Norges miljø- og biovitenskapelige universitet

Master's thesis 2020 30 ECTS

Fakultet for kjemi, bioteknologi og matvitenskap

Developing and validating tests for a metabolic model of Atlantic salmon (*Salmo salar*)

Ingunn Marie Verne Ruud

Chemistry and biotechnology

Acknowledgements

This thesis marks the end of a five-year education on NMBU, Ås. These years have been wonderful and filled with laughter, challenges and good memories. First, I would like to thank the DigiSal group for being so welcoming and friendly. I would like to thank my supervisors Jon Olav Vik and Ove Øyås for guiding me through this process. Special thanks to Filip Rotnes and Marie Gulla for infinite patience for my unending flow of questions. I would also like to thank Kathrine Frey Frøslie for introducing me to the Biostatistics and DigiSal group.

I would like to thank my flat mates Anniken and Juliane for supporting each other and keeping our spirits up through this very special and out-of-the-ordinary spring. Being a trio in the same situation really helped on motivation. Special thanks to Åsblæst'n for making these five years a blast. Finally, I would like to thank my family for always supporting and believing in me.

Developing and validating tests for a metabolic model of Atlantic salmon (*Salmo salar*)

Ingunn Marie Verne Ruud

June 2, 2020

Abstract

The fish farming industry is expanding, and to achieve economic and ecological sustainability, new fish feeds are being developed. When developing new feeds, it can be useful to first simulate on a computer how the biological network will react. This can be done with metabolic models. Metabolic models consist of the reactions and metabolites arranged into a stoichiometric matrix. Constraints on the network are imposed in the form of stoichiometric coefficients and bounds on reaction rates. To trust the results from a simulation, it is important that the model is well annotated and internally consistent, i.e. of high quality. The software Memote tests the model for a set of quality criteria and presents the score in a report. This thesis will discuss the application, development and validation of Memote's tests. This is done by implementing three possible improvements iteratively to a model and testing with Memote for each iteration, eventually composing a Memote history report to inspect the change in score for the different model versions. There is an overall emphasis on annotations to databases in the tests; a wide range of annotations for genes, metabolites and reactions will increase the score. Including physiologically important reactions, such as secretion of CO_2 will also increase the score. Memote is a good tool to show what the model contains, the scope and notify you if a feature you think was added to the model, was in fact not added. To more thoroughly review the models, adding organism-specific tests could be a possibility.

Sammendrag

Akvakulturnæringen er i vekst, og for å oppnå økonomisk og økologisk bærekraftighet må nytt fôr utvikles. Under fôrutviklingen kan det være nyttig å først simulere på datamaskin hvordan det biologiske nettverket vil reagere. Dette kan gjøres med metabolske modeller. Metabolske modeller inneholder reaksjonene og metabolittene satt i en støkiometrisk matrise. Begrensninger på nettverket blir påført i form av støkiometriske koeffisienter og grenser for reaksjonsrater. For å kunne stole på resultatene fra en simulering er det viktig at modellen er godt annotert og internt konsistent, med andre ord av god kvalitet. Programvaren Memote kan teste en modell for bestemte kvalitetskrav og presenterer poengene i en rapport. Denne avhandlingen vil diskutere anvendelsen, utviklingen og validering av Memote sine tester. Dette gjøres ved å implementere tre mulige forbedringer iterativt til en modell og teste med Memote for hver iterasjon, og til slutt lage en Memote historierapport for å se endringene i score for de forskjellige modellversjonene. Det legges stor vekt på databaseannoteringer i testene og et bredt spekter av annoteringer for gener, metabolitter og reaksjoner vil øke poengene. Memote er et bra verktøy for å vise hva modellen inneholder, hva den kan gjøre og varsle hvis en egenskap man trodde ble lagt til i modellen, ikke ble lagt til allikevel. For en enda grundigere gjennomgang av modellene kan det være en mulighet å legge til organismespesifikke tester.

Table of contents

List of Figures

1 Introduction

1.1 Fish farming

The aquaculture market is an important economical asset. Following the industry's rapid growth (1), resources for traditional feeds such as fish meal and fish oil have become scarce and increasingly expensive. Other potential alternatives have been tested, such as insect-based feed and protein sources from land animals (2), but due to considerations regarding ecological and economical sustainability, the feed is now more plant-based (3). Atlantic salmon (*Salmo salar*) can eat feed containing up to 50% plant proteins without any negative effects on growth or issues regarding welfare (4). However, salmon has evolved as a carnivore and plants are not a natural diet for the fish. If the portion of plant proteins in the feed exceeds 50% it causes amino acid deficiency, non-beneficial changes to the gut microbiota, and lower growth rate (4)(5), but this has so far been mitigated by food processing and dietary supplements. To achieve ecological and economical sustainability researchers are trying out more plant-based feeds, especially ones that are inedible for humans such as sawdust and seaweed. To determine how the fish can be fed with these novel feeds while animal health and welfare is sustained will require detailed insight into the systems biology of the salmon.

1.2 Systems biology and mathematical models

Systems biology is an approach to biological research that tries to understand how different processes in the cell are interconnected (6). Rather than looking at individual genes or proteins one at a time, it investigates the behavior and relationships of all the elements in a particular biological system while it is functioning (7). To better understand the biological system of choice the molecules and reactions are often systemized into mathematical models.

Mathematical models are a way to describe a system using mathematical concepts and language. A mathematical model is not a perfect representation of reality but can be useful for prediction and increase our understanding of the system. When the model has correctly predicted results for known conditions, it can be used to predict outcomes of conditions not yet investigated. Mathematical models can be used to simulate processes within the cell, or bigger networks such as the whole cell (6)(8). An example of such a network is the metabolism of an organism, which can be analyzed with metabolic modeling.

1.3 Metabolic modelling

Metabolic networks are complex and consist of hundreds or thousands of metabolites and reactions (6). These form pathways and the reactions and metabolites in the pathways can be arranged in the stoichiometric matrix, which has become an indispensable tool for studying the systems biology of metabolism (9). The rows in the matrix represent the metabolites and the columns represent the reactions (Figure 1). The metabolites and the stoichiometric coefficients of the metabolites impose constraints on the rates of reactions in the network. The matrix is then a model of the metabolic network and depending on what is to be studied, the model can encompass a varying degree of the metabolic network. A small model would encompass just the core carbon metabolism and on the other end is the genome-scale reconstruction which models the entire metabolic network. Eukaryotes have different compartments within the cell which must be considered and are therefore more difficult to model than prokaryotes. Multicellular systems further complicate the reconstruction.

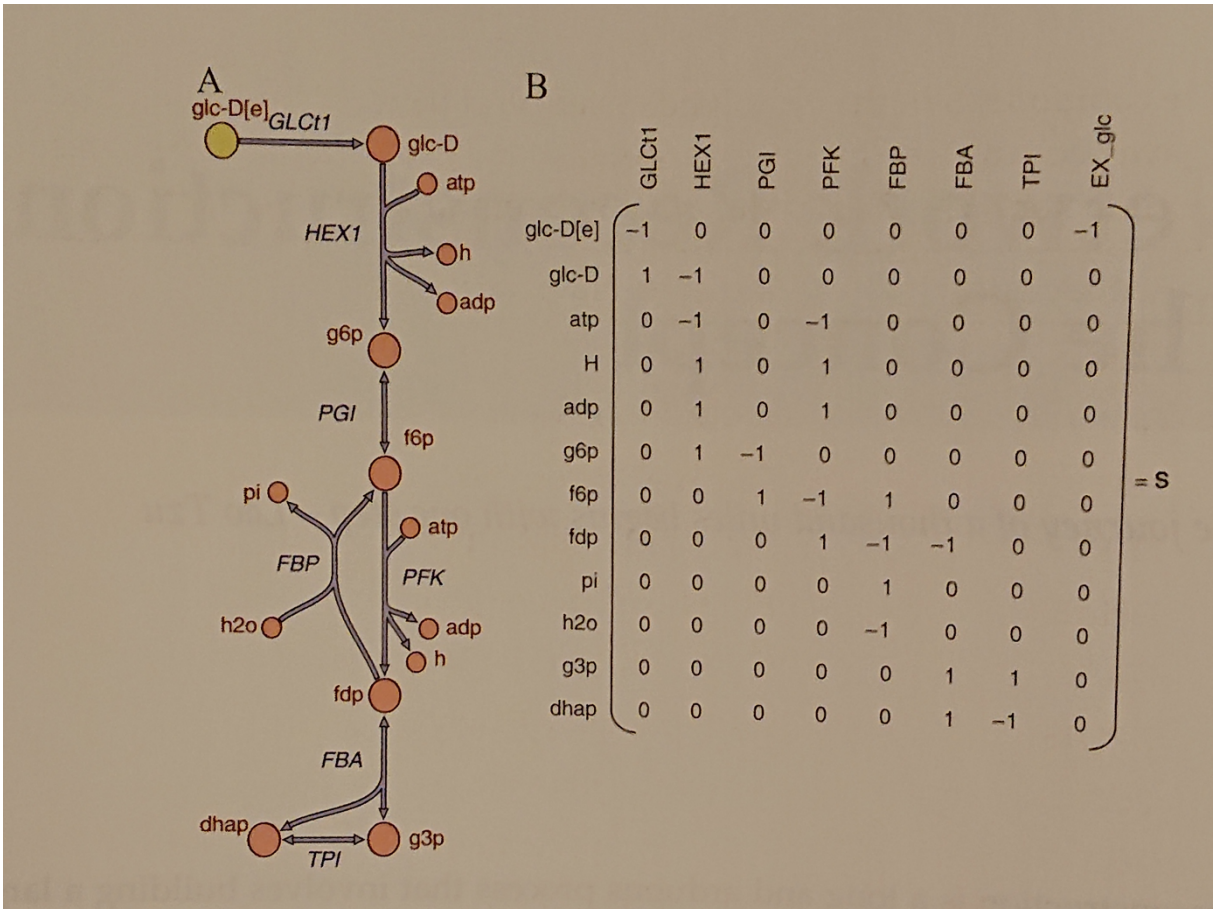


Figure 1. Panel (A) shows the first few reactions of glycolysis in a graphical form as a network of interacting reactions (arrows) with shared metabolites (dots). Panel (B) shows the stoichiometric matrix corresponding to panel (A). As indicated, the columns correspond to reactions and the rows to metabolites. Figure taken from ref. (6).

A metabolic system will in most cases have more reactions than metabolites. Consequently, the stoichiometric matrix ***S*** contains more columns than rows. In other words, there are more unknown variables than equations, so there is no unique solution to the system of equations (10). The mass balances of metabolites can be expressed as a system of differential equations, with metabolite concentrations **c** (11):

$$\frac{d\boldsymbol{c}(t)}{dt} = \boldsymbol{S} \cdot \boldsymbol{v}(t) \quad (1)$$

Where $\boldsymbol{S}$ is the stoichiometric matrix and $\boldsymbol{v}$(t) is the vector of reaction rates. However, since this equation is difficult to solve, we assume a quasi-steady-state on the system (11). This leads to the system of linear equations shown below:

$$\boldsymbol{S} \cdot \boldsymbol{v} = 0 \quad (2)$$

The solutions to this system of linear equations define the null space of $\boldsymbol{S}$, in which each point is a feasible combination of reaction rates at steady state known as a flux distribution. The solution space of the model consists of the portion of the null space that also satisfies other linear equality and inequality constraints on the network such as upper and lower flux bounds

1.3.1 Flux balance analysis

As Orth et al. (10) write, flux balance analysis (FBA) is a mathematical approach for analyzing the flow of metabolites through a metabolic network. This method uses an objective function to find an optimal solution within the solution space. This means that the output of FBA is a particular flux distribution which maximizes or minimizes the objective function. As for the constraints on the network, there are constraints from the coefficients in the stoichiometric matrix as well as capacity constraints distinguishing between reversible and irreversible reactions. The latter are in the form of upper and lower bounds. These constraints create an allowable solution space (Figure 2). Because the constraints are linear, the formed solution space is convex. This means that wherever you are in the solution space, you can always move to any other solution via a straight line. This also helps to find the optimal solution, as it will always be in a corner (Figure 2).

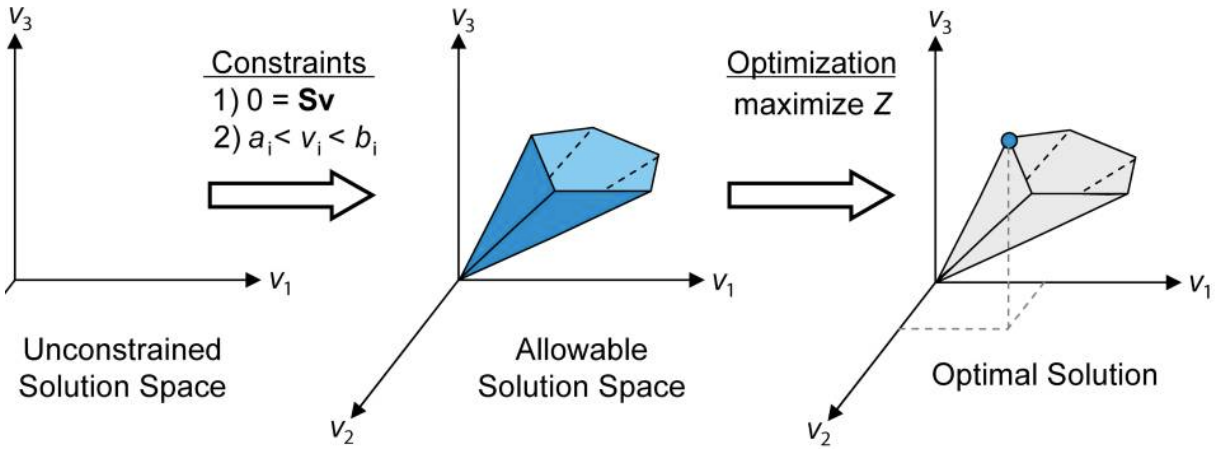


Figure 2. Diagram showing an unconstrained solution space, allowable solution space after constraints are imposed, and after the system is optimized with objective function to maximize Z. Figure taken from ref. (10).

The objective function is often related to cellular growth in models simulating the metabolic networks of microorganisms, because cells with a maximized cell growth tend to outcompete the other cells in the population (12). Since the aquaculture industry is interested in fast-growing fish, having biomass production as the objective function seems fitting. The biomass function has a reaction rate which is defined to be equal to the specific growth rate and has the unit 'per hours'. Specific growth rate is defined as the percentage of size increase per day.

Before you give the fish food that it is not evolved to digest, it can be helpful to simulate this on a computer to see how the metabolic network responds. Simulations can increase the understanding of processes that take place in the fish and can guide scientists to choose a feed to test or experiment to run. Such a simulation can be done using FBA. The molecules and reactions are, as mentioned, represented by a stoichiometric matrix. The environment or growth medium is represented by the rates of the uptake and secretion reactions. You can then simulate different feed compositions by regulating the uptake flux of metabolites into the metabolic network. This makes it possible to study how new feeds, e.g. differing in amino acid composition, can affect the biological network. In salmon farming, we are interested in large fish, which would require high growth rate for the cells in the body of the fish, so this is an important criterion for potential feeds.

1.4 Testing of metabolic models

To ensure that a metabolic model is as close to the metabolic network as possible, the quality of the model should be evaluated. There are protocols for constructing models, but a standardized means of quality control for metabolic models has been lacking (9)(13)(14). There is, however, a general consensus that the quality of the model is reflected in some fundamental features, such as the presence of a biomass reaction, multiple database annotations for genes and reactions, reactions being charge and mass-balanced, the presence of reactions and metabolites, and the portion of genes per reaction (13). Testing the model for the presence of these features and how they function in the model would give a score as to the quality of the model. Besides these model features, having a standard file format would encourage reuse, reproducibility and collaboration. Lieven et al. work towards having the Systems Biology Markup Language (SBML) as an official community standard (13). Additionally, having a standard file format would ensure that the test suite code can read the information in the model.

1.5 Memote

Memote (MEtabolic MOdel TEsting) is a Python software that was introduced for quality control of metabolic models (13). This software runs a series of tests to evaluate the quality of a model but is only a few years old and still in development. Version 0.1.0 of Memote was released in January 2017 (15). Memote tries to collect the quality criteria from the general consensus into four standardized categories: biomass reaction, annotations, stoichiometric consistency and basic tests. There are still shortcomings, such as no organism-specific and few functional tests (16). A functional test would for example be checking if the model can produce biomass. The tests give scores for different features in a metabolic model and present them in a report. Before each test result there is also a small section explaining the reasoning behind adding a specific test to the test suite and why this test is important to consider. See Figure 3 for the report for Mass balance. The report will often list which metabolites or reactions do not

meet the criteria in the test. This is because the test framework is designed to illuminate issues in the model so these can be fixed. At the end of the report the total score is presented, based on the score for the four categories. Memote uses Git, an open-source system for version control, to keep track of changes done to models. This is especially important in the Memote history report, which is a report where different scores for different versions of the models are displayed chronologically in graphs.

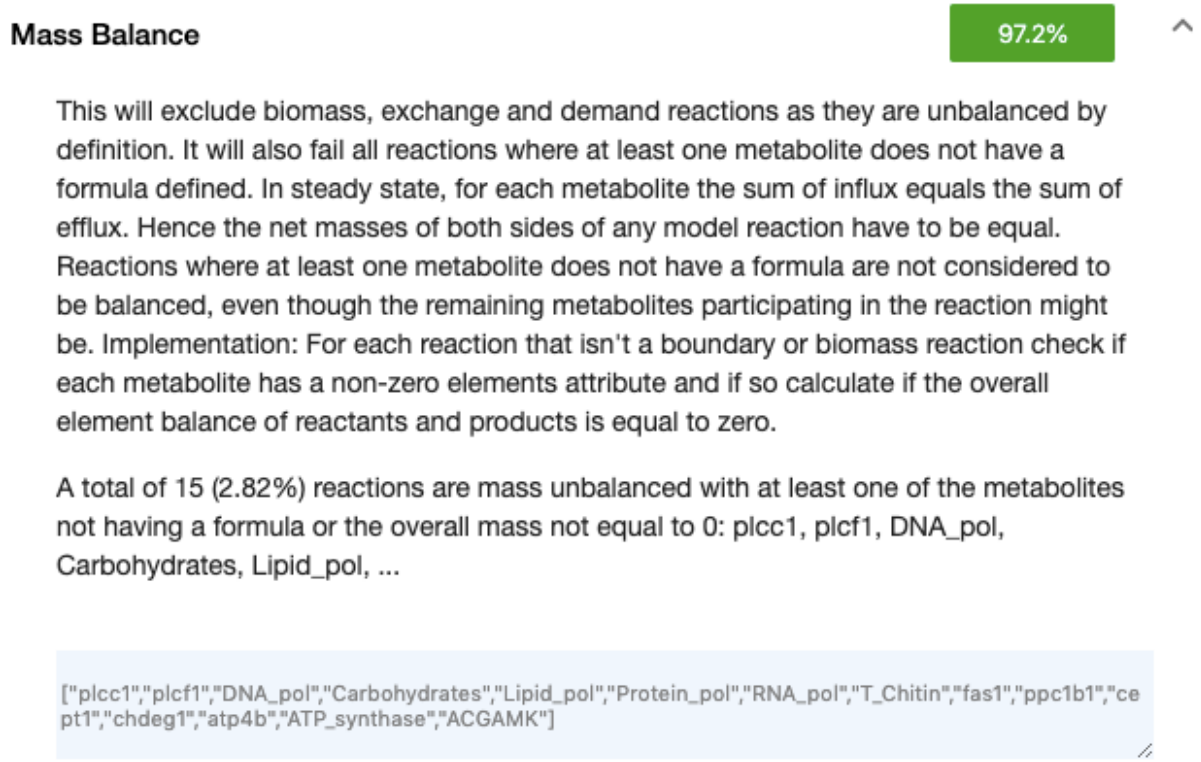


Figure 3. An example of how the score for a test in a Memote snapshot report is presented. There is a small section with information about the test, such as which metabolites or reactions are included. The green box in the upper right corner shows a percentage score for the test, in this case how many of the total reactions are mass balanced. At the bottom is also a list of reactions the test found to be not mass balanced. This picture is from a Memote snapshot report.

1.6 BiGG reference database

BiGG is a database consisting of genome-scale metabolic network reconstructions (17). Each network, as well as the components in the network, have an identifier called a BiGG ID. The genes in the BiGG models are mapped to NCBI genome annotations and metabolites are linked to external databases such as KEGG, PubChem and many more (17). This makes it easy to, for example, look up a metabolite in KEGG to inspect which other reactions the metabolite is connected to. You can search the BiGG database by typing in the name of a metabolite, reaction, gene or organism in the search bar. When working with metabolic models, we try to follow the BiGG ID conventions and make sure metabolites and reactions have IDs conforming to BiGG patterns.

1.7 Outline of problem

This thesis will discuss the development, application and validation of metabolic tests for a metabolic model of Atlantic salmon. This will be done by adding suggested improvements iteratively to the model and test it with Memote for each iteration, to see how Memote responds to and displays the model changes. To keep track of the changes, Git will be used.

2 Methods

2.1 Python and COBRApy

Python version 3.7.4 was used to make changes to the SBML-formatted metabolic model. The Python package COBRApy (COnstraint-Based Reconstruction and Analysis) was also necessary to work with the models (18). COBRApy version 0.17.1 was used. This package contains metabolic models for various organisms and software for refinement and analysis of the models. The toolbox is community-generated, allowing improvements on metabolic models to be added by every user. The coding that implemented changes to the model was done with Python in Jupyter Notebooks. A Jupyter Notebooks is an open-source web application for Python that allows live code, narrative text and visualizations of plots as well as tables.

2.2 Memote testing

For this work, Memote version 0.9.13 was used. It was run locally on a computer in the Terminal window. The unaltered model was first committed to Git for version control. When the model received a new feature in the Jupyter Notebook and was saved to keep the changes, Git would notice the file had changed and the new file had to be committed to version control. Then you could run the command “memote run” in the Terminal and Memote would evaluate the model and store the results in a JSON file. When all the planned additions were implemented and evaluated with Memote after each addition, the Memote history report could be composed using the command “memote report history” in the Terminal. Memote would then use Git to find the commits in which the model had been altered and compose a history report. It was important that the file kept the exact same file name throughout the editing, otherwise Git would view it as a new file if it had a different name and you would lose the file tracking.

2.2.1 Stoichiometry

A metabolic network in a living cell will be mass-balanced (19), but this will not automatically be the case in a model of such a network. Therefore, this category checks the consistency of stoichiometry and mass in the model. Errors in the stoichiometry can result in metabolites being produced from nothing which is not the case for a living cell or any other mass-balanced system (13). The mass balance of reactions is checked by counting how many metabolites have a mass equal to zero and counting reactions where overall mass is not equal to zero, see the Memote report attachment. The consistency tests also look for gaps in the network by checking for universally blocked reactions, orphan metabolites and dead-end metabolites. Universally blocked reactions are reactions that cannot carry any fluxes while all model boundaries are open. Orphan metabolites are metabolites that are consumed but not produced by any reactions in the model, and dead-end metabolites are produced but not consumed by reactions in the model.

2.2.2 Annotation

The annotation testing checks annotations for metabolites, reactions, genes and Systems Biology Ontology (SBO) terms and whether these annotations conform to specific patterns defined in the MIRIAM guidelines, i.e. matching the patterns on https://identifiers.org/ (13). Only when the patterns can be identified consistently is the ID truly machine-readable. Some of the databases that are included in the testing are Rhea, KEGG, MetaNetX and BiGG, see the Memote report attachment. The testing checks whether the model has included annotations for at least one of these, and the more annotations, the higher the score. The Systems Biology Ontology (SBO) annotations are also included in the testing. SBO annotations are controlled vocabularies of terms used in systems biology (20). This ensures standard terms for components in the models so there are fewer misunderstandings when comparing different models.

With the recent explosion of bioinformatics information, the number of unannotated genes is rapidly increasing (21). Further, Griesemer at al. (21) state that 30-50% of genes in a typical genome are still lacking annotation. More than 30% of these unannotated genes are estimated to have some metabolic function, which leaves a gap in our understanding of the underlying metabolic processes. In other words, there are still lots of models made based on genes lacking annotations, and yet, annotations are essential for collaboration and sharing as well as providing proof of the existence of the metabolite or reaction. The database annotations make it possible to identify metabolites, reactions and genes and enable cross-referencing between databases. Furthermore, collaborating, comparing and combining models is more manageable when the annotations are according to community standards. Another reason why annotations are valuable is that they make it possible to compare different model systems, by saying which parts of a model corresponds to parts in another model.

2.2.3 Biomass reaction

This test looks for the presence of a biomass reaction. This is a pseudo reaction in the model accounting for biomass synthesis in the modeled organism (13). This is biologically very important, since all organisms have evolved to produce biomass with the intention to grow and multiply. This is especially important in single-celled organisms, as the organism with the highest growth rate will often outnumber and therefor outcompete the other organisms in the environment (12). The test also looks for the biomass reaction precursors, if they have chemical formulas assigned and whether the model can synthesize them. The tests also check whether the growth rate is realistic, which means that it cannot exceed the growth rate of the fastest growing organism, *Vibrio natriegens*, with a reported doubling time of 14.8 minutes (22).

2.2.4 Basic tests

These tests verify the presence of metabolites, reactions and genes as well as gather information about them. They also calculate the metabolic coverage (13), which indicates the modeling detail of a reconstruction. This is tested because even though there are more and more metabolic network reconstructions released every year, the number of new reactions added to the models is not increasing (23). That means that the metabolic coverage in models has not progressed in line with the rising number of publications. Above 1 metabolic coverage is good and indicates high level of detail in the modeling. Below 1 in metabolic coverage indicates low

level of detail, and implies that many gene products and their enzymatic transformations are lumped together (23).

The number of counted reactions and metabolites indicate how big the model is, i.e. whether the model covers a small part of the metabolic network such as the central carbon metabolism or the full genome-scale network of a cell (13). Gene-protein-reaction (GPR) associations are also assessed. GPR annotations are important to justify the existence of reactions in the model, see the Memote report attachment. There can, however, be valid reactions that lack GPR. This can be the case in spontaneous reactions and known reactions with yet undiscovered genes.

2.3 Adding new features the model

The model received three new features that were added through three iterations and tested with Memote for each new addition, see the Jupyter Notebook attachment. The results for each Memote test were stored as JSON formatted files and were eventually used to generate a history report, where the score for the additions to the model could be viewed in graphs.

2.4 Essentiality of amino acids

In an early phase of the work, to demonstrate testing of metabolic functions, we looked at essentiality of amino acids. To test essentiality, we iterated through all the uptake reactions for amino acids, cancelled the uptake rate of the current amino acid into the model and optimized the model with maximal growth as the objective function. Amino acids that were required for growth were identified as essential.

3 Results

3.1 Memote history report

The model to be developed was missing features and under development. Two of the many shortcomings were missing BiGG database annotations and transport reactions. Suggested improvements that were to be implemented included addition of BiGG IDs, transport and exchange reactions of CO_2, and addition of automatically generated transport reactions. They were implemented in that order.

In the following plots, the leftmost dot represents the first version of the model, and each dot following to the right represents a new version of the model. Now, there are three features added to the model, but four dots in the plots (not counting the very first dot). The two rightmost dots represent the same feature: the automatically generated transport reactions. They were unknowingly added unsuccessfully first, and the Memote history report was composed. It was discovered when looking at the history report, that the number of reactions was not increasing when they should be. The transport reactions were then added and the score altered accordingly.

The Memote report is interactive when viewed on the computer. When you first open the report, all the test results are hidden. If you click on the title of a test, it will expand and reveal the result and informational text, such as in Figure 3, showing Mass balance from a snapshot report. When the mouse cursor hovers over the points in the plots, a small box with information appears, such as exact value for the point, commit identification as well as to which Git branch the commit was made. The branch overview on the right side of the graph shows with color-coding which commits are from which branches, see Figure 4. This is especially handy if you have separate work in two different branches. In the plots in this thesis, however, the branches master_work and origin/master_work contain the same work. The work was done in the branch master_work, but also pushed to the branch origin/master_work.

As Memote was run in the terminal and needed Git version control, it was tricky to get started and get the hang of it. Both Memote and Git needed to be installed through the terminal as well as run from there. Memote can also be run on the Memote web page, but then only a snapshot report of a single model. This somewhat difficult method may deter potential users.

After adding the BiGG IDs, the CO_2 transport and exchange reactions and the automatically generated transport reactions, the Memote history report was composed. Upon inspecting it, there is an overall score increase following the BiGG ID addition to the model. In the category “BiGG annotations” for reactions there is an increase from 0% to 75%. After version 2 the score decreases (Figure 4). Another annotation category, Systems Biology Ontology (SBO) annotations, are barely present in the model.

Figure 4. A graph showing the change in score for BiGG annotations for reactions. The exact score for the points is, respectively, 0, 75.38, 75.13, 75.13 and 30.9.

In the category "Transport reactions", there is an increase from 137 to 138 from version 2 to version 3, and then from 138 to 784 from version 4 to version 5 (Figure 5). Total reactions increase from 593 to 1246 (Figure 6). Total metabolites increase by 1 from version 2 to 3 and by 192 from version 4 to version 5 (Figure 7). Although these results are not unexpected, it is reassuring to see the Memote report confirming that reactions and metabolites have been added to the model. It also shows that the additions have been coded properly in the SBML-file so the model and the Memote tests can read it.

In Figure 5, the two rightmost dots, which both represent the automatically generated transport reactions, illustrate how Memote works. It tells you whether the feature, in this case reactions, was added to the model or not. When the automatically generated transport reactions initially were added, it was unsuccessful, but there was no error message to alert us. In this instance, Memote was very useful in helping to discover the mistake. It would have taken longer without Memote as one would have to manually inspect the number of reactions in the model. The mistake could possibly have gone by unnoticed. A failed addition to the model going unnoticed can cause problems further down the line, for example in giving an unexpected value for the optimal solution.

Transport Reactions

Cellular metabolism in any organism usually involves the transport of metabolites across a lipid bi-layer. This test reports how many of these reactions, which transports metabolites from one compartment to another, are present in the model, as at least one transport reaction must be present for cells to take up nutrients and/or excrete waste. Implementation: A transport reaction is defined as follows: 1. It contains metabolites from at least 2 compartments and 2. at least 1 metabolite undergoes no chemical reaction, i.e., the formula and/or annotation stays the same on both sides of the equation. A notable exception is transport via PTS, which also contains the following restriction: 3. The transported metabolite(s) are transported into a compartment through the exchange of a phosphate. An example of transport via PTS would be pep(c) + glucose(e) -> glucose-6-phosphate(c) + pyr(c) Reactions similar to transport via PTS (referred to as "modified transport reactions") follow a similar pattern: A(x) + B-R(y) -> A-R(y) + B(y) Such modified transport reactions can be detected, but only when the formula is defined for all metabolites in a particular reaction. If this is not the case, transport reactions are identified through annotations, which cannot detect modified transport reactions.

Figure 5. A graph showing the transport reactions for the different versions of the model. The exact score for the points is 137, 137, 138, 138 and 784.

Total Reactions

To be useful a metabolic model should consist at least of a few reactions. This test simply checks if there are more than zero reactions. Implementation: Check if the cobra.Model object has non-empty "reactions" attribute, this list is populated from the list of sbml:listOfReactions which should contain at least one sbml:reaction.

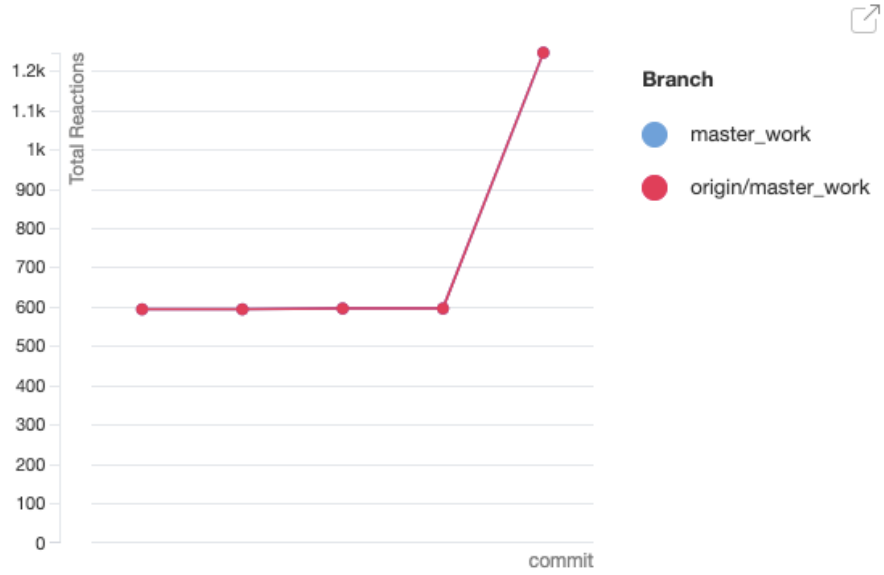


Figure 6. A graph showing the total reactions in the different versions of the model. The exact score for the points is, respectively, 593, 593, 595, 595 and 1 246.

Total Metabolites

To be useful a metabolic model should consist at least of a few metabolites that are converted by reactions. This test simply checks if there are more than zero metabolites. Implementation: Check if the cobra.Model object has non-empty "metabolites" attribute, this list is populated from the list of sbml:listOfSpecies which should contain at least one sbml:species.

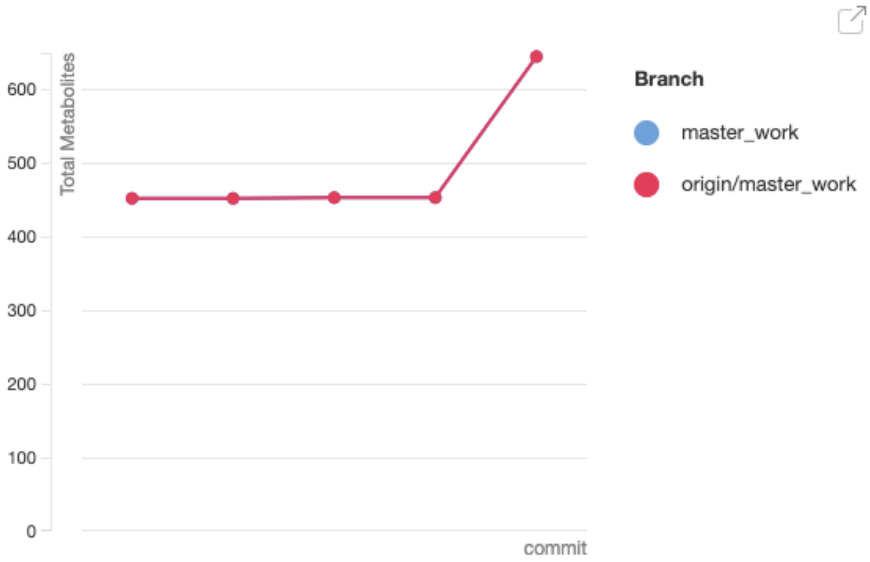


Figure 7. A graph showing the total metabolites for the different versions of the model. The exact score for the points is, respectively, 452, 452, 453, 453 and 645.

The total score for the model versions is at the highest for version 2, after which it decreases (Figure 8 and Figure 9). In the figures the three leftmost dots with the value zero are early versions of the model that were discarded.

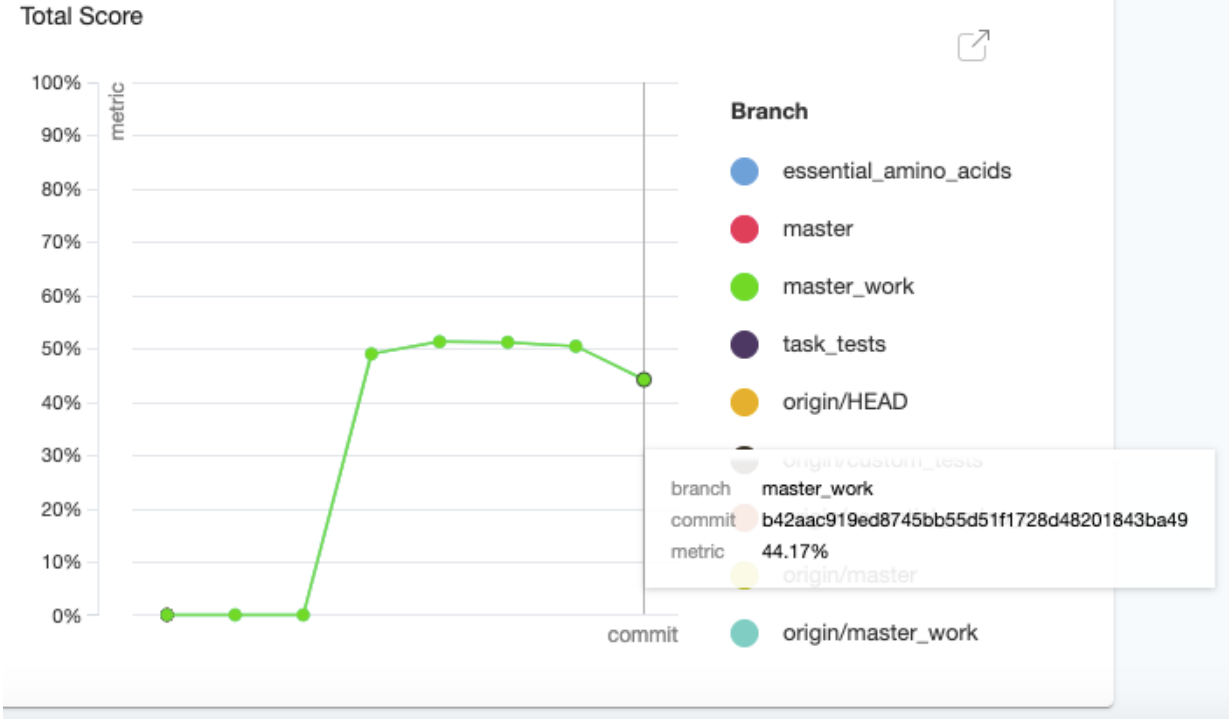


Figure 8. The total score on the Memote test for the last version of the model. The highlighted dot shows the score for the model version after adding the automatically generated transport reactions. The exact score for the points is, respectively, 49, 51.29, 51.19, 50.43 and 44.17.

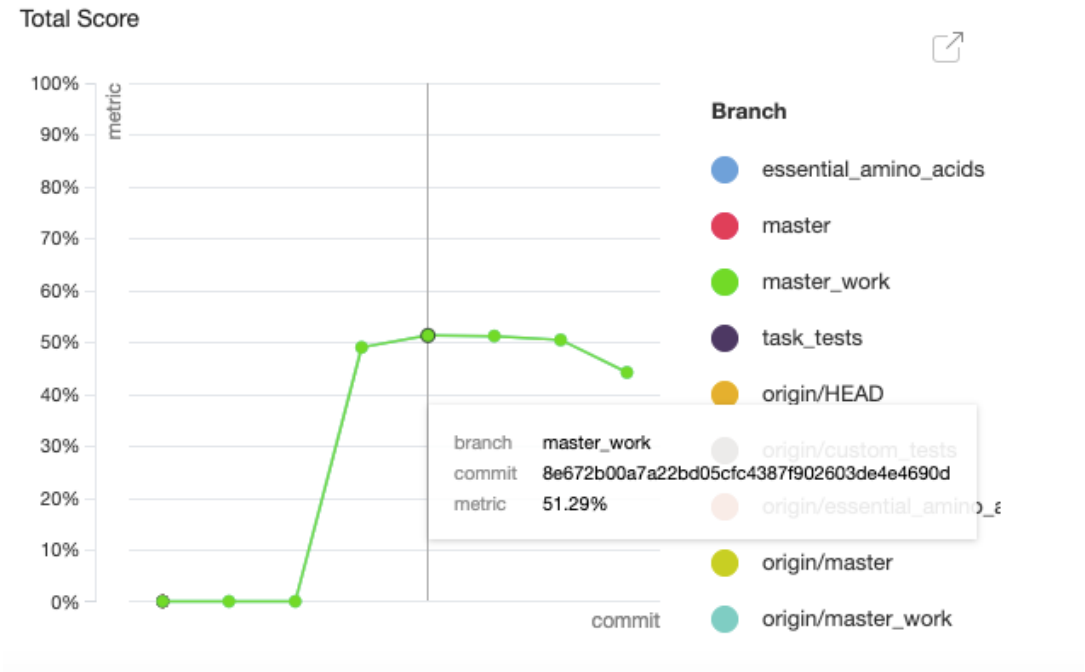


Figure 9. The total score on the Memote test. The highlighted dot shows the score for the model version after adding BiGG IDs. The exact score for the points is, respectively, 49, 51.29, 51.19, 50.43 and 44.17.

3.2 Optimal solution

The optimal solution, with maximal biomass production as objective, also increases due to the additions to the model. Even so, this does not occur until the CO_2 transport and exchange reactions are added. From the unaltered version of the model to the version with BiGG IDs, the objective value remains the same. Upon adding the CO_2 reactions, the solution increases from 78.358 per hour to 80.691 per hour, see Figure 10 and Figure 11. After adding the automatically generated transport reactions, the optimal solution increases to 114.139 per hour, see Figure 12.

```
In [11]: ver2.optimize()
```

Out[11]: ***Optimal* solution with objective value 78.358**

	fluxes	reduced_costs
EX_pchol_cho_e	0.000000	0.000000e+00
EX_akg_e	71.422365	0.000000e+00
EX_adn_e	0.000000	0.000000e+00
EX_Butyrate_e	0.000000	0.000000e+00
EX_chol_e	-11.897725	0.000000e+00
...	...	...
DNA_pol	0.015577	8.881784e-16
Glycogen_pol	0.035319	-8.881784e-16
Lipid_pol	0.118977	0.000000e+00
Protein_pol	1.204576	-2.220446e-14
RNA_pol	0.005802	-2.842171e-14

593 rows × 2 columns

Figure 10. A code chunk from a Jupyter Notebook showing optimal solution for the model after adding BiGG IDs.

```
In [12]: ver3.optimize()
```

Out[12]: ***Optimal* solution with objective value 80.691**

	fluxes	reduced_costs
EX_pchol_cho_e	0.000000	0.000000e+00
EX_akg_e	-131.738793	0.000000e+00
EX_adn_e	0.000000	0.000000e+00
EX_Butyrate_e	0.000000	0.000000e+00
EX_chol_e	-12.251954	0.000000e+00
...	...	...
Lipid_pol	0.122520	0.000000e+00
Protein_pol	1.240440	-6.217249e-15
RNA_pol	0.005975	-4.529710e-14
EX_co2_e	1000.000000	2.954120e-03
CO2t	-1000.000000	-0.000000e+00

595 rows × 2 columns

Figure 11. A code chunk from a Jupyter Notebook showing optimal solution for the model after adding the CO2 transport and exchange reactions.

```
In [4]: ver4.optimize()
```

Out[4]: ***Optimal*** **solution with objective value 114.139**

	fluxes	reduced_costs
EX_pchol_cho_e	0.000000	0.0
EX_akg_e	-558.347400	0.0
EX_adn_e	918.384172	0.0
EX_Butyrate_e	0.000000	0.0
EX_chol_e	-17.330620	0.0
...	...	...
FOLTle	0.000000	0.0
MLTHFte	0.000000	0.0
5MTHFt	1000.000000	0.0
r0963	0.000000	0.0
MLTHFte3	0.000000	0.0

1246 rows × 2 columns

Figure 12. A code chunk from a Jupyter Notebook showing optimal solution for the model after adding the automatically generated transport reactions.

3.3 Essentiality of amino acids

Arginine is supposed to be an essential amino acid in Atlantic salmon (24)(25), and therefore when cutting off arginine uptake and then optimizing the model, should give an optimal solution of zero. This was however not the case, as arginine was, according to the model, non-essential. In other words, the solution when optimizing the model was not zero. However, the optimal solution for arginine is slightly lower than the optimal solution for the other non-essential amino acids (Figure 13). Version 2 of the model was used in these computations. The model characterized all the other amino acids correctly as essential or non-essential.

In [6]:
```
for r in aax.members:
    old_bounds = r.bounds # setting the default bounds as the variable old_bounds
    r.bounds = (0,1000) # setting uptake of r-th amino acid to 0
    print(r.id, sasa.optimize().objective_value) # printing the r-th amino acid along with the objective value
    #when the model is optimized
    r.bounds = old_bounds # setting the bounds back to the default value
```

```
EX_trp__L_e 0.0
EX_orn__L_e 78.39269391555355
EX_asn__L_e 77.76695390722615
EX_his__L_e 0.0
EX_tyr__L_e 78.39269391555354
EX_phe__L_e 0.0
EX_ile__L_e 0.0
EX_asp__L_e 78.39269391555354
EX_val__L_e 0.0
EX_pro__L_e 78.39269391555347
EX_leu__L_e 0.0
EX_cys__L_e 78.30607999911464
EX_ala__L_e 78.39269391555356
EX_ser__L_e 78.39269391555356
EX_lys__L_e 0.0
EX_gly_e 77.78096336434935
EX_glu__L_e 78.3926939155536
EX_thr__L_e 0.0
EX_arg__L_e 75.00467289066887
EX_met__L_e 0.0
EX_gln__L_e 78.39269391555355
```

Figure 13. A code chunk showing a for loop iterating through a list of all the amino acid transport reactions, setting the uptake to zero for whichever reaction the loop is on and lastly optimizing the model. The reaction identification is in the left column and the value for the optimal solution is in the right column. Arginine is the third from the bottom.

4 Discussion

4.1 Memote history report

The score increase for BiGG annotations for metabolites exceeded that of reactions. This may be because it was easier to add IDs for metabolites than reactions, as there is not always one answer to which metabolites a reaction contains. Perhaps the same reaction is in two databases but in one of the reactions there is a proton that has been left out in the other database. This can also be seen in the code for adding BiGG IDs: the chunk regarding the reactions is longer and more extensive than the chunk for metabolites. Sometimes the BiGG IDs for reactions are not following the same pattern and therefore the code must be more extensive to recognize the different IDs.

The importance of collaboration and sharing is heavily emphasized in the Memote report, where the category for annotations of metabolites, reactions and genes make up a large portion of the report. This is reflected in the total score change after adding BiGG IDs, it increases by 2% and confirms the importance of a wide range of annotations. Similarly, when the score increases when annotations are added, it will also decrease when reactions and metabolites are added without annotations. This is evident in the total score variation in Figure 8. After version 2 the total score declines. The two last alterations consisted of adding multiple reactions and metabolites, but since they were added after the BiGG IDs additions, they did not receive BiGG IDs and thus there are less annotations percentage-wise. Version 4 has the most reactions without BiGG annotations, so that may explain why the score is the lowest at the last version. Additionally, the model does not include many Systems Biology Ontology (SBO) annotations. SBO annotations are also important for collaboration and comparison, as they provide standard terms for components in the models. The lack of SBO annotations may negatively affect the score.

In Figure 8, there are three dots to the left with the value 0. These are from previous work with the model. This work was however discarded, but the points still show up in the report. When composing the history report, Git finds all the commits in which the model has been altered and looks for the results of "memote run" in a JSON file. These JSON files from the old commits were moved to another folder and so Git couldn't find them and the score for that commit in the report was consequently zero. Why these old commits only show up in the Total score graph in the Memote report may be because the Total score graph is composed slightly differently than the other graphs in the report. When composing the history report, Git looks for all the commits in which the model has been changed and then looks for the JSON files containing test results. Then it uses the located result files to extract test results, for example for Total reactions, and assemble graphs. When making the Total score graph, all the commits are included, regardless of whether there are corresponding result files. That might be the reason why the old commits only show up in the Total score plot.

There are tools to help reconstructing metabolic networks (26). These tools are developed to speed up the reconstruction process by automating several tasks, such as gap filling and draft network generation. Mendoza et al. have evaluated these tools (26), and evaluating their performance can help researchers choose the best tool to help their reconstruction. However, when the reconstruction is finished and you have a metabolic model, there are fewer tools to choose from to evaluate the model. Now, Memote can evaluate a metabolic model, but it is still very new, and before Memote there were few quality control systems for metabolic models (27). Working with and trying to improve a metabolic model without having a test system is challenging. The network reconstruction work is in itself time-consuming and cumbersome,

and chances are you will overlook a reaction or metabolite. The same can be said about adding new features to the model. Additionally, when you think you have added a feature, but it was in fact unsuccessful, it can go unnoticed. Without a test system like Memote, you have to use your own knowledge to search for and find solutions for issues in the model, which can be frustrating and time-consuming. Another issue with which Memote really helps, is if features for some reason disappear during the model development. COBRApy is still under development and bugs can happen (18). This can cause components to disappear from the model when writing and reading models. Components disappearing is not something you think to check for, so Memote notifying us is very useful.

Another question is how much you trust the model when it gives unexpected results, especially beyond known conditions. When the marine flagellate *Chrysochromulina* blooms it can cause mortality in marine organisms, including fish in aquaculture (28). This of course has financial consequences for the fish breeders and it could perhaps be useful to simulate how much of this toxin the fish can withstand to get an idea as to what to do to prevent fish death. However, one must exert caution if such a simulation were to take place, because if the fish breeders thought the model was of high quality, but the model produced inaccurate results, e.g. indicating that the fish could handle more toxin than what was actually true, it would result in high fish mortality. If a metabolic model were to be used for this kind of simulation one must be extremely certain of the model quality. Metabolic models do not always contain the components with which toxins react, but it is an example of a situation where one must be very careful when interpreting the results, and even more cautious to trust them.

Simulations are useful as a preliminary round of experiments. The more you trust your model, the more you can trust the simulation results to reflect reality. Even so, it is important to remember that a simulation can only give pointers and not replace *in vivo* experiments. Memote can aid in validating the quality of the model, but only to a certain point, only as far as the Memote tests go. Beyond that, you have to use your own knowledge and manually inspect the code of the model. Only what is presented in the report is tested, so you will know which aspects and areas of the model it is more likely that you will have to inspect yourself. Moreover, Memote can perhaps shorten the distance between simulations and *in vivo* experiments by providing a quick and trusted quality control of the model.

Simulations done with this metabolic model, such as optimizing for biomass growth, are reproducible in the sense that the model follows a standard format: SBML, and it is not hard-coded for a single experiment (29). It is possible that Memote can help in ensuring that the model meets these two requirements. Memote encourages models to be in the SBML format, as the Memote tests are coded to read SBML. By testing the model for the agreed-upon quality criteria, it contributes to making a model less specific for only one experiment by reporting which of the general quality criteria are lacking and needs to be added or adjusted in the model. Furthermore, a model hard-coded for a single experiment will perhaps lack annotations, which Memote will report.

Adding organism-specific tests would increase our knowledge of what the model is capable of. Some examples are: For a model on an anaerobic organism, testing whether the model requires oxygen would be beneficial for the quality of the model. For a model describing a biological network of a eukaryote, checking for reactions in mitochondria would be central. The same could be applied for networks of plant cells, but then also checking for photosynthesis reactions. As well as testing essentiality of amino acids in the model. Additionally, perhaps a test that checked whether waste-product metabolites were consumed would be a good idea to add. This would require a list in a database and it would contain known waste products such as

CO_2 and urea. A similar test could be added for metabolites that should not be produced in the model, such as essential amino acids and other essential nutrients.

This metabolic model covers the metabolic network of Atlantic salmon and can hopefully be used in predictions for novel feeds once the model quality is good. During the work for this thesis, a small contribution was made to improve the model quality, but seeing as the total score is below 50%, the model still needs plenty of development. As we worked with this model, the objective function was always biomass growth, but there may be additional criteria contributing to the meat quality. Around the world, fish is an extremely important food because of its nutritional value (30). Perhaps the objective function in metabolic models can be reformed to a weighted sum of different equations from the model, for example biomass growth counting 70% and nutritional content counting 30% of the total. Further, the aquaculture industry could then consider, based on simulations, whether a novel feed will not only produce fish with large muscles, but also if the meat contains sufficient nutrients. Fat content can maybe also be interesting to consider as a factor in the objective function. Fish breeders could subsequently decide if a particular feed is worth testing or developing further.

4.2 Usability of Memote

The more Memote is used by different researchers, the lower the threshold for sharing models will become. This will increase the use of metabolic models as well as our understanding of biological systems and biology as a whole. But many users may find it challenging to use Memote. The Memote history report is very useful for looking at how the model score develops, but a history report can only be composed from the Terminal window on a computer. A function on the Memote website to make a history report will make it available for a larger user group.

It would be very beneficial if the Memote report could be converted to a PDF file and still be neat and readable. On my system setup, I had to go through the “Print”-function on the computer to save it as a PDF file, and the resulting file becomes unreadable. The graphs and text boxes are piled on top of each other. This was something my colleagues also experienced. A nice PDF file would make it easier to share the Memote report. Additionally, in a Memote snapshot report, when showing the total score, there is also a graph showing the percentage of total possible score for each category. This provides a good overview of which categories are lacking the most in score and which areas in the model need fixing or additions. The Memote history report does not include this, but it would be beneficial if it did. If the total score decreases, it would be useful if there was a graph displaying the scores in each category for each version of the model and one could see exactly where it decreased. Further development will then be more targeted for a wider user group.

4.3 Optimizing the model

The increase of 2.333 in the optimal solution from version 2 and 3 of the model suggests the CO_2 transport and exchange reactions were beneficial to include in the model. From version 3 to 4 the optimal solution increases by 33.448. The transport reactions are from the extracellular area to cytosol, for the metabolites already in cytosol. This increase in biomass production can be caused by more available pathways to transport the metabolites into the model. With these

new transport reactions, if there now are two new pathways available for a metabolite, the total flux of that metabolite into the network would increase three-fold, given the flux is equal in all the transport reactions. An abundance of metabolites in the network could increase the biomass production. Additionally, if an essential metabolite could now access a pathway that required less fuel, such as ATP or NADP, it would contribute to the increasing biomass growth.

4.4 Essentiality of amino acids

The model showed arginine as a non-essential amino acid. According to the Food and Agriculture Organization of the United Nations (FAO), arginine is an essential amino acid in Atlantic salmon (24). In channel catfish (*Ictalurus punctatus*) there are indications that arginine is a conditionally essential amino acid. If the diet of the catfish contained a surplus of glutamine, dietary arginine requirements were reduced (25). This is however not the case for Atlantic salmon as there has not been observed a pathway connection between arginine and glutamine yet (31). Arginine should therefore behave like an essential amino acid when processed in the model, even though it did not. Furthermore, the optimal solution was slightly lower than the optimal solution for the other non-essential amino acids, see Figure 13. This was interesting because it showed that even though the model could give a feasible solution, it was not ideal for the model to manage without arginine. Since the optimal solution was lower than for the other non-essential amino acids, the model has perhaps used alternative and less effective pathways when arginine was cut off, which would result in a slightly lower value. On the positive side, this shows that we know enough about the metabolic processes of salmon to create a model that works, since many of the other amino acids were processed normally by the model.

This could be an example of a metabolic function which is relevant to test. It would then be an organism-specific test, since essential amino acids differ from different organisms. As this incident revealed a big flaw in the model, such a test would be useful to add to the model development. Adding such a custom test to the test suite would be quite straight-forward, you simply make a Python script with the test and place it in the Memote test suite folder on your computer.

5 Conclusions and outlook

In this thesis, I have added three new features iteratively to an SBML-formatted metabolic model, and tested the model with the software Memote. We found that the model did not handle arginine correctly, and this error was not reported by Memote. We also found that the biomass growth increased after adding CO_2 transport and exchange reactions. The automatically generated transport reactions were first unsuccessfully added unsuccessfully by me, and they were correctly reported missing by Memote. Further developments for this model could include adding lipid pathways into the model and SBO annotations.

Adding BiGG annotations to the model made the total score increase, this indicates the importance of annotations in the model. It also implies that as more models are developed, there needs to be a focus on including several database annotations, as this will enable cooperation and comparison of models between different research groups and environments.

Testing metabolic models is important because it contributes to verifying the quality and clarifying the scope of the model. Having a clear view of the scope of the model will make it easier to choose in which experiments the model can be used. Even though simulations will only act as a guide to choose which wet lab experiment to run, it is important that the simulations give as accurate results as possible. Testing with Memote can aid researchers in this process. Additionally, having a solid and trusted model which follows a standard file format will contribute to reproducibility (29)(32). This means that a high quality model will be a good machinery for testing robustness in the results, using different conditions.

Memote can help scientists choose what direction to take in the model development. Knowing what the model contains is a good starting point for further development. The addition of organism-specific metabolic tasks could also be important for improvement. Furthermore, there is a possibility that Memote can aid in the integration of new knowledge into existing models, i.e. provide a quick testing of the model after the new features are implemented to ensure that they are added correctly. Besides, it is also possible that with Memote's support, the speed of model development will rapidly escalate.

High quality models can contribute to an increased understanding of biology. With Memote, the model can be developed until it gives accurate results for known conditions. Given that the model then is near identical to the metabolic network it covers, it can then be used to simulate unknown conditions. Moreover, there could even be a possibility to use the model in simulations with conditions that would be unethical to carry out *in vitro* or even *in vivo*, such as how the metabolic system reacts to a possible toxin, or studying which feeds are lethal or not to an organism. This could open up possibilities for the aquaculture industry to try out even stranger new feeds, if the simulation results seem promising. If it is possible to change the objective function to a weighted sum of different equations, fish breeders can also get an estimate of other qualities in the meat, e.g. the nutritional value or fat content.

If a software could be developed which integrates reconstruction tools, such as the ones Mendoza et. al (26) evaluate, and Memote to help in the reconstruction of a genome. It could then be a possibility to regularly test the model during the early development. Imaginably, a high-quality model could open new possibilities for simulations. It could be possible to use the model in a reverse way to search for new metabolites, i.e. add different non-existing metabolites with an invented molecular formula to the model to see if any of them increase the biomass growth substantially, or in other ways affect the network positively. The next challenge would then be to synthesize such a metabolite in a laboratory.

With a model of good quality, new possibilities for simulations may open and our understanding of biology may increase. To achieve good quality, many users must use and develop the model. Memote has a contributing role in both the quality and increased use of metabolic models. Memote is a great tool for systems biology and metabolic modelling, but it can be even better with further development.

6 References

1. Directory of fisheries. Totalt, hele næringen [Internet]. Fiskeridirektoratet. 2016 [cited 2020 Mar 27]. Available from: https://www.fiskeridir.no/Akvakultur/Tall-og-analyse/Akvakulturstatistikk-tidsserier/Totalt-hele-naeringen

2. Belghit I, Liland NS, Waagbø R, Biancarosa I, Pelusio N, Li Y, et al. Potential of insect-based diets for Atlantic salmon (Salmo salar). Aquaculture. 2018 Apr;491:72–81.

3. Ayadi FY, Rosentrate KA, Muthukumar K. Alternative Protein Sources for Aquaculture Feeds. J Aquac Feed Sci Nutr. 2012 Jan 1;4(1):1–26.

4. Egerton S, Wan A, Murphy K, Collins F, Ahern G, Sugrue I, et al. Replacing fishmeal with plant protein in Atlantic salmon (Salmo salar) diets by supplementation with fish protein hydrolysate. Sci Rep. 2020 Dec;10(1):4194.

5. Collins SA, Øverland M, Skrede A, Drew MD. Effect of plant protein sources on growth rate in salmonids: Meta-analysis of dietary inclusion of soybean, pea and canola/rapeseed meals and protein concentrates. Aquaculture. 2013 Jun 20;400–401:85–100.

6. Palsson BO. Systems Biology: Constraint-based Reconstruction and Analysis. First. Cambridge University Press;

7. Ideker T, Galitski T, Hood L. A new approach to decoding life: systems biology. Annu Rev Genomics Hum Genet. 2001;2:343–72.

8. Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, Bolival B, et al. A whole-cell computational model predicts phenotype from genotype. Cell. 2012 Jul 20;150(2):389–401.

9. Thiele I, Palsson BØ. A protocol for generating a high-quality genome-scale metabolic reconstruction. Nat Protoc. 2010;5(1):93–121.

10. Orth JD, Thiele I, Palsson BØ. What is flux balance analysis? Nat Biotechnol. 2010 Mar;28(3):245–8.

11. Terzer M, Maynard ND, Covert MW, Stelling J. Genome-scale metabolic networks. Wiley Interdiscip Rev Syst Biol Med. 2009 Nov;1(3):285–97.

12. Schuetz R, Kuepfer L, Sauer U. Systematic evaluation of objective functions for predicting intracellular fluxes in Escherichia coli. Mol Syst Biol [Internet]. 2007 Jul 10 [cited 2020 Apr 8];3. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1949037/

13. Lieven C, Beber ME, Olivier BG, Bergmann FT, Ataman M, Babaei P, et al. MEMOTE for standardized genome-scale metabolic model testing. Nat Biotechnol. 2020 Mar;38(3):272–6.

14. Ravikrishnan A, Raman K. Critical assessment of genome-scale metabolic networks: the need for a unified standard. Brief Bioinform. 2015 Nov 1;16(6):1057–68.

15. History — memote [Internet]. [cited 2020 May 23]. Available from: https://memote.readthedocs.io/en/stable/history.html#id49

16. Uhlén M, Fagerberg L, Hallström BM, Lindskog C, Oksvold P, Mardinoglu A, et al. Tissue-based map of the human proteome. Science [Internet]. 2015 Jan 23 [cited 2020 May 4];347(6220). Available from: https://science.sciencemag.org/content/347/6220/1260419

17. Schellenberger J, Park JO, Conrad TM, Palsson BØ. BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. BMC Bioinformatics. 2010 Apr 29;11(1):213.

18. Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst Biol. 2013 Aug 8;7(1):74.

19. Nelson DL, Nelson DL, Lehninger AL, Cox MM. Lehninger principles of biochemistry. New York: W.H. Freeman; 2008.

20. Novère NL. BioModels.net, tools and resources to support Computational Systems Biology. :9.

21. Griesemer M, Kimbrel JA, Zhou CE, Navid A, D'haeseleer P. Combining multiple functional annotation tools increases coverage of metabolic annotation. BMC Genomics [Internet]. 2018 Dec 19 [cited 2020 May 2];19. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6299973/

22. Lee HH, Ostrov N, Wong BG, Gold MA, Khalil AS, Church GM. *Vibrio natriegens* , a new genomic powerhouse [Internet]. Genomics; 2016 Jun [cited 2020 Apr 13]. Available from: http://biorxiv.org/lookup/doi/10.1101/058487

23. Monk J, Nogales J, Palsson BO. Optimizing genome-scale network reconstructions. Nat Biotechnol. 2014 May;32(5):447–52.

24. FAO: Nutritional requirements [Internet]. [cited 2020 Apr 28]. Available from: http://www.fao.org/fishery/affris/species-profiles/atlantic-salmon/nutritional-requirements/en/

25. Espe M. Functional amino acids in fish nutrition health and welfare. Front Biosci. 2016;8(1):143–69.

26. Mendoza SN, Olivier BG, Molenaar D, Teusink B. A systematic assessment of current genome-scale metabolic reconstruction tools. Genome Biol. 2019 Aug 7;20(1):158.

27. Gilbert J, Pearcy N, Norman R, Millat T, Winzer K, King J, et al. Gsmodutils: a python based framework for test-driven genome scale metabolic model development. Bioinformatics. 2019 Sep 15;35(18):3397–403.

28. Simonsen S, Moestrup Ø. Toxicity tests in eight species of *Chrysochromulina* (Haptophyta). Can J Bot. 1997 Jan 1;75(1):129–36.

29. Cooper J, Vik JO, Waltemath D. A call for virtual experiments: Accelerating the scientific process. Prog Biophys Mol Biol. 2015 Jan;117(1):99–106.

30. Pal J, Shukla B, Maurya AK, Verma HO. A review on role of fish in human nutrition with special emphasis to essential fatty acid. :4.

31. Andersen SM, Holen E, Aksnes A, Rønnestad I, Zerrahn J-E, Espe M. Adult Atlantic

salmon (Salmo salar L.) adapts to long-term surplus dietary arginine supplementation. Aquac Nutr. 2015;21(3):355–63.

32. Drummond C. Replicability is not Reproducibility: Nor is it Good Science. :4.

7 Attachments

Jupyter Notebook – 9pages

Memote history report – 22pages

Adding new features to a metabolic model

This Jupyter Notebook contains the code for adding three new features to a metabolic model.

In [1]:

```
import cobra # importing the package for constraint based reconstruction and ana
lysis
```

Adding BiGG IDs

Will add BiGG IDs to the existing metabolites and reactions in the model.

In [12]:

```
from ontology_translator import * # importing all the functions from the python
 script ontology_translator
```

In [13]:

```
from addBiggIDs import * # importing all the functions from the python script ad
dBiggIDs
```

In [ ]:

```
m = cobra.io.read_sbml_model('Salmo_salar.sbml') # reading the model

for met in m.metabolites:
    add_bigg_metabolite(met) # using a function from the addBiggIDs script
m = convertMetIDsBiGG(m)
id_to_bigg(m)

d = bigg_rxn_set()
for rxn in m.reactions:
    try:
        results = reaction_to_sets(rxn)
    except:
        results=[False]
    for result in results:
        bigg = d.get(result, False)
        if bigg:
            rxn.annotation['bigg.reaction'] = bigg

[add_bigg_reaction(r) for r in m.reactions]
# m = biggify_exchanges(m)  # update exchanges, must do before convertMetIdsBiGG
m = biggify_metabolites(m)  # Add BiGG IDs from curation text file
for reaction in m.reactions:
    bigg = reaction.annotation.get('bigg.reaction', False)
    if bigg:
        try:
            reaction.id = bigg
        except:
            continue
for r in m.reactions:
    if r.annotation.get('alternative.bigg',False):
        r.annotation.pop('alternative.bigg')

cobra.io.write_sbml_model(m, 'Salmo_salar.sbml')
```

In [ ]:

```
ver2 = cobra.io.read_sbml_model('Salmo_salar.sbml')
```

In [ ]:

```
ver2.optimize()
```

Will then run memote on the updated model. Have committed and pushed the model to origin. Will do this step after every addition.

Adding manual curations

Transport and exchange reaction for CO2

In [ ]:

```
# importing the necessary packages
from cobra import io
import cobra
from cobra import Model, Reaction, Metabolite
import libsbml
from cobra.core import Group
#import memote
import pytest
#import memote.support.basic as basic
import copy
%matplotlib inline
#import plot_helper
import cobra.test
from cobra.flux_analysis.loopless import add_loopless, loopless_solution
from cobra.flux_analysis import pfba

import hashlib

from collections import defaultdict
from copy import copy, deepcopy
from functools import partial
from operator import attrgetter
from warnings import warn

from six import iteritems, iterkeys, string_types

from cobra.exceptions import OptimizationError
from cobra.core.gene import Gene, ast2str, parse_gpr, eval_gpr
from cobra.core.metabolite import Metabolite
from cobra.core.object import Object
from cobra.util.context import resettable, get_context
from cobra.util.solver import (
    linear_reaction_coefficients, set_objective, check_solver_status)
from cobra.util.util import format_long_string
```

In [ ]:

```
model = cobra.io.read_sbml_model('Salmo_salar.sbml') # reading the newest versio
n of the model
# add co2 metabolite
co2_e = cobra.Metabolite(
    'co2_e',
    formula = 'CO2',
    name = 'CO2',
    compartment= 'e'
)
model.add_metabolites(co2_e)

#Add exchange of co2 reaction
rxn_co2_exchange = cobra.Reaction('EX_co2_e')
rxn_co2_exchange.name = 'Exchange of CO2'
rxn_co2_exchange.lower_bound = -1000.0
rxn_co2_exchange.upper_bound = 1000.0
model.add_reaction(rxn_co2_exchange)

model.reactions.EX_co2_e.add_metabolites({
    model.metabolites.co2_e : -1.0
})

model.reactions.EX_co2_e.upper_bound = 1000.0
model.reactions.EX_co2_e.lower_bound = -1000.0

#Add transport of co2
rxn_co2_transport = cobra.Reaction('CO2t')
rxn_co2_transport.name = 'Transport of CO2'
rxn_co2_transport.lower_bound = -1000.0
rxn_co2_transport.upper_bound = 1000.0

model.add_reaction(rxn_co2_transport)

model.reactions.CO2t.add_metabolites({
    model.metabolites.co2_e: -1.0,
    model.metabolites.co2_c: 1.0
})

# check if EX is in exchanges
```

In [8]:

```
model = cobra.io.read_sbml_model('Salmo_salar.sbml') # reading the newest versio
n of the model
```

Checking that the reactions and metabolites are in the model

In [3]:

```
model.exchanges.EX_co2_e
```

Out[3]:

Reaction identifier	EX_co2_e
Name	Exchange of CO2
Memory address	0x01022dc41d0
Stoichiometry	co2_e <=> CO2 <=>
GPR	
Lower bound	-1000.0
Upper bound	1000.0

In [4]:

```
model.reactions.CO2t
```

Out[4]:

Reaction identifier	CO2t
Name	Transport of CO2
Memory address	0x01022dc48d0
Stoichiometry	co2_e <=> co2_c CO2 <=> CO2
GPR	
Lower bound	-1000.0
Upper bound	1000.0

In [5]:

```
model.metabolites.co2_e.reactions
```

Out[5]:

```
frozenset({<Reaction CO2t at 0x1022dc48d0>,
           <Reaction EX_co2_e at 0x1022dc41d0>})
```

In [ ]:

```
cobra.io.write_sbml_model(model, 'Salmo_salar.sbml')
```

In [ ]:

```
ver3 = cobra.io.read_sbml_model('Salmo_salar.sbml')
```

In [ ]:

```
ver3.optimize()
```

Adding automatically generated transport reactions

Here transport reactions for Atlantic salmon and human is compared using KEGG annotations. Transport reactions found to be in human and Atlantic salmon are added to the model.

In [3]:

```
from transporters_from_kegg_and_recon import * #importing functions from the scr
ipt
```

In [15]:

```
sasa = cobra.io.read_sbml_model('Salmo_salar.sbml')
hsa = cobra.io.read_sbml_model('Recon3D_301.xml')
```

In [6]:

```
for m in [met for met in sasa.metabolites if met.compartment == 'c']:
    try:
        sasa.metabolites.get_by_id(m.id[:-1]+'e') #changing the last letter in t
he id to 'e'
    except KeyError: #if the last letter is not 'c'
        mb = m.copy() #copy the information about m
        mb.compartment = 'e' #set compartment to 'e'
        mb.id = mb.id[:-1] + 'e' #set the last letter in the id to 'e'
        sasa.add_metabolites([mb]) #add this metabolite to the list of metabolit
es in sasa
```

In [9]:

```
OT = orthologous_transporters(hsa, sasa, 'hsa02000.json', 'sasa02000.json')
```

```
Read LP format model from file /var/folders/j1/y7hwk0w50p78nrjhcn7p_
xwr0000gq/T/tmp7jzzkak3.lp
Reading time = 0.02 seconds
: 645 rows, 1190 columns, 4852 nonzeros
```

In [10]:

```
OT.transporter_model
```

Out[10]:

Name	Salmo_salar
Memory address	0x01028b1aa10
Number of metabolites	645
Number of reactions	1246
Number of groups	0
Objective expression	1.0*Biomass_pol - 1.0*Biomass_pol_reverse_d3f73
Compartments	mitochondria, cytosol, extracellular space, nucleus

In [11]:

```
cobra.io.write_sbml_model(OT.transporter_model, 'Salmo_salar.sbml')
```

Checking whether the number of reactions has increased

In [12]:

```
ver4 = cobra.io.read_sbml_model('Salmo_salar.sbml')
```

In [13]:

```
ver4
```

Out[13]:

Name	Salmo_salar
Memory address	0x0102124ce10
Number of metabolites	645
Number of reactions	1246
Number of groups	0
Objective expression	1.0*Biomass_pol - 1.0*Biomass_pol_reverse_d3f73
Compartments	mitochondria, cytosol, extracellular space, nucleus

In [ ]:

```
ver4.optimize()
```

Showing the essentiality of arginine

In [1]:

```
import cobra
```

In [3]:

```
sasa = cobra.io.read_sbml_model('Salmo_salar_BiGG_curated.sbml')
```

```
Using license file /Users/Ingunn/gurobi.lic
Academic license - for non-commercial use only
```

In [4]:

```
aax = cobra.core.Group(id='Amino acid exchange') # making a group for the amino
 acids
```

In [5]:

```
aax.add_members([sasa.reactions.get_by_id(r) for r in
['EX_gly_e',
'EX_ala__L_e',
 'EX_arg__L_e',
 'EX_asn__L_e',
 'EX_asp__L_e',
 'EX_cys__L_e',
 'EX_glu__L_e',
 'EX_gln__L_e',
 'EX_his__L_e',
 'EX_ile__L_e',
 'EX_leu__L_e',
 'EX_lys__L_e',
 'EX_met__L_e',
 'EX_orn__L_e',
 'EX_phe__L_e',
 'EX_pro__L_e',
 'EX_ser__L_e',
 'EX_thr__L_e',
 'EX_trp__L_e',
 'EX_tyr__L_e',
 'EX_val__L_e']]) # adding the uptake reactions for all the amino acids to the g
roup
```

In [6]:

```
for r in aax.members:
    old_bounds = r.bounds # setting the default bounds as the variable old_bound
s
    r.bounds = (0,1000) # setting uptake of r-th amino acid to 0
    print(r.id, sasa.optimize().objective_value) # printing the r-th amino acid
 along with the objective value
    #when the model is optimized
    r.bounds = old_bounds # setting the bounds back to the default value
```

```
EX_trp__L_e 0.0
EX_orn__L_e 78.39269391555355
EX_asn__L_e 77.76695390722615
EX_his__L_e 0.0
EX_tyr__L_e 78.39269391555354
EX_phe__L_e 0.0
EX_ile__L_e 0.0
EX_asp__L_e 78.39269391555354
EX_val__L_e 0.0
EX_pro__L_e 78.39269391555347
EX_leu__L_e 0.0
EX_cys__L_e 78.30607999911464
EX_ala__L_e 78.39269391555356
EX_ser__L_e 78.39269391555356
EX_lys__L_e 0.0
EX_gly_e 77.78096336434935
EX_glu__L_e 78.3926939155536
EX_thr__L_e 0.0
EX_arg__L_e 75.00467289066887
EX_met__L_e 0.0
EX_gln__L_e 78.39269391555355
```

Independent Section

Contains tests that are independent of the class of modeled organism, a model's complexity or types of identifiers that are used to describe its components. Parameterization or initialization of the network is not required. See readme for more details.

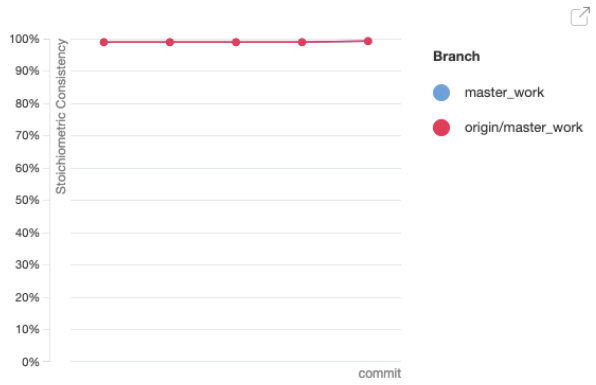
Specific Section

Covers general statistics and specific aspects of a metabolic network that are not universally applicable. See readme for more details.

Consistency

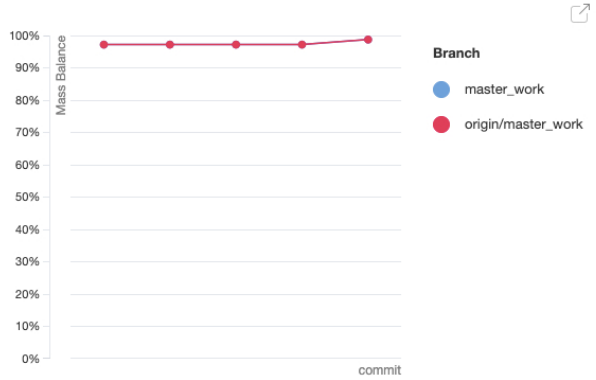
Stoichiometric Consistency

Stoichiometric inconsistency violates universal constraints: 1. Molecular masses are always positive, and 2. On each side of a reaction the mass is conserved. A single incorrectly defined reaction can lead to stoichiometric inconsistency in the model, and consequently to unconserved metabolites. Similar to insufficient constraints, this may give rise to cycles which either produce mass from nothing or consume mass from the model. Implementation: This test first uses an implementation of the algorithm presented in section 3.1 by Gevorgyan, A., M. G Poolman, and D. A Fell. "Detection of Stoichiometric Inconsistencies in Biomolecular Models." Bioinformatics 24, no. 19 (2008): 2245. doi: 10.1093/bioinformatics/btn425 Should the model be inconsistent, then the list of unconserved metabolites is computed using the algorithm described in section 3.2 of the same publication.



Mass Balance

This will exclude biomass, exchange and demand reactions as they are unbalanced by definition. It will also fail all reactions where at least one metabolite does not have a formula defined. In steady state, for each metabolite the sum of influx equals the sum of efflux. Hence the net masses of both sides of any model reaction have to be equal. Reactions where at least one metabolite does not have a formula are not considered to be balanced, even though the remaining metabolites participating in the reaction might be. Implementation: For each reaction that isn't a boundary or biomass reaction check if each metabolite has a non-zero elements attribute and if so calculate if the overall element balance of reactants and products is equal to zero.



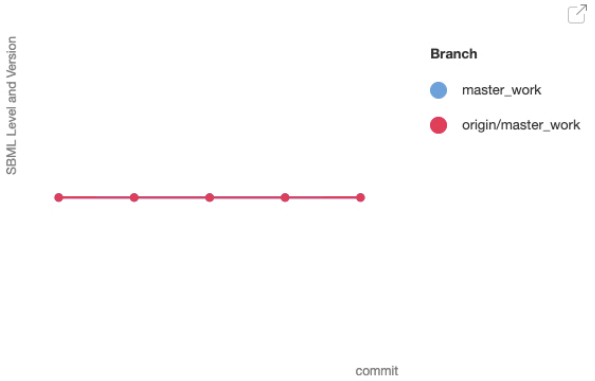
Charge Balance

This will exclude biomass, exchange and demand reactions as they are unbalanced by definition. It will also fail all reactions where at least one metabolite does not have a charge defined. In steady state, for each metabolite the sum of influx equals the sum of efflux. Hence the net charges of both sides of any model reaction have to be equal. Reactions where at least one metabolite does not have a charge are not considered to be balanced, even though the remaining metabolites participating in the reaction might be. Implementation: For each reaction that isn't a boundary or biomass reaction check if each metabolite has a non-zero charge attribute and if so calculate if the overall sum of charges of reactants and products is equal to zero.

SBML

SBML Level and Version

This test reports if the model file is represented in the latest edition (level) of the Systems Biology Markup Language (SBML) which is Level 3, and at least version 1. Implementation: The level and version are parsed directly from the SBML document.



FBC enabled

The Flux Balance Constraints (FBC) Package extends SBML with structured and semantic descriptions for domain-specific model components such as flux bounds, multiple linear objective functions, gene-protein-reaction associations, metabolite chemical formulas, charge and related annotations which are relevant for parameterized GEMs and FBA models. The SBML and constraint-based modeling communities collaboratively develop this package and update it based on user input. Implementation: Parse the state of the FBC plugin from the SBML document.

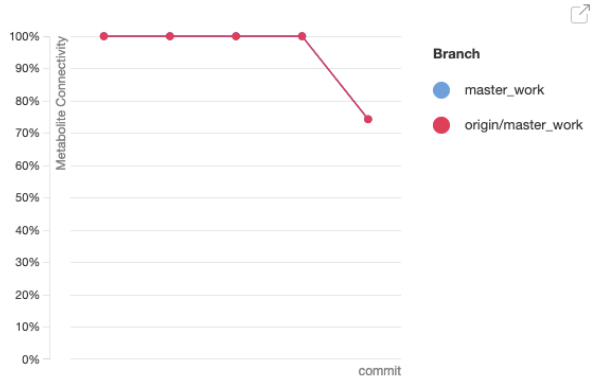
Basic Information

Model Identifier

The MIRIAM guidelines require a model to be identified via an ID. Further, the ID will be displayed on the memote snapshot report, which helps to distinguish the output clearly. Implementation: Check if the cobra.Model object has a non-empty "id" attribute, this value is parsed from the "id" attribute of the <model> tag in the SBML file e.g. <model fbc:strict="true" id="iJO1366">.

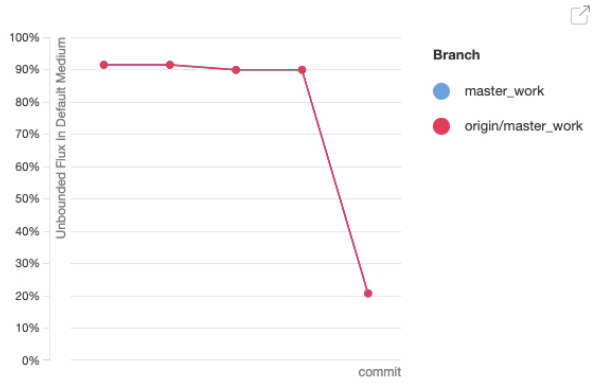
Metabolite Connectivity

Disconnected metabolites are not part of any reaction in the model. They are most likely left-over from the reconstruction process, but may also point to network and knowledge gaps. Implementation: Check for any metabolites of the cobra.Model object with emtpy reaction attribute.



Unbounded Flux In Default Medium

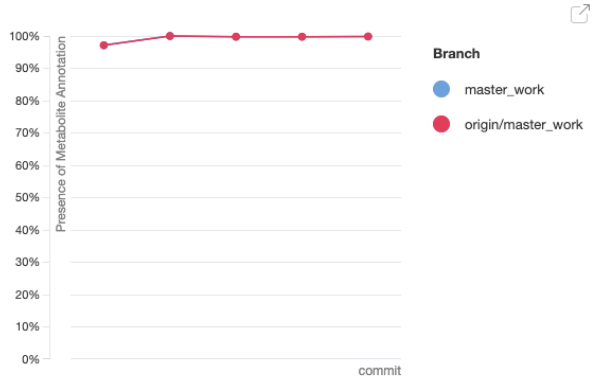
A large fraction of model reactions able to carry unlimited flux under default conditions indicates problems with reaction directionality, missing cofactors, incorrectly defined transport reactions and more. Implementation: Without changing the default constraints run flux variability analysis. From the FVA results identify those reactions that carry flux equal to the model's maximal or minimal flux.



Annotation - Metabolites

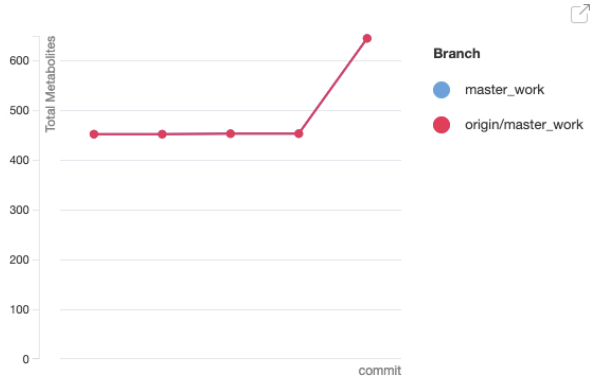
Presence of Metabolite Annotation

This test checks if any annotations at all are present in the SBML annotations field for each metabolite, irrespective of the type of annotation i.e. specific database cross-references, ontology terms, additional information. For this test to pass the model is expected to have metabolites and each of them should have some form of annotation. Implementation: Check if the annotation attribute of each cobra.Metabolite object of the model is unset or empty.



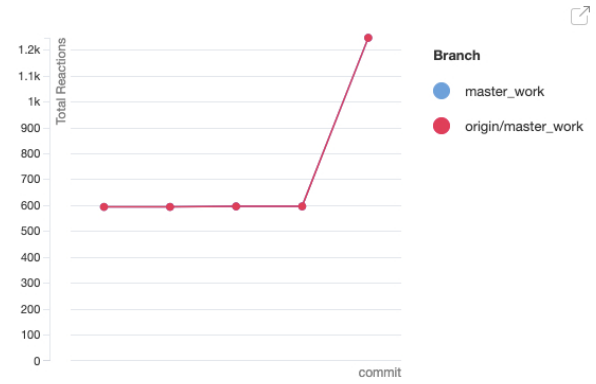
Total Metabolites

To be useful a metabolic model should consist at least of a few metabolites that are converted by reactions. This test simply checks if there are more than zero metabolites. Implementation: Check if the cobra.Model object has non-empty "metabolites" attribute, this list is populated from the list of sbml:listOfSpecies which should contain at least one sbml:species.



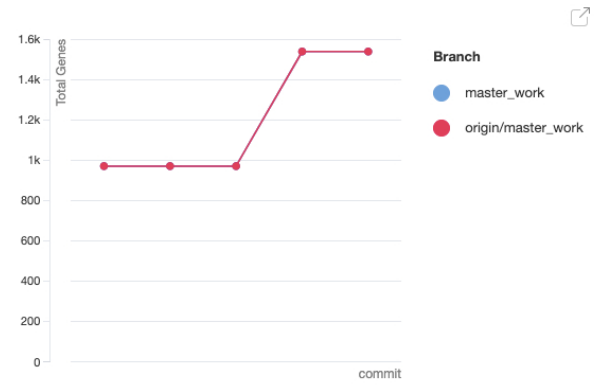
Total Reactions

To be useful a metabolic model should consist at least of a few reactions. This test simply checks if there are more than zero reactions. Implementation: Check if the cobra.Model object has non-empty "reactions" attribute, this list is populated from the list of sbml:listOfReactions which should contain at least one sbml:reaction.



Total Genes

A metabolic model can still be a useful tool without any genes, however there are certain methods which rely on the presence of genes and, more importantly, the corresponding gene-protein-reaction rules. This test requires that there is at least one gene defined. Implementation: Check if the cobra.Model object has non-empty "genes" attribute, this list is populated from the list of fbc:listOfGeneProducts which should contain at least one fbc:geneProduct.



Total Compartments

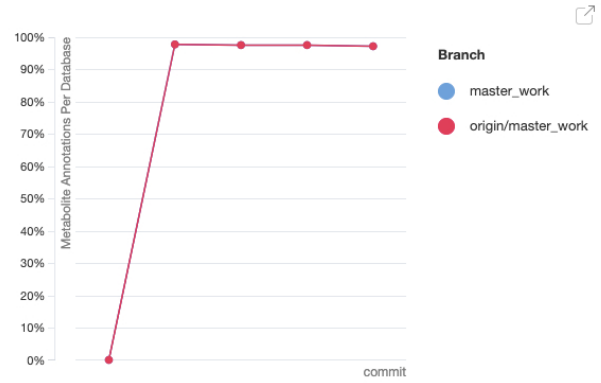
While simplified metabolic models may be perfectly viable, generally across the tree of life organisms contain at least one distinct compartment: the cytosol or cytoplasm. In the case of prokaryotes there is usually a periplasm, and eurkaryotes are more complex. In addition to the internal compartment, a metabolic model also reflects the extracellular environment i.e. the medium/ metabolic context in which the modelled cells grow. Hence, in total, at least two compartments can be expected from a metabolic model. Implementation: Check if the cobra.Model object has a non-empty "compartments" attribute, this list is populated from the list of sbml:listOfCompartments which should contain at least two sbml:compartment elements.

Metabolite Annotations Per Database

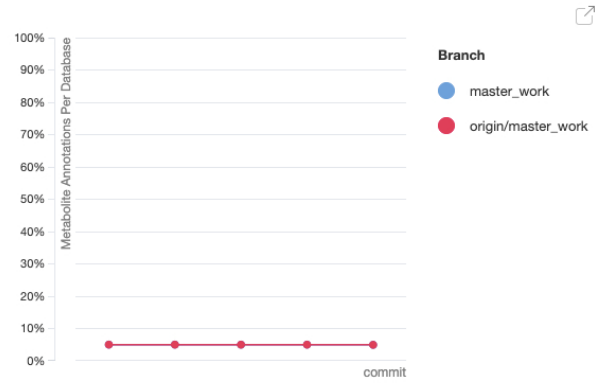
Info

Specific database cross-references are paramount to mapping information. To provide references to as many databases as possible helps to make the metabolic model more accessible to other researchers. This does not only facilitate the use of a model in a broad array of computational pipelines, it also promotes the metabolic model itself to become an organism-specific knowledge base. For this test to pass, each metabolite annotation should contain cross-references to a number of databases. The currently selection is listed in `annotation.py`, but an ongoing discussion can be found at https://github.com/opencobra/memote/issues/332. For each database this test checks for the presence of its corresponding namespace ID to comply with the MIRIAM guidelines i.e. they have to match those defined on https://identifiers.org/. Since each database is quite different and some potentially incomplete, it may not be feasible to achieve 100% coverage for each of them. Generally it should be possible, however, to obtain cross-references to at least one of the databases for all metabolites consistently. Implementation: Check if the keys of the annotation attribute of each cobra.Metabolite of the model match with a selection of common biochemical databases. The annotation attribute of cobrapy components is a dictionary of key:value pairs.

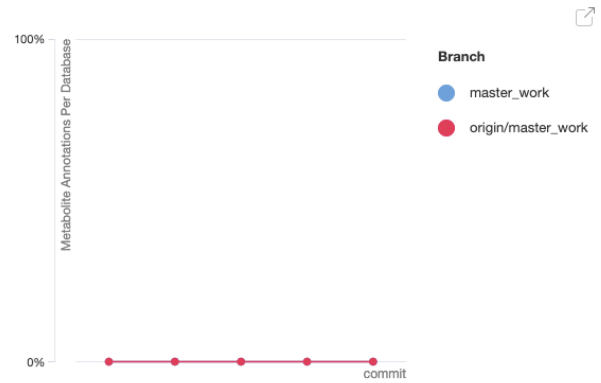
bigg.metabolite



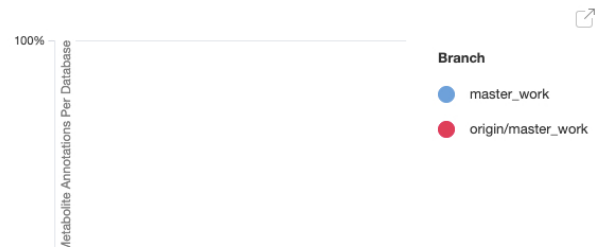
biocyc

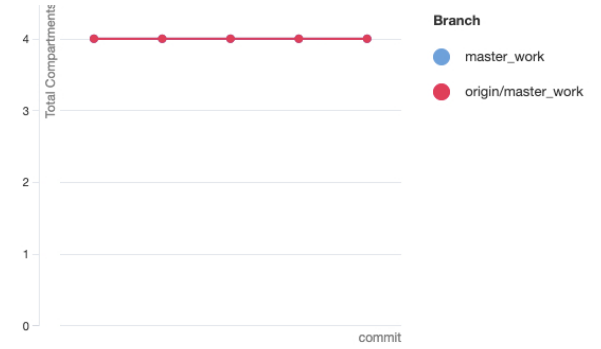


chebi



hmdb





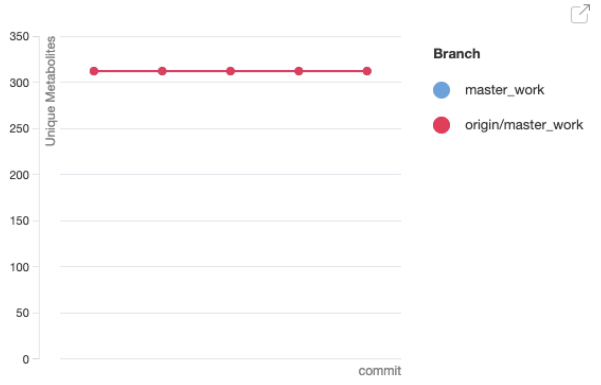
Metabolic Coverage

The degree of metabolic coverage indicates the modeling detail of a given reconstruction calculated by dividing the total amount of reactions by the amount of genes. Models with a 'high level of modeling detail have ratios >1, and models with a low level of detail have ratios <1. This difference arises as models with basic or intermediate levels of detail are assumed to include many reactions in which several gene products and their enzymatic transformations are 'lumped'. Implementation: Divides the amount reactions by the amount of genes. Raises an error if the model does not contain either reactions or genes.

Metabolite Information

Unique Metabolites

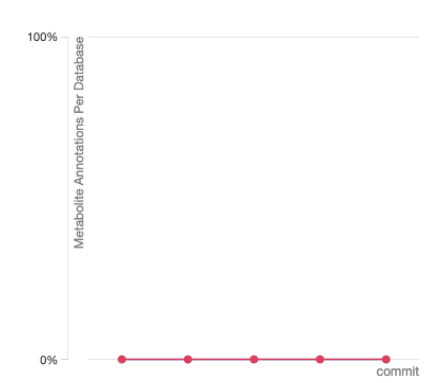
Metabolites may be transported into different compartments, which means that in a compartimentalized model the number of metabolites may be much higher than in a model with no compartments. This test counts only one occurrence of each metabolite and returns this as the number of unique metabolites. The test expects that the model is compartimentalized, and thus, that the number of unique metabolites is generally lower than the total number of metabolites. Implementation: Reduce the list of metabolites to a unique set by removing the compartment tag. The cobrapy SBML parser adds compartment tags to each metabolite ID.

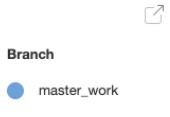


Duplicate Metabolites in Identical Compartments

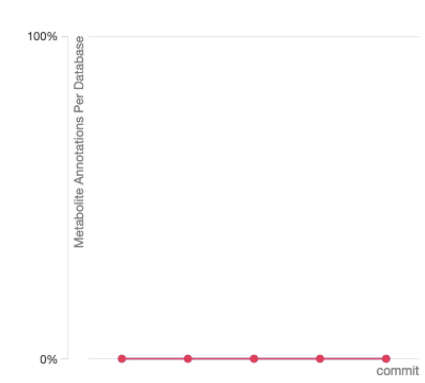
The main reason for having this test is to help cleaning up merged models or models from automated reconstruction pipelines as these are prone to having identical metabolites from different namespaces (hence different IDs). This test therefore expects that every metabolite in any particular compartment has unique inchikey values. Implementation: Identifies duplicate metabolites in each compartment by determining if any two metabolites have identical InChI-key annotations. For instance, this function would find compounds with IDs ATP1 and ATP2 in the cytosolic compartment, with both having the same InChI annotations.

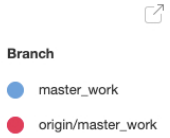
inchi



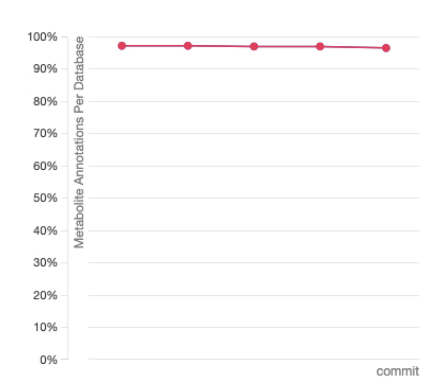


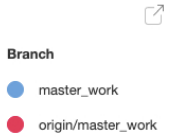
inchikey



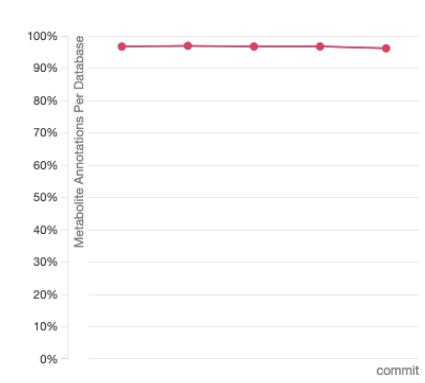


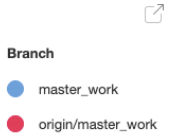
kegg.compound





metanetx.chemical

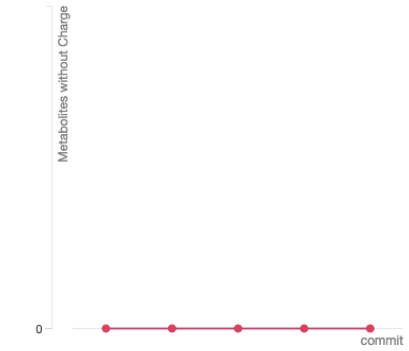


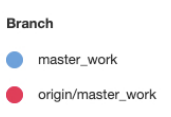


pubchem.compound

Metabolites without Charge

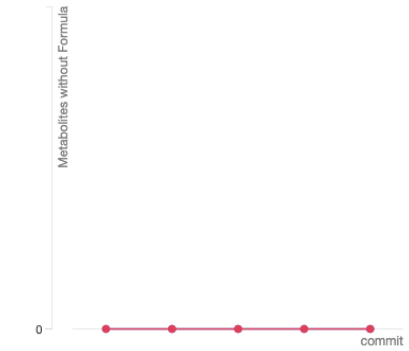
To be able to ensure that reactions are charge-balanced, all model metabolites ought to be provided with a charge. Since it may be difficult to obtain charges for certain metabolites this test serves as a mere report. Models can still be stoichiometrically consistent even when charge information is not defined for each metabolite. Implementation: Check if each cobra.Metabolite has a non-empty "charge" attribute. This attribute is set by the parser if there is an fbc:charge attribute for the corresponding species in the SBML.

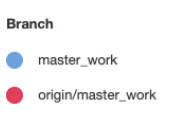




Metabolites without Formula

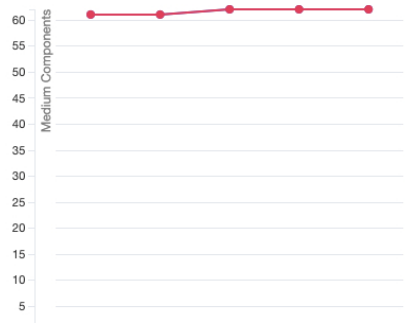
To be able to ensure that reactions are mass-balanced, all model metabolites ought to be provided with a chemical formula. Since it may be difficult to obtain formulas for certain metabolites this test serves as a mere report. Models can still be stoichiometrically consistent even when chemical formulas are not defined for each metabolite. Implementation: Check if each cobra.Metabolite has a non-empty "formula" attribute. This attribute is set by the parser if there is an fbc:chemicalFormula attribute for the corresponding species in the SBML.



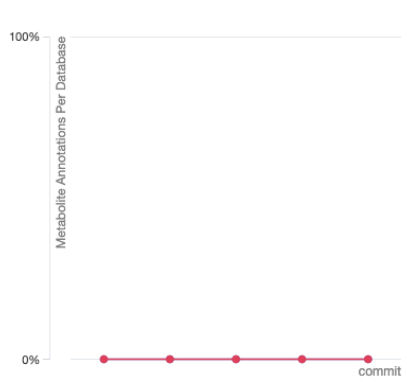


Medium Components

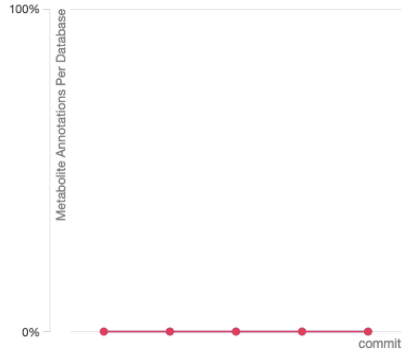
This test checks all boundary reactions in the model that permit flux towards creating a metabolite, and reports those metabolites. This test does not have any mandatory 'pass' criteria. Implementation: Identify the metabolite IDs of each reaction in the method cobra.Model.medium. Model.medium returns exchange reactions whose bounds permit the uptake of metabolites.



reactome



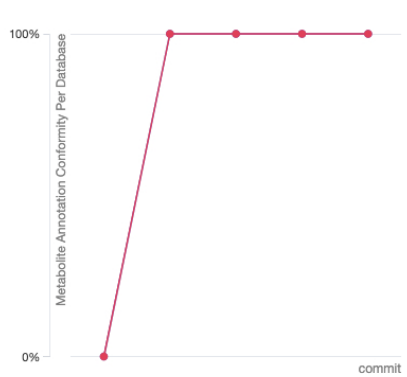
seed.compound



Metabolite Annotation Conformity Per Database Info

To identify databases and the identifiers belonging to them, computational tools rely on the presence of specific patterns. Only when these patterns can be identified consistently is an ID truly machine-readable. This test checks if the database cross-references in metabolite annotations conform to patterns defined according to the MIRIAM guidelines, i.e. matching those that are defined at https://identifiers.org/. The required formats, i.e., regex patterns are further outlined in `annotation.py`. This test does not carry out a web query for the composed URI, it merely controls that the regex patterns match the identifiers. Implementation: For those metabolites whose annotation keys match any of the tested databases, check if the corresponding values match the identifier pattern of each database.

bigg.metabolite

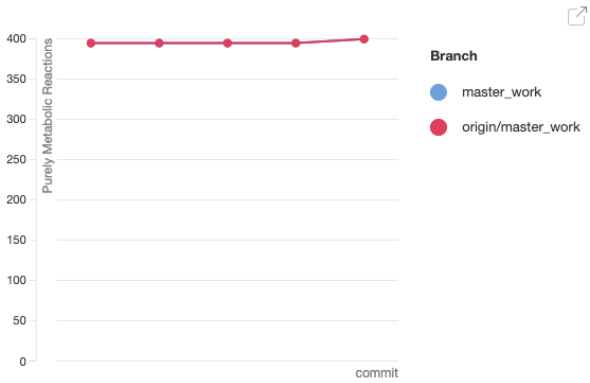


biocyc

Reaction Information

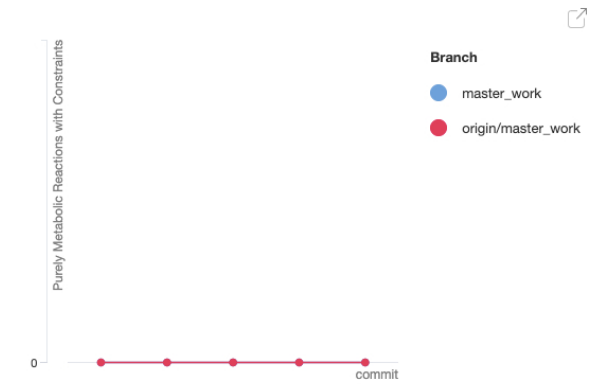
Purely Metabolic Reactions

If a reaction is neither a transport reaction, a biomass reaction nor a boundary reaction, it is counted as a purely metabolic reaction. This test requires the presence of metabolite formula to be able to identify transport reactions. This test is passed when the model contains at least one purely metabolic reaction i.e. a conversion of one metabolite into another. Implementation: From the list of all reactions, those that are boundary, transport and biomass reactions are removed and the remainder assumed to be pure metabolic reactions. Boundary reactions are identified using the attribute cobra.Model.boundary. Please read the description of "Transport Reactions" and "Biomass Reaction Identified" to learn how they are identified.



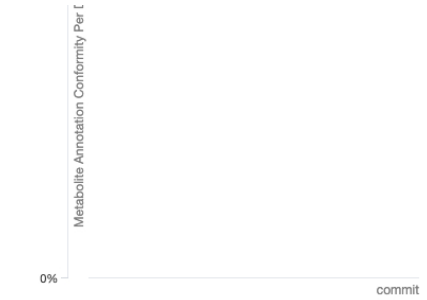
Purely Metabolic Reactions with Constraints

If a reaction is neither a transport reaction, a biomass reaction nor a boundary reaction, it is counted as a purely metabolic reaction. This test requires the presence of metabolite formula to be able to identify transport reactions. This test simply reports the number of purely metabolic reactions that have fixed constraints and does not have any mandatory 'pass' criteria. Implementation: From the pool of pure metabolic reactions identify reactions which are constrained to values other than the model's minimal or maximal possible bounds.

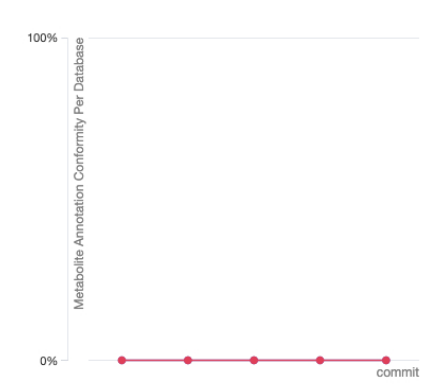


Transport Reactions

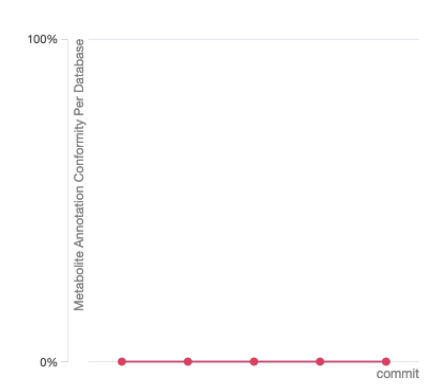
Cellular metabolism in any organism usually involves the transport of metabolites across a lipid bi-layer. This test reports how many of these reactions, which transports metabolites from one compartment to another, are present in the model, as at least one transport reaction must be present for cells to take up nutrients and/or excrete waste. Implementation: A transport reaction is defined as follows: 1. It contains metabolites from at least 2 compartments and 2. at least 1 metabolite undergoes no chemical reaction, i.e., the formula and/or annotation stays the same on both sides of the equation. A notable exception is transport via PTS, which also contains the following restriction: 3. The transported metabolite(s) are transported into a compartment through the exchange of a phosphate. An example of transport via PTS would be pep(c) + glucose(e) -> glucose-6-phosphate(c) + pyr(c) Reactions similar to transport via PTS (referred to as "modified transport reactions") follow a similar pattern: A(x) + B-R(y) -> A-R(y) + B(y) Such modified transport reactions can be detected, but only when the formula is defined for all metabolites in a particular reaction. If this is not the case, transport reactions are identified through annotations, which cannot detect modified transport reactions.



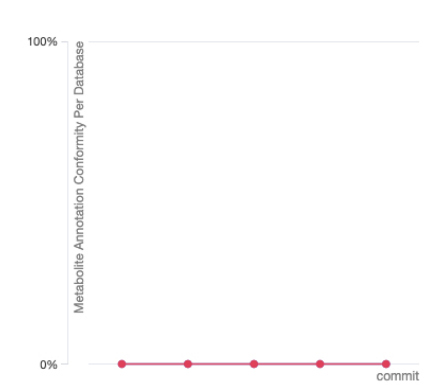
chebi



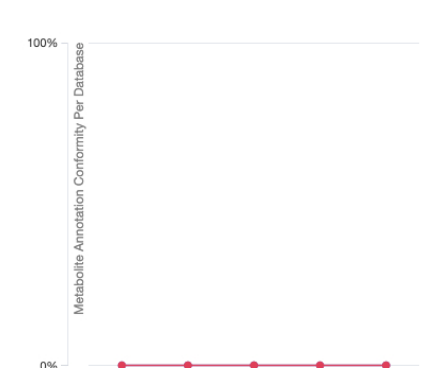
hmdb

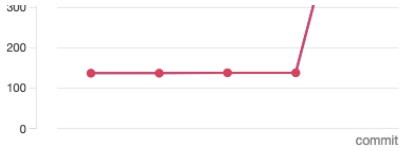


inchi



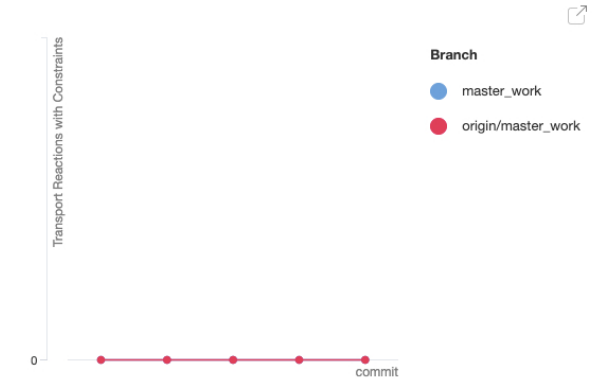
inchikey





Transport Reactions with Constraints

Cellular metabolism in any organism usually involves the transport of metabolites across a lipid bi-layer. Hence, this test reports how many of these reactions, which transports metabolites from one compartment to another, have fixed constraints. This test does not have any mandatory 'pass' criteria. Implementation: Please refer to "Transport Reactions" for details on how memote identifies transport reactions. From the pool of transport reactions identify reactions which are constrained to values other than the model's median lower and upper bounds.



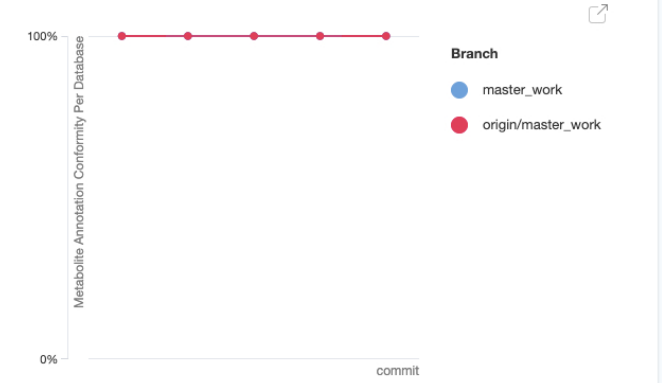
Thermodynamic Reversibility of Purely Metabolic Reactions

If a reaction is neither a transport reaction, a biomass reaction nor a boundary reaction, it is counted as a purely metabolic reaction. This test checks if the reversibility attribute of each reaction agrees with a thermodynamics-based calculation of reversibility. Implementation: To determine reversibility we calculate the reversibility index ln_gamma (natural logarithm of gamma) of each reaction using the eQuilibrator API. We consider reactions, whose reactants' concentrations would need to change by more than three orders of magnitude for the reaction flux to reverse direction, to be likely candidates of irreversible reactions. This assume default concentrations around 100 µM (~3 µM—3 mM) at pH = 7, I = 0.1 M and T = 298 K. The corresponding reversibility index is approximately 7. For further information on the thermodynamic and implementation details please refer to https://doi.org/10.1093/bioinformatics/bts317 and https://pypi.org/project/equilibrator-api/. Please note that currently eQuilibrator can only determine the reversibility index for chemically and redox balanced reactions whose metabolites can be mapped to KEGG compound identifiers (e.g. C00001). In addition to not being mappable to KEGG or the reaction not being balanced, there is a possibility that the metabolite cannot be broken down into chemical groups which is essential for the calculation of Gibbs energy using group contributions. This test collects each erroneous reaction and returns them as a tuple containing each list in the following order: 1. Reactions with reversibility index 2. Reactions with incomplete mapping to KEGG 3. Reactions with metabolites that are problematic during calculation 4. Chemically or redox unbalanced Reactions (after mapping to KEGG) This test simply reports the number of reversible reactions that, according to the reversibility index, are likely to be irreversible.

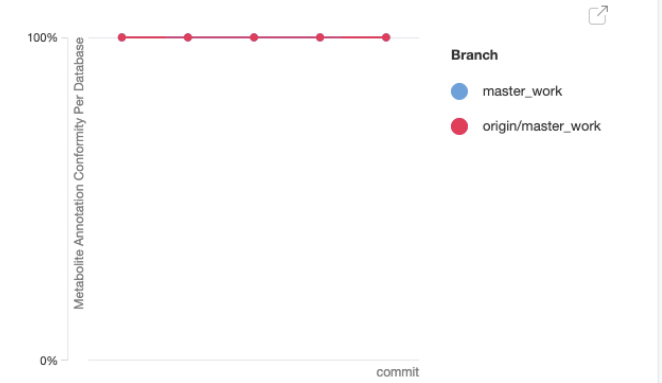
Reactions With Partially Identical Annotations

Identify reactions in a pairwise manner that are annotated with identical database references. This does not take into account a reaction's directionality or compartment. The main reason for having this test is to help cleaning up merged models or models from automated reconstruction pipelines as these are prone to having identical reactions with identifiers from different namespaces. It could also be useful to identify a 'type' of reaction that occurs in several compartments. Implementation: Identify duplicate reactions globally by checking if any two metabolic reactions have the same entries in their annotation attributes. The heuristic looks at annotations with the keys "metanetx.reaction", "kegg.reaction", "brenda", "rhea", "biocyc", "bigg.reaction" only.

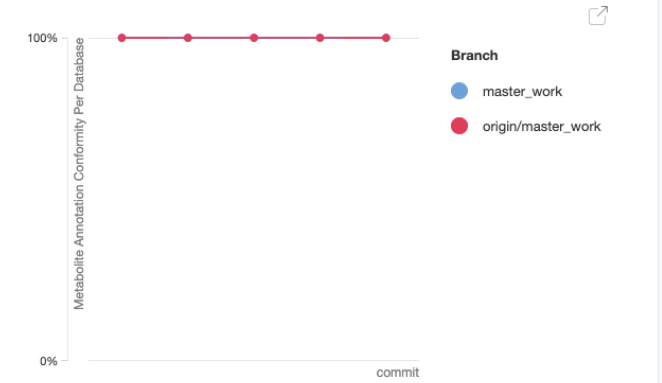
kegg.compound



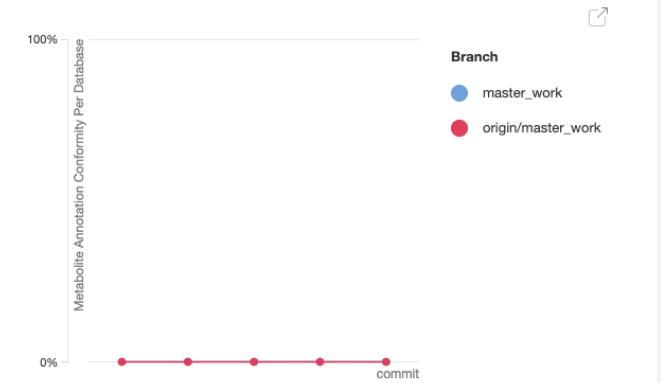
metanetx.chemical



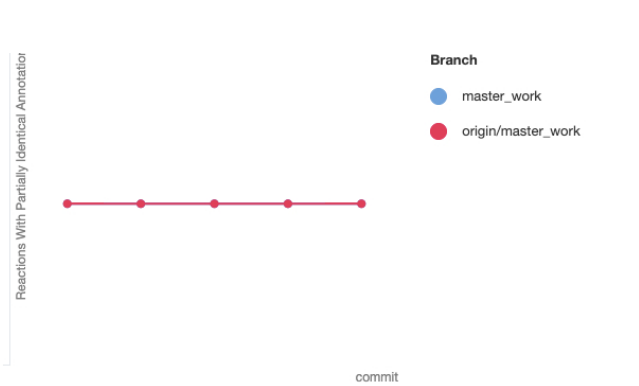
pubchem.compound



reactome



seed.compound



Duplicate Reactions

Identify reactions in a pairwise manner that use the same set of metabolites including potentially duplicate metabolites. Moreover, it will take a reaction's directionality and compartment into account. The main reason for having this test is to help cleaning up merged models or models from automated reconstruction pipelines as these are prone to having identical reactions with identifiers from different namespaces. Implementation: Compare reactions in a pairwise manner. For each reaction, the metabolite annotations are checked for a description of the structure (via InChI and InChIKey).If they exist, substrates and products as well as the stoichiometries of any reaction pair are compared. Only reactions where the substrates, products, stoichiometry and reversibility are identical are considered to be duplicates. This test will not be able to identify duplicate reactions if there are no structure annotations. Further, it will report reactions with differing bounds as equal if they otherwise match the above conditions.

Reactions With Identical Genes

Identify reactions in a pairwise manner that use identical sets of genes. It does *not* take into account a reaction's directionality, compartment, metabolites or annotations. The main reason for having this test is to help cleaning up merged models or models from automated reconstruction pipelines as these are prone to having identical reactions with identifiers from different namespaces. Implementation: Compare reactions in a pairwise manner and group reactions whose genes are identical. Skip reactions with missing genes.

Gene-Protein-Reaction (GPR) Associations

Reactions without GPR

Gene-Protein-Reaction rules express which gene has what function. The presence of this annotation is important to justify the existence of reactions in the model, and is required to conduct in silico gene deletion studies. However, reactions without GPR may also be valid: Spontaneous reactions, or known reactions with yet undiscovered genes likely lack GPR. Implementation: Check if each cobra.Reaction has a non-empty "gene_reaction_rule" attribute, which is set by the parser if there is an fbc:geneProductAssociation defined for the corresponding reaction in the SBML.

Uniform Metabolite Identifier Namespace

In well-annotated models it is no problem if the pool of main identifiers for metabolites consists of identifiers from several databases. However, in models that lack appropriate annotations, it may hamper the ability of other researchers to use it. Running the model through a computational pipeline may be difficult without first consolidating the namespace. Hence, this test checks if the main metabolite identifiers can be attributed to one single namespace based on the regex patterns defined at https://identifiers.org/ Implementation: Generate a table with each column corresponding to one database from the selection and each row to a metabolite identifier. A Boolean entry indicates whether the identifier matches the regular expression of the corresponding database. Since the Biocyc pattern matches broadly, we assume that any instance of an identifier matching to Biocyc AND any other database pattern is a false positive match for Biocyc and thus set it to `false`. Sum the positive matches for each database and assume that the largest set is the 'main' identifier namespace.

Annotation - Reactions

Presence of Reaction Annotation

This test checks if any annotations at all are present in the SBML annotations field for each reaction, irrespective of the type of annotation i.e. specific database cross-references, ontology terms, additional information. For this test to pass the model is expected to have reactions and each of them should have some form of annotation. Implementation: Check if the annotation attribute of each cobra.Reaction object of the model is unset or empty.

Reaction Annotations Per Database

Info

Specific database cross-references are paramount to mapping information. To provide references to as many databases as possible helps to make the metabolic model more accessible to other researchers. This does not only facilitate the use of a model in a broad array of computational pipelines, it also promotes the metabolic model itself to become an organism-specific knowledge base. For this test to pass, each reaction annotation should contain cross-references to a number of databases. The currently selection is listed in `annotation.py`, but an ongoing discussion can be found at https://github.com/opencobra/memote/issues/332. For each database this test checks for the presence of its corresponding namespace ID to comply with the MIRIAM guidelines i.e. they have to match those defined on https://identifiers.org/. Since each database is quite different and some potentially incomplete, it may not be feasible to achieve 100% coverage for each of them. Generally it should be possible, however, to obtain cross-references to at least one of the databases for all reactions consistently. Implementation: Check if the keys of the annotation attribute of each cobra.Reaction of the model match with a selection of common biochemical databases. The annotation attribute of cobrapy components is a dictionary of key:value pairs.

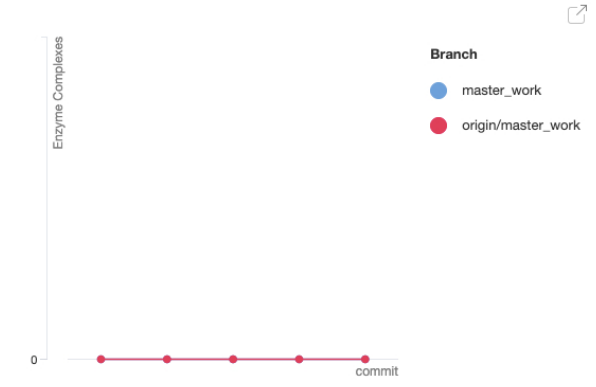
bigg.reaction

Fraction of Transport Reactions without GPR

As it is hard to identify the exact transport processes within a cell, transport reactions are often added purely for modeling purposes. Highlighting where assumptions have been made versus where there is proof may help direct the efforts to improve transport and transport energetics of the tested metabolic model. However, transport reactions without GPR may also be valid: Diffusion, or known reactions with yet undiscovered genes likely lack GPR. Implementation: Check which cobra.Reactions classified as transport reactions have a non-empty "gene_reaction_rule" attribute.

Enzyme Complexes

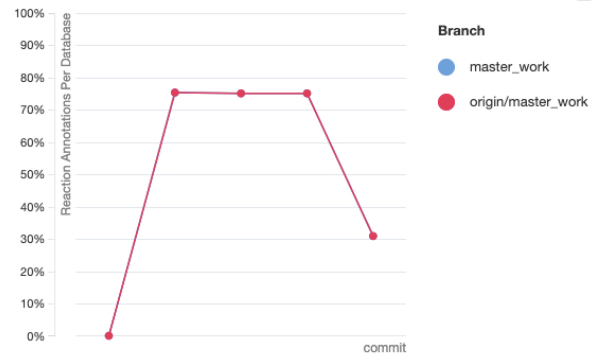
Based on the gene-protein-reaction (GPR) rules, it is possible to infer whether a reaction is catalyzed by a single gene product, isozymes or by a heteromeric protein complex. This test checks that at least one such heteromeric protein complex is defined in any GPR of the model. For S. cerevisiae it could be shown that "essential proteins tend to [cluster] together in essential complexes" (https://doi.org/10.1074%2Fmcp.M800490-MCP200). This might also be a relevant metric for other organisms. Implementation: Identify GPRs which contain at least one logical AND that combines two different gene products.



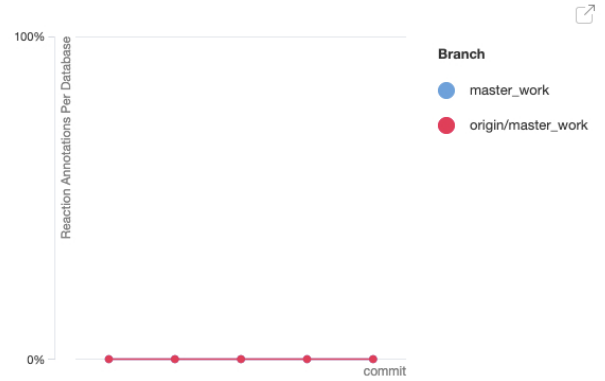
Biomass

Biomass Reactions Identified

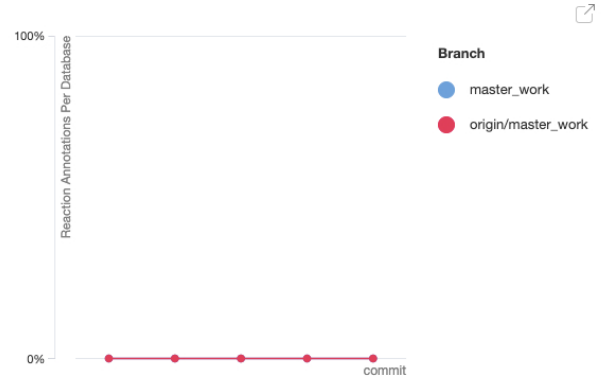
The biomass composition aka biomass formulation aka biomass reaction is a common pseudo-reaction accounting for biomass synthesis in constraints-based modelling. It describes the stoichiometry of intracellular compounds that are required for cell growth. While this reaction may not be relevant to modeling the metabolism of higher organisms, it is essential for single-cell modeling. Implementation: Identifies possible biomass reactions using two principal steps: 1. Return reactions that include the SBO annotation "SBO:0000629" for biomass. If no reactions can be identifies this way: 1. Look for the `buzzwords` "biomass", "growth" and "bof" in reaction IDs. 2. Look for metabolite IDs or names that contain the `buzzword` "biomass" and obtain the set of reactions they are involved in. 3. Remove boundary reactions from this set. 4. Return the union of reactions that match the buzzwords and of the reactions that metabolites are involved in that match the buzzword. This test checks if at least one biomass reaction is present.



biocyc



brenda



ec-code

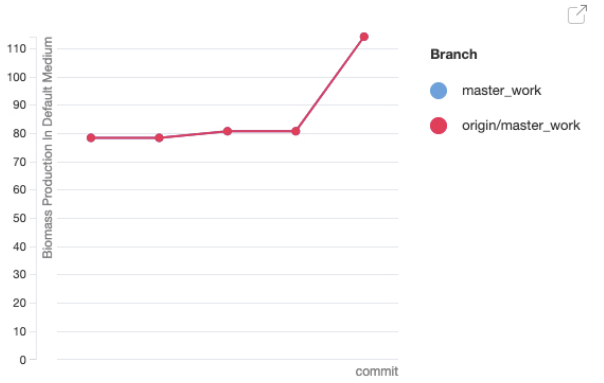
kegg.reaction

Biomass Consistency

This test only yields sensible results if all biomass precursor metabolites have chemical formulas assigned to them. The molecular weight of the biomass reaction in metabolic models is defined to be equal to 1 g/mmol. Conforming to this is essential in order to be able to reliably calculate growth yields, to cross-compare models, and to obtain valid predictions when simulating microbial consortia. A deviation from 1 - 1E-03 to 1 + 1E-06 is accepted. Implementation: Multiplies the coefficient of each metabolite of the biomass reaction with its molecular weight calculated from the formula, then divides the overall sum of all the products by 1000.

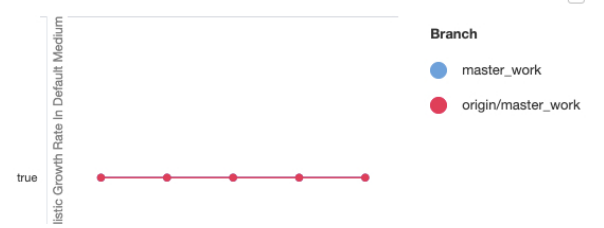
Biomass Production In Default Medium

Using flux balance analysis this test optimizes the model for growth in the medium that is set by default. Any non-zero growth rate is accepted to pass this test. Implementation: Calculate the solution of FBA with the biomass reaction set as objective function and the model's default constraints.

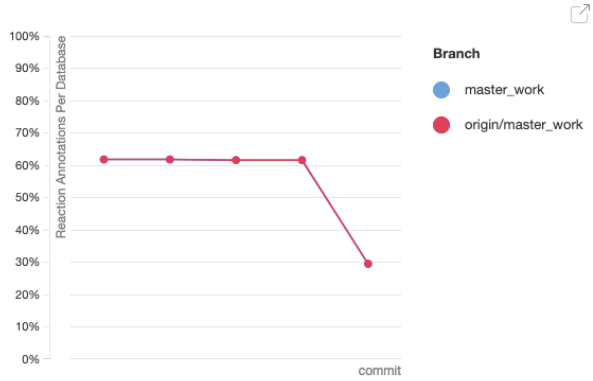


Unrealistic Growth Rate In Default Medium

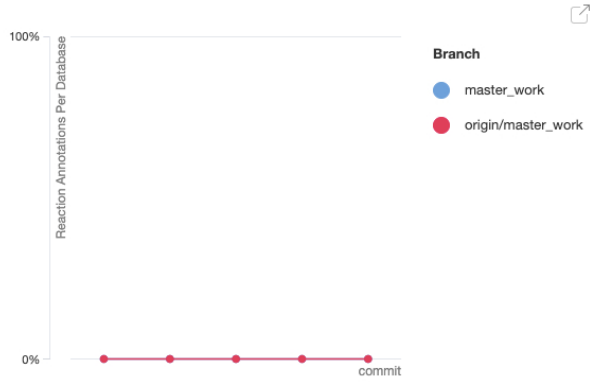
The growth rate of a metabolic model should not be faster than that of the fastest growing organism. This is based on a doubling time of Vibrio natriegens which was reported to be 14.8 minutes by: Henry H. Lee, Nili Ostrov, Brandon G. Wong, Michaela A. Gold, Ahmad S. Khalil, George M. Church in https://www.biorxiv.org/content/biorxiv/early/2016/06/12/058487.full.pdf The calculation ln(2)/(14.8/60) ~ 2.81 yields the corresponding growth rate. Implementation: Calculate the solution of FBA with the biomass reaction set as objective function and a model's default constraints. Then check if the objective value is higher than 2.81.



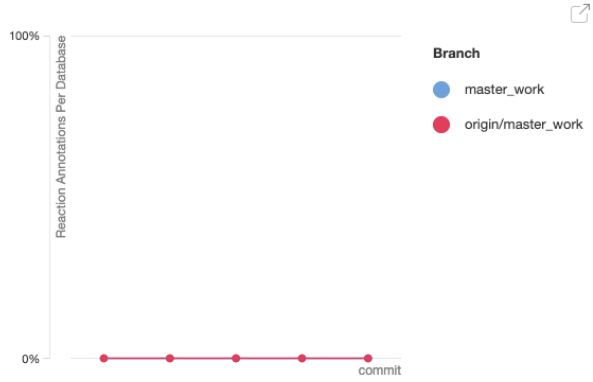
metanetx.reaction



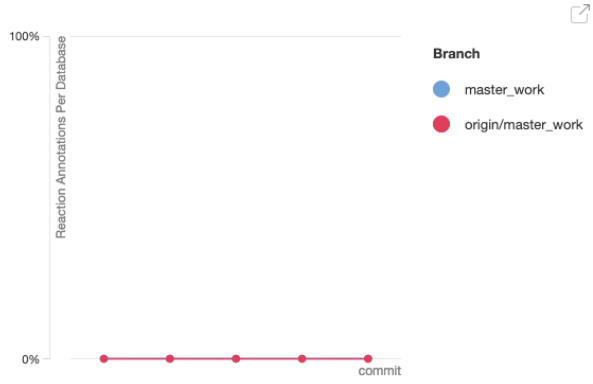
reactome



rhea



seed.reaction

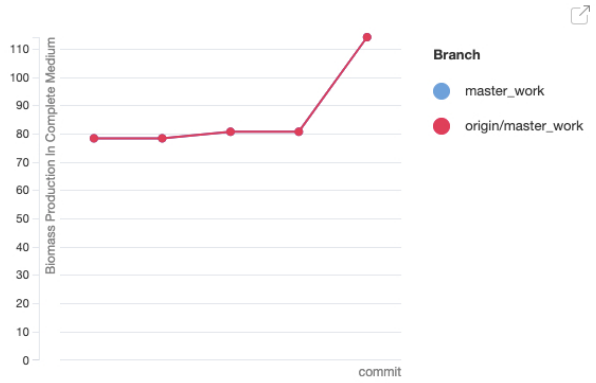


Reaction Annotation Conformity Per Database Info

To identify databases and the identifiers belonging to them, computational tools rely on the presence of specific patterns. Only when these patterns can be identified consistently is an ID truly machine-readable. This test checks if the database cross-references in reaction annotations conform to patterns defined according to the MIRIAM guidelines, i.e. matching

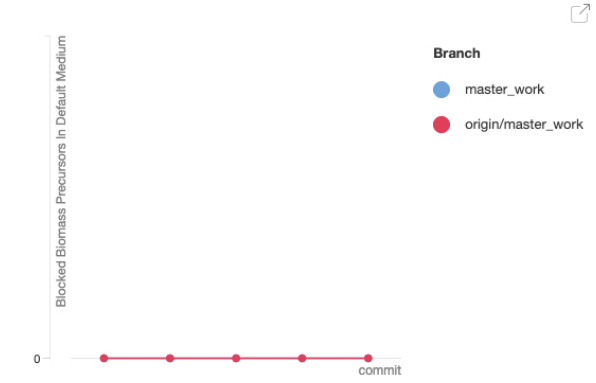
Biomass Production In Complete Medium

Using flux balance analysis this test optimizes the model for growth using a complete medium i.e. unconstrained boundary reactions. Any non-zero growth rate is accepted to pass this test. Implementation: Calculate the solution of FBA with the biomass reaction set as objective function and after removing any constraints from all boundary reactions.



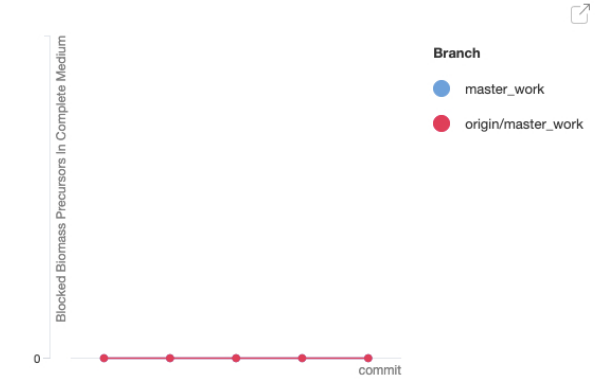
Blocked Biomass Precursors In Default Medium

Using flux balance analysis this test optimizes for the production of each metabolite that is a substrate of the biomass reaction with the exception of atp and h2o. Optimizations are carried out using the default conditions. This is useful when reconstructing the precursor biosynthesis pathways of a metabolic model. To pass this test, the model should be able to synthesis all the precursors. Implementation: For each biomass precursor (except ATP and H2O) add a temporary demand reaction, then carry out FBA with this reaction as the objective. Collect all metabolites for which this optimization is equal to zero or infeasible.



Blocked Biomass Precursors In Complete Medium

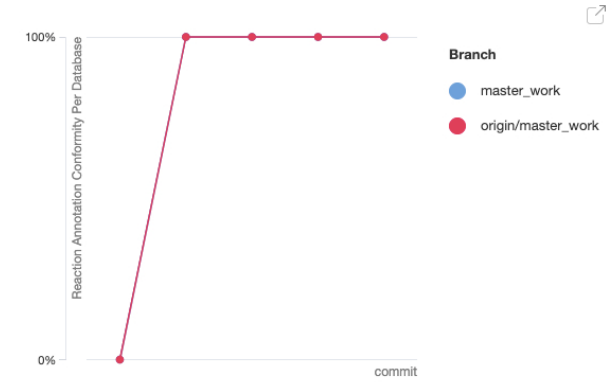
Using flux balance analysis this test optimizes for the production of each metabolite that is a substrate of the biomass reaction with the exception of atp and h2o. Optimizations are carried out using a complete medium i.e. unconstrained boundary reactions. This is useful when reconstructing the precursor biosynthesis pathways of a metabolic model. To pass this test, the model should be able to synthesis all the precursors. Implementation: First remove any constraints from all boundary reactions, then for each biomass precursor (except ATP and H2O) add a temporary demand reaction, then carry out FBA with this reaction as the objective. Collect all metabolites for which this optimization is below or equal to zero or is infeasible.



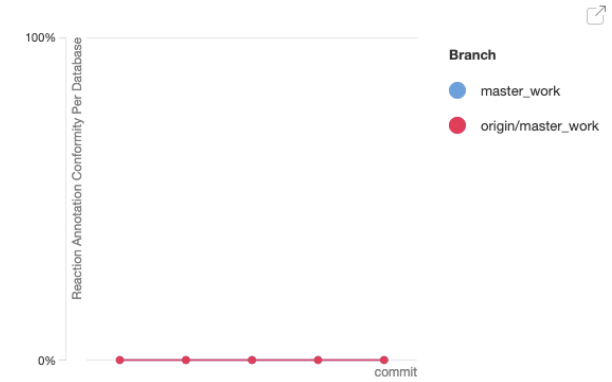
Ratio of Direct Metabolites in Biomass Reaction

those that are defined at https://identifiers.org/. The required formats, i.e., regex patterns are further outlined in `annotation.py`. This test does not carry out a web query for the composed URI, it merely controls that the regex patterns match the identifiers. Implementation: For those reaction whose annotation keys match any of the tested databases, check if the corresponding values match the identifier pattern of each database.

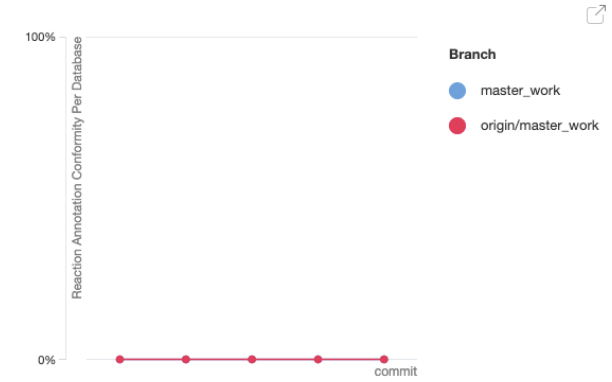
bigg.reaction



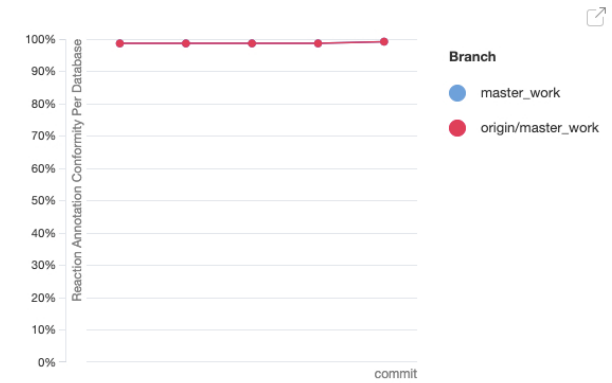
biocyc



brenda



ec-code

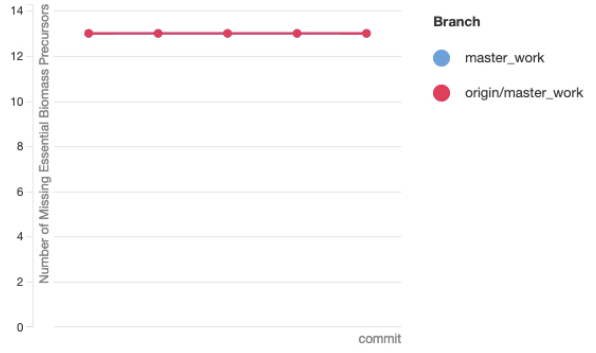


kegg.reaction

Some biomass precursors are taken from the media and directly consumed by the biomass reaction. It might not be a problem for ions or metabolites for which the organism in question is auxotrophic. However, too many of these metabolites may be artifacts of automated gap-filling procedures. Many gap-filling algorithms attempt to minimise the number of added reactions. This can lead to many biomass precursors being "direct metabolites". This test reports the ratio of direct metabolites to the total amount of precursors to a given biomass reaction. It specifically looks for metabolites that are only in either exchange, transport or biomass reactions. Bear in mind that this may lead to false positives in heavily compartimentalized models. To pass this test, the ratio of direct metabolites should be less than 50% of all biomass precursors. This is an arbitrary threshold but it takes into account that while certain ions do not serve a relevant metabolic function, it may still be important to include them in the biomass reaction to account for the impact of their uptake energy costs. This threshold is subject to change in the future. Implementation: Identify biomass precursors (excluding ATP and H+), identify cytosol and extracellular compartment from an internal mapping table. Then, determine which precursors is only involved in transport, boundary and biomass reactions. Using FBA with the biomass function as the objective then determine whether the metabolite is taken up only to be consumed by the biomass reaction.

Number of Missing Essential Biomass Precursors

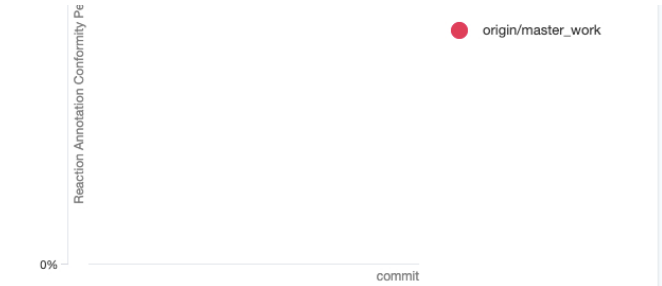
There are universal components of life that make up the biomass of all known organisms. These include all proteinogenic amino acids, deoxy- and ribonucleotides, water and a range of metabolic cofactors. This test reports the amount of biomass precursors that have been reported to be essential constituents of the biomass equation. All of the following precursors need to be included in the biomass reaction to pass the test: Aminoacids: trp__L, cys__L, his__L, tyr__L, met__L, phe__L, ser__L, pro__L, asp__L, thr__L, gln__L, glu__L, ile__L, arg__L, lys__L, val__L, leu__L, ala__L, gly, asn__L DNA: datp, dctp, dttp, dgtp RNA: atp, ctp, utp, gtp Cofactors: nad, nadp, amet, fad, pydx5p, coa, thmpp, fmn and h2o These metabolites were selected based on the results presented by DOI:10.1016/j.ymben.2016.12.002 Please note, that the authors also suggest to count C1 carriers (derivatives of tetrahydrofolate(B9) or tetrahydromethanopterin) as universal cofactors. We have omitted these from this check because there are many individual compounds that classify as C1 carriers, and it is not clear a priori which one should be preferred. In a future update, we may consider identifying these using a chemical ontology. Implementation: Determine whether the model employs a lumped or split biomass reaction. Then, using an internal mapping table, try to identify the above list of essential precursors in list of precursor metabolites of either type of biomass reaction. List IDs in the models namespace if the metabolite exists, else use the MetaNetX namespace if the metabolite does not exist in the model. Identifies the cytosol from an internal mapping table, and assumes that all precursors exist in that compartment.



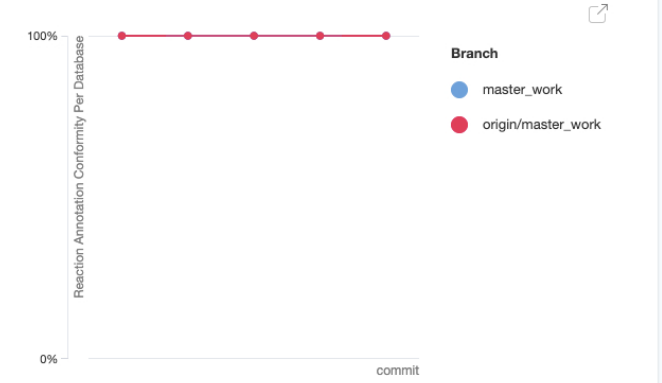
Energy Metabolism

Non-Growth Associated Maintenance Reaction

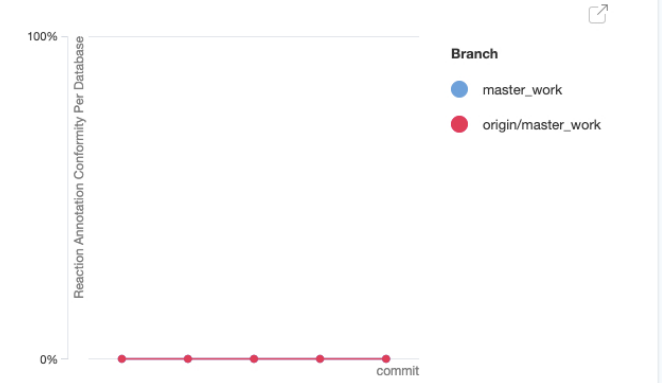
The Non-Growth Associated Maintenance reaction (NGAM) is an ATP-hydrolysis reaction added to metabolic models to represent energy expenses that the cell invests in continuous processes independent of the growth rate. Memote tries to infer this reaction from a list of buzzwords, and the stoichiometry and components of a simple ATP-hydrolysis reaction. Implementation: From the list of all reactions that convert ATP to ADP select the reactions that match the irreversible reaction "ATP + H2O -> ADP + HO4P + H+", whose metabolites are situated within the main model compartment. The main model compartment is assumed to be the cytosol, yet, if that cannot be identified, it is assumed to be the compartment with the most metabolites. The resulting list of reactions is then filtered further by attempting to match the reaction name with any of the following buzzwords ('maintenance', 'atpm',



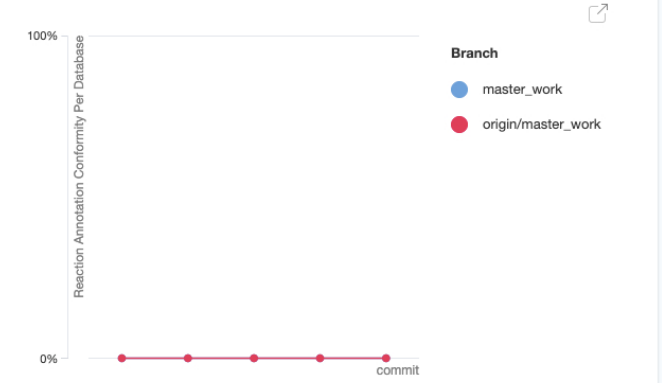
metanetx.reaction



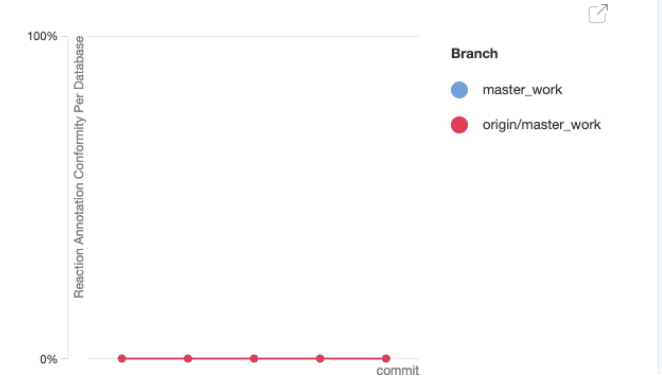
reactome



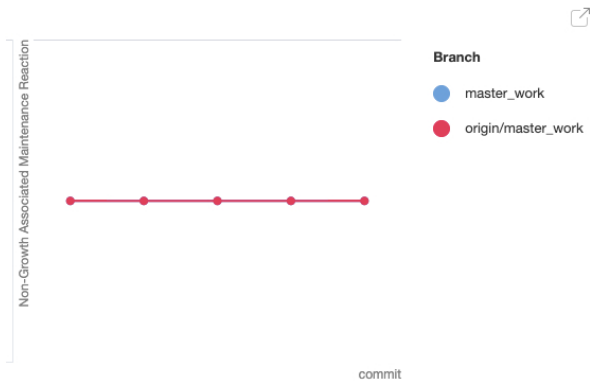
rhea



seed.reaction



'requirement', 'ngam', 'non-growth', 'associated'). If this is possible only the filtered reactions are returned, if not the list is returned as is.



Growth-associated Maintenance in Biomass Reaction

The growth-associated maintenance (GAM) term accounts for the energy in the form of ATP that is required to synthesize macromolecules such as Proteins, DNA and RNA, and other processes during growth. A GAM term is therefore a requirement for any well-defined biomass reaction. There are different ways to implement this term depending on what kind of experimental data is available and the preferred way of implementing the biomass reaction: - Chemostat growth experiments yield a single GAM value representing the required energy per gram of biomass (Figure 6 of [1]_). This can be implemented in a lumped biomass reaction or in the final term of a split biomass reaction. - Experimentally delineating or estimating the GAM requirements for each macromolecule separately is possible, yet requires either data from multi-omics experiments [2]_ or detailed resources [1]_ , respectively. Individual energy requirements can either be implemented in a split biomass equation on the term for each macromolecule, or, on the basis of the biomass composition, they can be summed into a single GAM value for growth and treated as mentioned above. This test is only able to detect if a lumped biomass reaction or the final term of a split biomass reaction contains this term. Hence, it will only detect the use of a single GAM value as opposed to individual energy requirements of each macromolecule. Both approaches, however, have its merits. Implementation: Determines the metabolite identifiers of ATP, ADP, H2O, HO4P and H+ based on an internal mapping table. Checks if ATP and H2O are a subset of the reactants and ADP, HO4P and H+ a subset of the products of the biomass reaction. References: .. [1] Thiele, I., & Palsson, B. Ø. (2010, January). A protocol for generating a high-quality genome-scale metabolic reconstruction. Nature protocols. Nature Publishing Group. http://doi.org/10.1038/nprot.2009.203 .. [2] Hackett, S. R., Zanotelli, V. R. T., Xu, W., Goya, J., Park, J. O., Perlman, D. H., Gibney, P. A., Botstein, D., Storey, J. D., Rabinowitz, J. D. (2010, January). Systems-level analysis of mechanisms regulating yeast metabolic flux Science http://doi.org/10.1126/science.aaf2786

Number of Reversible Oxygen-Containing Reactions

The directionality of oxygen-producing/-consuming reactions affects the model's ability to grow anaerobically i.e. create faux-anaerobic organisms. This test reports how many of these oxygen-containing reactions are reversible. This test does not have any mandatory 'pass' criteria. Implementation: First, find the metabolite representing atmospheric oxygen in the model on the basis of an internal mapping table or by specifically looking for the formula "O2". Then, find all reactions that produce or consume oxygen and report those that are reversible.

Uniform Reaction Identifier Namespace

In well-annotated models it is no problem if the pool of main identifiers for reactions consists of identifiers from several databases. However, in models that lack appropriate annotations, it may hamper the ability of other researchers to use it. Running the model through a computational pipeline may be difficult without first consolidating the namespace. Hence, this test checks if the main reaction identifiers can be attributed to one single namespace based on the regex patterns defined at https://identifiers.org/ Implementation: Generate a pandas.DataFrame with each column corresponding to one database from the selection and each row to the reaction ID. A boolean entry indicates whether the metabolite ID matches the regex pattern of the corresponding database. Since the Biocyc pattern matches quite, assume that any instance of an identifier matching to Biocyc AND any other DB pattern is a false positive match for Biocyc and then set the boolean to ``false``. Sum the positive matches for each database and assume that the largest set is the 'main' identifier namespace.

Annotation - Genes

Presence of Gene Annotation

This test checks if any annotations at all are present in the SBML annotations field (extended by FBC package) for each gene product, irrespective of the type of annotation i.e. specific database, cross-references, ontology terms, additional information. For this test to pass the model is expected to have genes and each of them should have some form of annotation. Implementation: Check if the annotation attribute of each cobra.Gene object of the model is unset or empty.

Gene Annotations Per Database

Info

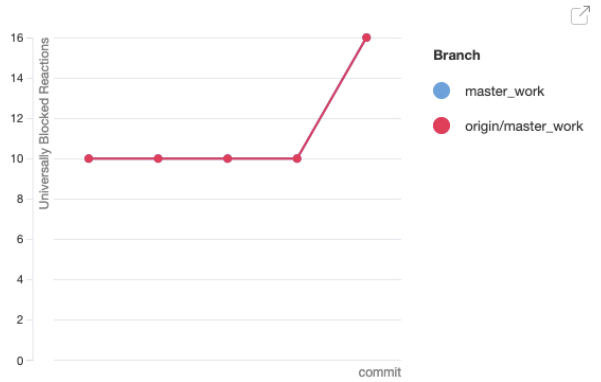
Specific database cross-references are paramount to mapping information. To provide references to as many databases as possible helps to make the metabolic model more accessible to other researchers. This does not only facilitate the use of a model in a broad array of computational pipelines, it also promotes the metabolic model itself to become an organism-specific knowledge base. For this test to pass, each gene annotation should contain cross-references to a number of databases. The currently selection is listed in `annotation.py`, but an ongoing discussion can be found at https://github.com/opencobra/memote/issues/332. For each database this test checks for the presence of its corresponding namespace ID to comply with the MIRIAM guidelines i.e. they have to match those defined on https://identifiers.org/. Since each database is quite different and some potentially incomplete, it may not be feasible to achieve 100% coverage for each of them. Generally it should be possible, however, to obtain cross-references to at least one of the databases for all gene products consistently. Implementation: Check if the keys of the annotation attribute of each cobra.Gene of the model match with a selection of common genome databases. The annotation attribute of cobrapy components is a dictionary of key:value pairs.

asap

Network Topology

Universally Blocked Reactions

Universally blocked reactions are reactions that during Flux Variability Analysis cannot carry any flux while all model boundaries are open. Generally blocked reactions are caused by network gaps, which can be attributed to scope or knowledge gaps. Implementation: Use flux variability analysis (FVA) implemented in cobra.flux_analysis.find_blocked_reactions with open_exchanges=True. Please refer to the cobrapy documentation for more information: https://cobrapy.readthedocs.io/en/stable/autoapi/cobra/flux_analysis/variability/index.html#cobra.flux_analysis.variability. find_blocked_reactions



Orphan Metabolites

Orphans are metabolites that are only consumed but not produced by reactions in the model. They may indicate the presence of network and knowledge gaps. Implementation: Find orphan metabolites structurally by considering only reaction equations and reversibility. FBA is not carried out.

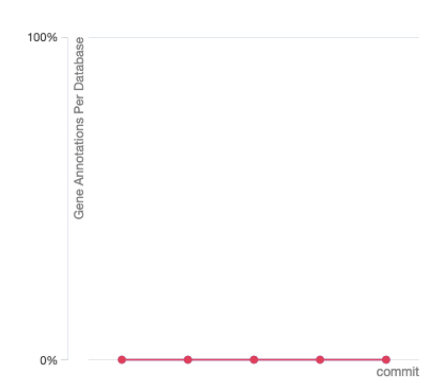
Dead-end Metabolites

Dead-ends are metabolites that can only be produced but not consumed by reactions in the model. They may indicate the presence of network and knowledge gaps. Implementation: Find dead-end metabolites structurally by considering only reaction equations and reversibility. FBA is not carried out.

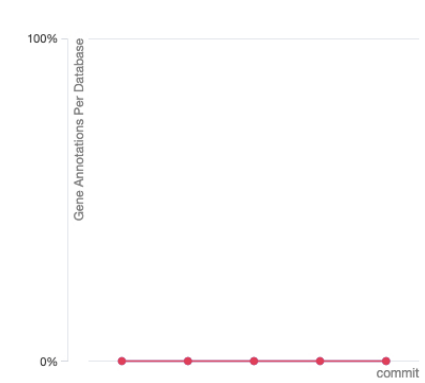
Stoichiometrically Balanced Cycles

Stoichiometrically Balanced Cycles are artifacts of insufficiently constrained networks resulting in reactions that can carry flux when all the boundaries have been closed. Implementation: Close all model boundary reactions and then use flux variability analysis (FVA) to identify reactions that carry flux.

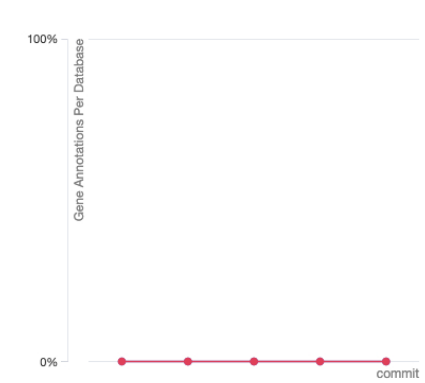
ccds



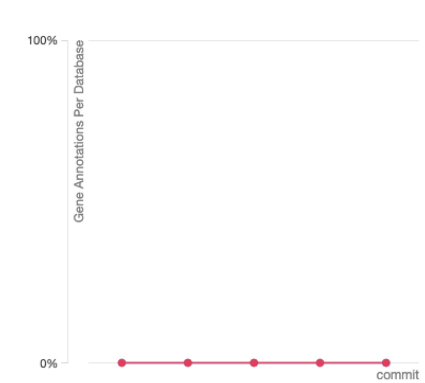
ecogene

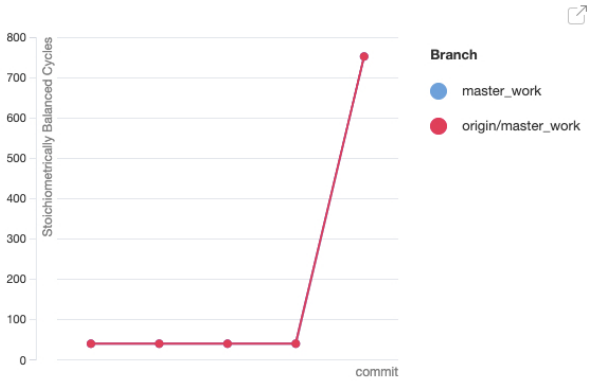


hprd



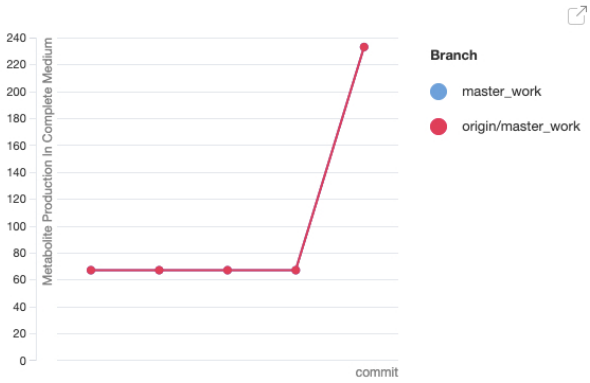
kegg.genes





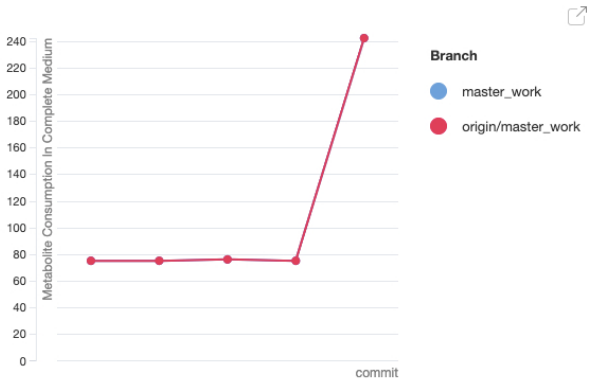
Metabolite Production In Complete Medium

In complete medium, a model should be able to divert flux to every metabolite. This test opens all the boundary reactions i.e. simulates a complete medium and checks if any metabolite cannot be produced individually using flux balance analysis. Metabolites that cannot be produced this way are likely orphan metabolites, downstream of reactions with fixed constraints, or blocked by a cofactor imbalance. To pass this test all metabolites should be producible. Implementation: Open all model boundary reactions, then for each metabolite in the model add a boundary reaction and maximize it with FBA.



Metabolite Consumption In Complete Medium

In complete medium, a model should be able to divert flux from every metabolite. This test opens all the boundary reactions i.e. simulates a complete medium and checks if any metabolite cannot be consumed individually using flux balance analysis. Metabolites that cannot be consumed this way are likely dead-end metabolites or upstream of reactions with fixed constraints. To pass this test all metabolites should be consumable. Implementation: Open all model boundary reactions, then for each metabolite in the model add a boundary reaction and minimize it with FBA.

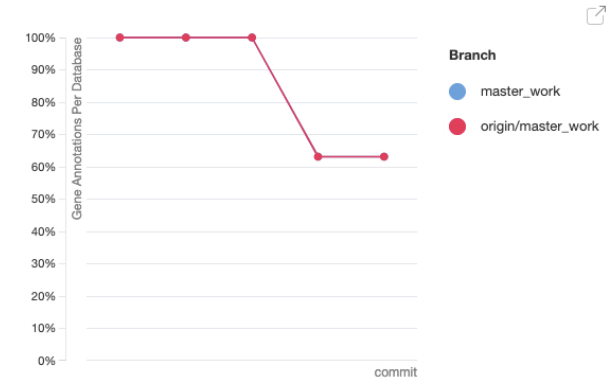


Matrix Conditioning

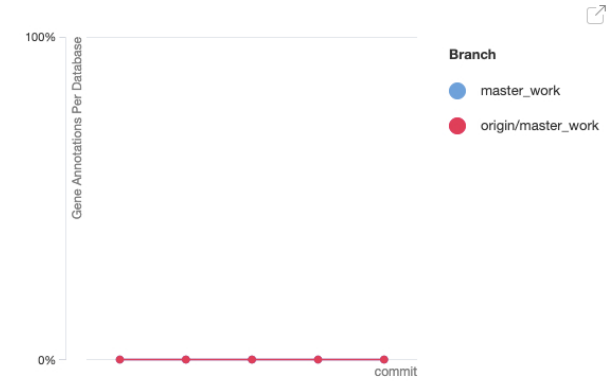
Ratio Min/Max Non-Zero Coefficients

This test will return the absolute largest and smallest, non-zero coefficients of the stoichiometric matrix. A large ratio of these values may point to potential numerical issues when trying to solve different mathematical optimization problems such as flux-balance analysis. To pass this test the ratio should not exceed 10^9. This threshold has been selected based on experience, and is likely to be adapted when more data on solver performance becomes available. Implementation: Compose the stoichiometric matrix, then calculate absolute coefficients and lastly use the maximal value and minimal non-zero value to calculate the ratio.

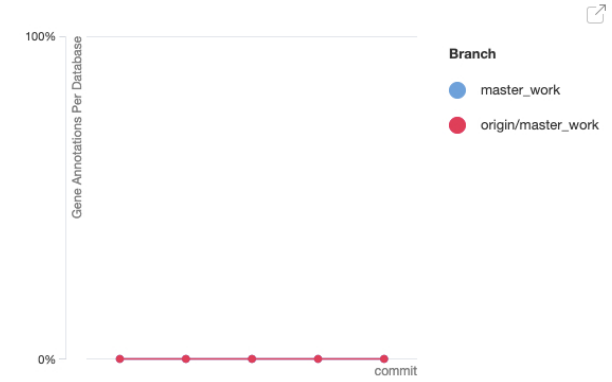
ncbigene



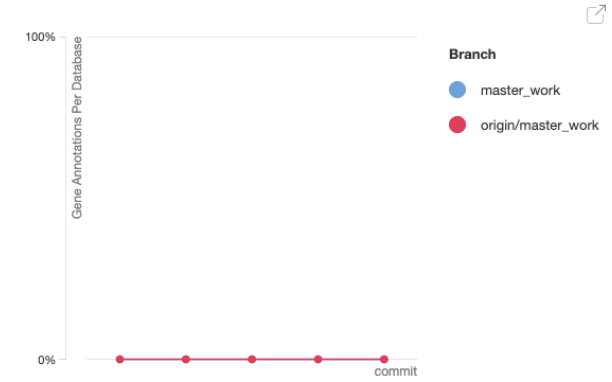
ncbigi



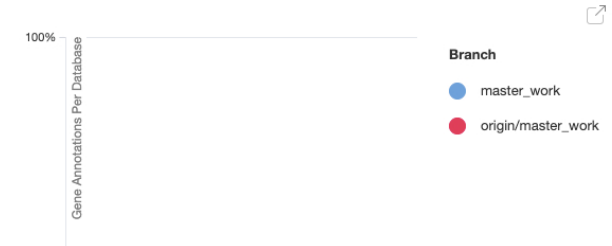
ncbiprotein

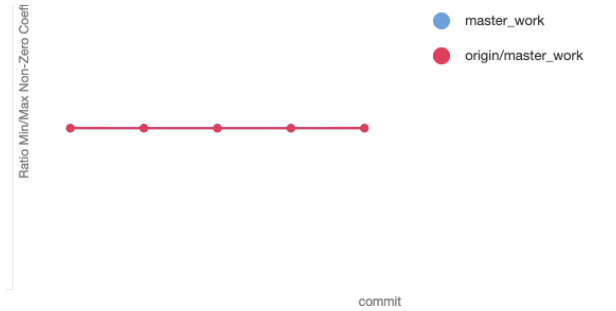


refseq



uniprot



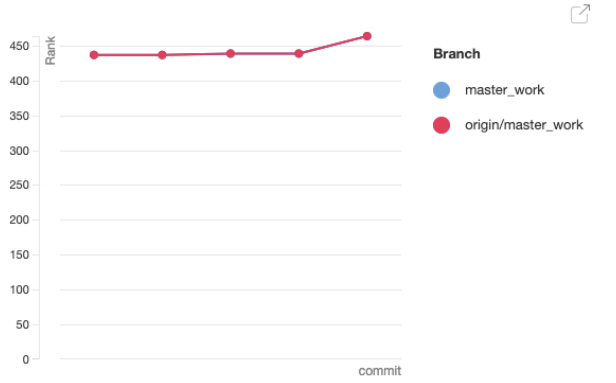


Independent Conservation Relations

This test will return the number of conservation relations, i.e., conservation pools through the left null space of the stoichiometric matrix. This test is not scored, as the dimension of the left null space is system-specific. Implementation: Calculate the left null space, i.e., the null space of the transposed stoichiometric matrix, using an algorithm based on the singular value decomposition adapted from https://scipy.github.io/old-wiki/pages/Cookbook/RankNullspace.html Then, return the estimated dimension of that null space.

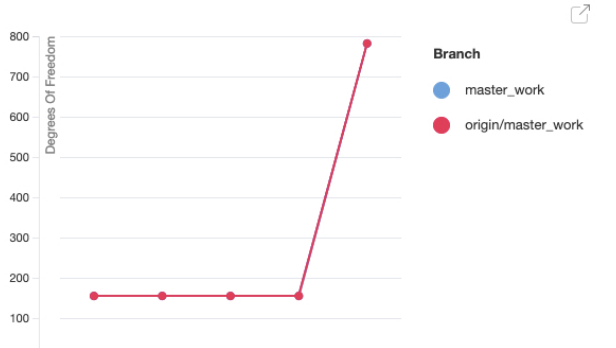
Rank

The rank of the stoichiometric matrix is system specific. It is calculated using singular value decomposition (SVD). Implementation: Compose the stoichiometric matrix, then estimate the rank, i.e. the dimension of the column space, of a matrix. The algorithm used by this function is based on the singular value decomposition of the matrix.



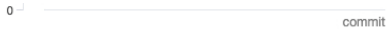
Degrees Of Freedom

The degrees of freedom of the stoichiometric matrix, i.e., the number of 'free variables' is system specific and corresponds to the dimension of the (right) null space of the matrix. Implementation: Compose the stoichiometric matrix, then calculate the dimensionality of the null space using the rank-nullity theorem outlined by Alama, J. The Rank+Nullity Theorem. Formalized Mathematics 15, (2007).



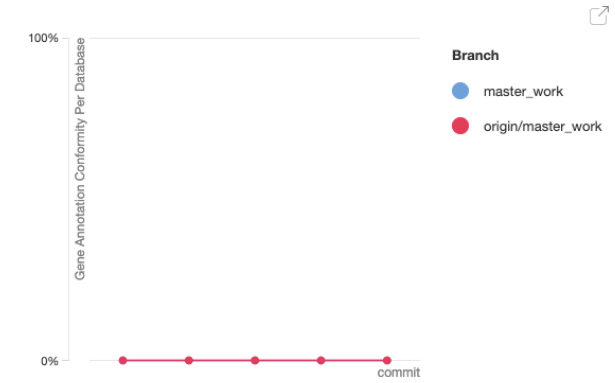
Gene Annotation Conformity Per Database **Info**

To identify databases and the identifiers belonging to them, computational tools rely on the presence of specific patterns. Only when these patterns can be identified consistently is an ID truly machine-readable. This test checks if the database cross-references in reaction annotations conform to patterns defined according to the MIRIAM guidelines, i.e. matching those that are defined at https://identifiers.org/. The required formats, i.e., regex patterns are further outlined in `annotation.py`. This test does not carry out a web query for the composed URI, it merely controls that the regex patterns match the identifiers.
Implementation: For those genes whose annotation keys match any of the tested databases, check if the corresponding values match the identifier pattern of each database.

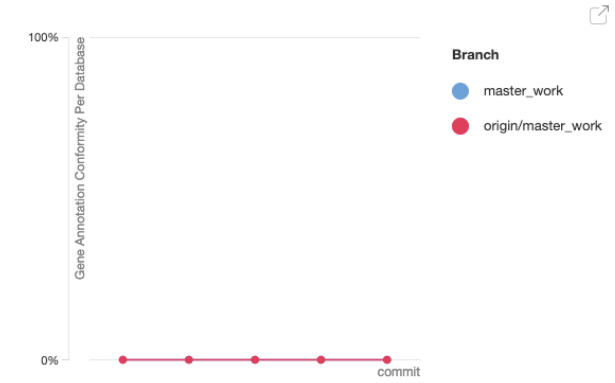


Experimental Data Comparison

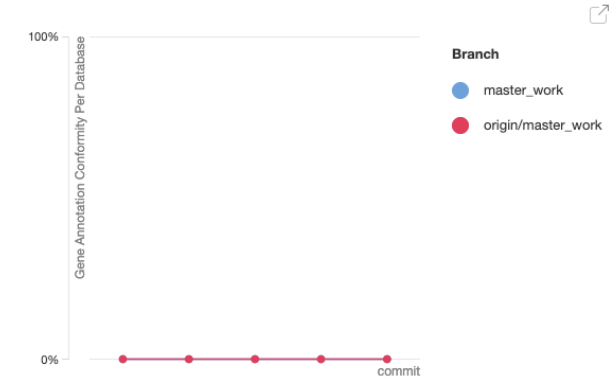
asap



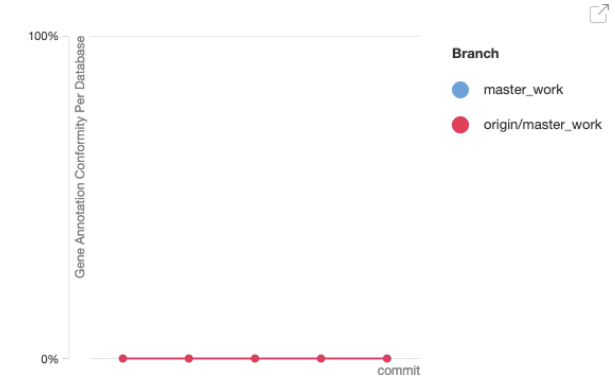
ccds



ecogene

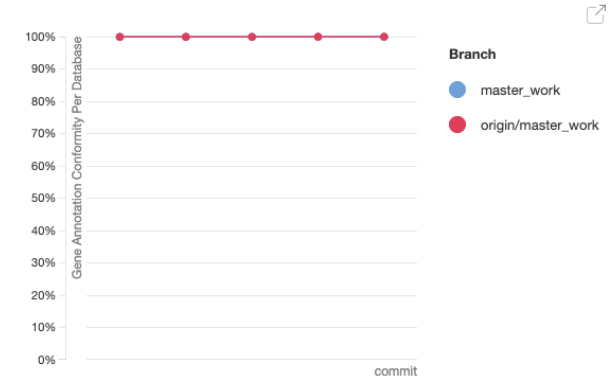


hprd

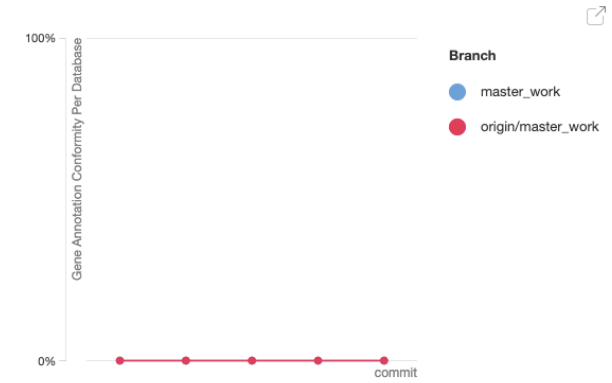


kegg.genes

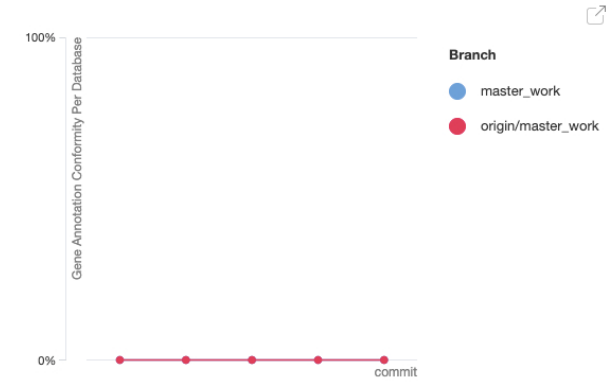
ncbigene



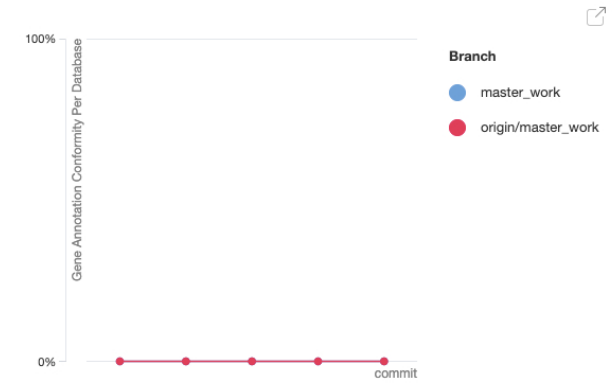
ncbigi



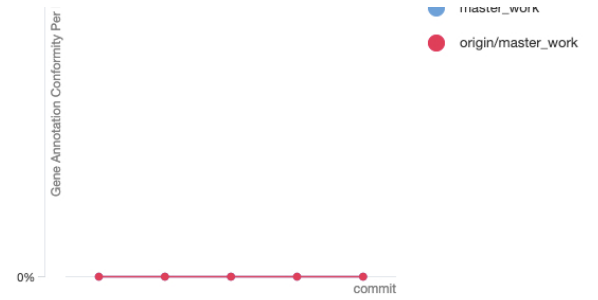
ncbiprotein



refseq



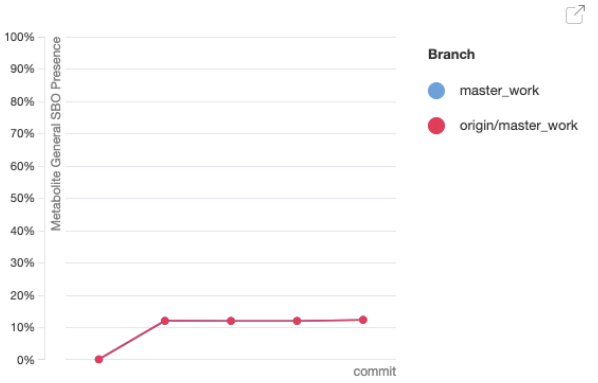
uniprot



Annotation - SBO Terms

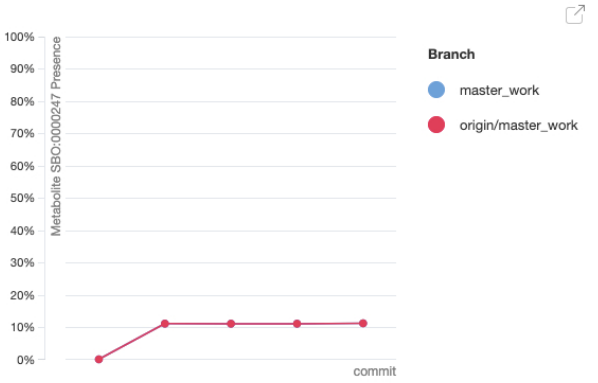
Metabolite General SBO Presence

The Systems Biology Ontology (SBO) allows researchers to annotate a model with terms which indicate the intended function of its individual components. The available terms are controlled and relational and can be viewed here http://www.ebi.ac.uk/sbo/main/tree. Implementation: Check if each cobra.Metabolite has a non-zero "annotation" attribute that contains the key "sbo".



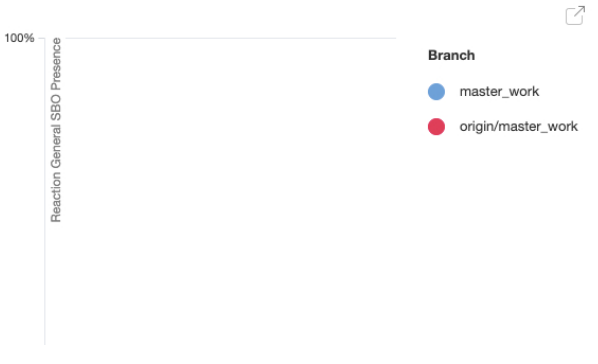
Metabolite SBO:0000247 Presence

SBO:0000247 represents the term 'simple chemical'. Every metabolite should be annotated with this. Implementation: Check if each cobra.Metabolite has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



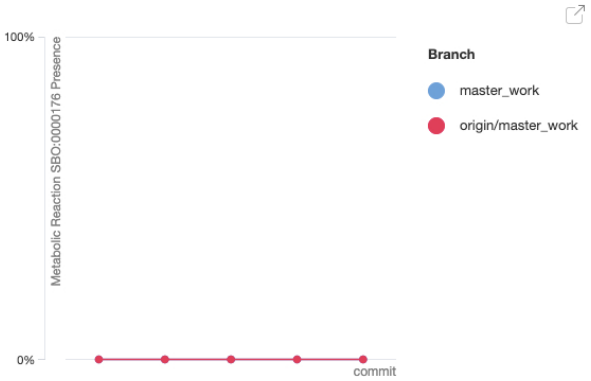
Reaction General SBO Presence

The Systems Biology Ontology (SBO) allows researchers to annotate a model with terms which indicate the intended function of its individual components. The available terms are controlled and relational and can be viewed here http://www.ebi.ac.uk/sbo/main/tree. Implementation: Check if each cobra.Reaction has a non-zero "annotation" attribute that contains the key "sbo".



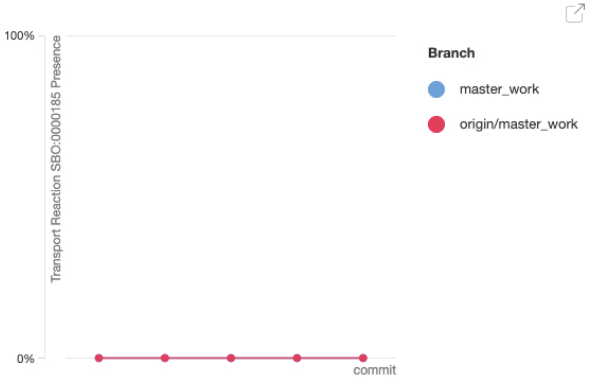
Metabolic Reaction SBO:0000176 Presence

SBO:0000176 represents the term 'biochemical reaction'. Every metabolic reaction that is not a transport or boundary reaction should be annotated with this. The results shown are relative to the total amount of pure metabolic reactions. Implementation: Check if each pure metabolic reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being the SBO term above.



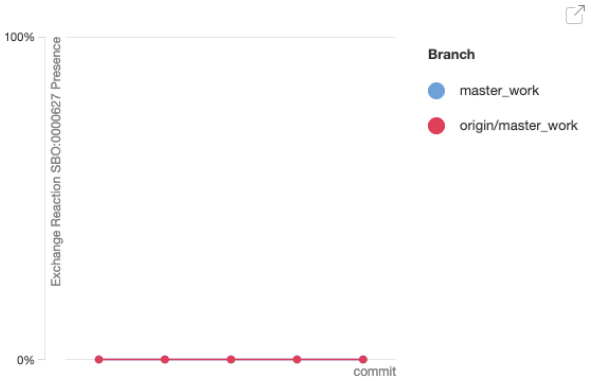
Transport Reaction SBO:0000185 Presence

'SBO:0000185', 'SBO:0000588', 'SBO:0000587', 'SBO:0000655', 'SBO:0000654', 'SBO:0000660', 'SBO:0000659', 'SBO:0000657', and 'SBO:0000658' represent the terms 'transport reaction' and 'translocation reaction', in addition to their children (more specific transport reaction labels). Every transport reaction that is not a pure metabolic or boundary reaction should be annotated with one of these terms. The results shown are relative to the total of all transport reactions. Implementation: Check if each transport reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



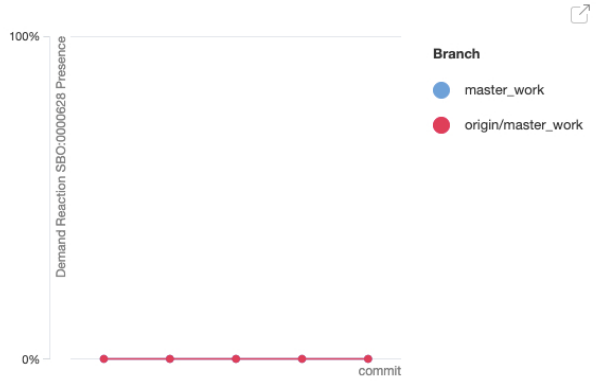
Exchange Reaction SBO:0000627 Presence

SBO:0000627 represents the term 'exchange reaction'. The Systems Biology Ontology defines an exchange reaction as follows: 'A modeling process to provide matter influx or efflux to a model, for example to replenish a metabolic network with raw materials (eg carbon / energy sources). Such reactions are conceptual, created solely for modeling purposes, and do not have a physical correspondence. Exchange reactions, often represented as 'R_EX_', can operate in the negative (uptake) direction or positive (secretion) direction. By convention, a negative flux through an exchange reaction represents uptake of the corresponding metabolite, and a positive flux represent discharge.' Every exchange reaction should be annotated with this. Exchange reactions differ from demand reactions in that the metabolites are removed from or added to the extracellular environment only. Implementation: Check if each exchange reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



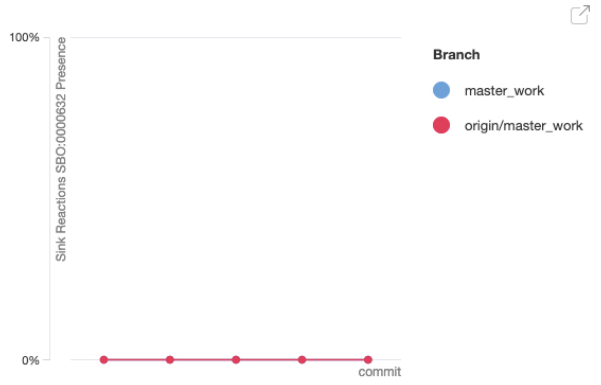
Demand Reaction SBO:0000628 Presence

SBO:0000628 represents the term 'demand reaction'. The Systems Biology Ontology defines a demand reaction as follows: 'A modeling process analogous to exchange reaction, but which operates upon "internal" metabolites. Metabolites that are consumed by these reactions are assumed to be used in intra-cellular processes that are not part of the model. Demand reactions, often represented 'R_DM_', can also deliver metabolites (from intra-cellular processes that are not considered in the model).' Every demand reaction should be annotated with this. Demand reactions differ from exchange reactions in that the metabolites are not removed from the extracellular environment, but from any of the organism's compartments. Demand reactions differ from sink reactions in that they are designated as irreversible. Implementation: Check if each demand reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



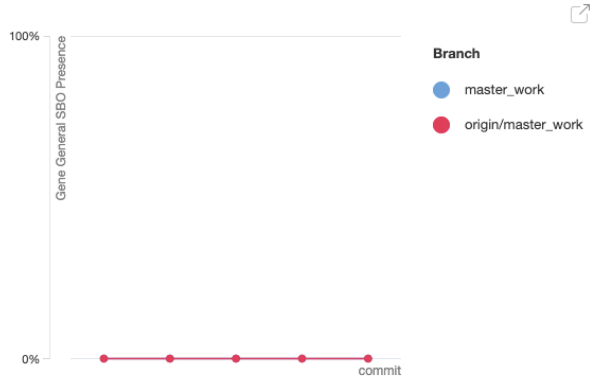
Sink Reactions SBO:0000632 Presence

SBO:0000632 represents the term 'sink reaction'. The Systems Biology Ontology defines a sink reaction as follows: 'A modeling process to provide matter influx or efflux to a model, for example to replenish a metabolic network with raw materials (eg carbon / energy sources). Such reactions are conceptual, created solely for modeling purposes, and do not have a physical correspondence. Unlike the analogous demand (SBO:....) reactions, which are usually designated as irreversible, sink reactions always represent a reversible uptake/secretion processes, and act as a metabolite source with no cost to the cell. Sink reactions, also referred to as R_SINK_, are generally used for compounds that are metabolized by the cell but are produced by non-metabolic, un-modeled cellular processes.' Every sink reaction should be annotated with this. Sink reactions differ from exchange reactions in that the metabolites are not removed from the extracellular environment, but from any of the organism's compartments. Implementation: Check if each sink reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



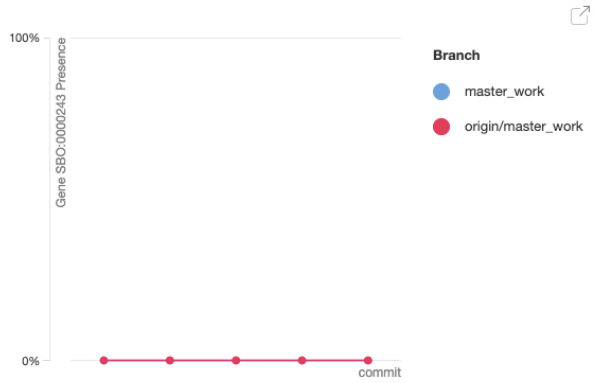
Gene General SBO Presence

The Systems Biology Ontology (SBO) allows researchers to annotate a model with terms which indicate the intended function of its individual components. The available terms are controlled and relational and can be viewed here http://www.ebi.ac.uk/sbo/main/tree. Check if each cobra.Gene has a non-zero "annotation" attribute that contains the key "sbo".



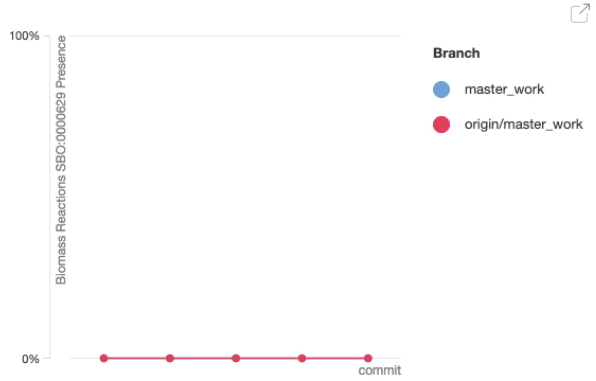
Gene SBO:0000243 Presence

SBO:0000243 represents the term 'gene'. Every gene should be annotated with this. Implementation: Check if each cobra.Gene has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.

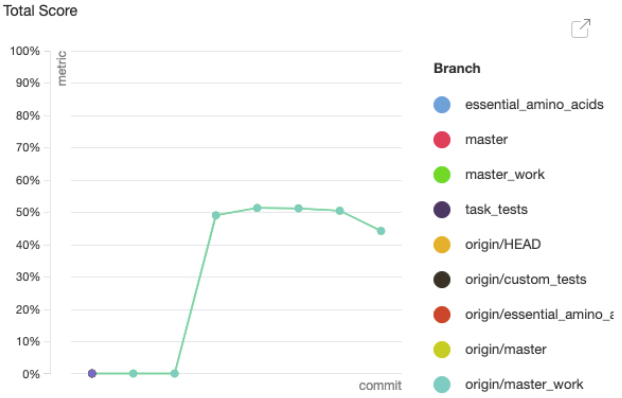


Biomass Reactions SBO:0000629 Presence

SBO:0000629 represents the term 'biomass production'. The Systems Biology Ontology defines an exchange reaction as follows: 'Biomass production, often represented 'R_BIOMASS_', is usually the optimization target reaction of constraint-based models, and can consume multiple reactants to produce multiple products. It is also assumed that parts of the reactants are also consumed in unrepresented processes and hence products do not have to reflect all the atom composition of the reactants. Formulation of a biomass production process entails definition of the macromolecular content (eg. cellular protein fraction), metabolic constitution of each fraction (eg. amino acids), and subsequently the atomic composition (eg. nitrogen atoms). More complex biomass functions can additionally incorporate details of essential vitamins and cofactors required for growth.' Every reaction representing the biomass production should be annotated with this. Implementation: Check if each biomass reaction has a non-zero "annotation" attribute that contains the key "sbo" with the associated value being one of the SBO terms above.



Total Score



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway