coordinate the reconstruction of ME-models.

Parameters

- **`*args`** – Positional arguments are passed as paths to JSON files that update the configuration of the parent class.
- **`**kwargs`** – Further keyword arguments are passed on as dictionaries to update the configuration of the parent class.

Role: Coordinate the roles of other classes.

This class acts as the main coordinator between other objects, e.g. Organism, Homology, MEProcesser, and METroubleshooter. It contains methods to manipulate class Organism by using attributes in class Homology, and manually curated files in the folder containing the main organism. Moreover, it is called by objects to access stored information in other objects.

2.3.3 MEReconstruction()

`class coralme.builder.main.`**`MEReconstruction`**`(`*builder*`)`

MEReconstruction class for reconstructing a ME-model from user/automated input

Parameters

`MEBuilder` (`coralme.builder.main.MEBuilder`) –

Role: Reconstruct a ME-model from the information contained in class Organism.

This class was based almost entirely from the original ECOLIme code in build_me_model.py. Adaptations to this code were necessary to make it applicable to other organisms.

2.3.4 Homology()

`class coralme.builder.homology.`**`Homology`**`(`*org*, *ref*, *evalue=False*, *verbose=False*`)`

Homology class for storing information about homology of the main and reference organisms.

This class contains methods to predict and process homology of the main and reference organisms. Homology is inferred from the reciprocal best hits of a BLAST. The results are used to update and complement the attributes of the class Organism.

Parameters

- **`org`** (*`str`*) – Identifier of the main organism. Has to be the same as its containing folder name.
- **`ref`** (*`str`*) – Identifier of the reference organism. Has to be the same as its containing folder name.
- **`evalue`** (*`float`*) – E-value cutoff to call enzyme homologs from the BLAST. Two reciprocal best hits are considered homologs if their E-value is less than this parameter.

Role: Generate and store information about homology of the main and reference organisms.

This class contains methods to predict and process homology of the main and reference organisms. Homology is inferred from the reciprocal best hits of a BLAST. The results are used to update and complement the attributes of the class Organism.

2.4 How to manually curate a ME-model using coralME

2.4.1 Manual input files

After you run coralME for the first time the following files are generated in building_data/</code>. Most of them are also automatically filled by the algorithm and saved in building_data/reference_files/</code>. These _reference **files_** are meant to guide manual curation as they contain all information mapped by coralME formatted as manual input files.

__coralME does not overwrite any file in building_data/</code>, but it will always overwrite files in building_data/reference_files/</code>__

- **termination_subreactions.txt**
  Input here will define translation termination subreactions and their machinery.

`class coralme.builder.curation.TerminationSubreactions(org, id='termination_subreactions', config={}, file='termination_subreactions.txt', name='Translation termination subreactions')`

Reads manual input to define translation termination subreactions.

This class creates the property "termination_subreactions" from the manual inputs in termination_subreactions.txt in an instance of Organism.

Input here will define translation termination subreactions and their machinery.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

termination_subreactions.txt :
subreaction enzymes PrfA_mono_mediated_termination PrfA_mono …

- **peptide_release_factors.txt**
  Input here will define peptide release factors.

`class coralme.builder.curation.PeptideReleaseFactors(org, id='peptide_release_factors', config={}, file='peptide_release_factors.txt', name='Peptide release factors')`

Reads manual input to define peptide release factors.

This class creates the property "peptide_release_factors" from the manual inputs in peptide_release_factors.txt in an instance of Organism.

Input here will define peptide release factors.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

peptide_release_factors.txt :
release_factor enzyme UAA generic_RF …

- **rna_degradosome.txt**
  Input here will define the composition of the RNA degradosome.

`class coralme.builder.curation.RNADegradosome(org, id='rna_degradosome', config={}, file='rna_degradosome.txt', name='RNA degradosome composition')`

Reads manual input to add RNA degradosome composition.

This class creates the property "rna_degradosome" from the manual inputs in rna_degradosome.txt in an instance of Organism.

Input here will define the composition of the RNA degradosome.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

rna_degradosome.txt :
enzymes Eno_dim_mod_mg2(4) …

- **special_trna_subreactions.txt**
  Input here will define special tRNA subreactions, such as tRNA-Sec (selenocysteine) synthesis from tRNA-Ser.

class coralme.builder.curation.**SpecialtRNASubreactions**(*org*, *id='special_trna_subreactions'*, *config={}*, *file='special_trna_subreactions.txt'*, *name='Special tRNA subreactions'*)

Reads manual input to define special tRNA subreactions.

This class creates the property "special_trna_subreactions" from the manual inputs in special_trna_subreactions.txt in an instance of Organism.

Input here will define special tRNA subreactions, such as tRNA-Sec (selenocysteine) synthesis from tRNA-Ser.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

special_trna_subreactions.txt :
subreaction enzymes PrfA_mono_mediated_termination PrfA_mono …

- **lipoprotein_precursors.txt**
  Input here will add lipoprotein precursors.

class coralme.builder.curation.**LipoproteinPrecursors**(*org*, *id='lipoprotein_precursors'*, *config={}*, *file='lipoprotein_precursors.txt'*, *name='Lipoprotein precursors'*)

Reads manual input to add lipoprotein precursors.

This class creates the property "lipoprotein_precursors" from the manual inputs in lipoprotein_precursors.txt in an instance of Organism.

Input here will add lipoprotein precursors.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

lipoprotein_precursors.txt :
name,gene AcrA,b0463 …

- **special_modifications.txt**
  Input here will define machinery for special modifications. These modifications are a set of pre-defined modifications that are used in ME-models.

class coralme.builder.curation.**SpecialModifications**(*org*, *id='special_modifications'*, *config={}*, *file='special_modifications.txt'*, *name='Special protein modifications'*)

Reads manual input to define machinery for special modifications.

This class creates the property "special_modifications" from the manual inputs in special_modifications.txt in an instance of Organism.

Input here will define machinery for special modifications. These modifications are a set of pre-defined modifications that are used in ME-models.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

special_modifications.txt :
modification enzymes stoich fes_transfer CPLX0-7617,IscA_tetra,CPLX0-7824 …

- **excision_machinery.txt**
  Input here will define machinery for excision.

`class coralme.builder.curation.`**`ExcisionMachinery`**(*org*, *id='excision_machinery'*, *config={}*, *file='excision_machinery.txt'*, *name='Excision machinery'*)

Reads manual input to define machinery for excision.

This class creates the property "excision_machinery" from the manual inputs in excision_machinery.txt in an instance of Organism.

Input here will define machinery for excision.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

excision_machinery.txt :
mechanism enzymes rRNA_containing RNase_E_tetra_mod_zn2(2), … …

- **orphan_and_spont_reactions.txt**
  Input here will mark reactions as orphan or spontaneous. Orphan reactions will be associated with CPLX_dummy, and spontaneous ones will not require enzymes for flux.

`class coralme.builder.curation.`**`OrphanSpontReactions`**(*org*, *id='orphan_and_spont_reactions'*, *config={}*, *file='orphan_and_spont_reactions.txt'*, *name='Orphan and spontaneous reactions'*)

Reads manual input to add reactions to the ME-model.

This class creates the property "orphan_and_spont_reactions" from the manual inputs in orphan_and_spont_reactions.txt in an instance of Organism.

Input here will mark reactions as orphan or spontaneous. Orphan reactions will be associated with CPLX_dummy, and spontaneous ones will not require enzymes for flux.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

orphan_and_spont_reactions.txt :
name description is_reversible is_spontaneous subsystems CODH_Fe_loading Loading of Fe false true …

- **enzyme_reaction_association.txt**
  Input here will create the association between enzymes and reactions in the ME-model.

`class coralme.builder.curation.`**`EnzymeReactionAssociation`**(*org*, *id='enz_rxn_assoc_df'*, *config={}*, *file='enzyme_reaction_association.txt'*, *name='Enzyme-reaction associations'*)

Reads manual input to specify enzyme-reaction associations.

This class creates the property "enz_rxn_assoc_df" from the manual inputs in enzyme_reaction_association.txt in an instance of Organism.

Input here will create the association between enzymes and reactions in the ME-model.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

enzyme_reaction_association.txt :
Reaction Complexes ADNt2pp NUPG-MONOMER OR NUPC-MONOMER …

- **peptide_compartment_and_pathways.txt**
  Input here will modify protein locations, and translocation pathways in the ME-model.

`class coralme.builder.curation.`**`ProteinLocation`**(*org*, *id='protein_location'*, *config={}*, *file='peptide_compartment_and_pathways.txt'*, *name='Protein location'*)

Reads manual input to add protein locations.

This class creates the property "protein_location" from the manual inputs in peptide_compartment_and_pathways.txt in an instance of Organism.

Input here will modify protein locations, and translocation pathways in the ME-model.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

peptide_compartment_and_pathways.txt :
Complex Complex_compartment Protein Protein_compartment translocase_pathway BSU02690-MONOMER Plasma_Membrane BSU02690() Plasma_Membrane s …

- **translocation_pathways.txt**
  Input here will define translocation pathways and their machinery.

`class coralme.builder.curation.`**`TranslocationPathways`**(*org*, *id='translocation_pathways'*, *config={}*, *file='translocation_pathways.txt'*, *name='Translocation pathways'*)

Reads manual input to define translocation pathways.

This class creates the property "translocation_pathways" from the manual inputs in translocation_pathways.txt in an instance of Organism.

Input here will define translocation pathways and their machinery.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

translocation_pathways.txt :
pathway enzyme sec BSU27650-MONOMER sec BSU35300-MONOMER sec secYEG . . .

- **rna_modification.txt**
  Input here will define enzymes that perform RNA modifications for either rRNA or tRNA in the ME-model.

`class coralme.builder.curation.`**`RNAModificationMachinery`**`(`*org*, *id='rna_modification_df'*, *config={}*, *file='rna_modification.txt'*, *name='RNA Modification machinery'*`)`

Reads manual input to add RNA modification machinery.

This class creates the property "rna_modification_df" from the manual inputs in rna_modification.txt in an instance of Organism.

Input here will define enzymes that perform RNA modifications for either rRNA or tRNA in the ME-model.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

rna_modification.txt :
modification positions type enzymes source D 16,17,20,20A,21 tRNA DusB_mono . . .

- **ribosomal_proteins.txt**
  Input here will define the composition of the ribosome.

`class coralme.builder.curation.`**`RibosomeStoich`**`(`*org*, *id='ribosome_stoich'*, *config={}*, *file='ribosomal_proteins.txt'*, *name='Ribosomal proteins'*`)`

Reads manual input to define ribosome composition.

This class creates the property "ribosome_stoich" from the manual inputs in ribosomal_proteins.txt in an instance of Organism.

Input here will define the composition of the ribosome.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

ribosomal_proteins.txt :
subunits proteins 30S RpsD_mono,... 50S generic_23s_rRNAs,generic_5s_rRNAs,RplA_mono,...

- **rho_independent.txt**
  Input here will mark genes with rho independent transcription termination.

`class coralme.builder.curation.`**`RhoIndependent`**`(`*org*, *id='rho_independent'*, *config={}*, *file='rho_independent.txt'*, *name='Genes with rho-independent termination'*`)`

Reads manual input to define genes with rho independent termination.

This class creates the property "rho_independent" from the manual inputs in rho_independent.txt in an instance of Organism.

Input here will mark genes with rho independent transcription termination.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

rho_independent.txt :
id b0344 ...

- **sigma_factors.txt**
  Input here will mark proteins for N-terminal methionine cleavage in the ME-model.

`class coralme.builder.curation.`**`Sigmas`**`(`*org*, *id='sigmas'*, *config={}*, *file='sigma_factors.txt'*, *name='Sigma factors'*`)`

Reads manual input to modify or add sigma factors.

This class creates the property "sigmas" from the manual inputs in sigma_factors.txt in an instance of Organism.

Input here will mark proteins for N-terminal methionine cleavage in the ME-model.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

sigma_factors.txt :
sigma,complex,genes,name RpoH_mono,RNAP_32H,b3461,"RNA polymerase, sigma 32 (sigma H) factor" ...

- **cleaved_methionine.txt**
  Input here will mark proteins for N-terminal methionine cleavage in the ME-model.

`class coralme.builder.curation.`**`CleavedMethionine`**`(`*org*, *id='cleaved_methionine'*, *config={}*, *file='cleaved_methionine.txt'*, *name='Proteins with N-terminal methionine cleavage'*`)`

Reads manual input to mark proteins that undergo N-terminal methionine cleavage.

This class creates the property "cleaved_methionine" from the manual inputs in cleaved_methionine.txt in an instance of Organism.

Input here will mark proteins for N-terminal methionine cleavage in the ME-model.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

cleaved_methionine.txt :
cleaved_methionine_genes b4154 …

- **folding_dict.txt**
  Input here will define folding pathways for proteins.

`class coralme.builder.curation.FoldingDict`(*org*, *id='folding_dict'*, *config={}*, *file='folding_dict.txt'*, *name='Protein to folding machinery associations'*)

Reads manual input to define folding pathways for proteins.

This class creates the property "folding_dict" from the manual inputs in folding_dict.txt in an instance of Organism.

Input here will define folding pathways for proteins.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

folding_dict.txt :
mechanism enzymes GroEL_dependent_folding b0014, … …

- **translocation_multipliers.txt**
  Input here will modify how many pores are required for the translocation of a protein.

`class coralme.builder.curation.TranslocationMultipliers`(*org*, *id='translocation_multipliers'*, *config={}*, *file='translocation_multipliers.txt'*, *name='Translocation multipliers'*)

Reads manual input to define translocation multipliers.

This class creates the property "translocation_multipliers" from the manual inputs in translocation_multipliers.txt in an instance of Organism.

Input here will modify how many pores are required for the translocation of a protein.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

translocation_multipliers.txt :
Gene,YidC_MONOMER,TatE_MONOMER,TatA_MONOMER b1855,2.0,0.0,0.0 …

- **subreaction_matrix.txt**
  Input here will define subreactions in the ME-model.

`class coralme.builder.curation.SubreactionMatrix`(*org*, *id='subreaction_matrix'*, *config={}*, *file='subreaction_matrix.txt'*, *name='Matrix of subreaction stoichiometries'*)

Reads manual input to add subreactions.

This class creates the property "subreaction_matrix" from the manual inputs in subreaction_matrix.txt in an instance of Organism.

Input here will define subreactions in the ME-model.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

subreaction_matrix.txt :
Reaction Metabolites Stoichiometry mod_acetyl_c accoa_c -1.0 mod_acetyl_c coa_c +1.0 …

- **me_metabolites.txt**
  Input here will mark metabolites in the M-model for replacement with their corrected E-matrix component.
- **elongation_subreactions.txt**
  Input here will define translation elongation subreactions and their machinery.

`class coralme.builder.curation.`**`MEMetabolites`**`(`*org*, *id='me_mets'*, *config={}*, *file='me_metabolites.txt'*, *name='Metabolites to substitute from M-model'*`)`

Reads manual input to replace metabolites in the M-model.

This class creates the property "me_mets" from the manual inputs in me_metabolites.txt in an instance of Organism.

Input here will mark metabolites in the M-model for replacement with their corrected E-matrix component.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

me_metabolites.txt :
id me_id name formula compartment type sufbcd_c CPLX0-1341 SufBCD complex REPLACE …

- **subsystem_classification.txt**
  Input here will classify subsystems in umbrella classifications which are then used to set a median Keff and correct it with the complex SASA.

`class coralme.builder.curation.`**`SubsystemClassification`**`(`*org*, *id='subsystem_classification'*, *config={}*, *file='subsystem_classification.txt'*, *name='Classification of subsystems'*`)`

Reads manual input to classify subsystems for Keff estimation.

This class creates the property "subsystem_classification" from the manual inputs in subsystem_classification.txt in an instance of Organism.

Input here will classify subsystems in umbrella classifications which are then used to set a median Keff and correct it with the complex SASA.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

subsystem_classification.txt : subsystem central_CE central_AFN intermediate secondary other S_Amino_acids_and_related_molecules 0 1 0 0 0 …

- **reaction_matrix.txt**
  Input here will define reactions directly in the ME-model. Definitions here will be added to the ME-model after processing the M-model into the ME-model.

class `coralme.builder.curation.`**`ReactionMatrix`**(*org*, *id='reaction_matrix'*, *config={}*, *file='reaction_matrix.txt'*, *name='Matrix of reaction stoichiometries'*)

Reads manual input to add reactions to the ME-model.

This class creates the property "reaction_matrix" from the manual inputs in reaction_matrix.txt in an instance of Organism.

Input here will define reactions directly in the ME-model. Definitions here will be added to the ME-model after processing the M-model into the ME-model.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

reaction_matrix.txt :
Reaction Metabolites Stoichiometry Cs_cyto_import cs_p -1.0 Cs_cyto_import h_c 1.0 Cs_cyto_import cs_c 1.0 Cs_cyto_import h_p -1.0 …

- **lipid_modifications.txt**
  Input here will define enzymes that perform lipid modifications.
- **amino_acid_trna_synthetase.txt**
  Input here will define amino acid tRNA ligases.
- **initiation_subreactions.txt**
  Input here will define translation initiation subreactions and their machinery.

class `coralme.builder.curation.`**`AminoacidtRNASynthetase`**(*org*, *id='amino_acid_trna_synthetase'*, *config={}*, *file='amino_acid_trna_synthetase.txt'*, *name='Amino acid to tRNA synthetase associations'*)

Reads manual input to define amino acid tRNA ligases.

This class creates the property "amino_acid_trna_synthetase" from the manual inputs in amino_acid_trna_synthetase.txt in an instance of Organism.

Input here will define amino acid tRNA ligases.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

amino_acid_trna_synthetase.txt :
amino_acid enzyme ala__L_c Ala_RS_tetra_mod_zn2(4) …

- **post_transcriptional_modification_of_RNA.txt**
  Input here will define RNA genes that undergo modifications.

`class coralme.builder.curation.`**`RNAModificationTargets`**`(`*org*, *id='rna_modification_targets'*, *config={}*, *file='post_transcriptional_modification_of_RNA.txt'*, *name='RNA modification targets'*`)`

Reads manual input to add RNA modification targets.

This class creates the property "rna_modification_targets" from the manual inputs in post_transcriptional_modification_of_RNA.txt in an instance of Organism.

Input here will define RNA genes that undergo modifications.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

post_transcriptional_modification_of_RNA.txt :
bnum position modification b0202 20A D …

- **protein_corrections.txt**
  Input here will add, modify complexes in the ME-model, as well as add, modify their modifications. You can add a complex modification ID in the replace column, which will remove that modified complex and replace it with your manually added one.
- **reaction_median_keffs.txt**
  Input here will define median Keffs for estimation of Keffs using the SASA method.
- **transcription_subreactions.txt**
  Input here will define machinery for transcription subreactions. These subreactions are a set of pre-defined subreactions that are used in ME-models.

`class coralme.builder.curation.`**`TranscriptionSubreactions`**`(`*org*, *id='transcription_subreactions'*, *config={}*, *file='transcription_subreactions.txt'*, *name='Transcription subreactions'*`)`

Reads manual input to define transcription subreactions.

This class creates the property "transcription_subreactions" from the manual inputs in transcription_subreactions.txt in an instance of Organism.

Input here will define machinery for transcription subreactions. These subreactions are a set of pre-defined subreactions that are used in ME-models.

Parameters
`org` (*`coralme.builder.organism.Organism`*) – Organism object.

Examples

transcription_subreactions.txt :
mechanism enzymes Transcription_normal_rho_independent Mfd_mono_mod_mg2(1),NusA_mono,NusG_mono,GreA_mo
…

- **generic_dict.txt**
  Input here will define generics.

`class coralme.builder.curation.GenericDict(org, id='generic_dict', config={}, file='generic_dict.txt', name='Dictionary of generic complexes')`

Reads manual input to define generics.

This class creates the property “generic_dict” from the manual inputs in generic_dict.txt in an instance of Organism.

Input here will define generics.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

generic_dict.txt :
generic_component enzymes generic_16Sm4Cm1402 RsmH_mono,RsmI_mono …

- **ribosome_subreactions.txt**
  Input here will define enzymes that perform a ribosome subreaction.

`class coralme.builder.curation.RibosomeSubreactions(org, id='ribosome_subreactions', config={}, file='ribosome_subreactions.txt', name='Ribosomal subreactions')`

Reads manual input to define ribosome subreactions.

This class creates the property “ribosome_subreactions” from the manual inputs in ribosome_subreactions.txt in an instance of Organism.

Input here will define enzymes that perform a ribosome subreaction.

Parameters
`org` (*coralme.builder.organism.Organism*) – Organism object.

Examples

ribosome_subreactions.txt :
subreaction enzyme gtp_bound_30S_assembly_factor_phase1 BSU16650-MONOMER …

- **reaction_corrections.txt**
  Input here will modify reactions at the M-model stage before ME-model building.

`class coralme.builder.curation.ReactionCorrections(org, id='reaction_corrections', config={}, file='reaction_corrections.txt', name='Reaction corrections')`

Reads manual input to modify reactions in the M-model.

This class creates the property “reaction_corrections” from the manual inputs in reaction_corrections.txt in an instance of Organism.

Input here will modify reactions at the M-model stage before ME-model building.

Parameters
: **org** (*coralme.builder.organism.Organism*) – Organism object.

Examples

reaction_corrections.txt :
reaction_id,name,gene_reaction_rule,reaction,notes COBALT2tpp,cobalt transport in via permease (no H+),BSU24740,cobalt2_e –> cobalt2_c,No notes …

- **TUs_from_biocyc.txt**
  Input here will modify transcriptional unit information.

2.4.2 How to curate?

1. If you have not run coralME yet, go back to *GettingStarted.ipynb*.
2. **Copy** all of the generated *reference files* in building_data/reference_files and replace accordingly in building_data/
3. **Go one by one** through the files in building_data/ curating as needed! Important flags are risen in curation_notes.json to further guide you through curation.
4. Everytime you make a change, **run the model through the troubleshooter!** It will show you remaining gaps to look at, and the new curation notes might show new warnings.
5. **Keep iterating!** You will have finished when no gaps are present, and all remaining warnings in curation notes are irrelevant.

2.5 Frequently Asked Questions

2.5.1 My gene identifiers are not consistent, what should I do?

We know that consistent gene ID conventions are a problem across all platforms of bioinformatics. We tried to generalize as much as possible what the gene conventions could be, but often different genome assemblies or M-model reconstructions yield inconsistent files.

What you should do depends on your problem, so we will classify the gene ID convention issues considering there are three main sources of information that must be consistent:

- **M-model** *gene identifiers*
- **Genome** *locus_tag*
- **Optional file** column *Accession-1*

Overall, you can assume that modifying the genome genbank file is the hardest approach and thus, the last resort.

1. M-model and Genbank are consistent, but they are not consistent with the BioCyc files.

Make sure that you looked for the correct BioCyc database, which corresponds to the M-model reconstruction. One quick way to ensure that is to copy one gene from your genbank or M-model and paste it in the search bar of BioCyc. Best case scenario, you microbe will appear in the list. Download the files from there and your problems are solved!

That didn't help?

It is possible that even when ensuring the BioCyc database is correct, the **Accession-1** column of genes.txt is still not consistent. However, you can assume that the correct IDs are somewhere in the database, since you found it looking for a gene id that follows your conventions (see *Getting Started*).

Try:

- Adding new columns in the gene **SmartTable**, **Accession-2** or **Synonyms** could contain your IDs.
- Maybe your IDs and BioCyc's only differ by an underscore, e.g. "PP0001" and "PP_0001". Use a text editor to change the IDs accordingly in **Accession-1** of genes.txt. Make sure not to make a mistake by editing gene IDs!

2. M-model and Genbank are not consistent

Make sure that you downloaded the same genbank file that was used to reconstruct the M-model, that is critical! If this is happening, you probably have the wrong genbank.

If you have a gene dictionary to convert between conventions, change the files to the IDs that are consistent with BioCyc.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX