MCMC_CLIB

Documentation

Andrei Kramer

May 15, 2019

Here, we explain the model structure assumed by `mcmc_clib`. In Section 1 we define the ODE model, its input/output structure and its parameter sensitivity. We describe how the model is passed to the sampling software in Section 2 and how to build a model as a shared library. All data sets have to be loaded in `hdf5` form, with specific naming conventions. This is described in Section 3. Some of the symbols have shorthands defined in the text and the meaning of any variable can be overloaded by such a shorthand symbol.

1 Model Specifications

We consider a deterministic ODE model and a stochastic measurement model. System states are captured by the state variables $x(t; \rho, u) \in \mathbb{R}^n$. The model parameters $\rho = \exp(\theta) \in \mathbb{R}^m_{++}$ are for the most part reaction rate coefficients and equilibrium parameters in systems biology; they describe interactions between the model state variables and are *unknown*. If any coefficients are exactly known, they should be defined as known constants in the model rather than parameters perhaps.

A second set of parameters $u \in \mathbb{R}^v$ describes the conditions of an experimental setup. These parameters are considered inputs and we assume that they can be set by the experimenter (and therefor *known*). They can be external parameters, e.g. the temperature, or describe modifications to the system, e.g. inhibitions to some of the interactions. A special kind of inputs are measured data sets of compounds that the model lacks a mechanism for, so their state has to be replayed, perhaps using interpolation, during simulations. Such an interpolation has to be implemented by the user, in the model; the software has no automated interpolation of input signals by itself.

Any experiment observes data y that is analogous[1] to the model's output function $g(x) \in \mathbb{R}^r$. The modeling assumption is that the data can be explained by the model output aside from measurement noise:

$$y_{ijk} - c_k g_i(x(t_j;\rho,u_k)) \sim \mathcal{N}(0,\varsigma^2_{ijk}) \qquad x(0;\rho,u) = x_0\,, \tag{1}$$

$$\tag{2}$$

where c_k is a possibly unknown scaling constant that accounts for experiments with *arbitrary units*. Data from such an experiment is called *relative data*. In any case, the data is obtained at measurement time-points t_{jk} ($j = 1,\dots,\nu_k$); these can be different in each experiment, and $t_{jk} \neq t_{jk'}$ is allowed. But we'll omit this from the notation and just use t_j.

There are several ways to deal with relative data. Our choice was to consider a second level of output function h:

$$h_i(g_{ijk}, g_{ij'0}) = \frac{g_i(x(t_j;\rho,u_k))}{g_i(x(t_{j'};\rho,u_0))}\,, \qquad g_{ijk} := g_i(x(t_j;\rho,u_k))\,, \tag{3}$$

$$z_{ijk} - h_i(x(t_j;\rho,u_k)) \sim \mathcal{N}(0,\sigma^2_{ijk}) \qquad z_{ijk} = \frac{y_{ijk}}{y_{ij'0}}\,, \tag{4}$$

such that scale constants c always cancel; and we define the shorthand $g_{ijk} := g_i(x(t_j;\rho,u_k))$. Here we assume that u_0 determines the so called *control* experiment. It is the reference experiment that experiment k is *relative to*. But, it can be any k' in principle; this notation choice neatly implies that if $k = 0,\dots,n_\mathrm{E}$ the *number of experiments* is still effectively n_E (not $n_\mathrm{E}+1$) as one of them is merely a *normalization* (control). The difference between g and h is that g corresponds to exactly one model simulation, while h processes several simulations. The output g is defined within the model, while h is specified through data annotation.

A data set can contain several (unrelated) *controls* if needed. Experiments have an annotation to express the relationship between them. The *reference time* $\hat{t} = t_{j'}$ does not have to be the same as t_j and can also be defined through annotation of data sets. In principle h can also mix output functions and devide $g_{i\cdot\cdot}$ by $g_{i'\cdots}$.

We consider *different experiments* to be distinguished in *input parameters*, *initial conditions*, or *output function* (at least one of these). All applicable inputs are enumerated as $u_k \in \mathbb{R}^v$ ($k = 0,\dots,n_\mathrm{E}$).

1.1 The Mechanistic Model

The model class the software can deal with is an ordinary differential equation (ODE) model:

$$\dot{x} = f(t,x;\theta,u)\,, \qquad x(t_0) = x_0 \tag{5}$$

$$\tag{6}$$

[1] up to measurement device specific scalings, offsets or other arbitrary constants

The initial conditions x_0 can be part of an experiment description, but most commonly they are not known for biological models and are assumed to be one of the steady states of the model. A common setup is:

$$t_0 = -T \qquad\qquad f(t, x(0;\rho,u);\rho,u) \stackrel{!}{=} 0\,, \tag{7}$$

with T being a large enough time to reach steady state and x_0 chosen suitably to hit the right steady state by whatever means.

1.2 Stochastic Measurement Model

Since the measurements are noisy the model's output is compared to the data using a statistical model. The right model is often unknown, in fact it is a typical case that no uncertainty analysis has been performed on the data at all and no repeated measurements have been made to estimate the parameters of an error model hypothesis.

Until data sets come with carefully justified error models and distribution parameters, we assume the error to be *Gaussian*.

Since the software calculates gradients, with hard coded partial derivatives, this choice is fairly static and cannot be changed easily. A different error model requires the implementation of an additional log-likelihood function and its partial derivatives. The currently used functions are *not supplied by the user*.

Similarly, since the Gaussian error model is merely an educated guess that performs well numerically, we don't transform the distribution into a possibly *heavy tail*ed *ratio distribution*.

We make the choice[2] that the error is modeled as a *Gaussian* distribution at the highest level output function h as well and we match that to the raw-distribution by appropriate choice of σ:

$$z_{ijk} - h_i(t_j;\rho,u_k) \stackrel{!}{\sim} \mathcal{N}(0,\sigma^2)\,, \tag{8}$$

where σ denotes the possible transformation of ς if the standard deviation of the raw data[3] ς has actually been estimated. The error propagation is calculated in the following way:

$$\sigma_{ijk} \approx \left|\frac{\partial h_i(g_{ijk}, g_{ij'0})}{\partial g_{ijk}}\right| \varsigma_{ijk} + \left|\frac{\partial h_i(g_{ijk}, g_{ij'0})}{\partial g_{ij'0}}\right| \varsigma_{ij'0}\,, \tag{9}$$

and in this case:

$$\sigma_{ijk} \approx \left|\frac{h_i(g_{ijk}, g_{ij'0})}{g_{ijk}}\right| \varsigma_{ijk} + \left|\frac{h_i(g_{ijk}, g_{ij'0})}{g_{ij'0}}\right| \varsigma_{ij'0}\,. \tag{10}$$

Our goal is to provide an estimate[4] of uncertainty in the data and propagate this

[2] rather than exactly transforming the distribution from g to h we match them by appropriate ς transformation, but not in shape.

[3] ς applies to the level-1 output $g_i(\cdot)$, σ applies to h.

[4] or perhaps even an upper bound

uncertainty to the model's parameters. We consider these decisions to be a middle ground between simplicity, numerical stability and accurate treatment of measurements. The statistical model is probably not quite right, but as it is almost always unknown we have nothing to replace it with.

Other sensible choices are to log-transform the data, such that an assumption of log-normal errors is made. But in practice, the above choices seem to be best compatible with biological data and numerically stable whenever no error bounds have been reported.

The first layer output is defined as part of the model, inside the model file. The characteristic of it is that one simulation at input u_k results in exactly one output g_{ijk} (nv_k matrix). The second layer, the normalisation of primary outputs: $g_{ijk}/g_{i'j'k'}$ is to some extent fixed and only the indexing rules (which i', j', k') can be supplied via the annotation of the data. The operation `ratio of g` is predefined and cannot be changed easily (only turned off).

Log-transformations inside of g are allowed already as that is defined in the model itself. Log-transformations on the level of h are not supported yet:

$$h_{ijk} \overset{\texttt{OK}}{=} \log(x_i(t_j;\rho,u_k))/\log(x_i(t_{j'};\rho,u_0))\,, \qquad g_{ijk} = \log(x_i(t_j;\rho,u_k)) \tag{11}$$

$$h_{ijk} \overset{\texttt{NO}}{=} \log(x_i(t_j;\rho,u_k)/x_i(t_{j'};\rho,u_0)))\,, \qquad g_{ijk} = x_i(t_j;\rho,u_k)\,. \tag{12}$$

This has to do with analytical, hard coded expressions during the calculations of gradients and the Fisher information; not a principal limitation.

1.3 Sensitivities

The model has sensitivities of x with respect to ρ and derived from that also sensitivities of g and h, with repsect to changes in the parameters ρ and on the log-scale: $\theta = \log(\rho)$.

The CVODES solver will return both the state sensitivities $S_x(t_j;\rho,u_k)_i^{\,l}$ and the output sensitivities $S_g(t_j;\rho,u_k)_i^{\,l}$. So, we will use these as given.

With a known normalisation indexing: i', j', k' for each i, j, k, the sensitivity of $h(\cdot)$ in terms of the (known) $g(\cdot)$ sensitivities $S_g(t_j;\rho,u_k)_i^{\,j}$ is:

$$g_{ijk}(\rho) := g_i(x(t_j;\rho,u_k))\,, \tag{13}$$

$$h_{ijk}(\rho) := \frac{g_{ijk}(\rho))}{g_{i'j'k'}(\rho)}\,, \tag{14}$$

$$\frac{dh_i(g_{ijk}(\rho)), g_{i'j'k'}(\rho))}{d\rho_l} = \frac{S_g(t_j;\rho,u_k)_i^{\,l} g_{i'j'k'}(\rho) - g_{ijk}(\rho) S_g(t_{j'};\rho,u_{k'})_{i'}^{\,l}}{g_{i'j'k'}(\rho)^2}$$

$$S_h(t_j;\rho,u_k)_i^{\,l} = \frac{S_g(t_j;\rho,u_k)_i^{\,l} - h_{ijk}(\rho) S_g(t_{j'};\rho,u_{k'})_{i'}^{\,l}}{g_{i'j'k'}(\rho)}\,. \tag{15}$$

The code for this operation is located in the function `LogLikelihood`.

1.4 Sampling in Logarithmic Space

In this type of ODE model, the parameters are positive. In some cases the model becomes unstable after sign flips, so non-negativity has to be enforced. Additionally, bio-chemical parameters are often unknown even in their magnitude. So, sampling θ will result in only positive values $\rho = \exp(\theta)$ to be passed to the model as parameters.

This has the additional benefit of covering several orders of magnitude more efficiently. Unfortunately, this choice implies that we have to modify the expressions for the model sensitivities. The model handling tool VFGEN and solver CVODES provide functions for sensitivity analysis with respect to the nominal model parameters ρ:

$$\begin{aligned}
\dot{x} &= f(t, x; \rho, u)\,, \\
g_{ijk}(\rho) &:= g_i(x(t_j; \rho, u_k))\,, \\
h_{ijk}(\rho) &:= \frac{g_i(x(t_j; \rho, u_k))}{g_{i'}(x(t_{j'}; \rho, u_{k'}))}\,, \\
S_h(t_j; \rho, u_k)_i{}^l &:= \frac{dh_i(g_i(x(t_j; \rho, u_k)), g_{i'}(x(t_{j'}, \rho, u_{k'})))}{d\rho_l}\,, \\
\frac{dh_{ijk}(\rho(\theta))}{d\theta_l} &= \frac{\partial h_{ijk}(\rho(\theta))}{\partial \rho_l}\frac{\partial \rho_l}{\partial \theta_l} \\
&= S_h(t_j; \rho, u_k)_i{}^l \rho_l\,,
\end{aligned} \tag{16}$$

for any input u_k (no summation implied).

2 Software Usage

The software has two major inputs: (i) the ODE model as a shared library, and (ii) all data sets (annotated) as an `hdf5` file.

2.1 Model `.so`

If the user has a way to generate CVODES compatible model source and header files (`C`) from some modeling language like `sbml` then nothing else is required. We provide no `sbml` to `C`/CVODES conversion scripts (currently).

One of the reasons is that to our knowledge the `sbml` standard does not provide any way to define model input parameters. In addition, no standard software tools are known to us to process `sbml` into `C` sources automatically, and certainly not including symbolically calculated Jacobians. Custom conventions for the definition of inputs and outputs can be made, yet `sbml` is quite difficult to parse, while tabular formats are easy to parse using line oriented tools. For all of these reasons we have decided to use `SBtab` for the editing and storage of the model and all data sets (if small enough) for our projects.

The model can be converted from `SBtab` using an R script we provide:

`sbtab_to_vfgen.R`,

as further explained in Section 3.3. It reads the SBtab file in *Open Document Spreadsheet* (`.ods`) format and converts the biologically motivated model into an ordinary dufferential equation model. This process strips biological meaning somewhat (species,compartment,etc.) and uses only general terms such as *Expression* and *State Variable*.

The result is a VFGEN (`.vf/.xml`) file that can be parsed by the VFGEN[5] software; VFGEN in turn uses GINAC to calculate the *Jacobian* and *sensitivity* terms symbolically (for this model). It outputs the model into a language of choice (e.g. MATLAB, R) one of which is C/{CVODES,GSL}.

The relevant commands are:

meaning	**command in** `bash`
vfgen to cvodes	`vfgen cvodes:sens=yes,func=yes ./C/Model_cvs.c`
make shared library	`gcc -shared -fPIC -o Model.so Model_cvs.c`

The shared library is loaded by the sampler `ode_smmala` and can be produced in any other way as long as it follows the interface requirements of `cvodes`.

As a workaround regarding solver failures on high dimensional problems with forward sensitivity analysis there is a sensitivity approximation routine that requires a *parameter* Jacobian $df(t,x;\rho,u)/d\rho$ to be available in the model `struct`.

3 Data Storage .h5

The problem set up is given as an hdf5 `.h5` file. The file shall contain annotated data sets, and prior probability density parameters. The prior parameters can be μ_i, ς_i for independent Guassian distributions for each model parameter, or μ, ς for a multivariate Gaussian.

3.1 Experiments

It is not immediately clear what aggregate of information can be said to belong to one *experiment*. We have so far identified two major types:

Dose Response this experiment type consists of model responses to varying *doses* of input. The data table consists of one or more columns of input and one or more columns of corresponding outputs. Each line of such a table requires a simulation of the model to opbtain the input/output relationship for a given parametrization. In such cases there can be only one measurement time to record an output. If more than one time was recorded, then we must treat the case as many[6] time series experiments.

[5] `github.com/WarrenWeckesser/vfgen`

[6] possibly very short

Time Series here, a data table describes one simulation of the model and gives a record of an output measured at discrete timepoints.

Both types of experiment have mandatory components aside from data points: an estimate of standard deviations[7] ς_{ijk}, which input to apply, time(s) to record outputs. As described in earlier sections, the sampler will always assume a normal distribution for output noise. While reading this file, `ode_smmala` will transform ς into σ, as described in Section 1.2.

The two types of experiment can be defined in the SBtab file if that is used. But, once the data is processed, it is stored in blocks that represent a *simultion unit*. So, time series data remains unchanged, while dose responses are converted into one *data block* per line. The SBtab file allows omission of many values, especially data points. The `sbtab_import` processing of the spreadsheet will replace missing values with defaults, the replacement can be inspected using standard hdf5 tools like

```
h5dump -d /data/data_block_0 DataFile.h5 .
```

The h5 file may not omit columns or rows[8].

3.2 Data File Structure

The terminology of hdf5 files can be understood as somewhat analogous to a zip archive with folders and files. In an hdf5 file, the folders are called *groups* and the files are *dataset*s. The datasets can have attributes. This is the structure the sampler expects to find:

data `GROUP` a group that contains measured data. All `DATASET`s inside this group are considered data matrices.

- **data_block_%i** `DATASET` Consequtively numbered data sets with attributes. A matrix of size $n_t \times r$ (`(size(t)*size(h))`). One line per measurement time point and one column per defined output. Missing values can be indicated by infinite standard deviations. The name data_block_* is not enforced or checked; the index attribute is used to order the data sets.
  - **input** `ATTRIBUTE` the input parameter vector for this data block (simulation unit).
  - **time** `ATTRIBUTE` the measurement times for this data block, one per row in the data matrix.
  - **LikelihoodFlag** `ATTRIBUTE` if this is 1, the experiment will contribute to the likelihood function. If it is 0, then it is assumed to be needed for the normalisation of another experiment. It will be simulated, can participate in normalisation operations but skipped during the calculations of: log-likelihood, its gradient and Fisher information.

[7] for output function i, time point j, and input vector k

[8] It is entirely untested what happens when NA or NaN values are passed to the sampler.

NormaliseByExperiment ATTRIBUTE *optional* Exactly one index that identifies which dataset this one is relative to. If this is missing or negative then the experiment is not relative to another. This corresponds to k' from Section 1.2.

NormaliseByOutput ATTRIBUTE *optional* a vector of indices, one per output, to indicate which output to devide by. This corresponds to i'. If negative, then the output is absolute, the normalising value is set to 1 ± 0.

NormaliseByTimePoint ATTRIBUTE *optional* if present, then each line of this data block will be devided by exactly on line from the reference experiment. This corresponds to j'. This cannot be a vector. If this is missing, but NormaliseByExperiment is present, then the two must have the same number of lines. No check is performed whether the lines represent the same time points.

index ATTRIBUTE a running index, numbering the data blocks, starting at 0. This index is used to find corresponding data blocks and standard deviation blocks (same index). There are two additional index numbers, major and minor. These can be used to check the correspondence between data blocks and SBtab sheets, where *dose response* sheets can contain several data blocks(simulation units).

major ATTRIBUTE an index that corresponds to data sheets, so all data blocks that come from the same *dose response* experiment will share the same major index.

minor ATTRIBUTE an index that numbers the data entry lines from a *dose reponse* experiment.

stdv GROUP the group that contains the standard deviation estimates for the data points.

stdv_block_%i DATASET a matrix of the same size as data_block_%i (these are matched by the index attribute). Infinite standard deviations indicate a missing value. These are omitted from calculations such as $\prod_{ijk} \sqrt{2\pi}\varsigma_{ijk}$. The standard deviations don't repeat the required annotation of the data matrices, they only contain the running index and possibly major & minor attribute.

index ATTRIBUTE a running index, numbering the data blocks, starting at 0. This index is used to find corresponding data blocks and standard deviation blocks (same index).

major ATTRIBUTE an index that corresponds to data sheets, so all data blocks that come from the same *dose response* experiment will share the same major index.

minor ATTRIBUTE an index that numbers the data entry lines from a *dose reponse* experiment.

prior GROUP Prior parameter group. The prior is either[9] a product of univariate *Guassians* or one multivariate Gaussian. These two cases are distinguished by name: sigma refers to ς, while Sigma refers to the multivariate case Σ (a symmetric matrix). Select one of the optional items.

mu DATASET a vector of one μ per paramete θ (size m), in logarithmic space (natural logarithm).

sigma DATASET *optional* prior parameter for *width* (standard deviation), same size as mu.

Sigma DATASET *optional* a matrix of size $m \times m$. The covariance matrix ς of the multivariate prior.

Precision DATASET *optional* Same as Σ^{-1}.

3.3 Tools

If the SBtab method is used, then the .ods file can be converted into a series of .tsv files and then converted into an hdf5 file using the sbtab_import program:

SBtab to hdf5 `sbtab_import *.tsv DataSet.h5`

The git repository github.com/a-kramer/SBtabVFGEN contains scripts for conversion of SBtab files between the .tsv and .ods formats and an R[10] script sbtab_to_vfgen.R that converts such a model into a VFGEN readable[11] .xml file.

An example model in the SBtab form is provided here: github.com/a-kramer/DemoModel. VFGEN can output the same model into many other languages for post processing of sampling results.

[9] not implemented yet: generalised Gaussian
[10] r-project
[11] also fairly human readable