Systems Biology Markup Language for R

SBMLR Version 1.16 (2/1/2005)

Tomas Radivoyevitch
Department of Epidemiology and Biostatistics
Case Western Reserve University
Cleveland, Ohio 44106
Email: radivot@hal.cwru.edu
Website: http://epbi-radivot.cwru.edu/

Systems biology markup language (SBML) (http://sbml.org/) is a standard for representing dynamical systems of biological interest (Finney and Hucka, 2003; Hucka *et al.*, 2003). SBMLR (Radivoyevitch, 2004, 2005) connects biochemical systems represented in SBML to R (Ihaka and Gentleman, 1996), the base language of Bioconductor (Gentleman *et al.*, 2004) tools used by biostatisticians. By connecting SBML to Bioconductor, SBMLR connects systems biology to biostatistics.

Package Contents

The SBMLR package defines an SBML-like R model structure and 4 functions: `write.sbml` for exporting SBMLR models to SBML level 2 models, `read.SBML` for importing a limited range of SBML level 2 models (events, algebraic rules and function definitions are not currently implemented in SBMLR), `getIncidenceMatrix` for creating stoichiometic matrices given the reactant and product lists in a specific SBMLR model, and `fderiv` as a generic state derivative function that gets passed to `lsoda` of the ODESOLVE package in SBMLR simulations. These 4 functions fall into two categories: model sharing functions (`write.SBML` and `read.SBML`) and model using functions (`getIncidenceMatrix` and `fderiv`). These SBMLR functions, and the objects which they act open, are illustrated in Figure 1.

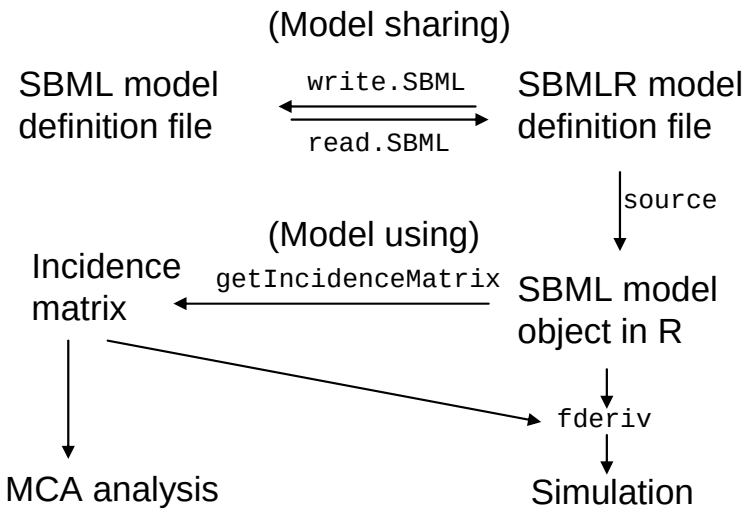

Figure 1. SBMLR objects and functions.

Documentation

Programming concepts and details as of the initial release (version 1.00) are given in (Radivoyevitch, 2004). Illustrative examples of SBMLR-based biochemical system analyses of DNA microarray data (Radivoyevitch, submitted in 2004) are given in the `BMCcancer04` directory. SBMLR models of purine metabolism (Curto *et al.*, 1997) and folate metabolism (Morrison and Allegra, 1989) are given in the `models` directory. R scripts that use these models are given in the `demos` directory, and also as examples under standard R help. This manual serves as a brief supplement to these examples. The Appendix given below contains up-to-date versions of out-dated portions of (Radivoyevitch, 2004).

Getting Started

The essential components of SBML, namely, compartments, chemical species and reactions, can all be readily identified in the SBMLR files in the `models` directory. The first step to learning SBMLR is to look at these files. The second step is to give the paper `BMC.BioInformatics04.pdf` (in the `doc` directory) a light reading. The third step is to explore and understand the standard R help examples that come with SBMLR. The next two sections support this third step.

File Conversions

File conversions between SBMLR model representations and SBML level 2 are carried out using `write.SBML` and `read.SBML`. Examples can be found through R help, e.g.

```
library(SBMLR)
setwd(file.path(.path.package("SBMLR"), "models"))
write.SBML("Curto") # writes model in Curto.r to SBML (level 2) file Curto.xml
read.SBML("Curto") # converts SBML file Curto.xml into SBMLR model file CurtoX.r
```

To see that the final file conversion executed at the end of this code was successful, the simulation `runCurto.r` in the package's `demo` directory can be changed to act on the newly created file `CurtoX.r` instead of `Curto.r`. The plots come out the same. The SBML file Curto.xml can also be validated through http://sbml.org/tools/htdocs/sbmltools.php.

SBMLR Model Simulations

The demo script `runCurto.r` simulates the response of Curto's purine metabolism model (Curto *et al.*, 1997, 1998) to a bolus 10-fold increase in phosphoribosylpyrophosphate (PRPP) at time t=0. Plots of inosine monophosphate (IMP) and hypoxanthine (HX) are shown in Figure 2. Since HX reacts with PRPP to form IMP in the purine salvage pathway, the model's response seems reasonable.

The demo script `runMorrison.r` simulates the response of the folate model (Morrison and Allegra, 1989) to a continuous exposure to 1 µM MTX (methotrexate). The results of this simulation (not shown) are consistent with the model validation plots given in (Morrison and Allegra, 1989). In these two examples given here, since no microarray data is being applied to the models, `fderiv` is passed `p=c(mod=0)` to indicate that the Vmax's are not being modulated, see `fderiv` help.

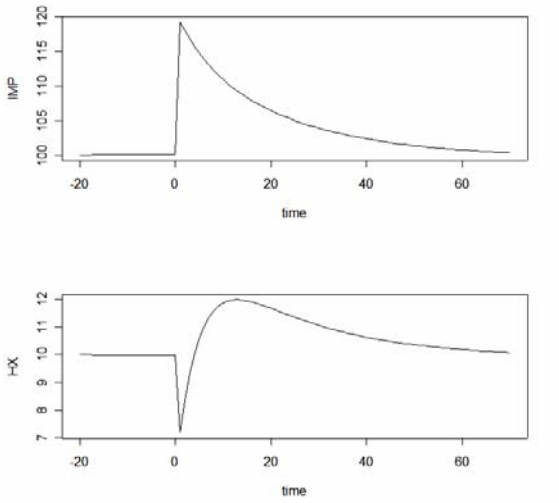

Figure 2. Model of Curto *et al.* responding to a 10-fold increase in PRPP treated as a new "initial state" at time t=0. Time is in minutes and concentration is in µM.

Microarray data driven simulations of Morrison's model

To learn SBMLR on an advanced level, the fourth step is to study the biochemical system analyses of DNA microarray data (Radivoyevitch, submitted in 2004) provided in the `BMCcancer04` directory. In these analyses, direct proportionality is assumed between enzyme mRNA levels and enzyme protein levels (Radivoyevitch, 2001). For steady state data, a matrix **`M`** containing V_{max} modulators must be

defined prior to calling lsoda. The matrix `M` is such that genes are in rows, patients are in columns, and values of 1 leave corresponding V_{max} values unchanged. The matrix **`M`** is passed globally to `fderiv` which must now be called with `p=c(mod=1)`to indicate that steady state data is being applied as a V_{max} step function perturbation at t=0. For time course data, a list of interpolation functions which map times t into V_{max} modulator values must be defined prior to calling `lsoda.` The list must be named **`Mt`** and must be passed globally to `fderiv`. In this case `p=c(mod=2)`.

Future Work

1. Add events and function definitions.
2. Provide graphic renderings of reaction rate equations in R.
3. Extend SBML to include unigene ID(s) for each reaction. If there is more than one gene involved in the production of a single functional enzyme [i.e. if the enzyme is a complex], a prescription of how gene expression values should be mapped into a modulator of V_{max} for the corresponding reaction rate law should be included in the SBML code.
4. It is anticipated that SBML level 3 will include graphical configuration information. When this happens, the SBMLR structure will be modified to include/carry such information.

Appendix

For those who are interested in modifying the SBMLR source codes, a fifth step to learning SBMLR would be to carefully reread (Radivoyevitch, 2004). The appendix here supplements this paper with updates of descriptions that have changed since SBMLR version 1.00. Thus, `fderiv` is updated below to reflect microarray data handling capability gained since version 1.10.

```
fderiv <-function(t, X, p)  # state derivative function sent to ODEsolve
{v=rep(0,nrxns);xp=rep(0,nStates)
St=S0
X[X<0]=0
St[BC==FALSE]=X
nrules=length(model$rules)
if (nrules>0)
 for (j in 1:nrules)
    St[model$rules[[j]]$output]=model$rules[[j]]$law(St[model$rule[[j]]$inputs])
if (p["mod"]==1) if (t<0) m=M[,"control"] else m=M[,patient]
for (j in 1:nrxns)
  if (model$rxns[[j]]$rever==FALSE){
   if (p["mod"]==0)   v[j]=model$rxns[[j]]$law(
      St[c(model$rxns[[j]]$reacts, model$rxns[[j]]$mods)],model$rxns[[j]]$params)
   if (p["mod"]==1)  v[j]=m[rIDs[j]]*model$rxns[[j]]$law(
      St[c(model$rxns[[j]]$reacts,model$rxns[[j]]$mods)],model$rxns[[j]]$params)
   if (p["mod"]==2) v[j]=Mt[[rIDs[j]]](t)*model$rxns[[j]]$law(
      St[c(model$rxns[[j]]$reacts,model$rxns[[j]]$mods)],model$rxns[[j]]$params)
   }
xp=incid%*%v
names(xp)<-names(y0)
names(v)<-rIDs
aux=c(v,St[BC==TRUE])
list(xp,aux)}    # ******************  END fderiv function
```

In this code, the current species vector **St** is created by overriding initial states $\mathbf{S_0}$ with current states **X** clipped to positive values, and by overriding any time-varying boundary conditions defined by rules. If `p["mod"]=1`, a step function at t=0 modulates the reaction rates in the subsequent for-loop. Before

the loop, the modulator **m** is set to the "control" column of **M** for negative times and the current patient column for positive times; the "control/average" patient corresponds to the model's initial steady state. `fderiv` then enters the reaction for-loop to compute the flux vector **v** based on the current species vector **St**. If `p["mod"]=0,` there are no modulators. If `p["mod"]=1,` the step function modulator is applied. And if `p["mod"]=2`, interpolating functions **Mt** evaluated at time *t* are applied. The current flux vector **v** then multiplies the incidence matrix to produce the current state derivative vector **xp**. At the end of each function call, the indexing names within **xp** and **v** are reset to override the problem of variables gaining new composite names from the names of their expression arguments.

Acknowledgements This work was supported by the Biostatistics Core Facility of the Comprehensive Cancer Center of Case Western Reserve University and University Hospitals of Cleveland (P30 CA43703), the Integrative Cancer Biology Program (1 P20 CA112963-01) and the American Cancer Society (IRG-91-022-09).

References

1. Curto, R., Voit, E. O., Sorribas, A. and Cascante, M. Validation and steady-state analysis of a power-law model of purine metabolism in man. *Biochem.J.* **324 (Pt 3)**, 761-775 (1997).
2. Curto, R., Voit, E. O., Sorribas, A. and Cascante, M. Mathematical models of purine metabolism in man. *Math.Biosci.* **151**, 1-49 (1998).
3. Finney, A. and Hucka, M. Systems biology markup language: Level 2 and beyond. *Biochem Soc Trans* **31**, 1472-3 (2003).
4. Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y. and Zhang, J. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol* **5**, R80 (2004).
5. Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, II, Hedley, W. J., Hodgman, T. C., Hofmeyr, J. H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novere, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J. and Wang, J. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**, 524-31 (2003).
6. Ihaka, R. and Gentleman, R. R:a language for data analysis and graphics. *Journal of Computational and graphical statistics* **5**, 299-314 (1996).
7. Morrison, P. F. and Allegra, C. J. Folate cycle kinetics in human breast cancer cells. *J.Biol.Chem.* **264**, 10552-10566 (1989).
8. Radivoyevitch, T. Sphingoid base metabolism in yeast: Mapping gene expression patterns into qualitative metabolite time course predictions. *Comparative & Functional Genomics* **2**, 289-294 (2001).
9. Radivoyevitch, T. A two-way interface between limited Systems Biology Markup Language and R. *BMC Bioinformatics* **5**, 190 (2004).
10. Radivoyevitch, T. SBMLR (2005) http://www.bioconductor.org/repository/devel/package/html/SBMLR.html.
11. Radivoyevitch, T. Folate system correlations in DNA microarray data. *BMC Cancer* (submitted in 2004).