Quick Introduction to the rsbml Package

Michael Lawrence

October 31, 2011

Introduction

The *rsbml* package supports the import, validation, and export of SBML data. It is similar in purpose to the *SBMLR* package [Radivoyevitch, 2004], except *rsbml* relies on the external library *libsbml* [Bornstein, 2006] for its speed and reliability. With *rsbml*, the user may import an SBML model as a *graph* object or an S4-based Document Object Model (DOM). Parsed models may be checked for consistency. *rsbml* is capable of checking models for consistency and exporting SBML to a character vector or directly to a file. This vignette provides a quick introduction to *rsbml*, demonstrating how to import an SBML model, manipulate the model in R as a graph or S4 object, validate the model, and export the model.

Importing an SBML document

Most users will begin an rsbml session by importing an SBML file into R. In the example below, we load an SBML file describing the glycolysis pathway. It is also possible to parse an R character vector instead of an external file.

```
> library(rsbml)
> doc <- rsbml_read(system.file("sbml", "GlycolysisLayout.xml", package = "rsbml"), dom = FA
```

If errors are encountered, the function throws an error along with warnings describing the specific problem(s) with the document. Otherwise, the result is an opaque object referring to a low-level libsbml data structure. From here, the user currently has two options for accessing the data: as an S4 object conforming to the SBML document object model or as a Bioconductor graph object.

The S4 object representation

Converting the opaque libsbml parse result to an S4 object is simple:

```
> dom <- rsbml_dom(doc)
```

The result contains all of the information from the SBML. Methods exist for accessing every element of the SBML specification (up to L2V1). Here is how one would retrieve all of the species ID's from the SBML model:

```
> sapply(species(model(dom)), id)
```

```
                   Glucose         Glucose_6_phosphate
                 "Glucose"       "Glucose_6_phosphate"
       Fructose_6_phosphate   Fructose_1_6_bisphosphate
     "Fructose_6_phosphate" "Fructose_1_6_bisphosphate"
  Dihydroxyacetonephosphate   Glyceraldehyd_3_phosphate
"Dihydroxyacetonephosphate" "Glyceraldehyd_3_phosphate"
   _1_3_Bisphosphoglycerate         _3_Phosphoglycerate
 "_1_3_Bisphosphoglycerate"       "_3_Phosphoglycerate"
        _2_Phosphoglycerate         Phosphoenolpyruvate
      "_2_Phosphoglycerate"       "Phosphoenolpyruvate"
                  Pyruvate                         ATP
                "Pyruvate"                       "ATP"
                       ADP                          H_
                     "ADP"                        "H_"
                      NAD_                       NADH
                    "NAD_"                     "NADH"
                       H2O                          Pi
                     "H2O"                        "Pi"
```

The user may also modify every part of the model. Note that very little validation is performed in response to modifications. See **??** for a guide to checking SBML models for consistency problems.

The graph representation

All SBML models have an implicit graphical structure. To extract this into a Bioconductor graph object, type the following:

```
> g <- rsbml_graph(doc)
> graph::nodes(g)
```

```
 [1] "Glucose"                    "Glucose_6_phosphate"
 [3] "Fructose_6_phosphate"       "Fructose_1_6_bisphosphate"
 [5] "Dihydroxyacetonephosphate"  "Glyceraldehyd_3_phosphate"
 [7] "_1_3_Bisphosphoglycerate"   "_3_Phosphoglycerate"
 [9] "_2_Phosphoglycerate"        "Phosphoenolpyruvate"
[11] "Pyruvate"                   "ATP"
[13] "ADP"                        "H_"
[15] "NAD_"                       "NADH"
[17] "H2O"                        "Pi"
```

```
[19] "Hexokinase"                  "Phosphoglucoseisomerase"
[21] "Phosphofructokinase"         "Aldolase"
[23] "triose_phosphate_isomerase" "GAP_Dehydrogenase"
[25] "Phosphoglyceratekinase"      "Phosphoglyceratemutase"
[27] "Enolase"                     "Pyruvatekinase"
```

Note that the list of node ID's contains all of the species ID's retrieved from the S4 object above. The additional ID's are for the reaction nodes. At this point, the graph can be passed to other packages, such as RBGL, Rgraphviz, etc.

Checking SBML models for consistency

The SBML specification provides many complex rules that ensure an SBML model is internally consistent. The following is an example of checking a document against those rules.

```
> rsbml_check(doc)
```

```
[1] TRUE
```

If there were any problems with the document, they would be communicated as R console messages, warnings or errors. If you would like to compute on the problems, you will need to catch the conditions thrown by the `rsbml_read` function. There are three different types of conditions: *SBMLFatal*, *SBMLError* and *SBMLWarning*. *SBMLFatal* and *SBMLError* both inherit from *error*, while *SBMLWarning* inherits from *warning*. The following code catches the error resulting from a non-existent file and prints the message:

```
> tryCatch(rsbml_read("non-existent-file.xml"),
+          error = function(err) err$msg)
```

```
NULL
```

Writing SBML documents

After creating/manipulating SBML objects in R, the result may be translated back to XML in two different ways: directly to a file or to an R character vector. This example is of the latter:

```
> xml <- rsbml_xml(doc)
```

The result is a string XML representation of the SBML model.

More information

For more details, please see the online help for the rsbml package. If you encounter problems, please email lawremi at the domain iastate.edu.

References

B. Bornstein. libsbml, 2006. URL http://sbml.org/software/libsbml/.

T. Radivoyevitch. A two-way interface between limited systems biology markup language and R. *BMC Bioinformatics*, 5(1):190, 2004. ISSN 1471-2105. doi: 10.1186/1471-2105-5-190. URL http://www.biomedcentral.com/1471-2105/5/190.