findCPcore

Release 0.1.1

Alex Oarga

Aug 07, 2021

Contents

1 1. findCPcore

1.1 Import core modules

In order to use **findCPcore** library you have to import the following core modules.

[1]:
```
import sys

from findCPcore import CobraMetabolicModel
```

1.2 Read metabolic model

Read input metabolic model in Systems Biology Markup Language (SBML)[1] format. SBML is a XML-based standard for systems biology model's exchange.

Allowed files formats are: * xml * json * yml

[2]:
```
model = CobraMetabolicModel("aureus.xml")
```

1.3 Get model info

The following methods allow us to print on the command line data from the model.

1.3.1 Model info

[3]:
```
model.print_model_info()
```

```
MODEL INFO
------------------------------------------------------
MODEL:  MODEL1507180070
REACTIONS:  743
METABOLITES:  655
GENES:  619
COMPARTMENTS:  c
               e
```

1.3.2 Metabolites

[4]:
```
model.print_metabolites()
```

```
MODEL:  MODEL1507180070  - NUMBER OF METABOLITES:  655
METABOLITE  |  COMPARTMENT      |  REACTION ID
------------------------------------------------------
10fthf_c    |  c                |  biomass_SA_7a
            |                   |  biomass_SA_8a
            |                   |  FTHFL
            |                   |  biomass_SA_7b
            |                   |  MTHFC
            |                   |  AICART
            |                   |  GARFT
12dgr_EC_c  |  c                |  biomass_SA_2a
```

(continues on next page)

[1] http://sbml.org

(continued from previous page)

```
            |                   |  biomass_SA_2b
            |                   |  biomass_SA_3b
            |                   |  biomass_SA_lipids_only
            |
```

1.3.3 Reactions

[5]:
```
model.print_reactions()
```

```
MODEL:  MODEL1507180070  - NUMBER OF REACTIONS:  743
REACTION ID | UPPER BOUND | LOWER BOUND | REACTION
-----------------------------------------------------
3M2OBLOXRD  |   999999.0  |  -999999.0  |  3mob_c + h_c + lpam_c <=> 2mpdhl_c + co2_c
3M2OPLOXRD  |   999999.0  |  -999999.0  |  3mop_c + h_c + lpam_c <=> 2mbdhl_c + co2_c
4M2OPLOXRD  |   999999.0  |  -999999.0  |  4mop_c + h_c + lpam_c <=> 3mbdhl_c + co2_c
6PGALSZ     |   999999.0  |  0.0        |  h2o_c + lac6p_c --> dgal6p_c + glc_DASH_D_c
6PHBG       |   999999.0  |  0.0        |  h2o_c + salc6p_c --> 2hymeph_c + g6p_c
ABTAr       |   999999.0  |  0.0        |  4abut_c + akg_c --> glu_DASH_L_c + sucsal_c
ACACT1r     |   999999.0
```

1.3.4 Genes

[6]:
```
model.print_genes()
```

```
MODEL:  MODEL1507180070  - NUMBER OF GENES:  619
GENE ID     |  GENE NAME   |  REACTION ID |  GPR RELATION
-----------------------------------------------------
SA0008      |  SA0008      |  HISDr       |  SA0008
SA0009      |  SA0009      |  SERTRS      |  SA0009
SA0011      |  SA0011      |  HSERTA      |  SA0011
SA0016      |  SA0016      |  ADSS        |  SA0016
SA0036      |  SA0036      |  GPDDA2      |  SA0036 or SA0820 or SA1542 or SA0969 or SA0220
            |              |  GPDDA5      |  SA0036 or SA0820 or SA1542 or SA0969 or SA0220
            |              |  GPDDA3      |  SA0036 or SA0820 or SA1542 or SA0969 or SA0220
            |
```

1.4 Dead-end metabolites

Dead-end metabolites[2] are those metabolites which are not consumed or not produced by any reaction of a given compartment of the model including exchange reactions.

1.4.1 Finding Dead-end metabolites

Dead-end metabolites of the model are calculated with the method `find_dem()`. The method `dem()` returns a python dict[3] with the following content:

- **key**: string with compartment name.
- **value**: list with cobra.core.metabolites[4].

[2] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0075210
[3] https://docs.python.org/2/tutorial/datastructures.html#dictionaries
[4] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Metabolites

[7]:
```
model.find_dem()
model.dem()
```

[7]:
```
{'c': [<Metabolite 2pglyc_c at 0x7fe3dae1c110>,
  <Metabolite adcobdam_c at 0x7fe3dae4e190>,
  <Metabolite 2ombz_c at 0x7fe42cece250>,
  <Metabolite fdxox_c at 0x7fe40fdde750>,
  <Metabolite nop_c at 0x7fe40fd90910>,
  <Metabolite acmama_c at 0x7fe3dae48950>,
  <Metabolite adprib_c at 0x7fe3dae4e950>,
  <Metabolite DGDG_SA_c at 0x7fe3dae42990>,
  <Metabolite udcp_c at 0x7fe40fd3cb10>,
  <Metabolite Sptmyn_c at 0x7fe3dae42b50>,
  <Metabolite mhpglu_c at 0x7fe40fd86b50>,
  <Metabolite 12dgr_EC_c at 0x7fe3dae24b90>,
  <Metabolite o2s_c at 0x7fe40fd90bd0>,
  <Metabolite Stmyn_c at 0x7fe3dae42bd0>,
  <Metabolite drib_c at 0x7fe40fdd2c50>,
  <Metabolite mi1p_DASH_D_c at 0x7fe40fd86c50>,
  <Metabol
```

Dead-end metabolites can be printed on the command line with `print_dem()` method.

[8]:
```
model.print_dem()
```

```
MODEL:  MODEL1507180070  - NUMBER OF DEM:  2  - COMPARTMENT:  ALL
METABOLITE  |  COMPARTMENT      |  REACTION ID
-----------------------------------------------------------
2pglyc_c    |  c                |  PGLYCP
adcobdam_c  |  c                |  ADCYRS
2ombz_c     |  c                |  URFGTT
fdxox_c     |  c                |  OOR3r
nop_c       |  c                |  NOPD
acmama_c    |  c                |  AMAA
adprib_c    |  c                |  ADPRDP
DGDG_SA_c   |  c                |  biomass_SA_lipids_only
            |                   |  biomass_SA_2a
udcp_c      |  c                |  UDCPKr
Sptmyn_c    |  c                |  Spt3AdT
mhpglu_c    |  c                |  MHPGLUT
12dgr_E
```

Dead-end metabolites printed can be limited to a specific compartment. Available compartments are given by `model.compartments()` method.

[9]:
```
model.print_dem(compartment="e")
```

```
MODEL:  MODEL1507180070  - NUMBER OF DEM:  2  - COMPARTMENT:  e
METABOLITE  |  COMPARTMENT      |  REACTION ID
-----------------------------------------------------------
nh4_e       |  e                |  NH4t4
            |                   |  EX_nh4_e
lcts_e      |  e                |  LACpts
            |                   |  EX_lcts_e
mnl_e       |  e                |  EX_mnl_e
            |                   |  MNLpts
man_e       |  e                |  EX_man_e
            |                   |  MANpts
mobd_e      |  e                |  MOBDabc
```

(continues on next page)

(continued from previous page)

```
           |                  |  EX_mobd_e
ni2_e      |  e               |  NIabc
           |                  |  EX_ni2_e
gua_e      |  e
```

1.4.2 Removing dead-end metabolites

Method `remove_dem()` removes dead-end metabolites from the model. Once dead-end metabolites are deleted, some reactions might not produce a metabolite anymore so the method deletes also these reactions and loops again until no dead-end metabolite is found:

```
while number of metabolites does not change:
    delete all dead-end metabolites
    for reaction that produced or consumed dead-end metabolites:
        if reaction produces or consumes 0 metabolites [and is not exchange nor demand]:
            delete reaction
    find dead-end metabolites
```

The reactions that are deleted on the method can be modified by the following params:

- **delete_exchange** :
  - **True** : all the reactions that are produce or consume 0 metabolites are deleted whether they are exchange/demand or not
  - **False** : (default) deleted according to 'keep_all_incomplete_reactions' param.
- **keep_all_incomplete_reactions** :
  - **False**: if a reaction is a cobra boundary reaction[5] (calculated by heuristics) that reaction can't be deleted.
  - **True**: (default) if a reaction initially doesn't produce or consume any metabolite that reaction can't be deleted.

[10]:
```
print("Metabolites: ", len(model.metabolites()), "\nReactions: ", len(model.reactions()))
```

```
Metabolites:  655
Reactions:  743
```

[11]:
```
model.remove_dem()
```

[12]:
```
print("Metabolites: ", len(model.metabolites()), "\nReactions: ", len(model.reactions()))
```

```
Metabolites:  486
Reactions:  739
```

1.5 Dead reactions

Dead reactions are those reactions with upper and lower flux equal to zero.

[5] https://cobrapy.readthedocs.io/en/latest/media.html#Boundary-reactions

1.5.1 Finding Dead reactions

Dead reactions of the model are calculated with the method `dead_reactions()`, which returns a list of cobra.core.reactions[6] with dead reactions.

[13]:
```
model.dead_reactions()
```

[13]:
```
[<Reaction SA_biomass_1a at 0x7fe40f8359d0>,
 <Reaction biomass_SA_2a at 0x7fe40f7025d0>,
 <Reaction biomass_SA_2b at 0x7fe40f702a50>,
 <Reaction biomass_SA_3a at 0x7fe40f702ed0>,
 <Reaction biomass_SA_3b at 0x7fe40f702c90>,
 <Reaction biomass_SA_4a at 0x7fe40f7160d0>,
 <Reaction biomass_SA_5a at 0x7fe40f716050>,
 <Reaction biomass_SA_6a at 0x7fe40f716fd0>,
 <Reaction biomass_SA_6b at 0x7fe40f721fd0>,
 <Reaction biomass_SA_7a at 0x7fe40f730f90>,
 <Reaction biomass_SA_7b at 0x7fe40f6bff90>,
 <Reaction biomass_SA_lipids_only at 0x7fe40f721f90>,
 <Reaction biomass_SA_nuc_only at 0x7fe40f710c90>,
 <Reaction biomass_SA_only_AA at 0x7fe40f730fd0>]
```

1.6 Chokepoint reactions

Chokepoint reactions[7] are those reactions that are the only consumer or producer of a given metabolite that isn't a dead-end metabolite.

1.6.1 Finding chokepoint reactions

Chokepoint reactions are calculated with the method `find_chokepoints()`. The method `chokepoints()` returns a list of tuples of types (cobra.core.reactions[8], cobra.core.metabolites[9]) representing a chokepoint reactions object and the metabolite it only produces/consumes.

[15]:
```
# Read initial model again
model = CobraMetabolicModel("aureus.xml")

model.find_chokepoints()
model.chokepoints()
```

[15]:
```
[(<Reaction PAPA_SA at 0x7fe40f2e4a10>,
  <Metabolite 12dgr_SA_c at 0x7fe40fd65590>),
 (<Reaction PROD2 at 0x7fe40f249cd0>, <Metabolite 1pyr5c_c at 0x7fe40fd65bd0>),
 (<Reaction DHDPRy at 0x7fe40f4d1d90>,
  <Metabolite 23dhdp_c at 0x7fe40fdb1310>),
 (<Reaction DHDPS at 0x7fe40f4d34d0>, <Metabolite 23dhdp_c at 0x7fe40fdb1310>),
 (<Reaction DHAD1 at 0x7fe40f4ce8d0>, <Metabolite 23dhmb_c at 0x7fe40fde34d0>),
 (<Reaction KARA1i at 0x7fe40f3ce910>,
  <Metabolite 23dhmb_c at 0x7fe40fde34d0>),
 (<Reaction DHAD2 at 0x7fe40f4d1490>, <Metabolite 23dhmp_c at 0x7fe40fde3910>),
 (<Reaction KARA2i at 0x7fe40f3ceb10>,
  <Metabolite 23dhmp_c at 0x7fe40fde3910>),
 (<Reaction DHPPDA at 0x7fe40f4d9950>,
  <Met
```

[6] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Reactions
[7] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2174424/
[8] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Reactions
[9] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Metabolites

Dead reactions can be excluded from the computation of chokepoints with the `exclude_dead_reactions=True` parameter. This is strongly recommended as dead reactions are considered forward reactions by default an this can lead to misinterpretations.

[16]:
```
model.find_chokepoints(exclude_dead_reactions=True)
model.chokepoints()
```

[16]:
```
[(<Reaction PAPA_SA at 0x7fe40f2e4a10>,
  <Metabolite 12dgr_SA_c at 0x7fe40fd65590>),
 (<Reaction PROD2 at 0x7fe40f249cd0>, <Metabolite 1pyr5c_c at 0x7fe40fd65bd0>),
 (<Reaction DHDPRy at 0x7fe40f4d1d90>,
  <Metabolite 23dhdp_c at 0x7fe40fdb1310>),
 (<Reaction DHDPS at 0x7fe40f4d34d0>, <Metabolite 23dhdp_c at 0x7fe40fdb1310>),
 (<Reaction DHAD1 at 0x7fe40f4ce8d0>, <Metabolite 23dhmb_c at 0x7fe40fde34d0>),
 (<Reaction KARA1i at 0x7fe40f3ce910>,
  <Metabolite 23dhmb_c at 0x7fe40fde34d0>),
 (<Reaction DHAD2 at 0x7fe40f4d1490>, <Metabolite 23dhmp_c at 0x7fe40fde3910>),
 (<Reaction KARA2i at 0x7fe40f3ceb10>,
  <Metabolite 23dhmp_c at 0x7fe40fde3910>),
 (<Reaction DHPPDA at 0x7fe40f4d9950>,
  <Met
```

Chokepoint reactions can also be printed on the command line with `print_chokepoints`.

[17]:
```
model.print_chokepoints()
```

```
MODEL:  MODEL1507180070  - NUMBER OF CHOKEPOINTS:  448
METABOLITE ID |  METABOLITE NAME                              | REACTION ID | REACTION NAME
--------------------------------------------------------------
12dgr_SA_c    |  1,2-Daicylglycerol (Saureus)                 |  PAPA_SA    |  Phosphatidate␣
↪phosphatase
1pyr5c_c      |  1-Pyrroline-5-carboxylate                    |  PROD2      |  Proline␣
↪dehydrogenase
23dhdp_c      |  2,3-Dihydrodipicolinate                      |  DHDPRy     | ␣
↪dihydrodipicolinate reductase (NADPH)
23dhdp_c      |  2,3-Dihydrodipicolinate                      |  DHDPS      | ␣
↪dihydrodipicolinate synthase
23dhmb_c      |  (R)-2,3-Dihydroxy-3-methylbutanoate          |  DHAD1      |
```

1.7 Flux Balance Analysis

Flux Balance Analysis (FBA)[10] is a mathematical procedure used to calculate the growth rate of a metabolic model considering an objective function to maximize and the reactions flux as constraints.

The method `get_growth()` calculates the objective value (growth rate) that maximizes the objective function. This method uses cobra.core.model.slim_optimize[11] and was obtained from cobra.flux_analysis.deletion[12].

The objective value obtained with FBA can be accessed with `objective_value()`.

[18]:
```
model.get_growth()
model.objective_value()
```

[10] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3108565/
[11] https://cobrapy.readthedocs.io/en/latest/autoapi/cobra/core/model/index.html?highlight=slim#cobra.core.model.Model.slim_optimize
[12] https://cobrapy.readthedocs.io/en/latest/_modules/cobra/flux_analysis/deletion.html#_get_growth

[18]:
```
0.1580502916027849
```

The objective function that is maximized during FBA can be accessed by `objective()`

[19]:
```
model.objective()
```

[19]:
```
'1.0*biomass_SA_8a - 1.0*biomass_SA_8a_reverse_4ce77'
```

This objective function can be changed by another reaction of the model with `set_objective()`. This method receives the id of the reaction that will be set as the new objective value.

[20]:
```
model.set_objective("DHAD1")
model.objective()
```

[20]:
```
'1.0*DHAD1 - 1.0*DHAD1_reverse_39dca'
```

1.8 Flux Variability Analysis

Flux Vatiability Analysis (FVA)[13] is a mathematical procedure used to calculate the ''minimum and maximum flux for reactions in the network while maintaining some state of the network, e.g., supporting 90% of maximal possible biomass production rate".

The method `fva()` runs Flux Variability Analysis on the model. This method runs cobra.flux_analysis.variability[14] and as so it allows the same parameters (see the previous link):
- **loopless**: (default False) return only loopless solutions.
- **threshold**: (float default None. In cobrapy 'fraction_of_optimum'): Requires that the objective value is at least the fraction times maximum objective value.
- **pfba_factor**: (float default None) the total sum of absolute fluxes must not be larger than this value times the smallest possible sum of absolute fluxes.

An extra parameter:
- **verbose**: (default False) if True prints on the command line the result of FVA while running the analysis.

Method `fva()` returns an error list if there was an error while running FVA or an empty list `[]` otherwise.

[21]:
```
model.fva(threshold=0.95)
```

[21]:
```
[]
```

[22]:
```
model.fva(threshold=0.95, verbose=True)
```

```
FLUX VARIABILITY ANALYSIS:  MODEL1507180070
REACTION:  3-Methyl-2-oxobutanoate:lipoamide oxidoreductase(decarboxylating and acceptor-2-
↪methylpropanoylating)
   fva ranges:   [ 0.0        ,  0.0        ]
REACTION:  3-Methyl-2-oxopentanoate:lipoamide oxidoreductase(decarboxylating and acceptor-
↪2-methylpropanoylating)
   fva ranges:   [ 0.0        ,  0.0        ]
```

(continues on next page)

[13] https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-489
[14] https://cobrapy.readthedocs.io/en/latest/autoapi/cobra/flux_analysis/variability/index.html?highlight=flux_varia#cobra.flux_analysis.variability.flux_variability_analysis

(continued from previous page)

```
REACTION:  4-Methyl-2-oxopentanoate:lipoamide oxidoreductase(decarboxylating and acceptor-
↪2-methylpropanoylating)
    fva ranges:   [ 0.0        ,  0.0        ]
REACTION:  6-phospho-beta-galactosidase
    fva ranges:   [ 0.0        ,  0.0        ]
REACTION:  6-phospho-beta-glucosidase
    fva ranges:   [ 0.0        ,  0.0        ]
RE
```

The result obtained with FVA can be accessed with `get_fva()`. This method returns a list of tuples: (cobra.core.reaction[15], [float] maximum flux, [float] minimum flux)

[23]:
```
model.get_fva()
```

[23]:
```
[(<Reaction 3M2OBLOXRD at 0x7fe40fc3d2d0>, 0.0, 0.0),
 (<Reaction 3M2OPLOXRD at 0x7fe40fc3d790>, 0.0, 0.0),
 (<Reaction 4M2OPLOXRD at 0x7fe40fc3d390>, 0.0, 0.0),
 (<Reaction 6PGALSZ at 0x7fe40fca2ed0>, 0.0, 0.0),
 (<Reaction 6PHBG at 0x7fe40fca2f90>, 0.0, 0.0),
 (<Reaction ABTAr at 0x7fe40f6df490>, 0.0, 0.0),
 (<Reaction ACACT1r at 0x7fe40fcb6d50>,
  0.00032511320071648854,
  -2065.6249999999704),
 (<Reaction ACACT2r at 0x7fe40fcb6850>, 0.0, -2065.624999999973),
 (<Reaction ACACT3r at 0x7fe40fcb6e90>, 0.0, -2065.624999999973),
 (<Reaction ACACT4r at 0x7fe40fcb6cd0>, 0.0, -2065.624999999973),
 (<Reaction ACACT5r at 0x7fe40fcb6b90>, 0.0, -2065.624999999973),
 (<Reaction ACACT6r at 0x7fe40fcb6b
```

1.8.1 Updating reaction's flux with FVA

findCritical adds an extra parameter to this method which is `update_flux`. If `True` the method `fva()` updates the model reaction's flux with the maximum and minimum flux values obtained with FVA (see the example below).

[24]:
```
# Print initial reactions of the model with initial flux values.
model.print_reactions()
```

```
MODEL:  MODEL1507180070  - NUMBER OF REACTIONS:  743
REACTION ID | UPPER BOUND | LOWER BOUND | REACTION
-----------------------------------------------------
3M2OBLOXRD  |   999999.0  |  -999999.0  |  3mob_c + h_c + lpam_c <=> 2mpdhl_c + co2_c
3M2OPLOXRD  |   999999.0  |  -999999.0  |  3mop_c + h_c + lpam_c <=> 2mbdhl_c + co2_c
4M2OPLOXRD  |   999999.0  |  -999999.0  |  4mop_c + h_c + lpam_c <=> 3mbdhl_c + co2_c
6PGALSZ     |   999999.0  |  0.0        |  h2o_c + lac6p_c --> dgal6p_c + glc_DASH_D_c
6PHBG       |   999999.0  |  0.0        |  h2o_c + salc6p_c --> 2hymeph_c + g6p_c
ABTAr       |   999999.0  |  0.0        |  4abut_c + akg_c --> glu_DASH_L_c + sucsal_c
ACACT1r     |   999999.0
```

[25]:
```
# Update reactions flux values with FVA
model.fva(update_flux=True)
```

[25]:
```
[]
```

[15] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Reactions

[26]:
```
# Print reactions of the model with refined reactions flux values.
model.print_reactions()
```

```
MODEL:  MODEL1507180070  - NUMBER OF REACTIONS:  743
REACTION ID | UPPER BOUND | LOWER BOUND | REACTION
-----------------------------------------------------
3M2OBLOXRD  |   0.0       |  0.0        |  3mob_c + h_c + lpam_c --> 2mpdhl_c + co2_c
3M2OPLOXRD  |   0.0       |  0.0        |  3mop_c + h_c + lpam_c --> 2mbdhl_c + co2_c
4M2OPLOXRD  |   0.0       |  0.0        |  4mop_c + h_c + lpam_c --> 3mbdhl_c + co2_c
6PGALSZ     |   0.0       |  0.0        |  h2o_c + lac6p_c --> dgal6p_c + glc_DASH_D_c
6PHBG       |   0.0       |  0.0        |  h2o_c + salc6p_c --> 2hymeph_c + g6p_c
ABTAr       |   0.0       |  0.0        |  4abut_c + akg_c --> glu_DASH_L_c + sucsal_c
ACACT1r     |   -3.26642
```

1.9 Essential genes

Essential genes[16] are those genes that cause a zero growth rate when knocked out.

1.9.1 Finding essential genes

Essential genes are calculated with the method `find_essential_genes_1()`. This method returns an error list if there was an error during the computing or an empty list `[]` otherwise.

[27]:
```
model.find_essential_genes_1()
```

[27]:
```
[]
```

If essential genes were computed with the method above, they can be accessed with `essential_genes()`. This method return a list of cobra.core.gene[17] with essential genes.

[28]:
```
model.essential_genes()
```

[28]:
```
{<Gene SA0183 at 0x7fe40fb95450>,
 <Gene SA0512 at 0x7fe40fb7a550>,
 <Gene SA0728 at 0x7fe40fbe8690>,
 <Gene SA0729 at 0x7fe40fbe8710>,
 <Gene SA0823 at 0x7fe40fbe0fd0>,
 <Gene SA0910 at 0x7fe40fbdae10>,
 <Gene SA0911 at 0x7fe40fbdafd0>,
 <Gene SA0912 at 0x7fe40fbdab50>,
 <Gene SA0913 at 0x7fe40fbda050>,
 <Gene SA0937 at 0x7fe40fbd4e50>,
 <Gene SA0938 at 0x7fe40fbd4ad0>,
 <Gene SA0994 at 0x7fe40fbccf50>,
 <Gene SA0995 at 0x7fe40fbccb10>,
 <Gene SA0996 at 0x7fe40fbcc650>,
 <Gene SA1088 at 0x7fe40fbc7f10>,
 <Gene SA1089 at 0x7fe40fbc79d0>,
 <Gene SA1244 at 0x7fe40fc30850>,
 <Gene SA1245 at 0x7fe40fc30a10>,
 <Gene SA1255 at 0x7fe40fc30510>,
 <Gene SA1521 at 0x7fe40fc14310>,
 <Gene SA1585 at 0x7
```

[16] https://link.springer.com/chapter/10.1007/978-3-642-36546-1_43
[17] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Genes

1.10 Essential reactions

Essential reactions are those genes that cause a zero growth rate when knocked out.

1.10.1 Finding essential reactions

Essential reactions are calculated with the method `find_essential_reactions_1()`. This method returns an error list if there was an error during the computing or an empty list `[]` otherwise.

[29]:
```
model.find_essential_reactions_1()
```

[29]:
```
[]
```

If essential genes were computed with the method above, they can be accessed with `essential_reactions()`. This method return a `dict` with keys cobra.core.reaction[18] with all the reactions of the model, and values `float` with the result of computing FBA with the reaction knocked-out.

[30]:
```
model.essential_reactions()
```

[30]:
```
{<Reaction TECA1S at 0x7fe40f1d7410>: 100.00000000000126,
 <Reaction HETZK at 0x7fe40f410bd0>: 100.00000000000126,
 <Reaction YUMPS at 0x7fe40f164990>: 100.00000000000126,
 <Reaction NO3R1 at 0x7fe40f3179d0>: 100.00000000000126,
 <Reaction GLYD at 0x7fe40f45ce10>: 100.00000000000126,
 <Reaction GMHEPPA at 0x7fe40f465e90>: 100.00000000000126,
 <Reaction THRAr at 0x7fe40f1edad0>: 100.00000000000126,
 <Reaction ADPRDP at 0x7fe40f525890>: 100.00000000000126,
 <Reaction SUCBZL at 0x7fe40f1bd950>: 100.00000000000126,
 <Reaction EX_ser_DASH_L_e at 0x7fe40f493850>: 100.00000000000126,
 <Reaction biomass_SA_4a at 0x7fe40f164e90>: 100.00000000000126,
 <Reaction KAS8 at 0x7fe40f3e8e90>: 100.00000000000
```

1.11 Essential genes reactions

Essential genes reactions are those reactions that are knocked-out when an essential gene is knocked-out.

1.11.1 Finding essential genes reactions

Essential genes reactions are calculated with the method `find_essential_genes_reactions()`. This method returns an error list if there was an error during the computing or an empty list `[]` otherwise.

[31]:
```
model.find_essential_genes_reactions()
```

[31]:
```
[]
```

If essential genes reactions were computed with the method above, they can be accessed with `essential_genes_reactions()`. This method returns a `dict` with keys cobra.core.reaction[19] with all the reactions of the model, and values a `list` of cobra.core.genes[20] with the essential genes that cause the knock-out of the reaction.

[18] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Reactions
[19] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Reactions
[20] https://cobrapy.readthedocs.io/en/latest/getting_started.html#Genes

[32]:
```
model.essential_genes_reactions()
```

[32]:
```
{<Reaction CYTBD at 0x7fe40f4c4890>: [<Gene SA0913 at 0x7fe40fbda050>,
  <Gene SA0938 at 0x7fe40fbd4ad0>,
  <Gene SA0912 at 0x7fe40fbdab50>,
  <Gene SA0911 at 0x7fe40fbdafd0>,
  <Gene SA0910 at 0x7fe40fbdae10>,
  <Gene SA0937 at 0x7fe40fbd4e50>],
 <Reaction AKGDb at 0x7fe40f52c3d0>: [<Gene SA1244 at 0x7fe40fc30850>],
 <Reaction P5CD at 0x7fe40f2e4910>: [<Gene SA2341 at 0x7fe40fc4e9d0>],
 <Reaction SUCOAS at 0x7fe40f1bde90>: [<Gene SA1089 at 0x7fe40fbc79d0>,
  <Gene SA1088 at 0x7fe40fbc7f10>],
 <Reaction PROD2 at 0x7fe40f249cd0>: [<Gene SA1585 at 0x7fe40fc0f9d0>],
 <Reaction AKGDa at 0x7fe40f52c510>: [<Gene SA1245 at 0x7fe40fc30a10>],
 <Reaction FUM at 0x7fe40f4a98d0>: [<Gene SA1669 at 0x7fe4
```

2 2. Facade & FacadeUtils

Modules `FacadeUtils` and `Facade` from `findCPcore` provides specific methods for the generation of spreadsheets regarding the computation of critical reactions taking into account Flux Variability Analysis and Dead End Metabolites.

2.1 Import FacadeUtils modules

Import the module

[3]:
```
from findCPcore import Facade
from findCPcore import FacadeUtils
```

2.2 Generate chokepoint summary spreadsheet

The method `run_summary_model` generates a spreadsheet file from a model file with the following data:

- List of metabolites of the model.
- List of reactions of the model.
- List of genes of the model.
- Upper and lower flux bound of each reaction obtained with Flux Variability Analysis.
- Upper and lower flux bound of each reaction obtained with Flux Variability Analysis grouped by metabolite.
- List of reversible reactions of the model before and after Flux Variability Analysis refinement (see *refinement*).
- List of Dead End Metabolites before and after FVA refinement.
- For the following models:
    * Initial model
    * Model without DEM
    * Model refined with FVA.
    * Model refined with FVA and with DEM removed.
- The following set of assets is computed for each and included in one sheet:
    * List of chokepoint reactions with the metabolite they produce/consume.
    * List of essential genes of the models.

* List of essential reactions of the models.
* Comparison of chokepoint reactions, essential reactions and essential gene reactions.

- Summary comparing the size of the previous sets and their intersections.

Method declaration:

```
run_summary_model(self, model_path, print_f, arg1, arg2, objective=None,
fraction=1.0)
```

Parameters:

- `model_path`: Path of the SBML metabolic model file.
- `print_f`: Callback function to inform of the computation progression. The function must have a declaration like the following:
- `custom_function_name(message, arg1, arg2)`
- `arg1`: First parameter to pass to the function if any.
- `arg2`: Second parameter to pass to the function if any.
- `objective`: Reactions id to be used as objective function.
- `fraction`: Fraction of optmimum to be used with FVA.

[4]:
```
def callback_print_ignore(message, arg1, arg2):
    pass

def callback_print_logger(message, arg1, arg2):
    logger = arg1
    logger.info(message)

def callback_print(message, arg1, arg2):
    print(arg1 + message)

facadeUtils = FacadeUtils()
spreadsheet = facadeUtils.run_summary_model("aureus.xml", callback_print, "LOG:", None,␣
↪fraction=0.95)
facadeUtils.save_spreadsheet("output.xls", spreadsheet)
```

```
LOG:Reading model...
LOG:Generating models...
LOG:Searching Dead End Metabolites (D.E.M.)...
LOG:Searching chokepoint reactions...
LOG:Searching essential reactions...
LOG:Searching essential genes...
LOG:Searching essential genes reactions...
LOG:Removing Dead End Metabolites (D.E.M.)...
LOG:Searching essential reactions...
LOG:Searching new chokepoint reactions...
LOG:Searching essential genes...
LOG:Searching essential genes reactions...
LOG:Running Flux Variability Analysis...
LOG:Searching Dead End Metabolites (D.E.M.)...
LOG:Searching new chokepoint reactions...
LOG:Searching essential genes...
LOG:Searching essential genes reactions...
LOG:Searching essential reactions...
LOG:Removing Dead End Metabolites (D.E.M.)...
LOG:Searching essential reactions...
```

(continues on next page)

(continued from previous page)

```
LOG:Searching new chokepoint reactions...
LOG:Searching essential genes...
LOG:Searching essential genes reactions...
LOG:Generating spreadsheet...
```

[4]:
```
(True, 'output.xls')
```

2.3 Generate chokepoint growth sensibility reports

The method `generate_growth_dependent_report` of `Facade` computes and returns growth dependent chokepoints.

The output can be saved in a spreadsheet file or html report. The resulting files include an analysis of the effects of varying the values of fraction of optimum with FVA has on the folloing sets: - Reversible Reactions (RR) - Non-reversible Reactions (NR) - Dead Reactions (DR) - Chokepoint reactions (CP)

Methods declaration:

`generate_growth_dependent_report(self, config)`

Parameters:

- `config`: `findCPcore.utils.GrowthDependentCPConfig`

class `findCPcore.utils.GrowthDependentCPConfig`

- `model_path`: Path of the SBML metabolic model file.
- `print_f`: Callback function to inform of the computation progression. The function must have a declaration like the following: `custom_function_name(message, arg1, arg2)`
- `arg1`: First parameter to pass to the function if any.
- `arg2`: Second parameter to pass to the function if any.
- `objective`: Reactions id to be used as objective function.
- `fraction`: Fraction of optmimum to be used with FVA.
- `output_path_spreadsheet`: Output path to save spreadsheet report. If None no file is generated.
- `output_path_html`: Output path to save html report. If None no file is generated.

[6]:
```
# import config class
from findCPcore.utils import GrowthDependentCPConfig


def callback_print(message, arg1, arg2):
    print(message)

config = GrowthDependentCPConfig()
config.print_f          = callback_print
config.model_path       = "aureus.xml"
config.model_path       = "report.xls"
config.output_path_html = "report.html"

facade = Facade()
facade.generate_growth_dependent_report(config)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-4bebcdd4e316> in <module>
      6     print(message)
      7
----> 8 config = GrowthDependentCPConfig()
      9 config.print_f          = callback_print
     10 config.model_path       = "aureus.xml"

TypeError: 'module' object is not callable
```

[ ]: