# Running an Existing Model

## Installing a Model from Elsewhere

1. Download the model from the GitHub page.

## Running the Model

# Building an RBA Model

My reasoning for each step is explained in the indented paragraph following it, where needed.

1. Build or download the genome-scale metabolic (GSM) model of your chosen organism. Make sure it has been adequately tested beforehand (e.g., via MEMOTE). Put it in the build\_model/input folder.
2. Assemble the PROTEIN\_stoich\_curation.xlsx sheet. Using Uniprot or another database, find all genes in your model (and, if alternative splicing is a factor, all proteins that arise from it) that encode enzymes (or subunits of them), transporters, or ribosomes; then, find all subcellular compartments (e.g., mitochondria, ER, peroxisome) where they are present. Create versions of each protein for each compartment they appear in, noting their location under subloc\_assigned with your model’s notation for that compartment (e.g., if “\_m” represents the mitochondria, put “m” in subloc\_assigned for proteins translated there). Assign their cofactors (using the cofactor\_manual column for easier typing or directly into the cofactor\_stoich column). Input their names, Uniprot IDs, amino acid sequences, and the compartment they are translated in (under translation\_loc).
3. For all proteins not under PROTEIN\_stoich\_curation.xlsx but in your organism, use them to build a “dummy” protein under PROTEIN\_dummy\_prot\_calc.xlsx. If you have protein abundance data and sequences, input them into “example\_data” and let the sheet “example\_calculation\_procedure” recalculate; copy the sheet’s contents into Sheet1. If not, then find another way to create a protein that represents the typical contents of proteins not individually included in your model.
   1. Why: To accurately reflect the nutrient demands placed on the organism, the costs of producing proteins not involved in enzymes, transporters, or ribosomes (e.g., DNA polymerases) must be accounted for. While every such protein (henceforth dubbed “nonmodel proteins”) could be given a constraint forcing it to comprise at least a specific percentage of the proteome determined by its abundance *in vivo*, this would add many more variables for the solver to consider, making it much slower while underrepresenting the need for nonmodeled proteins in organisms without abundance data or sequences for every protein. Aggregating nonmodel proteins into a dummy protein simplifies solving and model building, since not every protein must be explicitly modeled. A dummy protein could thus be constructed even from amino acid abundance data alone, even if this would hardly be ideal.
   2. In this case, the dummy protein’s amino acid composition is determined by finding the molar fraction of each amino acid among all nonmodel proteins. The spreadsheet divides the mass fraction of the proteome comprised of each nonmodel protein by its molar mass, to find a molar fraction. It finds the percentage of each protein’s sequence comprised of each amino acid and multiplies it by that protein’s molar fraction to give the molar fraction of the nonmodel proteome comprised of each amino acid from that protein. These values are then summed across all nonmodel proteins for each amino acid, to determine the amino acid composition of the dummy protein.
   3. The dummy protein’s length is the median of all nonmodeled protein lengths, to ensure the protein is long enough to represent the nonmodel proteome’s amino acid composition with minimal rounding.
   4. An additional dummy protein 'BIO-protdummyunidentified' could be included to account for proteomics measurements that do not identify all proteins present (in this case, 0.3% of proteins by mass). These proteins are modeled using a dummy protein with a length equal to the median length of all proteins in the organism’s proteome, and its amino acid composition is the average of such proteins.
      1. How to do this: In phenotype.txt, add the constraint Equation dummyUnidentified; dummyUnidentified.. v('BIOSYN-PROTDUMMYUNIDENTIFIED') =e= 0.003 \* v('BIOSYN-PROTTOBIO'); replace 0.003 with whatever fraction of the proteome you want this protein to comprise.
      2. Why: The unidentified 0.3% could stem from measurement errors or other proteins not listed in the proteomics dataset (whether due to only being present in undetectably low quantities, fragmentation during measurement making identification difficult or impossible, etc.), though it is unclear which is the case and if so, what proteins it would comprise. Thus, like in scRBA, the median length and average amino acid composition of all proteins is used as to not bias it towards any subset of the proteome when there is no evidence that supports doing so.
4. In ENZYME\_stoich\_curation.xlsx, make a row with each reaction’s name (“rxn\_src” column). If that reaction occurs spontaneously or the proteins needed for it are unknown, put “SPONT” or “UNKNOWN” respectively in its “enz” column and “zeroCost” in its “protein\_stoich” column. Otherwise, make a row for every combination of proteins (and how many of them are needed) that can catalyze it (“protein\_stoich” column) and assign each row a unique enzyme name (“enz” column).
5. In RNA\_stoich.xlsx, fill out a row for each RNA molecule in the model with its name (“RNAid”) and sequence. All other info is optional.
6. In RIBOSOME\_nucleus.xlsx and RIBOSOME\_mitochondria.xlsx, add rows for the rRNAs and proteins – along with their respective weights – for the respective mitochondria types, if present in your organism.
7. In BIOMASS\_RBA.xlsx, each biomass component has its own section with columns like “Metabolite”, “Coeff” (stoichiometric coefficient from biomass equation), “Formula”, and “MW” (in g/mol). Replace the entries in these with the respective biomass components from your organism; do the same for the GAM entries in cells I18 and below.
8. Run build\_model/A01\_build\_excel\_stoich\_for\_GAMS.ipynb; change the line containing “**model = cobra.io.load\_json\_model('./input/**” to include the name of your GSM model (e.g., if using **model.json**, it would say “**model = cobra.io.load\_json\_model('./input/model.json')**”)
9. Run build\_model/A02\_build\_GAMS\_Sij\_and\_fluxBounds.ipynb then build\_model/A03\_build\_GAMS\_RBA\_constraints.ipynb.
10. Copy the “build\_model/model” folder over to “/GAMS”; replace the existing folder.
11. Run your model to ensure it works. See [Running the Model](#_Running_the_Model) for how to do that.
12. Build your media file.
13. Build your phenotype.txt file. Add NGAM.

## kapp calculations

Do this step after determining all other parameters/constraints, to ensure values can be easily reproduced.

Check min\_flux\_violation/min\_flux\_violation.prosynSlack.txt for any slack variables applied to protein production. These variables are meant to account for cases where a protein’s measurements may be inaccurate (e.g., due to excessive rounding of numbers) or paralogs of it may have not been identified. The solver tries to minimize their usage, but checking your dataset for other proteins that should be modeled would also help. Infeasibility may stem from such cases (e.g., ribosome subunits with unidentified paralogs).