RBA pipeline

S. Fischer - Biosys - MAIAGE

February 1, 2017

Contents

1 Description of workflow

Figure 1 shows the global workflow of the pipeline. It contains three parts:

1. **preRBA**: Parsing of biological data into RBA compatible XML files. Parts of the process are semi-automated, meaning the user is needed to help solve ambiguous annotations.

2. **buildRBA**: XML files are transformed into matrices used by the RBA solver and stored into optimized matlab structures.

3. **solveRBA**: matrices are used to compute the optimal resource allocation.

As a first step, this workflow should run for any prokaryote.

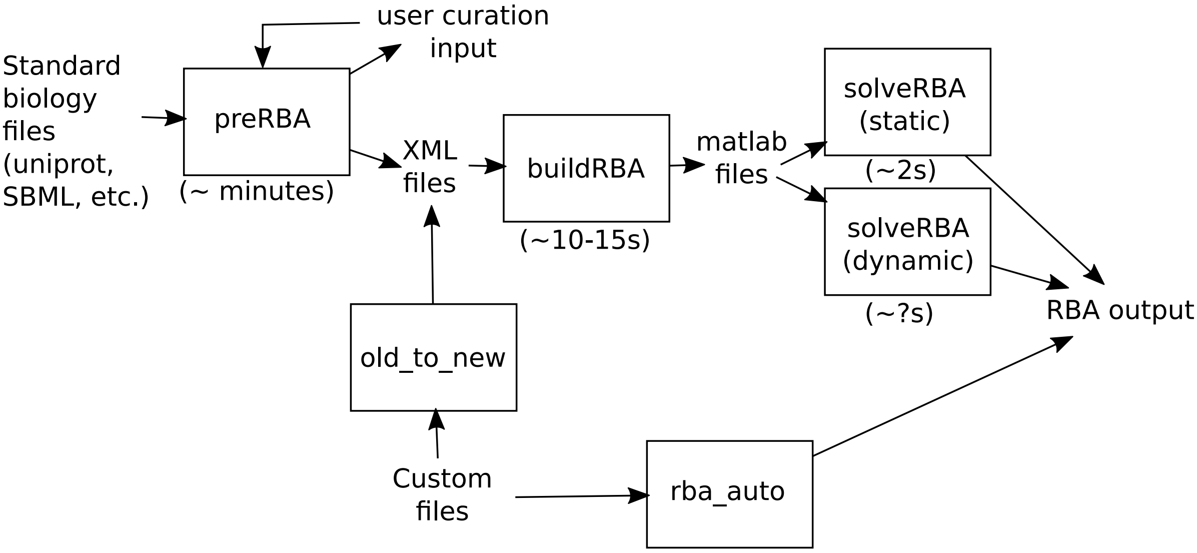

Figure 1: Workflow of pipeline. The lower part shows how compatibility between the workflow and the former version of RBA were maintained (used for consistency checks).

The conversion of biological data is the heaviest part. Some of the effort is on the user's side: the files they provide need to follow some specifications. On the other hand, we provide scripts that will automatically transform standard files into standard RBA XML files. The process is semi-automatic: everything works well as long as the annotations are not ambiguous. When an annotation is ambiguous, the user will be asked for help in a (hopefully) user-friendly way. Everything that is input by the user is stored so the user does not have to provide the same information twice.

Once the first step has been taken, the user will have ready-to-use RBA input files. Because user curated information is stored, re-runnig the workflow is way quicker than at first use. The user may modify some information at any time, the workflow will detect it and only run specific subparts and modify files where needed.

2 preRBA: converting biological data

In order to get the workflow working, the user has to provide an SBML file describing the metabolism of their organism and a Uniprot file describing the proteins of their organism. This section lists the requirements that these files need to meet and how the user's help will be prompted while parsing them.

2.1 SBML: extraction of metabolism and enzyme information

2.1.1 Requirements

Requirement 2.1. File should contain *only* metabolic reactions. User should remove the biomass reaction and reactions used to assemble non-metabolites (*e.g.* proteins, rnas, etc.).

Requirement 2.2. Every reaction should have a note containing a PROTEIN_ASSOCIATION in standard form. This field should describe the composition of the enzymatic complex catalyzing the reaction in terms of proteins.

Requirement 2.3. Name of proteins used in the PROTEIN_ASSOCIATION field should correspond to the names listed in the `Gene names (ordered locus )` field of the uniprot file.

Requirement 2.4. All proteins listed in the PROTEIN_ASSOCIATION field should be referenced in the uniprot file.

2.1.2 User interactions

Everything is automated, no help needed.

2.2 Uniprot: extraction of protein information

2.2.1 Requirements

Requirement 2.5. Uniprot file should be standard uniprot in TSV format.

2.2.2 User interactions

Cofactor stoichiometry Cofactor stoichiometry (and sometimes names) can be ambiguous. When necessary, user is prompted to read the `Cofactor` field and provide stoichiometry of cofactors. In order to limit interactions, note that we use the following rules to parse cofactor information:

- If field is empty, we assume there is no cofactor.
- If we find exactly one name and its associated CHEBI identifier, and there is only one occurrence of the keyword `Binds`, we assume stoichiometry is the number the follows `Binds`.

- If we find exactly one name and its associated CHEBI identifier, and there is no stoichiometry information using keyword `Binds`, we assume stoichiometry is 1.
- In any other case, user is asked for help.

Subunit structure Subunit structure is often ambiguous. When necessary, user is prompted to type how many copies of the proteins are usually found in the enzymatic complex. In order to limit interactions, note that we use the following rules to parse cofactor information:

- If field is empty, we assume there is one subunit in the complex.
- If field contains exactly one occurence of the form "*prefix*mer", we look at the prefix. If prefix is mono or heterodi, we assume stoichiomery is one. If prefix is homodi, homotri, homotera, homopenta, homohexa, hepta, homoocta, homodeca, homododeca, we assume the number of subunits corresponds to the prefix.
- In any other case, user is asked for help.