Drug target prediction using context-specific metabolic models reconstructed from rFastcormics

Running title: Drug target prediction with rFastcormics

Tamara Bintener[1], Maria Pires Pacheco[1], Ali Kishk[1], Jeff Didier[1], Thomas Sauter[1*]

[1]Department of Life Sciences and Medicine, University of Luxembourg, Esch-Alzette, Luxembourg

[*]Corresponding Author : Thomas Sauter (thomas.sauter@uni.lu)

Abstract

Metabolic modelling is a powerful computational tool to analyse metabolism. It has not only been used to identify metabolic rewiring strategies in cancer but also to predict drug targets and candidate drugs for repurposing. Here, we will elaborate on the reconstruction of context-specific metabolic models of cancer using rFASTCORMICS and the subsequent prediction of drugs for repurposing using our drug prediction workflow.

Keywords Metabolic Modelling, Cancer, Drug prediction, Drug repurposing

1. Introduction

De novo drug discovery is expensive and time-consuming. On average, from the discovery of a new compound to the approval of a drug, 14 years pass, and the approval rates are still dismal while the necessary investments are increasing. Drug repurposing, on the other hand, requires fewer resources while searching for new indications of already approved drugs. As the drug profiles are already established, the first clinical trial phases can be skipped, greatly reducing the costs and approval times ***(1)***.

Computational and systems biology approaches have been developed to predict novel drug targets, their mechanisms of action as well as their interaction that can be depicted via gene regulatory ***(2)***, protein interaction ***(3, 4)***, signalling ***(5)***, and metabolic networks ***(6–8)***.

For example, genome-scale metabolic networks are a computational representation of metabolism and describe the relationship between genes, proteins, reactions, and the involved metabolites. Metabolic models have many applications that enabled, among others, predicting biomarkers for inborn errors of metabolism ***(9)***, identifying metabolic changes and mechanisms in cancer ***(10, 11)***, and cancer-specific drug targets ***(6–8, 12–14)***.

Model-building algorithms, such as the FASTCORE family ***(14–16)***, extract a subnetwork from a generic metabolic reconstruction that best reflects the input experimental data by maximizing the inclusion of reactions under the control of expressed genes and the exclusion of reactions associated with inactive genes. Therefore, it is possible to reconstruct context-specific models that are specific for a tissue, cancer type, or even patient.

By reconstructing and comparing the metabolism between cancer-specific and healthy models, metabolic vulnerabilities and drug targets can be discovered and exploited for drug repurposing using the here explained drug target and repurposing workflow that has previously been used to identify drugs for colorectal cancer ***(14)*** and melanoma ***(17)***. But the drug repurposing workflow based on rFASTCORMICS for RNA-seq data can be applied without modifications to find, analyse, and overcome drug resistance mechanisms in other cancers and diseases, given the appropriate experimental data is provided.

Within the drug repurposing workflow, *in silico* gene knockouts are performed to simulate the inhibiting effects of drugs as gene knockouts can be a surrogate for drug effects. Even though there exist options to perform single, double, and multiple knockouts in metabolic models, we have slightly altered the original code to only delete those genes that are inhibited by a drug. As such, the input of the *in silico* deletion is no longer genes but drugs.

2. Materials

Please download and install the following programs and toolboxes on your computer.

2.1. Matlab and toolboxes

rFASTCORMICS was developed for Matlab (MathWorks) and requires the Statistics and Machine Learning Toolbox as well as the Curve Fitting Toolbox. Both toolboxes can be downloaded from the “Add-On Explorer” within Matlab.

Additionally, a compatible IBM CPLEX installation is needed (or an equivalent solver). A free academic version of the IBM ILOG CPLEX Optimization Studio can be requested for university staff and students from https://www.ibm.com/products/ilog-cplex-optimization-studio.

The COBRA (COnstraint-Based Reconstruction and Analysis) Toolbox ***(18)*** is available at https://opencobra.github.io/cobratoolbox/stable/ and needs to be installed according to the respective tutorial. The COBRA toolbox is a community effort to collect computational scripts that allow performing quantitative predictions on the metabolism that include flux balance analysis, *in silico* knockouts, and robustness analysis, among others.

Detailend manuals for the these installations on Windows and macOS are provided at https://github.com/sysbiolux/rFASTCORMICS.

2.2. rFASTCORMICS

rFASTCORMICS for RNA-seq can be downloaded from https://wwwen.uni.lu/research/fstm/dlsm/research_areas/systems_biology/software/rfastcormics or https://github.com/sysbiolux/rFASTCORMICS and needs to be added to the working path of Matlab (see Note 1).

2.3. Input Model

Any genome-scale reconstruction can be taken as an input model as long as it is in a COBRA-compatible format. Note that different input models have different gene identifiers. For

example, the Recon family reconstructions use NCBI gene identifiers, whereas HMR 2 and Human 1 use Ensembl identifiers. Therefore, a gene conversion file (dictionary) is needed to map the genes from the expression data onto the metabolic model. For the following example, this file is provided with rFASTCORMICS and contains the official gene symbols as well as ENTREZ Gene IDs to allow mapping between the gene expression data and the input model. In general, the conversion file can be compiled using the Ensembl Biomart tool available at https://m.ensembl.org/biomart/martview/.

2.4. Expression data

Whereas FASTCORMICS was designed for microarray data, rFASTCORMICS was designed to take RNA-seq data as input. So far, rFASTCORMICS was tested with unfiltered FPKM transformed RNA-seq data. The data can be provided in any readable format for Matlab (Excel, text file).

3. Methods

A running example script (Molecular_Biology_Bintener_et_al_181121.m/.mlx) for the model reconstruction as described in detail here is provided with the download of rFASTCORMICS (https://github.com/sysbiolux/rFASTCORMICS).

3.1. Expression data input

1. Start Matlab and make sure all needed files and toolboxes are properly installed and added to the Matlab path, see Note 1.
2. Import the gene expression data into Matlab. Make sure to get three distinct variables
   - colnames: cell array with as many columns as samples
   - rownames: cell array with as many genes as measured
   - fpkm: measurements per gene and sample. The provided fpkm_BRCA_cancer.txt contains 10 breast cancer samples from the TCGA and fpkm_BRCA_control.txt 10 breast control samples and will be loaded separately into Matlab and merged afterwards.

   If your data comes in a single text format, you can use

   ```
   data = readtable('fpkm.txt' ,"ReadRowNames",true);
   ```

   As we have two different example files, we use

   ```
   data_cancer = readtable('fpkm_BRCA_cancer.txt',
   "ReadRowNames",true);

   data_control = readtable('fpkm_BRCA_control.txt',
   "ReadRowNames",true);
   ```

   to combine the data:

   ```
   data = [data_cancer, data_control];
   ```

   followed by

```
fpkm = table2array(data);

rownames = data.Properties.RowNames;

colnames = data.Properties.VariableNames;
```

3. With the data loaded, we can proceed with the discretization step that will determine if a gene is expressed (value of 1), not expressed (value of -1), or if the gene has an unknown expression status (value of 0). For this we use

```
discretized = discretize_FPKM(fpkm, colnames, 1);
```

The output will be an array with -1, 0, or 1 for each gene in each sample. Setting the third argument of the `discretized_FPKM` function to 1 will enable the generation of figures that show the gene expression distribution per sample as well as the discretization threshold (see Figure 1). These figures will be saved automatically in the Figures folder in your working directory. It is always advised to check the density plots, sometimes the density plot is too skewed and some slight adjustment needs to be made, see Note 3. In case this function is used in MATLAB's command window with the figure argument set to 1, it is important to let all the figures pop-up first, as closing these windows during the process will lead to an error.

[insert FIGURE 1]

Figure 1: Density plot showing the log2(FPKM) gene expression distribution of one sample. In black, the original gene expression is shown. In blue and red, the left and right half-Gaussians are shown, respectively. For more information on these steps, please refer to the material an methods section of rFASTCORMICS ***(14)****.*

3.2. Context-specific model reconstruction

1. Here we are using the human reconstruction Recon 2.04 that can be downloaded from https://www.vmh.life/files/reconstructions/Recon/2.04/Recon2.v04.mat .zip. Unzip the

compressed archive and add it to your working path in Matlab. Load the reconstruction with:

```
load Recon2.v04.mat
```

2. To reconstruct a context-specific model, the input model must be consistent. To achieve this, we use `fastcc` (included in the rFASTCORMICS download) on the loaded generic reconstruction whose variable is called modelR204 as follows:

```
A = fastcc_4_rfastcormics(modelR204, 1e-4,0);

consistent_model = removeRxns(modelR204,
modelR204.rxns(setdiff(1:numel(modelR204.rxns),A)))
```

   Here, `consistent_model` is the flux consistent part of the input reconstruction which means that every reaction can carry a non-zero flux

3. Before reconstructing the models based on the expression data via rFASTCORMICS, a few parameters need to be set:
   a. **dico**: a file that allows mapping gene identifiers between the input data and the model. An example is provided with the rFASTCORMICS download:

```
load dico_ML.mat
```

   b. **medium**: if the cells have been grown in a specific media it can be defined here to further constrain the model. The medium variable contains the identifiers of the metabolites that are allowed to be taken up by the model and hence are present in the medium.

```
load medium_example.mat
```

   c. **already_mapped_tag**: 1 if the data has previously been mapped to the model identifier and 0 if the mapping was not done yet.

```
already_mapped_tag = 0
```

   d. **consensus_proportion**: in case consensus models are built, this variable defines the fraction of samples required to have expressed a gene or to not

have expressed a gene to consider the gene and associated reaction to be expressed (1) and not expressed (-1), respectively. Genes that do not reach the consensus proportion are tagged with unknown expression status (0).

```
consensus_proportion = 0.9
```

e. **epsilon**: flux threshold to distinguish active from non-active reactions. Usually, epsilon is a small number close to 0 such as 0.0001 to avoid numerical issues with the solver.

```
epsilon = 1e-4
```

f. **biomass_rxn**: the biomass reaction ID in the model which will be used as an objective reaction by the `fastcormics_RNAseq` function.

```
biomass_rxn = {'biomass_reaction'};
```

g. **optional_settings** is an object that can contain the following fields: unpenalized systems:

- **unpenalizedSystems**: systems (also called pathways) for which the inclusion of reactions is not penalized. The inclusion of reactions that are associated with genes of unknown status is penalized to favour the inclusion of reactions under the control of expressed genes. It is however possible to define an unpenalized set of reactions that will not be forced in the reconstruction but would be favoured over non-core reactions.
- **func**: also called objective function. These are reactions that have to be included as they are optimized for during the analysis. Often the biomass and ATP maintenance are set as objective functions in a model.

- **not_medium_constrained**: metabolites that are not in the medium but, due to shortcomings of the model, they have to be taken up to allow the objective function(s) to carry a flux.
- **medium**: a list of metabolites identifiers that can be taken up by the model according to the medium composition.

```
unpenalizedSystems = {'Transport, endoplasmic reticular';
'Transport, extracellular'; 'Transport, golgi apparatus';
'Transport, mitochondrial'; 'Transport, peroxisomal';
'Transport, lysosomal'; 'Transport, nuclear'};

unpenalized =
consistent_model.rxns(ismember(consistent_model.subSystem
s,unpenalizedSystems));

optional_settings.unpenalized = unpenalized;

optional_settings.func =
{'DM_atp_c_';'biomass_reaction'};

not_medium_constrained = 'EX_tag_hs(e)';

optional_settings.not_medium_constrained =
not_medium_constrained;

optional_settings.medium = medium_example;
```

4. The command to reconstruct a context-specific model looks as follows:

```
[model_out, A_final] =
fastcormics_RNAseq(consistent_model, discretized,
rownames, dico, biomass_rxn, already_mapped_tag,
consensus_proportion, epsilon, optional_settings)
```

In general, the variable model_out is not saved because the generated models can take up a lot of memory. Hence, if many models are created, model_out should be replaced with ~. The variable A_final (or A_keep) contains the indices of reactions that have to be included in the model and can later be used to quickly reconstruct the context-specific model.

5. We can further distinguish between two scenarios during the model reconstruction process with rFASTCORMICS.

   We can reconstruct a model for each sample:

```
for i = 1:numel(colnames)

[~, A_keep{i}] = fastcormics_RNAseq(consistent_model,
discretized(:,i), rownames, dico, biomass_rxn,
already_mapped_tag, consensus_proportion, epsilon,
optional_settings)

end
```

   or we can reconstruct a consensus model for each condition. Only reactions that are active in at least 90% of the samples will be included for the reconstruction of the consensus model. As we have 10 cancer samples:

```
[model_cancer, A_final_cancer] =
fastcormics_RNAseq(consistent_model, discretized(:,1:10),
rownames, dico, biomass_rxn, already_mapped_tag,
consensus_proportion, epsilon, optional_settings)
```

   and 10 control samples:

```
[model_control, A_final_control] =
fastcormics_RNAseq(consistent_model,
discretized(:,11:20), rownames, dico, biomass_rxn,
```

```
already_mapped_tag, consensus_proportion, epsilon,
optional_settings)
```

6. The variable “A_keep” saves the indices of active reactions and can be saved in a more comparable array, called models_keep. Here, for the sample-specific models:

```
models_keep = zeros(numel(consistent_model.rxns),
numel(colnames));

for i=1:numel(colnames)

      models_keep(A_keep{i},i) = 1;

end
```

Similarly, for the consensus models that will be saved in a different variable called models_keep_consensus:

```
models_keep_consensus =
zeros(numel(consistent_model.rxns), 2);

models_keep_consensus(A_final_cancer,1) = 1

models_keep_consensus(A_final_control,2) = 1
```

3.3. Basic model and pathway analysis

Based on the models_keep variables, basic analyses based on the reaction presence can be performed.

1. Firstly, the similarity between the two models can be assessed via the Jaccard similarity index. Generally speaking, the Jaccard similarity index compares the intersection of two sets over their union:

$$\frac{M1 \cap M2}{M1 \cup M2}$$

In Matlab the Jaccard similarity score is calculated using:

```
J = squareform(pdist(models_keep','jaccard'));
```

The results can be represented in a clustergram. See Note 4 for the altcolor colorcode or use any other available colormap.

```
cgo_J = clustergram(1-J,...
  'RowLabels', colnames,...
  'ColumnLabels', colnames,...
  'ColumnLabelsRotate',270, ...
  'Cluster', 'all', ...
  'symmetric','False',...
  'Colormap', altcolor)
addTitle(cgo_J,{'Model similarity based on Jaccard distance','models_keep'})
```

Here, each model is represented on the x and the y axis and compared with every model. On the diagonal, a model is compared with itself which results in a similarity score of 1, represented by the dark colour. The darker the colour, the higher the similarity. Using the clustergram, we can determine models with a high similarity that also form clusters as can be observed by the dendrogram (see Figure 2).

[insert FIGURE 2]

Figure 2: Clustergram showing the model similarity for 10 breast cancer samples (BRCA_C) and 10 control samples (BRCA_H). Because each of the 20 models has been compared with each other, the darker diagonal line describes the case in which a model was compared with itself and is completely similar (score of 1). A relatively homogeneous cluster can be observed for BRCA_H whereas BRCA_C is more heterogeneous.

2. Pathway analysis can be performed by analysing the number of reactions present per pathway in the context-specific model in comparison to the input model. A score of 0.5 means that half of the reactions of the input model are present in the context-specific model for a given pathway. Within the model, pathways are often noted as “subsystems”. First, we will extract the number of reactions per pathway from the input model and compile the information in a table:

```
Pathways = table(unique(consistent_model.subSystems));

[pathways, ~, ub] = unique(consistent_model.subSystems);

path_counts = histc(ub, 1:length(pathways));

T = table(pathways, path_counts);

[I, ia, ib] = intersect(Pathways.Var1, T.pathways);

Pathways.consistent(ia) = T.path_counts(ib);

Pathways.Properties.VariableNames{1}='Pathways';
```

3. Then we can add the same pathway information for the consensus models:

```
[pathways, ~, ub] =

unique(consistent_model.subSystems(find(models_keep_conse

nsus(:,1))));

path_counts = histc(ub, 1:length(pathways));

T = table(pathways, path_counts);

[I, ia, ib] = intersect(Pathways.Pathways, T.pathways);

Pathways.Var2(ia) = T.path_counts(ib);

Pathways.Properties.VariableNames{3} =

'cancer_consensus';
```

```
[pathways, ~, ub] =
unique(consistent_model.subSystems(find(models_keep_conse
nsus(:,2))));
path_counts = histc(ub, 1:length(pathways));
T = table(pathways, path_counts);
[I, ia, ib] = intersect(Pathways.Pathways, T.pathways);
Pathways.Var2(ia) = T.path_counts(ib) ;
Pathways.Properties.VariableNames{4} =
'control_consensus';
```

and the sample-specific models:

```
for i=1:numel(colnames)
      [pathways, ~, ub] =
      unique(consistent_model.subSystems(find(models_keep(
      :,i))));
      path_counts = histc(ub, 1:length(pathways));
      T = table(pathways, path_counts);
      [I, ia, ib] = intersect(Pathways.Pathways,
      T.pathways);
      Pathways.Var2(ia) = T.path_counts(ib);
      Pathways.Properties.VariableNames{4+i} = colnames{i}
end
```

4. The pathway activity rates can also be represented by dividing each sample by the number of reactions in the consistent input model:

```
PathwayActivity = Pathways;
```

```
for i=3:size(PathwayActivity,2)

      PathwayActivity(:,i) =

      array2table(table2array(PathwayActivity(:,i))./table

      2array(PathwayActivity(:,2)));

end
```

5. If only 2 conditions or samples are compared, the pathway activity can be represented in a scatter plot. Here we will compare the consensus models saved in column 3 and 4 in the PathwayActivity variable. First, pathways with a difference higher than 20% will be identified for later plotting.

```
diff_idx = find(abs(table2array(PathwayActivity(:,3))-

table2array(PathwayActivity(:,4))) > 0.2)
```

6. All the pathways will be plotted in gray and surrounded by a black circle if the difference is higher than 20% and the pathway name will be added:

```
figure

hold on

scatter(table2array(PathwayActivity(:,3)),table2array(Pat

hwayActivity(:,4)),'filled','MarkerFaceColor',[0.9 0.9

0.9])

scatter(table2array(PathwayActivity(diff_idx,3)),table2ar

ray(PathwayActivity(diff_idx,4)), 'black')

ylabel('cancer consensus model')

xlabel('control consensus model')

title('Pathway presence rate in the consensus models')

line([0 1], [0,1],'Color','k')

line([0 0.8], [0.2,1],'Color','k','LineStyle','--')

line([0.2 1], [0,0.8],'Color','k','LineStyle','--')
```

```
legend({'All pathways','>20%'},"Location","best")

text(table2array(PathwayActivity(diff_idx,3)),table2array
(PathwayActivity(diff_idx,4)),
PathwayActivity.Pathways(diff_idx))
```

[insert FIGURE 3]

Figure 3: Pathway presence rate in the consensus models. Pathways whose activity difference is higher than 20% in either model are highlighted with a dark circle.

7. Alternatively, if several samples are compared, a heatmap can be useful. Here the consensus models are also included for completeness.

```
cgo =
clustergram(table2array(PathwayActivity(:,3:end)),...
  'RowLabels', PathwayActivity.Pathways,...
  'ColumnLabels',
PathwayActivity.Properties.VariableNames(3:end),...
  'ColumnLabelsRotate',270, ...
  'Cluster', 'column', ...
  'symmetric','False',...
  'Colormap', altcolor)
h = plot(cgo); set(h,'TickLabelInterpreter','none');
colorbar(h)
title(h,'Pathway activity')
```

[insert FIGURE 4]

Figure 4: Pathway presence rates for all models in a heatmap clustergram. Note that the number of shown pathways was reduced to 20 for a better representation.

3.4. In silico gene deletion and essential genes

After the context-specific model reconstruction, we can proceed with the *in silico* gene deletion. To this aim, an objective function needs to be set for the model. Generally, the biomass_reaction is used to describe quickly proliferating cells such as cancer cells. For healthy cells, the ATP demand reaction is considered here. Then, for each gene in the model, the corresponding reaction fluxes are set to zero and the flux through the objective function is calculated.

1. The objective function is set via the changeObjective function from the COBRA toolbox. To set the biomass reaction as objective function:

```
model_out =
changeObjective(consistent_model,'biomass_reaction')
```

   The set the ATP demand as objective function:

```
model_out = changeObjective(consistent_model,'DM_atp_c_')
```

   It can be verified with with the checkObjective function from the COBRA toolbox

```
checkObjective(model_out)
```

2. To perform the in silico gene deletion for every model for both objective functions, we first need to initialize the solver

```
changeCobraSolver('ibm_cplex')
```

3. Then we can run the single gene deletion adapted from the COBRA toolbox for the sample-specific models:

```
for i=1:size(models_keep,2)
```

```
ind = find(~cellfun(@isempty,
regexp(consistent_model.rxns, 'DM_atp_c_')));
model_out =
removeRxns(consistent_model,consistent_model.rxns(se
tdiff(1:numel(consistent_model.rxns),find(models_kee
p(:,i)))));
model_out =
changeObjective(model_out,consistent_model.rxns(ind)
);
[grRatio, grRateKO, grRateWT, hasEffect, delRxns,
fluxSolution] =
singleGeneDeletion_rFASTCORMICS(model_out,'FBA',[],0
,1);
grRatio_ATP(:,i)= grRatio;
grRateKO_ATP(:,i)    = grRateKO;
grRateWT_ATP(:,i)    = grRateWT;


ind = find(~cellfun(@isempty,
regexp(consistent_model.rxns, 'biomass_reaction')));
model_out =
removeRxns(consistent_model,consistent_model.rxns(se
tdiff(1:numel(consistent_model.rxns),find(models_kee
p(:,i)))));
```

```
        model_out =
        changeObjective(model_out,consistent_model.rxns(ind)
        );

        [grRatio, grRateKO, grRateWT, hasEffect, delRxns,
        fluxSolution, genelist] =
        singleGeneDeletion_rFASTCORMICS(model_out,'FBA',[],0
        ,1);

        grRatio_biomass(:,i)     = grRatio;

        grRateKO_biomass(:,i)    = grRateKO;

        grRateWT_biomass(:,i)    = grRateWT;

    end
```

The main outputs from this part are grRatio_ATP and grRatio_biomass. For both variables, the rows represent genes and the columns represent samples. grRatio stands for growth ratio and gives information on how much the gene deletion affects the objective function. Further information can be taken from the original COBRA documentation.

4. For the next analysis, it is easier to convert de geneList of ENTREZ IDs into Gene symbols using the dictionary:

```
[B, ia, ib] = intersect(genelist,dico.ENTREZ)

genelist(ia, 2) =   dico.SYMBOL(ib)
```

5. For the determination of essential cancer genes, we use a growth ratio of 0.5 as a cut-off. Additionally, the gene deletion should affect at least half of the samples. For the control samples, a more stringent cut-off is used; the ATP maintenance should only be minimally affected and not be below 0.9 in at least 10% of the control samples. This means that a gene whose deletion reduces the biomass production below 50% in at

least half of the samples compared to the wild type are considered to be essential. We can use the following code to find the essential genes for the cancer and control samples:

```
essential_cancer_genes = genelist
(sum(grRatio_biomass(:,1:10) < 0.5,2) > 5,2)

essential_control_genes = genelist
(sum(grRatio_ATP(:,11:20) < 0.9,2) > 1,2)
```

6. To find genes essential in cancer only:

```
cancer_specific_genes = setdiff(essential_cancer_genes,
essential_control_genes)
```

As a follow up for the predicted essential genes, an enrichment test can be performed by using experimentally validated essential genes from CRISPR Cas9 data with a hypergeometric test. Additionally, these genes can be looked up in several databases such as the DrugBank to potentially find drugs that allow inhibiting the protein product from the essential gene.

3.5. In silico drug deletion and drugs for repurposing

Instead of performing gene deletion, one can directly estimate the effect of a drug if the information on the gene-drug relation is available. For example, if a drug inhibits one or multiple gene products, these genes can be knocked down and the effect on the objective function can be measured. Therefore, we have modified the single gene deletion script to take as input a model and a list of drugs to be simulated.

1. Before starting the drug deletion, we need to define a list of drugs whose effect will be evaluated on the models. To this aim, we can use the provided gene-drug list called GeneDrugRelations to extract a list of drugs from the DrugBank. 1175 unique inhibiting drugs can be tested:

```
load GeneDrugRelations.mat
```

```
DrugList = unique(GeneDrugRelations.DrugName)
```

2. The DrugDeletion script is used in the same way as the singleGeneDeletion for rFASTCORMICS. First, the model is reconstructed from the models_keep variable, then the objective function is set and the analysis is run:

```
for i=1:size(models_keep,2)

    ind = find(~cellfun(@isempty,
    regexp(consistent_model.rxns, 'DM_atp_c_')));

    model_out =
    removeRxns(consistent_model,consistent_model.rxns(se
    tdiff(1:numel(consistent_model.rxns),find(models_kee
    p(:,i)))));

    model_out =
    changeObjective(model_out,consistent_model.rxns(ind)
    );

    [grRatio, grRateKO, grRateWT] =
    DrugDeletion(model_out,'FBA',DrugList);

    Drug_grRatio_ATP(:,i)     = grRatio;

    Drug_grRateKO_ATP(:,i)    = grRateKO;

    Drug_grRateWT_ATP(:,i)    = grRateWT;


    ind = find(~cellfun(@isempty,
    regexp(consistent_model.rxns, 'biomass_reaction')));

    model_out =
    removeRxns(consistent_model,consistent_model.rxns(se
```

```
        tdiff(1:numel(consistent_model.rxns),find(models_kee
        p(:,i)))));

        model_out =
        changeObjective(model_out,consistent_model.rxns(ind)
        );

        [grRatio, grRateKO, grRateWT] =
        DrugDeletion(model_out,'FBA',DrugList);

        Drug_grRatio_biomass(:,i)    = grRatio;

        Drug_grRateKO_biomass(:,i)   = grRateKO;

        Drug_grRateWT_biomass(:,i)   = grRateWT;

    end
```

3. Finding drugs for repurposing in cancer is similar to finding cancer-specific essential genes:

```
    cancer_drugs = DrugList(sum(Drug_grRatio_biomass(:,1:10)
    < 0.5,2) > 5)

    control_drugs = DrugList(sum(Drug_grRatio_ATP(:,11:20) <
    0.9,2) > 1)

    cancer_specific_drugs = setdiff(cancer_drugs,
    control_drugs)
```

Conclusions: The hereby presented workflow for the reconstruction of context-specific models coupled with a drug repurposing workflow allows for the discovery of interesting drugs for repurposing. After the drug prediction, additional information on the drugs should be sought to determine their feasibility in *in vitro* validation experiments. For example, the

availability, solubility, and concentration of the compounds are of high importance to perform sensible follow-up experiments.

Additionally, by comparing models from different conditions such as therapy-resistant and therapy-responding cells, metabolic mechanisms and rewiring strategies that play a role in the resistance can be discovered and possibly overcome in the future.

4. Notes

1. To add a folder to your Matlab working path use `addpath('directory')`. To add all subfolder from the current folder use `addpath(genpath(pwd))`.
2. If you get an error “Error: The input was too complicated or too big for MATLAB to parse” add `feature astheightlimit 2000` to the beginning of your code
3. If the peak of the distribution is shifted towards the left side, i.e. if the leftmost peak is higher than the rightmost curve, use the provided `discretize_FPKM_skewed` function.
4. Altcolor code:

```
altcolor= [255 255 255;255 204 204; 255 153 153; 255 102 102; 255 51 51; 255 0 0; 204 0 0; 152 0 0; 102 0 0; 51 0 0]/255;
```

5. References

1. Nosengo N (2016) Can you teach old drugs new tricks? Nature 534:314–316
2. Landtsheer S De, Trairatphisan P, Lucarelli P, et al (2017) FALCON: a toolbox for the fast contextualization of logical networks.
3. Zoraghi R and Reiner NE (2013) Protein interaction networks as starting points to identify novel antimicrobial drug targets. Curr Opin Microbiol 16:566–572
4. Badkas A, Landtsheer S De, and Sauter T (2020) Topological network measures for drug repositioning. Brief Bioinform 00:1–13
5. Mistro G Del, Lucarelli P, Müller I, et al (2018) Systemic network analysis identifies XIAP and IκBα as potential drug targets in TRAIL resistant BRAF mutated melanoma. npj Syst Biol Appl 4:39

6. Folger O, Jerby L, Frezza C, et al (2011) Predicting selective drug targets in cancer through metabolic networks. Mol Syst Biol 7:501

7. Frezza C, Zheng L, Folger O, et al (2011) Haem oxygenase is synthetically lethal with the tumour suppressor fumarate hydratase. Nature 477:225–228

8. Turanli B, Zhang C, Kim W, et al (2019) Discovery of therapeutic agents for prostate cancer using genome-scale metabolic modeling and drug repositioning. 42:386–396

9. Shlomi T, Cabili MN, and Ruppin E (2009) Predicting metabolic biomarkers of human inborn errors of metabolism. Mol Syst Biol 5:263

10. Resendis-Antonio O, Checa A, and Encarnación S (2010) Modeling core metabolism in cancer cells: Surveying the topology underlying the warburg effect. PLoS One 5:e12383

11. Yizhak K, Gaude E, Dévédec S Le, et al (2014) Phenotype-based cell-specific metabolic modeling reveals metabolic liabilities of cancer. Elife 3:1–23

12. Agren R, Mardinoglu A, Asplund A, et al (2014) Identification of anticancer drugs for hepatocellular carcinoma through personalized genome-scale metabolic modeling. Mol Syst Biol 10:721

13. Diener C and Resendis-Antonio O (2016) Personalized Prediction of Proliferation Rates and Metabolic Liabilities in Cancer Biopsies. Front Physiol 7:1–11

14. Pacheco MP, Bintener T, Ternes D, et al (2019) Identifying and targeting cancer-specific metabolism with network-based drug target prediction. 43:98–106

15. Vlassis N, Pacheco MP, and Sauter T (2014) Fast Reconstruction of Compact Context-Specific Metabolic Network Models. PLoS Comput Biol 10:e1003424

16. Pacheco MP, John E, Kaoma T, et al (2015) Integrated metabolic modelling reveals cell-type specific epigenetic control points of the macrophage metabolic network. BMC

Genomics 16:809

17. Bintener T, Pacheco MP, Kulms D, et al Melanoma in silico drug target prediction across patients and cell lines using metabolic modelling approaches. Under preparation.

18. Heirendt L, Arreckx S, Pfau T, et al (2019) Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0. Nat Protoc 14:639–702