`RBApy.estim` package

Ana Bulovic

August 2018

Contents

Chapter 1

Parameter estimation

1.1 Percentage of protein per compartment

In RBA, the amount of protein per compartment is represented by the concentration of amino acids in unit of $\frac{mmol.AA}{gCDW}$. This amount is later used in the optimization procedure as an inequality constraint limiting how much of protein produced for metabolic and process requirements can fit into each compartment. With the change in growth rate, bacteria change their size, and with it the amount of protein, and so we model these concentrations as functions of the growth rate. Because the amount of protein per cell has been shown to vary linearly with the growth rate, we initially assume that the estimated functions are either linear or constant. Within the RBApy.estim package, we offer an estimation procedure that can estimate the best linear fit of percentage of protein in each compartment, given that the user provides proteomics data and assignment of proteins to compartments. Because we assume the data will be provided for experiments of different growth rates, we also require a file in which the experiment ID is matched to a measured growth rate.

The estimation procedure expects the files to be in either CSV or Excel format. Three files are necessary: (1) a file describing the experiments, by linking the IDs to the growth rates (see Fig 1.1), (2) a proteomics file which needs to have all the columns with abundances (in any unit) corresponding to the experiments specified in the first file, and cellular localization column (see 1.1)and (3) the output file where the results of the fit will be stored. Also, the procedure allows for mapping of compartments, as one could choose to do in the case Uniprot offers two different compartments, Inner membrane, and Cell inner membrane, which the user would want to map onto one compartment.

The estimates of percentages of protein per compartment necessarily all need to be non-negative and need to sum to one. As this constraint cannot be ensured with ordinary least-squares, we transform the original least squares problem

Experiment file

Experiment_ID	Growth rate
glc_batch	0.58
gal_batch	0.4
fum_batch	0.3

ProteinData file

Protein	Location	glc_batch	gal_batch	fum_batch
p1	cytosol	3000	2510	0
p2	cytosol	0	0	4000
p3	periplasm	400	902	200
p4	cytosol	10800	12000	8000

Figure 1.1: Simple example of files needed to run the procedure to estimate the percentage of protein per compartment. The experiment IDs need to be matched to the corresponding growth rates and the protein data file needs to have columns that correspond to experiments. Protein data file needs to have a location column. The names of all the files and columns can be specified in the configuration file.

$$\begin{array}{ll} \min\limits_{x} & \|Ax - y\|_2 \\ \text{s.t.} & Ax - y \geq 0 \end{array} \tag{1.1}$$

into a quadratic programming problem:

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2}x^T Q x + c^T x \\ \text{s.t.} & Gx \leq h \end{array} \tag{1.2}$$

where $Q = A^T A$, $c = -A^T y$, $G = -A$, $h = y$.

The script to run this estimation is called `prot_per_compartment.py`, and the accompanying configuration file is `prot_per_compartment.cfg`.

1.2 Percentage of nonenzymatic protein per compartment

RBA model will rarely cover all cellular processes. Therefore, there will always be a portion of the proteome which will not be represented in the model. However, this portion still serves some cellular function. When we limit the amount of protein per each compartment, we need to take into an account that not all proteins can serve some function within the model, but that they in reallity take up only a portion of all available protein per compartment. Therefore, we estimate a percentage of nonenzymatic protein per each compartment as a function of the growth rate (or as a constant).

The algorithm and the configuration for the estimation are the same as in Section 1.1, the only difference being that, alongside cellular localization, proteins need to be assigned a functional category. One additional file is needed,

where the functional categories are mapped assigned True or False values: True if they are considered to be enzymatic, False if they are not. Enzymatic are those categories which are already functionally represented in the RBA model, such as metabolic enzymes or translation machinery, and nonenzymatic those which are missing from the model, such as for example DNA repair and signalling pathways.

The script to run this estimation is called `nonenz_per_compartment.py`, and the accompanying configuration file is `nonenz_per_compartment.cfg`.

1.3 Apparent catalytic rates

An apparent catalytic rate describes not the maximal, but the situation dependent catalytic rate. In case no data is available for parameter estimation, RBApy assumes an equal default apparent catalytic rate for all enzymes of $10s^{-1}$. In case that the user can provide proteomics and fluxomics data of sufficiently similar experimental conditions (regarding the strain, medium, culture type - batch vs. chemostat, growth rate, etc.), she can use a simple k_app estimation procedure provided in the `RBApy.estim` package. Data needed for estimation is:

- Genome-scale metabolic reconstruction
- Fluxomics data (in $\frac{mmol}{gCDW \times h}$)
- Proteomics data (can be provided in two units - copies per cell or $\frac{mmol}{gCDW}$. In the case it is provided in copies per cell, convert option of the configuration file needs to be set to True, and `dry_weight_per_cell` needs to be updated to a correct value)

.

The fluxomics data is used to restrain an FBA model (the same genome-scale metabolic model used for the generation of the RBA model). The simulation of the thus restrained FBA model yields a flux distribution to be used in the subsequent estimation. The proteomics data is used to estimate enzyme abundances. Since the same protein can be used for construction of different enzymes, we divide the protein abundance between the corresponding enzymes. The apparent catalytic rates are computed as a solution the the least-squares problem where the residues are defines as $r_i = \nu_i - k_{app_i} \times E_i$, where ν_i, E_i and k_{app_i} are the flux amount, enzyme concentration and an apparent catalytic rate of the i^{th} reaction. The solution for individual apparent catalytic rates then becomes: $k_{app_i} = \frac{\nu_i}{E_i}$.

The script to run this estimation is called `kapp.py`, and the accompanying configuration file is `kapp.cfg`.