Visualisation and map manipulation in Cell Designer

Authors:

Jennifer Modamio, Anna Danielsdottir, Systems Biochemistry group, University of Luxembourg.

Nicolas Sompairac, Bioinformatics and Computational Systems Biology of Cancer, Institut Curie.

Reviewer(s): Ronan Fleming, Inna Kuperstein and Andrei Zinovyev.

INTRODUCTION

Visualisation of data on top of biochemical pathways is an important tool for interpreting the results of constrained-based modeling. It can be an invaluable aid for developing an understanding of the biological meaning implied by a prediction. Biochemical network maps permit the visual integration of model predictions with the underlying biochemical context. Patterns that are very difficult to appreciate in a vector can often be much better appreciated by studying a generic map contextualised with model predictions. Genome-scale biochemical network visualisation is particularly demanding. No currently available software satisfies all of the requirements that might be desired for visualisation of predictions from genome-scale models.

Here we present a tool for the visualisation of computational predictions from The Constraint-based Reconstruction and Analysis Toolbox (COBRA Toolbox) [1] to available metabolic maps developed in CellDesigner [2].

Several maps are used in this tutorial for illustration: (i) a comprehensive mitochondrial metabolic map compassing 1263 metabolic reactions extracted from the latest version of the human cellular metabolism, Recon 3D [3]. (ii) Small map contaning Glycolisis and TCA for faster testing and manipulation. (iii) A mitochondria map combining metabolic pathways with protein-protein interactions (PPI). Proteins and complexes implicated in mitochondrial reactions have been extracted from the Parkinson Disease map (PDMap) [4].

In this tutorial, manipulation of CellDesgner maps in COBRA toolbox and visualising model predictions is explained. The main covered topics are:

- Loading metabolic models and models containing both metabolic and regulatory networks constructed in CellDesigner
- Detection and correction of discrepancies between map and models
- Basic map manipulation (change color and size of nodes and reactions, directionality of reactions, reaction types...)
- Basic model analysis visualisation (visualisation of Flux Balance Analysis)

EQUIPMENT SETUP

To visualise the metabolic maps it is necessary to obtain the version 4.4 of CellDesigner. This software can be freely downloaded from:

http://www.celldesigner.org/download.html

Initialise The Cobra Toolbox and set the solver.

If needed, initialise the cobra toolbox.

```
initCobraToolbox
```

The present tutorial can run with glpk package, which does not require additional installation and configuration. Although, for the analysis of large models it is recommended to use the GUROBI package.

```
if changeCobraSolver('gurobi', 'LP', 0)
    changeCobraSolver('gurobi6','all')
end
```

Model setup.

In this tutorial, we provided two exclusive metabolic models. A mitochondrial model and a small metabolic model specific for Glycolysis and Citric acid cycle. Both models were extracted from the latest version of the database of the human cellular metabolism, Recon 3D [3]. For extra information about metabolites structures and reactions, and to download the latest COBRA model releases, visit the Virtual Metabolic Human database (VMH, https://vmh.life).

Before proceeding with the simulations, load the model into the workspace:

```
modelFileName = 'Recon2.0model.mat';
modelDirectory = getDistributedModelFolder(modelFileName);
modelFileName = [modelDirectory filesep modelFileName];
model = readCbModel(modelFileName);
```

PROCEDURE

Aforementioned, two kind of maps will be used along the tutorial. Depending on the nature of the species in the map (metabolites or proteins), different functions will be used to import the XML file produced in CellDesigner into MATLAB.

1. Import a CellDesigner XML file to MATLAB environment

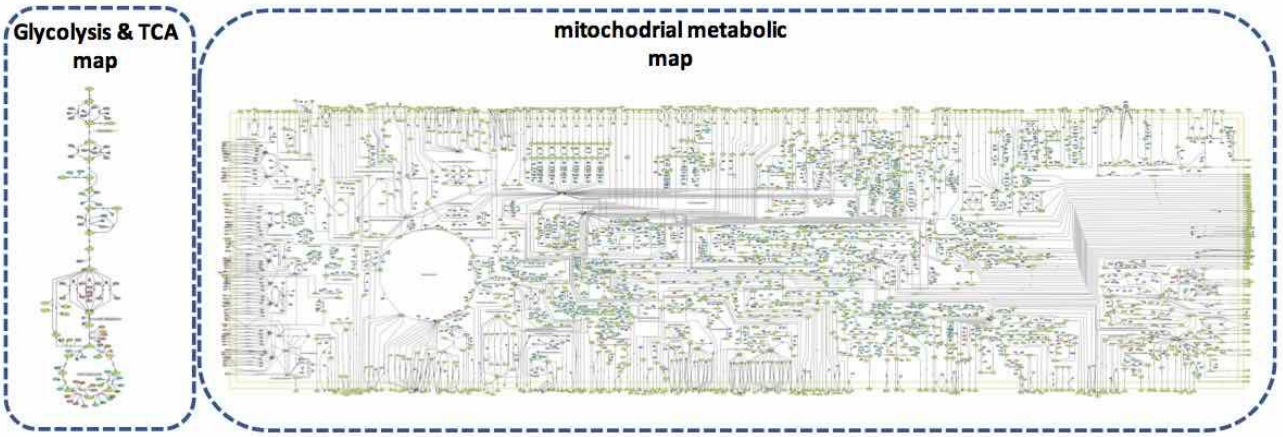
A) Parse a Metabolic map

The `transformXML2Map` function parses an XML file from Cell Designer (CD) into a Matlab structure. This structure is organised similarly to the structure found in the COnstraint-Base and Reconstruction Analysis (COBRA) models.

Load two types of metabolic maps:

1. A small metabolic model representative of glycolysis and citric acid cycle.
2. A bigger metabolic map representative of the mitochondrial metabolism.

```
[xmlGly, mapGly] = transformXML2Map('glycolysisAndTCA.xml');
[xmlMitoMetab, mapMitoMetab] = ...
    transformXML2Map('metabolicMitochondria.xml');
```



Metabolic maps provided in this tutorial. A mitochondrial map compassing 1263 metabolic reactions. A small map summarising glycolysis and citric acid cycle.

This function internally calls the function `xml2struct` which parses the XML file to a general Matlab structure. Afterwards, this general structure is reorganised. Due to this internal function, there are two outputs: "`xml`" is the general Matlab structure obtain from `xml2struct` function, whereas "`map`" is the desired final structure that could be later manipulated.

B) Parse a Metabolic map combined with Protein-Protein-Interactions (PPI)

The `transformFullXML2Map` function parses an XML file from Cell Designer (CD) into a Matlab structure. The resultant structure contains all the information commonly stored in a metabolic map, plus extra information corresponding to proteins and complexes.

```
[xmlPPI, mapPPI] = transformFullXML2Map('metabolicPPIMitochondria.xml');
```

NOTE! *The XML file to be parsed must be in the current folder in MATLAB when executing the function.*

TIMING

The time to parse a Cell Designer map from a XML file to a MATLAB structure or vice-versa depends on the size of the map, and can vary from seconds to minutes.

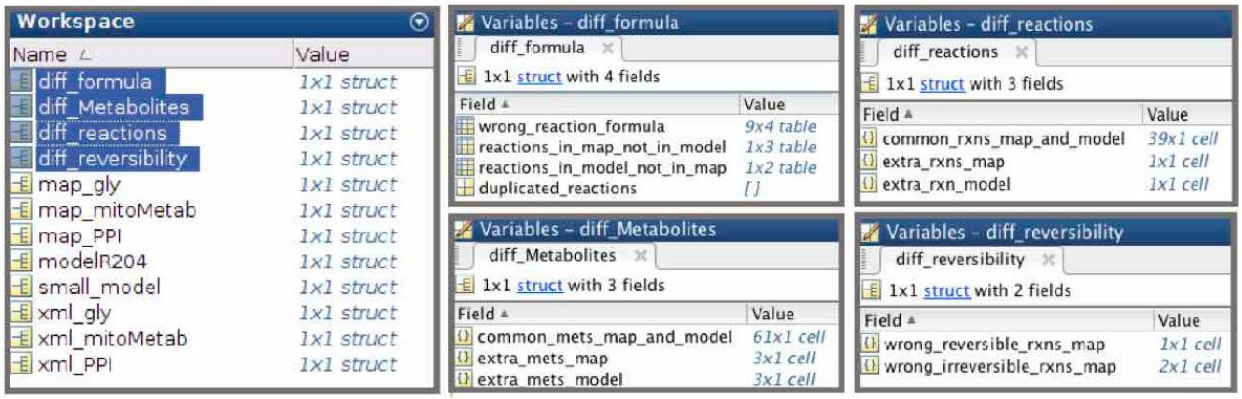
TROUBLESHOOTING (Control error check)

In order to properly visualise the modeling obtained during model analysis, the map used for the visualisation must match with the model under study. Errors in reaction or metabolite names are highly common, leading to mismatches and therefore wrong representation of data.

In order to ensure a proper visualisation of the outputs coming from model analysis, a control error check is highly recommended.

The function `checkCDerrors` gives four outputs summarising all possible discrepancies between model and map.

```
[diffReactions, diffMetabolites, diffReversibility, diffFormula] = ...
    checkCDerrors(mapGly, model);
```



`CheckCDerrors` output. This function compares reactions name, formula and reversibility as well as metabolite names between model and map. Four outputs are obtained from this function.

Four outputs are obtained from this function:

"`diffReactions`" summarises present and absent reactions between model and map.

"`diffMetabolites`" summarises present and absent metabolites.

NOTE! *Note that having more metabolites and reactions in the COBRA model is normal since the model can contain more elements than the map. From the other hand, the map should only contain elements present in the model.*

"`diffReversibility`" summarises discrepancies in defining the reversibility of reactions.

The last output "`diffFormula`" summarises discrepancies in reactions formulae (kinetic rates) and also lists duplicated reactions.

Some functions have been developed to modify attributes in the map automatically from MATLAB:

- Errors in reaction names can be manually corrected in the Matlab structure with the function `correctRxnNameCD`. In the example one of the most common errors is shown: spaces in names are identified as errors.

```
correctRxns = diffReactions.extraRxnModel;
mapGlyCorrected = correctRxnNameCD(mapGly, ...
    diffReactions, correctRxns);
```

- Errors in metabolites can be corrected manually or automatically by the function `correctErrorMets` by giving a list of correct metabolite names. In the example, "`diffMetabolites.extraMetsModel`" correspond to the correct name of wrong metabolites in "`diffMetabolites.extraMetsMap`".

```
correctMets = diffMetabolites.extraMetsModel;
mapGlyCorrected = correctMetNameCD(mapGlyCorrected, ...
    diffMetabolites, correctMets);
```

- Two functions can be used to solve errors in defining the reaction reversibility. The functions `transformToReversibleMap` and `transformToIrreversibleMap`, modify the reversibility of

reactions in the map. Reaction lists obtained from "`diffReversibility`" can be used as an input of the next functions.

To correct a reversible reaction in the map, irreversible in the model.

```
mapGlyCorrected = transformToIrreversibleMap(mapGlyCorrected, ...
    diffReversibility.wrongReversibleRxnsMap);
```

To correct a irreversible reaction in the map, reversible in the model.

```
mapGlyCorrected = transformToReversibleMap(mapGlyCorrected, ...
    diffReversibility.wrongIrreversibleRxnsMap);
```

NOTE! *Reversibility errors due to base direction of the arrow can only be manually fixed in Cell designer. When creating a "reversible" reaction in CellDesigner, first a "irreverisble" reaction is created and has a particular direction. This "base" direction can be interpreted as an error as it dictates what metabolites are reactants or products.*

In order to check the reaction reversibility, reaction formulae can be printed from the map and model using different functions.

```
wrongFormula = mapFormula(mapGly, ...
    diffReversibility.wrongIrreversibleRxnsMap);
```

Print the same formula in the model to see the corrected formula:

```
rightFormula = printRxnFormula(model, ...
    diffReversibility.wrongIrreversibleRxnsMap);
```

Print reaction formula from the corrected file mapGlyCorrected:

```
correctedFormula = mapFormula(mapGlyCorrected, ...
    diffReversibility.wrongIrreversibleRxnsMap);
```

- **Anticipated results**

Before correcting formula's errors, run the control check again. Probably several errors in the output "`diffFormula`" have already been taken care of when correcting previous outputs.

```
[diffReactions, diffMetabolites, diffReversibility, diffFormula] = ...
    checkCDerrors(mapGlyCorrected,model);
```

NOTE! *Formula errors can only be manually corrected from the XML file in Cell designer.*

2. Export the modified MATLAB structure to CellDesigner XML file

In order to save the corrections previously made into an XML file, two functions are available depending on the MATLAB structure used.

A) Parse a metabolic MATLAB structure

The "`transformMap2XML`" function parsed a MATLAB structure (from a simple metabolic map) into a XML file. In order to save the previous corrections made.

```
transformMap2XML(xmlGly, ...
    mapGlyCorrected,'GlycolysisAndTCACorrected.xml');
```

B) Parse a metabolic MATLAB structure combined with PPI

As in the parsing from XML to a MATLAB structure, a different function will be used when proteins and complexes are present in the map "`transformFullMap2XML`".

Visualisation of Metabolic networks

EQUIPMENT SETUP

CellDesigner uses the HTML-based colour format. This information if used to modify colors of pathways of interest such as Fluxes. The function `createColorsMap` contains all references for different colours HTML code to be directly recognised in Cell Designer and associated to a specific name. Therefore, users wont need to give a code but a colour name in capitals (143 colors are recognized).

```
% Check the list of available colours to use in Cell designer (retrieve 143 colors)
% open createColorsMap.m
```

PROCEDURE

Several modification can be done in the maps. All attributes can be easily reached in the COBRA type MATLAB structure and modified. The colour, name, type and size of nodes can be easily modified from MATLAB instead of doing it manually in CellDesigner. Furthermore, other attributes such as reaction type or reversibility (previously mentioned) can also be modified.

1. Change reaction colour and width

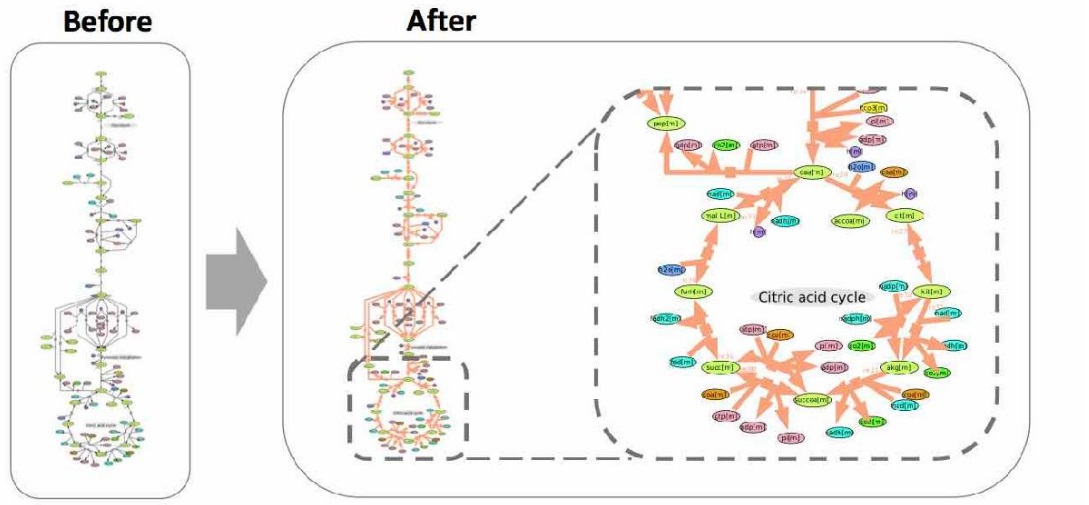
The function `changeRxnColorAndWidth` modifies the width and colour of reactions provided in the form of a list of names.

- **Anticipated results**

All reactions present in the map can be coloured if the list given is extracted from the map and not from the model. See the next example:

In the example, all reactions in the map will be coloured as Light-salmon and have a width of 10 (width=1 by default). Furthermore, the newly generated map will be transformed to be opened in CD.

```
mapGlyColoured = changeRxnColorAndWidth(mapGlyCorrected, ...
    mapGlyCorrected.rxnName, 'LIGHTSALMON',10);
transformMap2XML(xmlGly, mapGlyColoured,'mapGlyRxnColoured.xml');
```



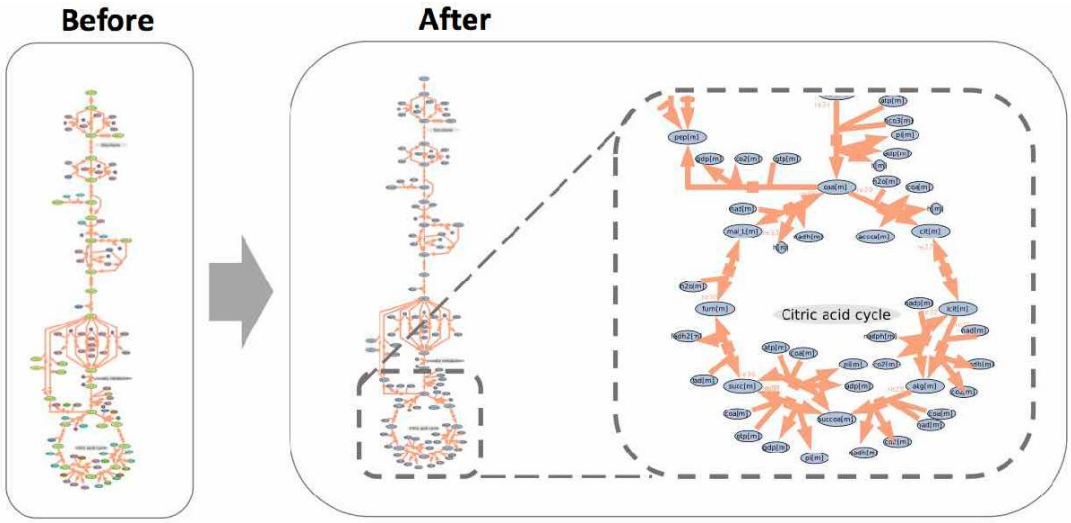
`changeRxnColourAndWidth`: changes the colour and with of reactions given from a list. Here all reactions present in map were selected (`map_gly.rxnName`).

NOTE! *in this example all reactions present in the map are being given as input list. However, this list can contain a set of reactions given by the user.*

2. Add colour to metabolites

The function `addColourNode` adds colour to all nodes linked to a specific list of reactions. Taking as an example the previous list we can modify the colour of all those nodes in the map in Light-steel-blue. Furthermore, the newly generated map will be transformed into a XML file.

```
mapGlyColouredNodes = addColourNode(mapGlyColoured, ...
    mapGlyColoured.rxnName, 'LIGHTSTEELBLUE');
transformMap2XML(xmlGly, ...
    mapGlyColouredNodes,'mapGlyMetColoured.xml');
```



`addColourNodes`: changes the colour of nodes linked to specific reactions (map_gly.rxnName).

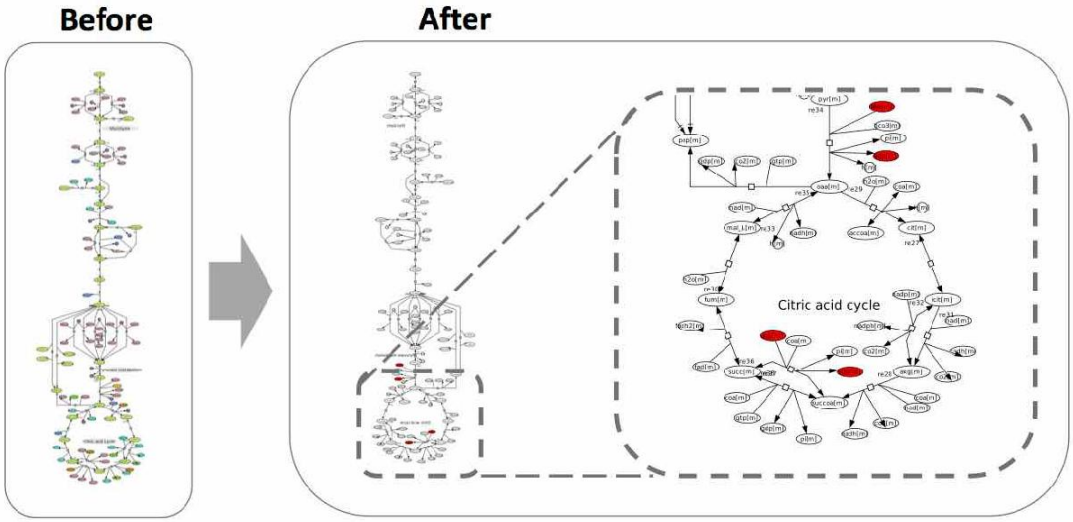
3. Changing the colour of individual metabolite

It is possible to change the colour of specific metabolites in the map given a list of metabolite names. Here for example we want to visualise where ATP and ADP appear in order to give a global visual image of where energy is being produced and consumed.

First, for an easier visualisation all metabolites present in the map will be coloured in white. Afterwards, selected metabolites "mitochondrial ATP and ADP" will be coloured in Red. Finally, the newly generated map will be transformed into a XML file.

```
mapATPADP = changeMetColor(mapGly, ...
    mapGly.specName, 'WHITE'); % Change the colour of all nodes in the map to white
mapATPADP = changeMetColor(mapATPADP, ...
    {'atp[m]'}, 'RED'); % Change specifically the colour of ATP and ADP
mapATPADP = changeMetColor(mapATPADP, {'adp[m]'}, 'RED');

transformMap2XML(xmlGly, mapATPADP, 'mapATPADPColoured.xml');
```

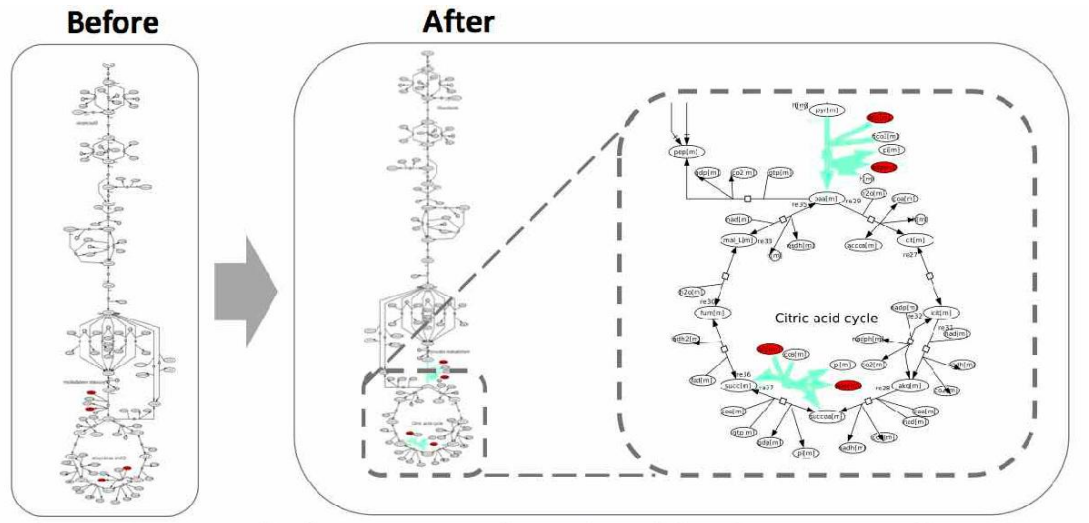


`changeMetColour`: changes the colour of nodes given a list of metabolite names. In this example, firstly all nodes are colour in white (map_gly.specName). Afterwards, two selected nodes “atp and adp” are coloured in red.

Furthermore, we can also color reactions linked to specific metabolites by combining functions for the COBRA models and Visualisation. The function `findRxnsFromMets` identify all reactions containing specific metabolites. Here we want to find reactions containing "mitochondrial ATP and ADP", and colour these reactions in "aquamarine". Moreover,the newly generated map will be transformed into a XML file.

```
rxnsATPADP = findRxnsFromMets(model,{'atp[m]';'adp[m]'});
mapATPADPRxns = changeRxnColorAndWidth(mapATPADP, ...
    rxnsATPADP, 'AQUAMARINE' ,10);

transformMap2XML(xmlGly, mapATPADPRxns, 'mapATPADPRxnsColoured.xml');
```



`findRxnsFromMets:` identify reactions associated to specific metabolites. These reactions can afterwards being coloured by using `changeRxnColourAndWidth` function.

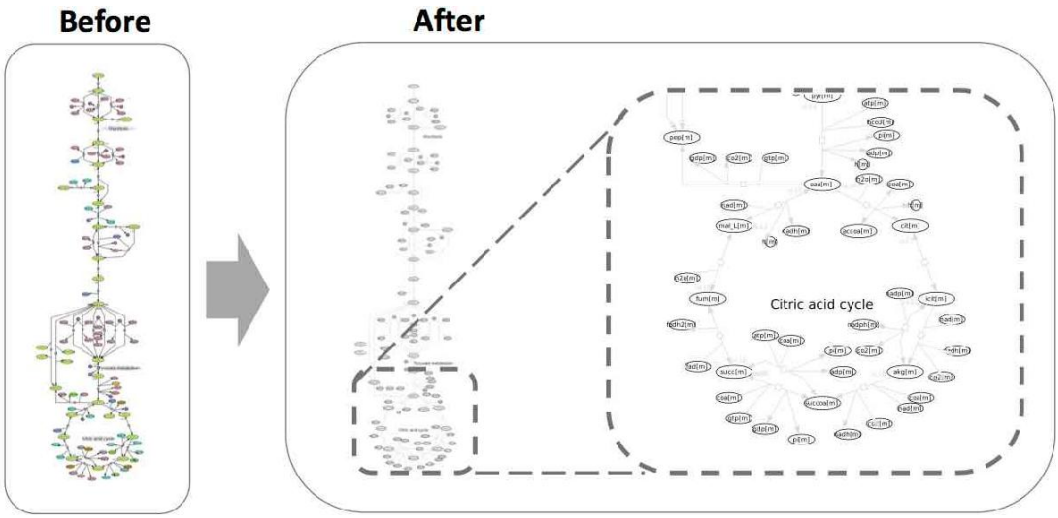
NOTE! *This funtion colors a list of specific metabolites whereas the function `addColourNode` color all nodes linked to a specific list of reactions.*

This combination of specific metabolites and reactions can be also directly done using the function `modifyReactionsMetabolites` mentioned before. However, using the functions described in this section, one can colour metabolites associated to the same reaction and chose colours in different ways.

4. Erase coloring of the map

In order to better visualize small changes it might be necessary to unify the rest of colours in the map. The function `unifyMetabolicMapCD` changes all nodes colour to white and reactions colour to light grey and width to 1 (usually set as default).

```
mapGlyUnified = unifyMetabolicMapCD(mapGly);
transformMap2XML(xmlGly, mapGlyUnified, 'mapGlyUnified.xml');
```

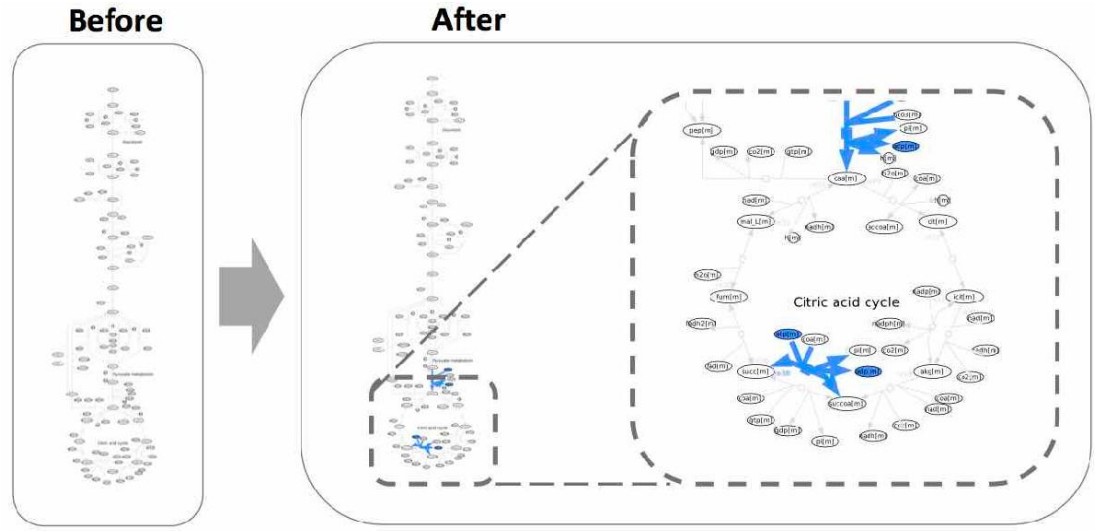


`unifyMetabolicMapCD:` turns to white all nodes and to light grey all reactions.

5. Change reactions and specific nodes color and width

The combination of previously mentioned functions can be summarise by the function `modifyReactionsMetabolites.` This function colours a list of reactions and specific metabolites linked to these reactions. In this example, reactions containing "mitochondrial ATP and ADP" will be coloured in Dodger-blue and have a width of 10. Furthremore, the two described metabolites will be also coloured.

```
mapGlyATPADPRxns = modifyReactionsMetabolites(mapGlyUnified, ...
    rxnsATPADP , {'atp[m]';'adp[m]'}, 'DODGERBLUE', 10);
transformMap2XML(xmlGly, ...
    mapGlyATPADPRxns, 'mapGlyATPADPRxnsAllColoured.xml');
```

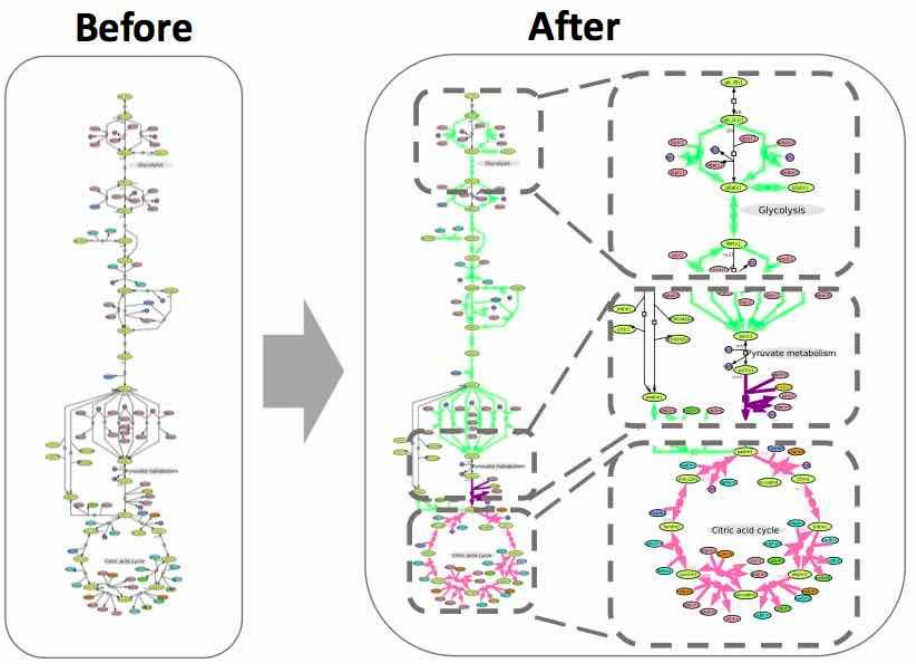


`changeColourRxnsAndMets:` given a list of reactions and metabolites, it changes the colour of reactions and specific metabolites.

6. Colour subsystems

The function `colorSubsystemCD` colours all reactions associated to a specific subsystem in the model (a subsystem is to be understood as a metabolic pathway). Furthermore changes in the width are also possible. Here three subsystems are differentially coloured: Glycolysis, TCA and Pyruvate metabolism.

```
mapSubSystems = colorSubsystemCD(mapGly, ...
    model, 'Citric acid cycle', 'HOTPINK', 10);
mapSubSystems = colorSubsystemCD(mapSubSystems, ...
    model, 'Pyruvate metabolism', 'DARKMAGENTA' , 10);
mapSubSystems = colorSubsystemCD(mapSubSystems, ...
    model, 'Glycolysis/gluconeogenesis', 'SPRINGGREEN', 10);
transformMap2XML(xmlGly, ...
    mapSubSystems,'mapGlySubsystemsColoured.xml');
```



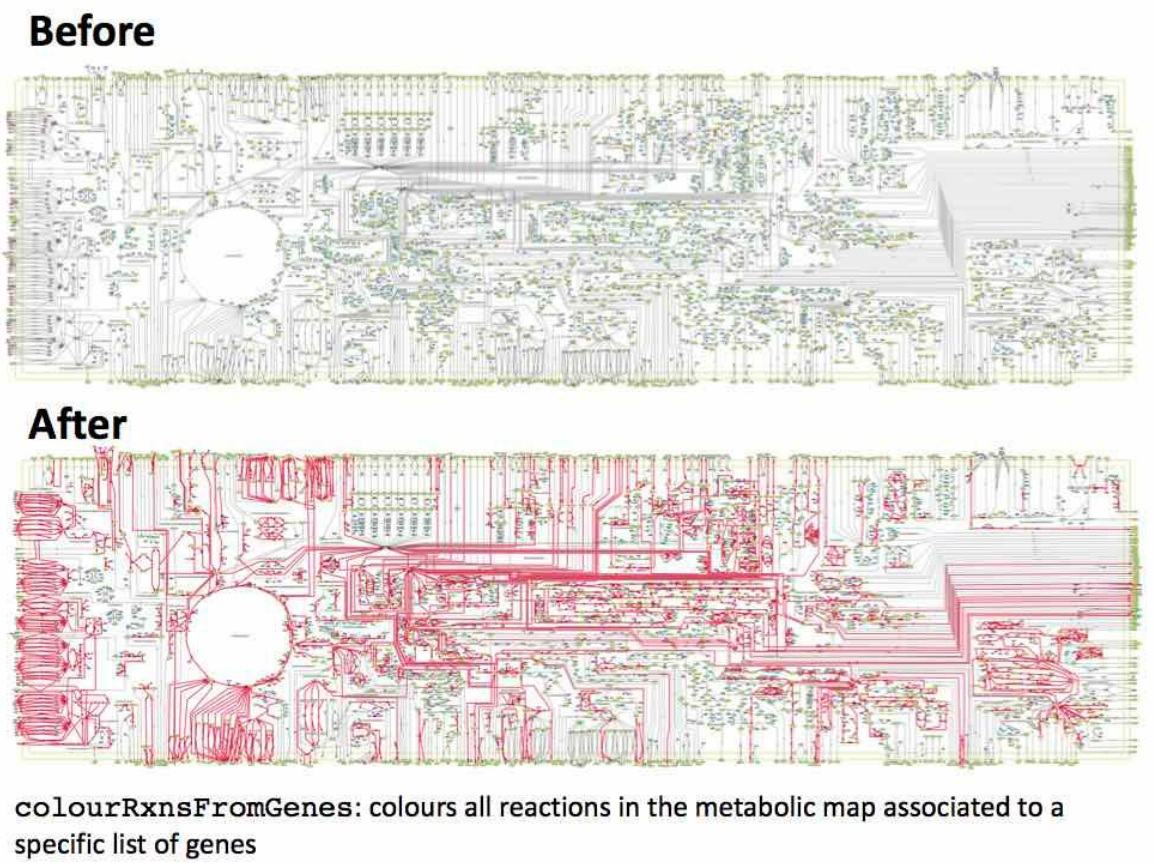
colourSubsystemCD: colour all reactions associates to a specific subsystem in the model.

7. Colour reactions associated to specific genes

The function `colorRxnsFromGenes` colours metabolic reactions linked to a gene based on a list of Entrez references. A list of mitochondria-associated genes was obtained from mitocarta 2.0 [5]. Based on this list of genes, we would like to know how many reactions in our metabolic map are associated to these genes.

```
load('mitocartaHumanGenes.mat')
mapMitocarta = colorRxnsFromGenes(mapMitoMetab, model, ...
    mitocartaHumanGenes, 'CRIMSON', 10);

transformMap2XML(xmlMitoMetab, ...
    mapMitocarta, 'mapMitocartaHumanGenes.xml');
```



`colourRxnsFromGenes`: colours all reactions in the metabolic map associated to a specific list of genes

8. Change and visualize reaction types

One of the characteristics of CD, is the possibility to represent different types of reactions depending on the interaction that needs to be ilustrated. As for example, the common reaction type is named "STATE TRANSITION" (check `map.rxnsType`). Reactions can be modified manually; however it can also be done automatically with the function `changeRxnType`. In the next example, we identify transport reactions in Recon2 model, and we change them to "TRANSPORT" type in the map.

First, we obtain reactions associated to all transport subsystems in Recon2 model:

```
transportMitochondria = ismember([model.subSystems{:}]', 'Transport');
index=find(transportMitochondria);
transportReactions = model.rxns(index,1);
```

Afterwards, we use this list of reactions to change the reaction type in the map. As a result, all "transport" reactions in the model will be converted to "transport" type in the map.

```
mitoMapTransport = changeRxnType(mapMitoMetab, transportReactions, 'TRANSPORT');
```

Furthermore, we would like to visualize these "transport" reactions:

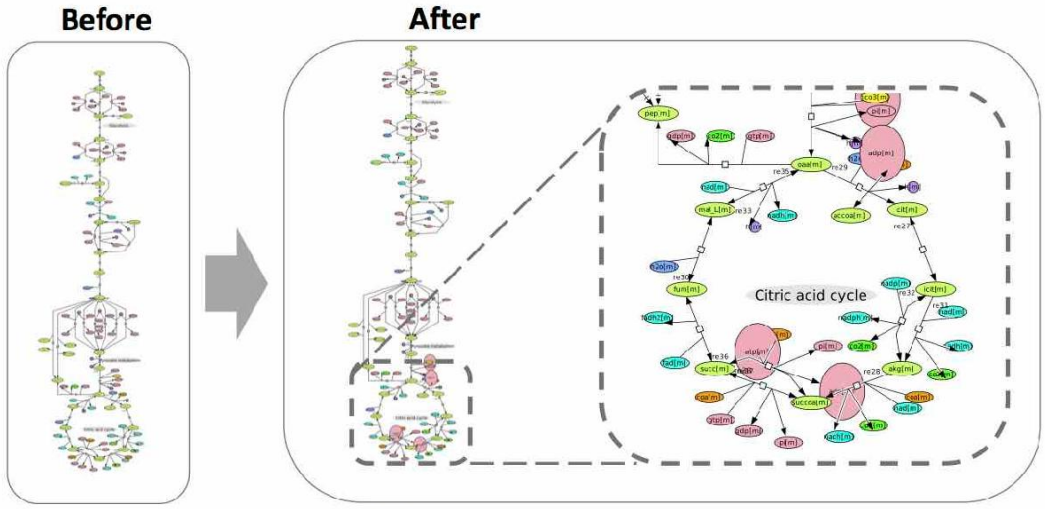
```
mitoMapTransportColoured = colorRxnType(mitoMapTransport, ...
    'TRANSPORT', 'DARKORCHID', 10);
transformMap2XML(xmlMitoMetab, mitoMapTransportColoured, ...
    'mapMitocartaTransportColoured.xml');
```

`colourRxnType:` colours a specific type of reactions

9. Changing the node sizes in CellDesigner

Molecule sizes can be modified by changing their height and width with the function `changeNodesArea` by giving a list of molecule names. Previously, we visualized mitochondrial ATP and ADP. Here we would like to modify their size in the map. The measures given are: heigth 100 and width 80.

```
mapNodesArea = changeNodesArea(mapGly, {'atp[m]'; 'adp[m]'}, 100, 80);
transformMap2XML(xmlGly, mapNodesArea, 'mapNodesArea.xml');
```



`changeNodeArea:` changes the area of selected nodes

NOTE! *Size of metabolites can be automatically modified. However coordinates of metabolites in the map will remain the same. Due to this, changes in size might lead to overlapping between entities.*

TROUBLESHOOTING

In order to come back to the default map, two functions revert the changes saved in the map. `defaultColourCD` reverts the changes in reactions width and colour (revert all reactions to black and width 1). Furthremore, `defaultLookMap` reverts to default reactions colour and width (black and 1) and metabolites colour and size (a specific colour is given depending on metabolite type, moreover the size is also given depending on metabolite relevance).

```
mapDefault = defaultColorCD(mapGly);
mapDefaultAll = defaultLookMap(mapGly);
```

Functions mimicking COBRA functions for model manipulation

1. Obtain logical matrices

Based on the COBRA model structure of the S matrix, similar matrices can be created based on the map. Three logical matrices will be added to the map structure:

- sID: Logical matrix with rows=speciesID and columns=reactionsID
- sAlias: Logical matrix with rows=speciesAlias and columns=reactionsID
- idAlias: Logical matrix with rows=speciesID and columns=speciesAlias

These matrices will be added to the map structure by the function `getMapMatrices`.

```
mapGlyCorrected = getMapMatrices(mapGlyCorrected);
```

`getMapMatrices`: This function generated three matrices based on the information compiled in the map

NOTE! *To use these matrices with COBRA, this function should be used with Metabolic only maps. An error will occur if used with PPI maps!*

2. Research of indexes in the map structure

Basic functions were created to simplify the manipulation of maps.

The function `findRxnsPerTypeInMap` gives a list of desired reactions based on the reaction type (2nd colum), and the index for those reactions in `map.rxnName` (1st colum).

```
transportReactionsIndexList = findRxnsPerTypeInMap(mapGlyCorrected, 'TRANSPORT');
```

The function `findMetsInMap` finds the IDs of specific metabolites in `map.specName` given a list of metabolites.

```
ATPADPIndexList = findMetsInMap(mapGlyCorrected, ...
    {'atp[c]', 'adp[c]', 'atp[m]', 'adp[m]'});
```

The function `findRxnsInMap` finds the IDs of specific reactions in `map.rxnName` given a list of reactions.

```
rxnOfInterestIndexList = findRxnsInMap(mapGlyCorrected, ...
    {'FBP', 'FBA', 'GAPD', 'PFK', 'ENO'});
```

The function `findMetsFromCompartInMap` finds all metabolites in the map associated to a specific compartment by looking at the composition of metabolite names (example: mitochondrial atp = atp**[m]**).

```
[mitochondrialMetsNameList,mitochondrialMetsIndexList] = ...
    findMetsFromCompartInMap(mapGlyCorrected, '[m]');
```

The function `findRxnsFromCompartInMap` finds all reactions in the map associated to a specific compartment by looking at the composition of metabolite names.

```
mitochondrialRxnsIndexList = ...
    findRxnsFromCompartInMap(mapGlyCorrected, '[m]');
```

Visualisation of Metabolic and PPI networks

Some of the aforementioned functions were addapted to be used in metabolic and PPi maps. Some examples are shown below:

1. Change protein colour based on a list of proteins

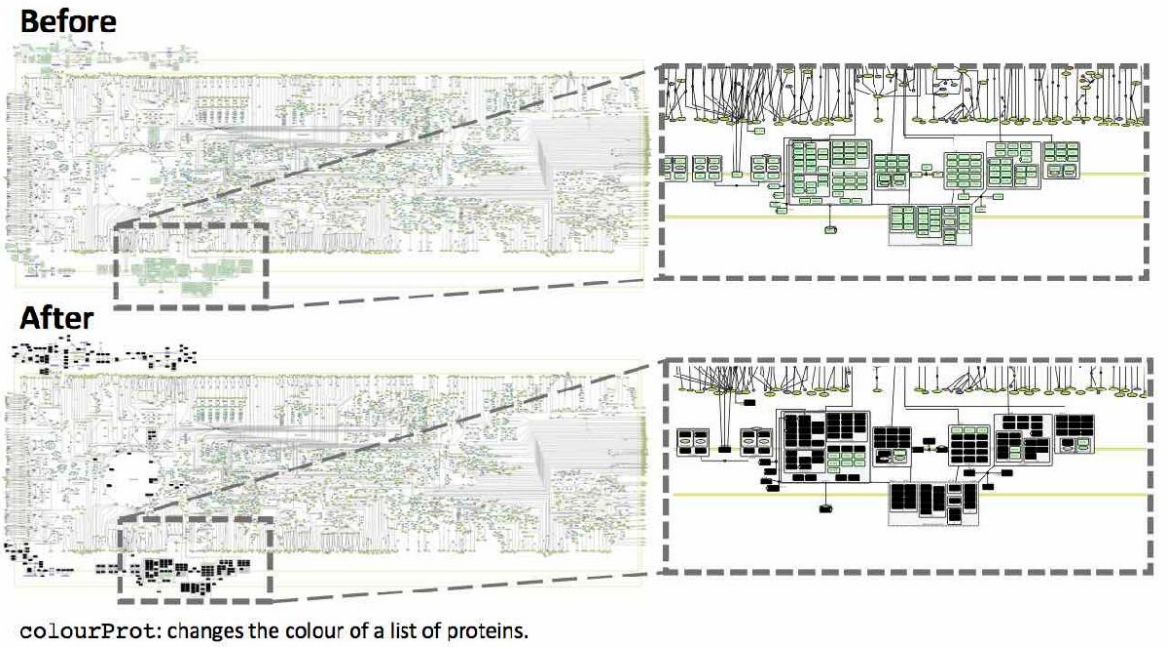
A list of proteins of interest can be highlighted in the map. Here we extract a list of proteins associated to the mitochondrial compartment from PDmap [4].

First, we load the protein list.

```
load('mitochondrialProteinsPDmap.mat')
```

Then, we would like to identify those proteins in our map.

```
mapColouredProteins = colorProtein(mapPPI, ...
    mitochondrialProteinsPDmap(:,1), 'BLACK');
transformFullMap2XML(xmlPPI, ...
    mapColouredProteins, 'mapColouredProteins.xml');
```



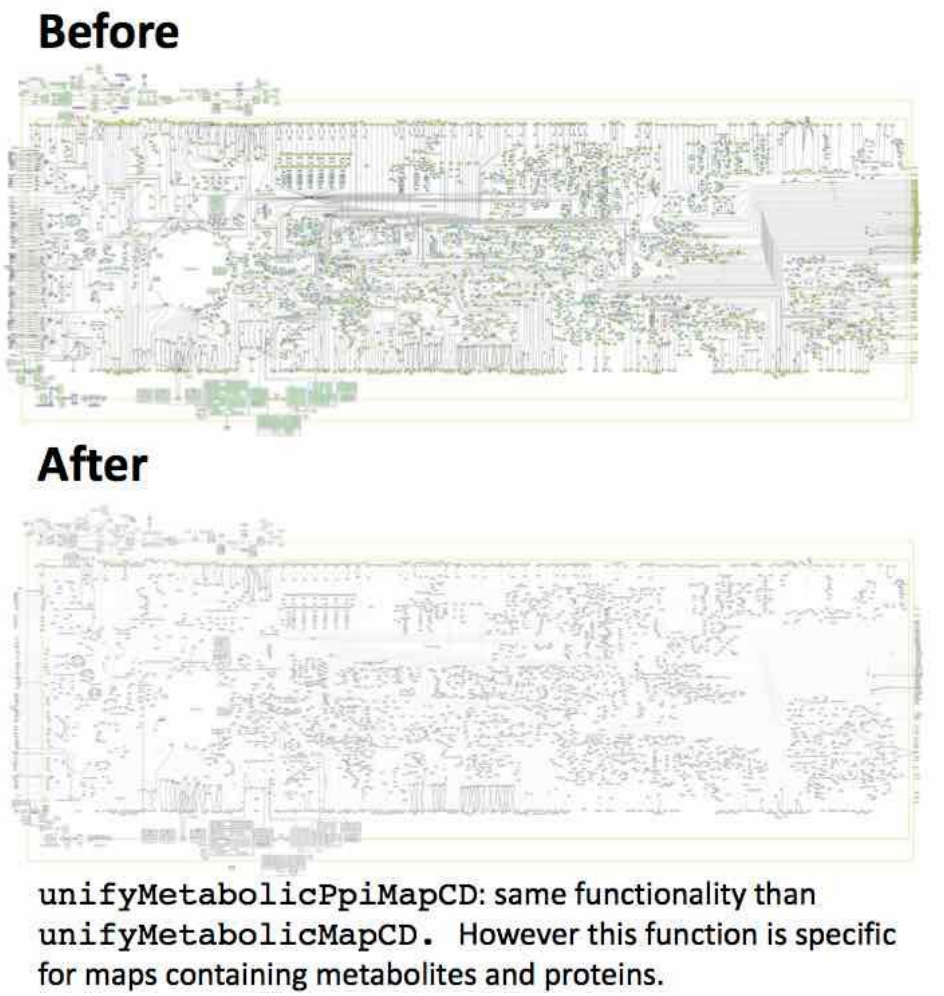
`colourProt`: changes the colour of a list of proteins.

NOTE! *A gene can codify for more than one protein. An association protein-gene-metabolicReaction doesn't have to be true. Manual curation would be required in this step.*

TROUBLESHOOTING

In order to come back to the default map, two functions revert the changes saved in the map. `defaultColorCD` reverts the changes in reactions width and colour (revert all reactions to black and width 1). Furthermore, `defaultColorAndSizeCDMap` reverts to default reactions and nodes (colour and size).

```
map2 = unifyMetabolicPPImapCD(mapPPI);
transformFullMap2XML(xmlPPI, map2, 'mapPPIUnified.xml');
```



`unifyMetabolicPpiMapCD`: same functionality than `unifyMetabolicMapCD`. However this function is specific for maps containing metabolites and proteins.

Specific visualisation for model analysis:

Visualise results from Flux Balance analysis (FBA)

Flux balance analysis is a mathematical approach for analysing the flow of metabolites through a metabolic network. This flow can be added to a parsed XML Matlab structure and visualised a posteriori in CellDesigner.

First, it would be necessary to select an objective function. Since the map represented is a mitochondria (main organelle responsible for energy production in the cell), we would maximize ATP production through complex V in the electron transport chain.

```
load('mitomodels.mat')
formulaATPS4m = printRxnFormula(modelMito3D, 'ATPS4m');
model = changeObjective(modelMito3D, 'ATPS4m');
fbaSolution = optimizeCbModel(model, 'max');
```

The output FBA`solution` will be afterwards used as input. The width assigned to each reaction in the map is directly related to the flux carried by the reaction in FBA.

Two types of visualisation are available:

- For a basic visualisation of fluxes `addFluxFBA` can be used:

```
mapGeneralATP = addFluxFBA(mapMitoMetab, modelMito3D, ...
    fbaSolution, 'MEDIUMAQUAMARINE');
transformMap2XML(xmlMitoMetab, mapGeneralATP, 'fbaFlux.xml');
```

- For a more specific visualisation including directionality of reactions, `addFluxFBAdirectionAndcolour` function can be used:

```
mapSpecificATP = addFluxFBAdirectionAndColor(mapMitoMetab, ...
    modelMito3D, fbaSolution);
transformMap2XML(xmlMitoMetab, ...
    mapSpecificATP, 'fbaFluxDirectionality.xml');
```

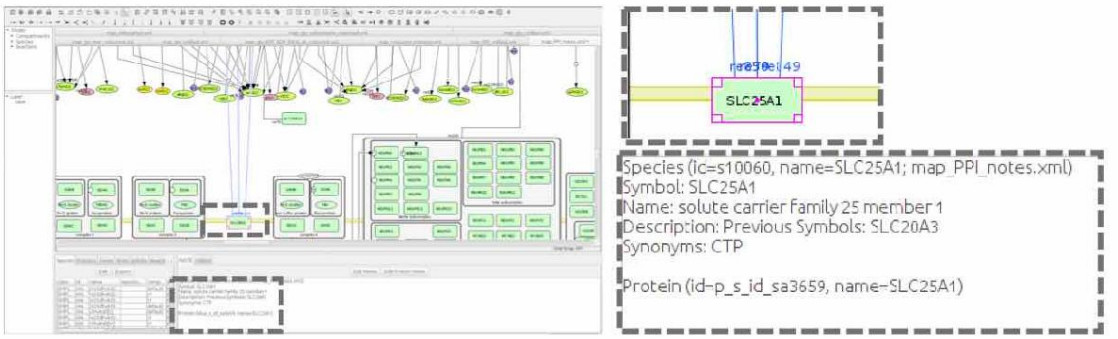
Visualize gene expression

A Cytoscape plugin [6] is available to visualize gene expression on top of a network map generated from CellDesigner: **BiNoM** [7].

Writing information about models to map

Information from the models specific to each reaction or metabolite can be retrieved from the models and added to the CellDesigner map in notes.

```
mapGlyCorrectedNotes = addNotes(model, mapGlyCorrected);
transformMap2XML(xmlGly, mapGlyCorrectedNotes, 'mapGlyNotes.xml');
```



`addNotes:` add notes into the map.

1. Hyduke D. COBRA Toolbox 2.0. *Nature Protocols* (2011).
2. Thiele, I., et al. "A community-driven global reconstruction of human metabolism". *Nat. Biotechnol.,* 31(5), 419-425 (2013).
3. Funahashi A. "CellDesigner: a process diagram editor for gene-regulatory and biochemical networks". *BIOSILICO*, 1:159-162, (2003).
4. Kazuhiro A. "Integrating Pathways of Parkinson's Disease in a Molecular Interaction Map". *Mol Neurobiol*. 49(1):88-102 (2014).
5. Calvo SE. "MitoCarta2.0: an updated inventory of mammalian mitochondrial proteins". *Nucleic Acids Res*. 4;44(D1):D1251-7 (2016).

6. Shannon, Paul et al. “Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks.” *Genome Research* 13.11 (2003): 2498–2504. *PMC*. Web. 5 Dec. (2017).
7. Bonnet E. et al, "BiNoM 2.0, a Cytoscape plugin for accessing and analyzing pathways using standard systems biology formats". *BMC Syst Biol.* 1;7:18 (2013).