BIOBAYES TUTORIAL

http://www.dcs.gla.ac.uk/BioBayes/

Inference Group
Department of Computing Science
University of Glasgow

Contents

1 Introduction

Inferring the structure of biochemical systems from experimental observations is one of the important challenges in Systems Biology (Burbeck and Jordan, 2006). Such structures are usually defined with mathematical models. One of the advantages of using formal mathematical models is the possibility to make predictions of system behaviour alongside explaining the observed processes. Systems of ordinary differential equations (ODE) are a widely used formalism for modelling biochemical systems (see, for example, de Jong, 2003; Voit, 2000). Inferring parameters of ODE models of biochemical systems can be achieved using methods of Bayesian inference (Lawrence *et al.*, 2007; Rogers *et al.*, 2006), and evidence-based ranking of alternative models is possible using Bayes factors (Vyshemirsky and Girolami, 2008).

The main benefit of adopting the Bayesian approach to model inference is the consistent propagation of uncertainty through all the stages of analysis and the formal way in which prior knowledge can be included in the modelling process. This approach allows one to consider noisy observations as a source of data for learning full distributions of beliefs rather than restricting oneself to the most plausible explanation of some phenomenon. So, instead of making future predictions based on one's best guess, the Bayesian approach considers all probable outcomes.

Implementing the methods of Bayesian inference for probabilistic analysis of biochemical models, however, requires addressing many technical problems such as solving initial value problems for stiff systems of differential equations (Press *et al.*, 2002), or estimating effective proposal distributions for satisfactory convergence of Markov Chain Monte Carlo (MCMC) algorithms (Gelman *et al.*, 1995). It is also important to mention, that in recent years the scientific community has formulated a number of standards for a unified description and exchange of data and models, for example, the SBML standard (M. Hucka *et al.*, 2003) for models of biochemical systems. At the same time working with *ad hoc* implementations of inference algorithms usually requires some fine tuning to each particular problem.

We herein present an extensible software package, BioBayes, which supports standard definitions of mathematical models, and provides a framework for applying methods of Bayesian inference to ODE models of biochemical systems. In addition to implementations of general inference and model comparison methods, BioBayes provides an infrastructure for plugging-in user specific methods using standard interfaces, thus enabling fine tailoring of the tool to user's specific requirements if needed.

2 Installation and Updates

BioBayes is available for download from the official web site http://www.dcs.gla.ac.uk/BioBayes/ for three platforms: Windows XP, Mac OS X, and Linux.

2.1 Installing BioBayes

To install BioBayes you first have to download an archive which corresponds to the platform you are running (see Figure 2.1).

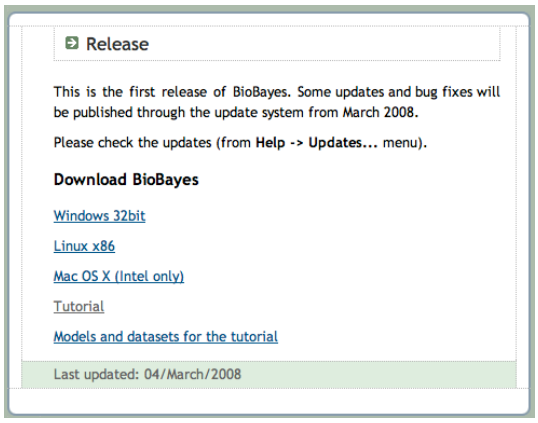

Figure 2.1: BioBayes is available for download from the official web site: http://www.dcs.gla.ac.uk/BioBayes/

After downloading an archive (a .zip file) you have to extract all the files and directories contained in it. You can open the archive using your operating systems' windowing interface, or use command line command `unzip` to do so. When you have extracted the contents of the archive, you can move the whole directory to a convenient place, for example to `Applications` folder or to your desktop folder.

This completes the installation procedure of BioBayes. No platform-specific operations are required to be performed. BioBayes does not store any information outside of its installation directory. Uninstalling BioBayes is as easy as removing the directory you have just extracted from the archive.

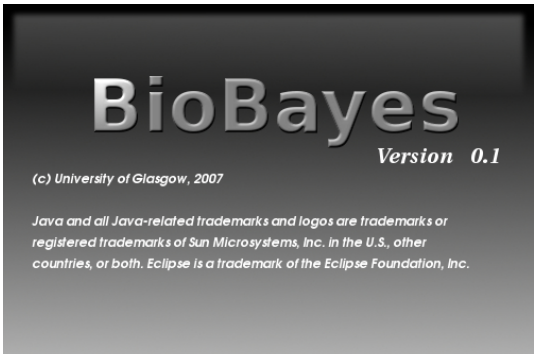

Figure 2.2: Splash screen is displayed for a few moments when BioBayes is started.

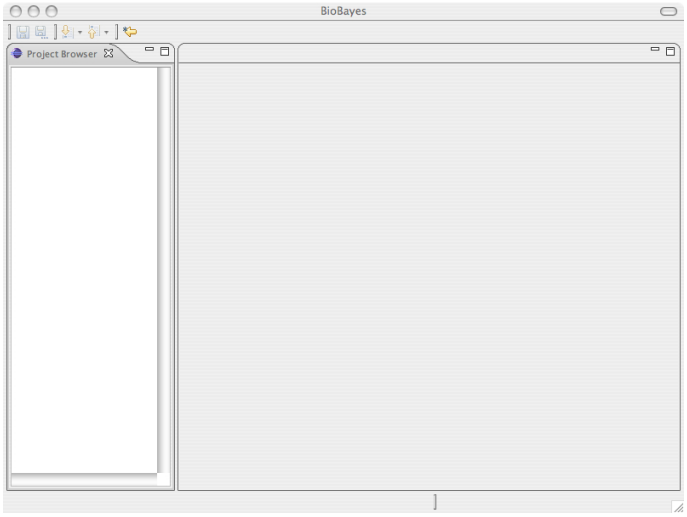

Figure 2.3: The main window of BioBayes when open for the first time.

> **ATTENTION:** *When deleting the installation directory make sure you have copied all your models and data. BioBayes projects are stored within the installation directory and will be lost when the directory is deleted.*

To run BioBayes, open this freshly extracted directory, and find an executable called:

- `BioBayes` for Mac OS X
- `BioBayes` for Linux
- `BioBayes.exe` for Windows XP.

Run this executable to start BioBayes. A splash screen is displayed for a few moments while the programme is starting (see Figure 2.2), and the main window is then displayed (see Figure 2.3).

> ***Hint:*** *If the programme fails to start or runs out of memory during execution, you might want to change the default memory management parameters by adding the following parameters to the BioBayes.ini file inside the installation directory:*

- *-XmsXXXm (where XXX is the amount of memory in megabytes the programme should take when started).*
- *-XmxXXXm (where XXX is the maximal amount of memory the programme can try to allocate).*

See the Java Virtual Machine parameters description for more information on these options.

2.2 Checking Software Updates

It is important to check for software updates, as bug fixes and new features will be distributed in this way.

To check if any software updates are available, and install them if necessary, you have to choose `Update...` item from the `Help` menu.

The programme then will try to contact BioBayes official web site and check for software updates. It is important that you have internet access, so the programme can reach the updates web site. If for some reason the connection cannot be established, contact your technical support to resolve the problem.

Hint: *If you have no way absolutely to bypass the proxy server at your location, manually configure the proxy settings in the BioBayes.ini file inside the installation directory. Please add the following options:*

- *-Dhttp.proxySet=true*
- *-Dhttp.proxyHost=<your_proxy_host>*
- *-Dhttp.proxyPort=<your_proxy_port>*

And for authentication set these parameters also

- *-Dhttp.proxyUser=<your_username>*
- *-Dhttp.proxyPassword=<your_password>*

2.3 Installing Additional Plugins

The users also can install additional features and plugins in the way similar to installing software updates. This can be done by selecting `Add New Features...` option from the `Help` menu.

3 Model Parameter Inference using Metropolis-Hastings Sampler

3.1 Metropolis-Hastings Sampler

The Metropolis-Hastings sampler is included in the default package distributed with BioBayes. It utilises Markov Chain Monte Carlo methods and enables model parameter inference from experimental data for simpler models of biochemical systems.

Users can define the desired prior distributions for model parameters, run this sampler to infer parameter posteriors using one or more experimental datasets, and progress can be monitored via the results pane.

We optimise the proposal distribution of the Metropolis-Hastings sampler for more effective convergence of Markov Chains by scaling the proposal variance proportionally to the local acceptance ratio and also by adjusting the proposal covariance matrix to a local approximation of the posterior distribution as described by Gelman *et al.* (1995).

This implementation of the Metropolis-Hastings sampler allows users to run several chains at the same time to monitor the convergence of the sampler to the true posterior distribution by comparing within-chain variance of the sample to between-chain variance as proposed by Gelman *et al.* (1995). The $\hat{R}$ statistic is computed for that purpose for each of the model parameters. Current values of $\hat{R}$ are displayed in the results pane of the programme. The $\hat{R}$ values approach one as the chains mix. The software allows users to define an acceptable threshold for the $\hat{R}$ values, and the programme assesses that the chains have converged to the true posterior after all the values fall below that threshold. Users, of course, can override that convergence criterion and use, for example, a simple limit on the number of steps during the initialisation of the chains.

When it is judged, using the $\hat{R}$ statistic, that the chains have converged to the posterior distribution, the programme produces the final posterior sample performing sample thinning if required by the user. The marginalised projections of this sample are then displayed, and the sample itself can be exported for further analysis using external tools.

3.2 An Example: Exponential Decay

The example we consider in this section is a simple reaction of decay. The following system of ordinary differential equations:

$$\begin{cases} \frac{dS}{dt} = -k_1 \cdot S \\ \frac{ddS}{dt} = k_1 \cdot S \end{cases} \tag{3.1}$$

describes the law by which the values of two variables, S and dS, change in time. This system of differential equations is parametric, and k_1 is the only parameter of this system. This system models the decay of some protein whose concentration is expressed by the value of the variable S.

So, equation (3.1) defines the model, and we can simulate some data by substituting a certain fixed value for the model parameter k_1. For example, selecting $k_1 = 0.1$, the initial values of the variables $S|_{t=0} = 1$, and $dS|_{t=0} = 0$, and adding some amount of noise (e.g. $\sigma = 0.1$) to that data produces the dataset in Table 3.1.

t	5	10	20	30
S	0.396246	0.2347	0.0303159	-0.0399773
dS	0.460572	0.494962	0.949708	0.975488

Table 3.1: Data generated from the system of ordinary differential equations (3.1) with $k_1 = 0.1$, $S|_{t=0} = 1$, $dS|_{t=0} = 0$, $\sigma = 0.1$.

3.3 Selecting Parameters and Running the Task

Run Biobayes and create a project for this tutorial by selecting `Projects` → `New...` menu, then selecting `General` → `Project` and clicking `Next >`. At the next stage, input the name for this new project (e.g. `Tutorial1`) and press `Finish`.

Now import the model and the dataset supplied with this tutorial. To do so, select `Projects` → `Import...` from the programme's menu. At the first stage you have to select `Biochemical Models and Data` → `Import SBML Model` and then press `Next >`. At the second stage select the project you want to import the model to, in this case `Tutorial1`, and browse for the model file, `decay.xml`, to be imported. Finally, press `Finish` to confirm the import operation. You will now be able to see the structure of the newly created project, including the imported model, in the `Project Browser` window. To import the dataset, select `Projects` → `Import...` menu again, and then select `Biochemical Models and Data` → `Import Dataset`. Confirm your choice by pressing `Next >`. Again, select the project to import the dataset to, and browse for the dataset file `decay.data`, confirm the import by pressing `Finish` button.

In the similar way (just selecting `Biochemical Models and Data` → `Import Task` in the first step) import the definition of parameter inference task for this decay model called `decay.task`.

Double click on `decay.task` in the project browser after the import is completed. This will open the task definition editor depicted in Figure 3.1.

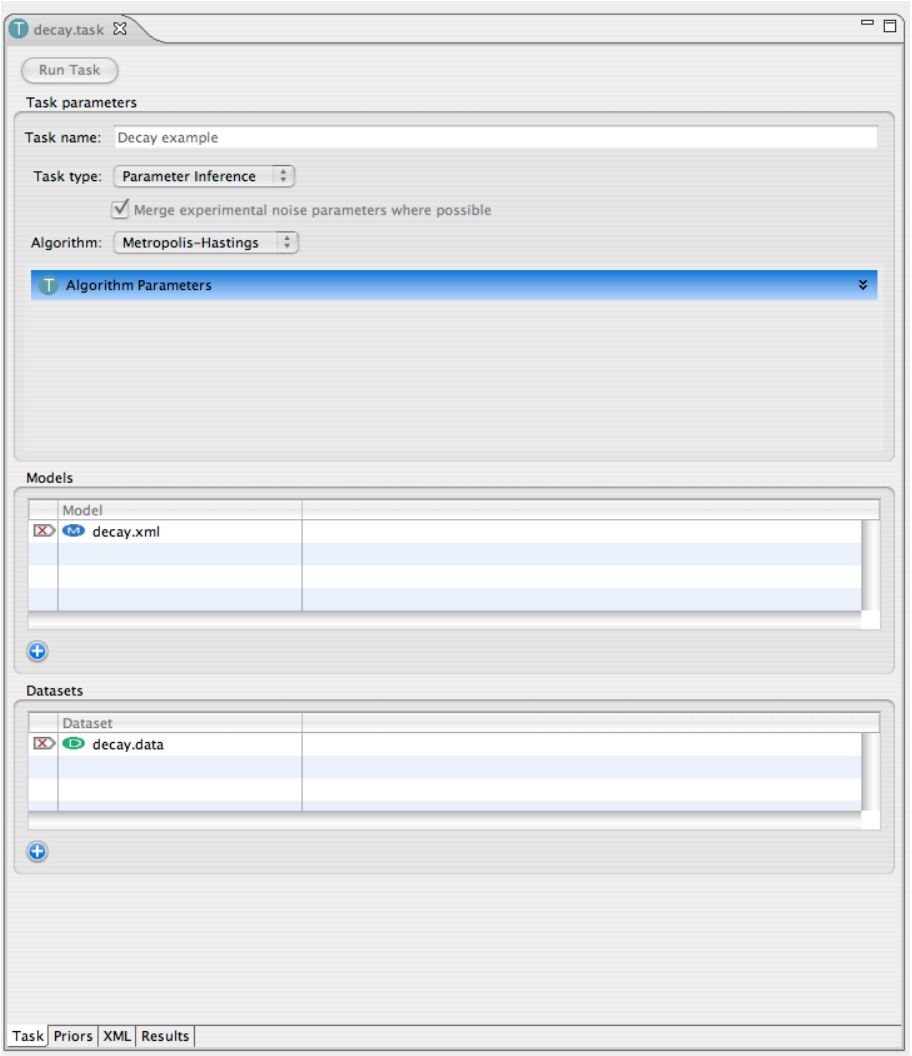

Figure 3.1: The task editor for `decay.task`.

Unfold the algorithm parameters, and see that the number of parallel chains selected for convergence monitoring is set to 3, the convergence monitoring using the $\hat{R}$ statistic is enabled, the threshold for $\hat{R}$ is set to 1.21, and the posterior sample size is set to 1000 with thinning of 1, which means that every sample produced by the Markov Chains will be taken into the posterior sample.

Now, switch to the `Priors` pane, which you can do by clicking on the `Priors` tab located in the lower edge of the editor (see Figure 3.2). The priors indicate that the initial

values for S and dS are fixed, and so is the volume of the compartment, while the kinetic parameter k_1, and the noise parameter `Sigma` have Gamma priors with parameters $a = 1$ and $b = 2$.

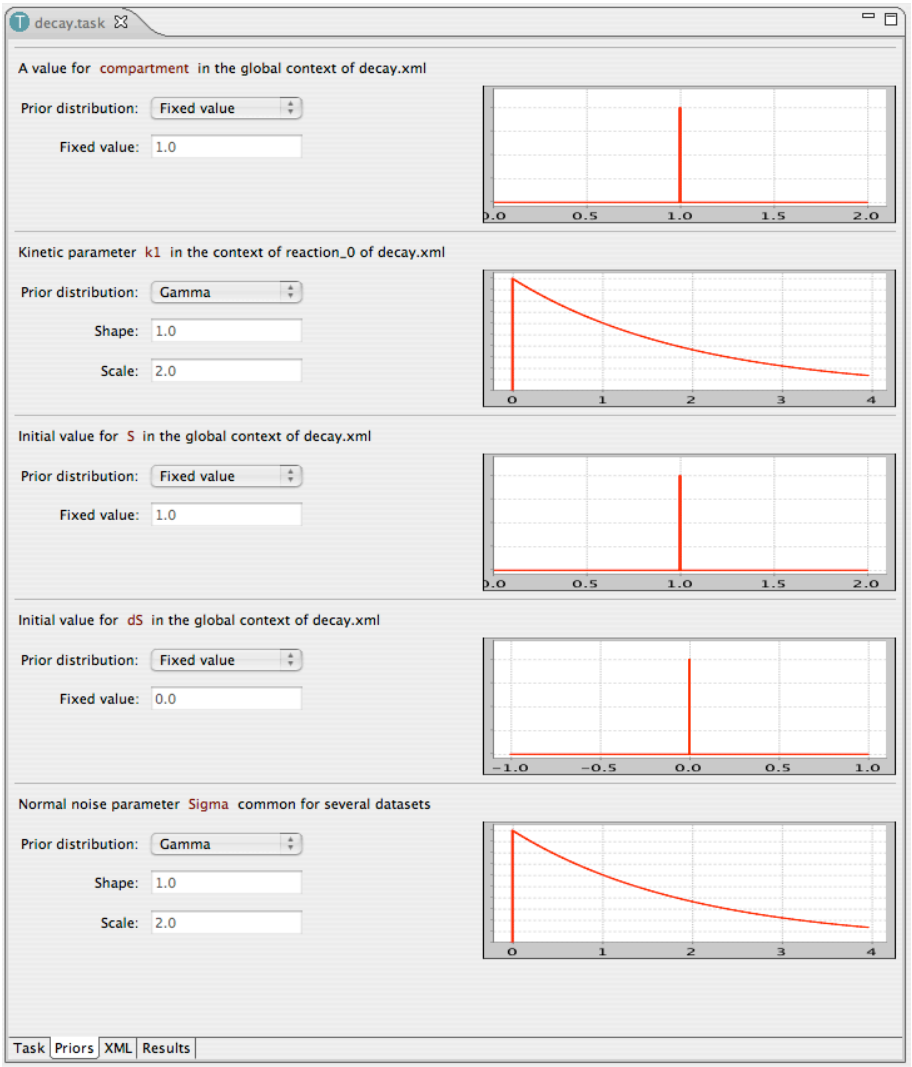

Figure 3.2: The task editor for `decay.task` with the `Priors` pane open.

You can start the sampler by pressing the `Run Task` button back in the `Task` pane.

3.4 Tracking Task Execution and Results

When the task is submitted, you can switch to the `Results` pane depicted in Figure 3.3.

In the top of the results pane, the status line informs how many chains are running in parallel, and how many proposals the sampler had already made. The `Convergence Monitor` plot depicts current values of the $\hat{R}$ statistic as they are computed. The

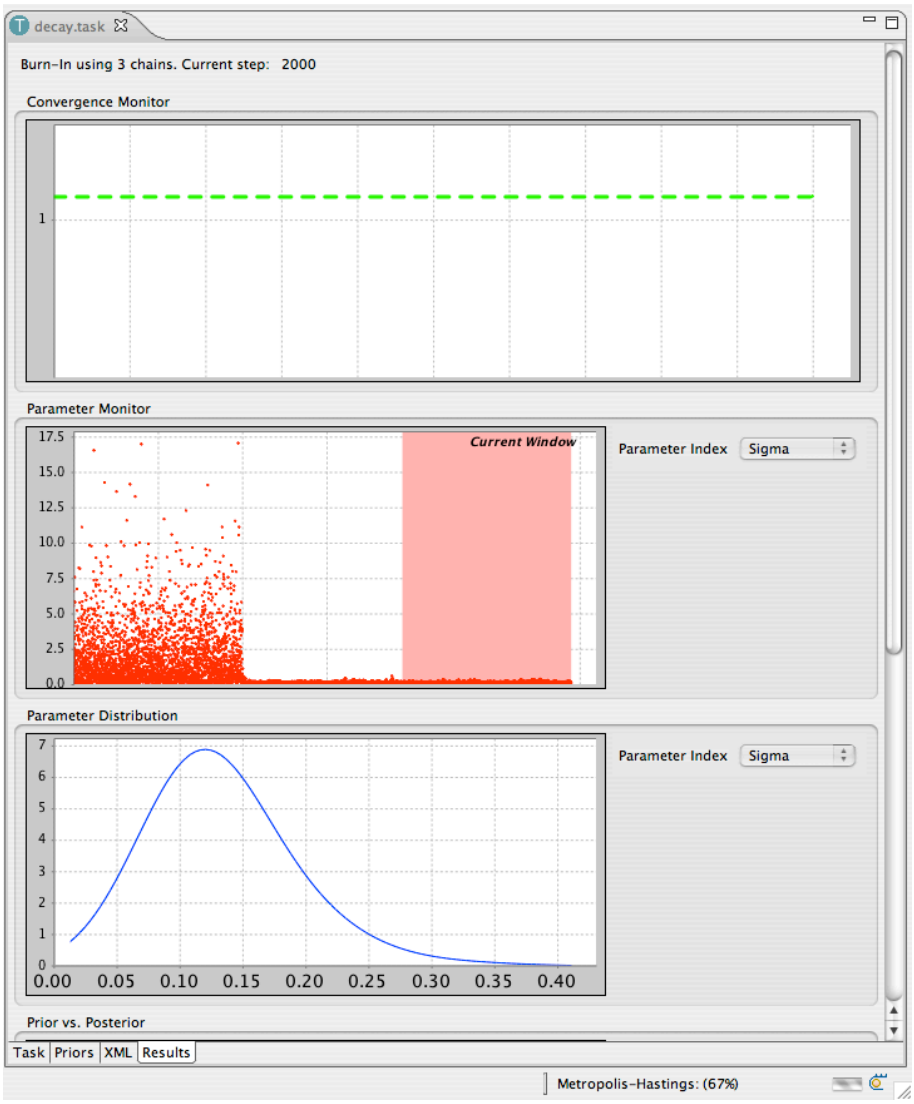

Figure 3.3: The task editor for `decay.task` with the `Results` pane open while running the Metropolis-Hastings sampler.

`Parameter Monitor` plot depicts the values sampled for each of the non-fixed parameters. The `Parameter Distribution` plot depicts current kernel-smoothed approximation to the parameter posterior distribution density function. The `Prior vs. Posterior` plot allows one to compare the kernel-smoothed plots of the parameter prior against the current approximation to the parameter posterior density. And the `Acceptance Rate` plot depicts how the acceptance rate of the Metropolis-Hasting sampler changes during the execution.

When the chains converge, and the sampling is finished, the projections of the parameter posterior are displayed as depicted in Figure 3.4.

As you can see, the true values of the model parameters used when simulating the dataset $k_1 = 0.1$ and $\sigma = 0.1$ fall into the high density area of the posterior distribution,

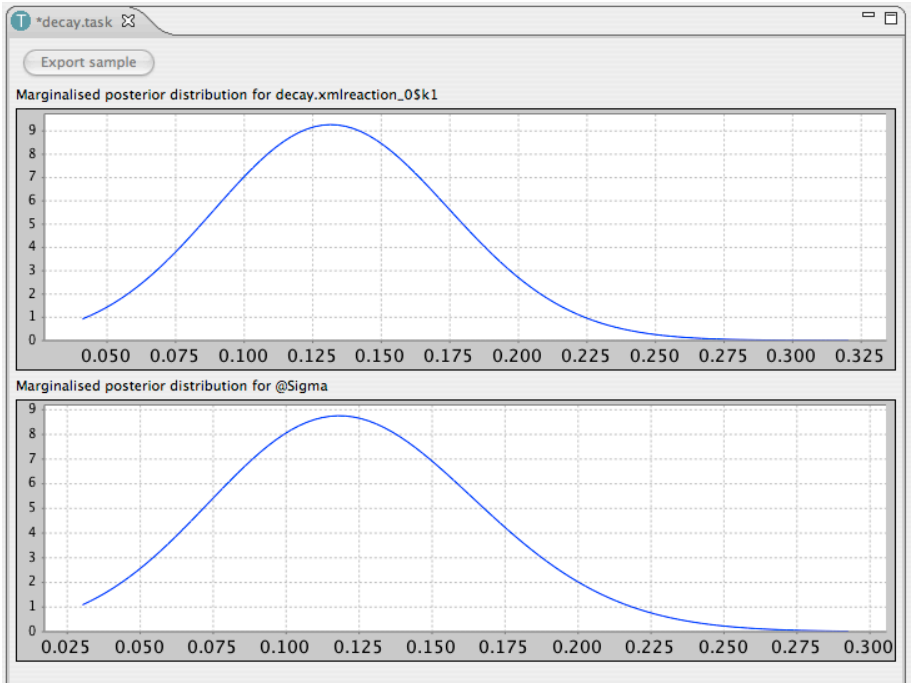

Figure 3.4: The task editor for `decay.task` with the `Results` pane open displaying the produced posterior sample projections.

which confirms that the parameters were properly inferred. The parameter posterior means are slightly shifted from the values used for data simulation, which demonstrates the impact of the prior when just a small amount of data is used.

Users can now click on the `Export sample` button to export the complete sample from the parameter posterior as a text file for future analysis with the third party tools (e.g. R or Matlab). For example, one can import the produced sample into *Matlab* and compute the correlation coefficient between parameters k_1 and σ in the parameter posterior, which should be about 0.08. This value of the correlation coefficient indicates that there is little correlation between parameters in the posterior sample.

4 Model Parameter Inference using Population-Based MCMC

4.1 Population-based MCMC

There is also a population-based MCMC sampler (Jasra *et al.*, 2007) available that can be applied to more complex problems when straightforward Metropolis-Hastings fails to converge, e.g. when using nonlinear oscillator models. This sampler runs several Markov chains in parallel using a tempered sequence of distributions as their targets. Moves between different chains in such a sequence of distributions help the sampler to overcome energy barriers and therefore sample more efficiently from multi-modal posterior distributions. The number of steps in such a sequence can be adjusted by the user.

The convergence of this sampler to the true posterior distribution is again judged by using the $\hat{R}$ statistic over several population-based MCMC samplers run simultaneously.

4.2 Nonlinear Oscillator Models

In this section we consider the nonlinear oscillator models of a circadian clock. We will consider the model with three variables defined by the system of ordinary differential equations (4.1), and the model with five variables defined with (4.2).

$$\begin{cases} \dfrac{dx_1}{dt} = \dfrac{k_1}{1 + x_3^{10}} - m \cdot x_1 \\ \dfrac{dx_2}{dt} = k_2 \cdot x_1 - m \cdot x_2 \\ \dfrac{dx_3}{dt} = k_3 \cdot x_2 - m \cdot x_3 \end{cases} \tag{4.1}$$

$$\begin{cases} \dfrac{dx_1}{dt} = \dfrac{k_1}{1 + x_5^{10}} - m \cdot x_1 \\ \dfrac{dx_2}{dt} = k_2 \cdot x_1 - m \cdot x_2 \\ \dfrac{dx_3}{dt} = k_3 \cdot x_2 - m \cdot x_3 \\ \dfrac{dx_4}{dt} = k_4 \cdot x_3 - m \cdot x_4 \\ \dfrac{dx_5}{dt} = k_5 \cdot x_4 - m \cdot x_5 \end{cases} \tag{4.2}$$

The models in SBML format are available in files `goodwin3.xml` and `goodwin5.xml` correspondently.

Two datasets are provided in the archive supplied with this tutorial. The first one, `goodwin3.data`, has been generated from model (4.1) by substituting $m = 0.4$, $k_1 = 1.2$, $k_2 = 1$, $k_3 = 1$, $\forall i : x_i|_{t=0} = 0$, and adding some normally distributed noise with standard deviation $\sigma = 0.02$. The second dataset, `goodwin5.data`, has been generated from model (4.2) by assigning $m = 0.4$, $k_1 = 1.2$, $k_2 = 1$, $k_3 = 0.8$, $k_4 = 1.3$, $k_5 = 1.2$, $\forall i : x_i|_{t=0} = 0$, and adding some normally distributed noise with standard deviation $\sigma = 0.02$. In both cases the datasets consist of 80 time points beginning with $t = 40$ and running for integer values of t up to 119. The data in those time series were collected from variables x_1 and x_2 only.

The likelihood landscape for such oscillating models is usually quite complex, and simple Markov Chain Monte Carlo samplers usually experience problems with convergence to the true posteriors. Very often such simple samplers stuck in the local modes of relatively high probability density and are very unlikely to escape such local modes and converge to the main mode of the posterior distribution. Population-based MCMC methods promise better convergence to the true posteriors for such models, and while being computationally more expensive in general for the simple models, these methods provide superior results in the cases when sticking to local modes becomes an issue.

We provide two predefined task definitions for parameter inference over the proposed oscillatory models using an implementation of the population-based Markov Chain Monte Carlo. The first one, `goodwin3.task`, defines parameter posterior inference for the model defined with (4.1), using the dataset `goodwin3.data`. This task definition requests to infer the parameters m, k_1, k_2, k_3 and σ, assuming that the prior for m, k_1, k_2 and k_3 is $\Gamma(4, 0.5)$, and the prior for σ is $\Gamma(1, 1)$. The second task definition, `goodwin5.task`, requests parameter posterior inference using model (4.2) and dataset `goodwin5.data`. The proposed priors are $\Gamma(4, 0.5)$ for m, $k_i, i = 1 \ldots 5$, and $\Gamma(1, 1)$ for σ. Both task definitions request 15 parallel chains to be run in the tempering schedule, and three independent instances of the sampler will be run in parallel to evaluate the convergence to the posterior distribution. The posterior samples will contain 5000 samples each with thinning of 2.

4.3 Running Population-Based MCMC

To run this example, begin with importing the following files into a project:

- goodwin3.xml
- goodwin3.data
- goodwin3.task
- goodwin5.xml
- goodwin5.data
- goodwin5.task

After importing all these files you can open one of the task definitions (`goodwin3.task` or `goodwin5.task`) in the editor. Unfold the algorithm parameters box to see the parameters for the population-based MCMC, and open the priors pane to see the priors.

You can start population-based MCMC by pressing the `Run task` button in the main pane.

You can monitor the progress of the sampler in the results pane (see Figure 4.1). Unlike the results pane for Metropolis-Hastigs sampler described in Section 3.4, the

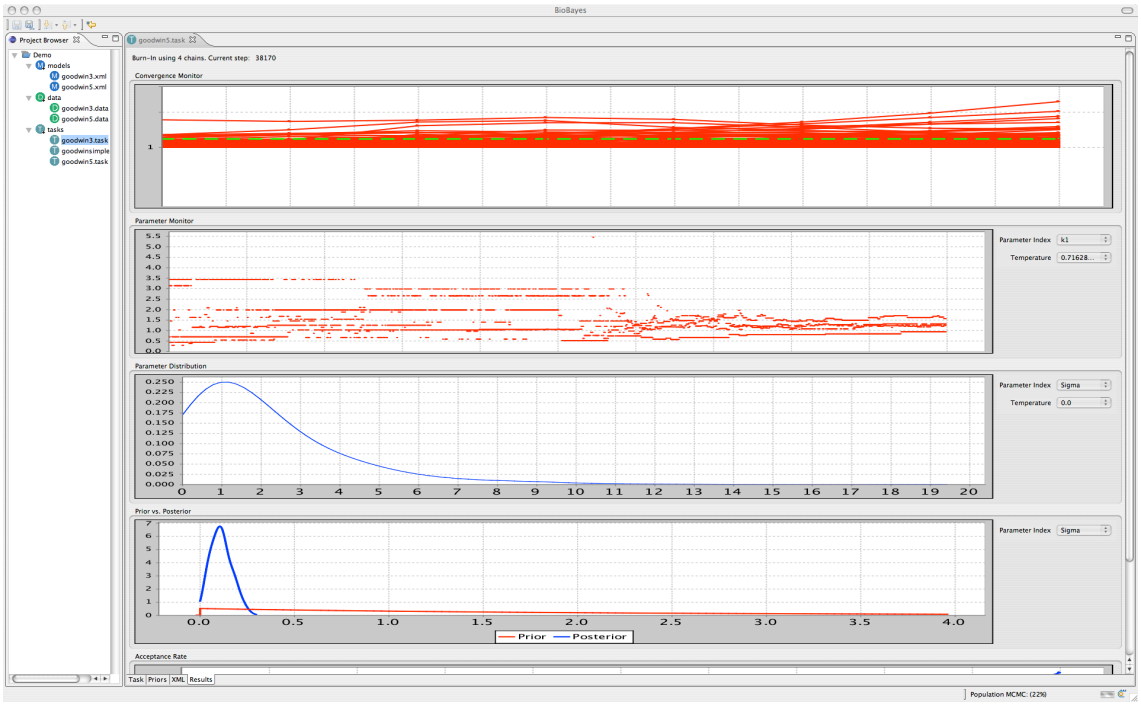

Figure 4.1: The task editor for `goodwin5.task` with the `Results` pane open.

plots in this pane allow the user to switch between different tempering temperatures as well as between different parameters.

The resulting parameter posterior from `goodwin3.task` is displayed in Figure 4.2, and is very sharp around the true parameter values used for data generation.

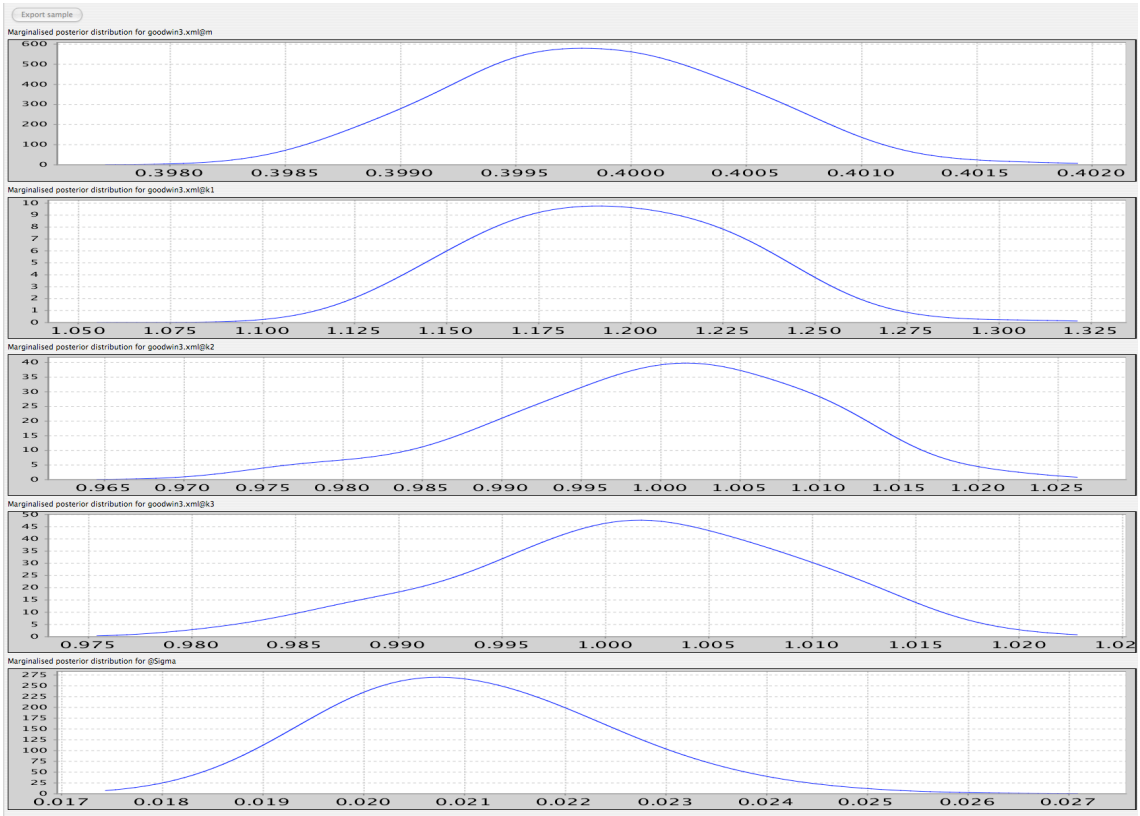

Figure 4.2: The parameter inference results for `goodwin3.xml` model. The parameter values used for data generation are $m = 0.4$, $k_1 = 1.2$, $k_2 = k_3 = 1$, $\sigma = 0.02$.

5 Model Ranking

Model ranking using the methods of Bayesian inference can be performed for consistent hypotheses testing (see Vyshemirsky and Girolami, 2008). To compare two alternative models by the weight of evidence supporting them, one needs to compute a value called the Bayes factor:

$$B_{12} = \frac{p(D|M_1)}{p(D|M_2)}$$

which is a ratio of the marginal likelihoods for these two alternative models.

In the case of nontrivial models these marginal likelihoods cannot be evaluated precisely, and have to be estimated using Monte Carlo integration procedures. We include such estimators with BioBayes.

5.1 Annealing-Melting Integration

Annealing-melting integration can be used to compute marginal likelihoods, the quantity used for evidence-based ranking of alternative models. This algorithm is based on the population-based MCMC sampler described in the previous chapter. The samples from the tempered sequence of target distributions are used to estimate the marginal likelihoods with thermodynamic integrals.

Several population-based MCMC samplers are run simultaneously to evaluate their convergence to the true posterior distribution, and at the same time the standard deviation of the final estimate is computed using this set of simultaneous samplers.

5.2 Running the estimator and using the results

If you have not imported the models and the datasets for the nonlinear oscillators while reading the previous chapter, do so. You will need the following files to be imported from the supplied tutorial package:

- `goodwin3.xml`
- `goodwin5.xml`
- `goodwin3.data`

- `goodwin5.data`

Also import the prepared task definitions for evaluating four marginal likelihoods:

- `goodwin3simple.task` — to evaluate the marginal likelihood of reproducing the data from model (4.1) with model (4.1).
- `goodwin3complex.task` — to evaluate the marginal likelihood of reproducing the data from model (4.2) with model (4.1).
- `goodwin5simple.task` — to evaluate the marginal likelihood of reproducing the data from model (4.1) with model (4.2).
- `goodwin5complex.task` — to evaluate the marginal likelihood of reproducing the data from model (4.2) with model (4.2).

It is generally reasonable to use more than 20 parallel chains in the tempering schedule, and from 5 to 10 simultaneous samplers to estimate the convergence of the chains and the variance of the estimate.

In some cases the convergence of all the chains to the true posterior may be difficult to achieve, so it might be helpful to disable the convergence monitoring and define some large enough limit for the number of burn-in steps. This will sacrifice the variance of the estimate, but provide the results faster. In practice we found that 200,000 steps usually serve well as a limit for the burn-in.

Monitoring of the task execution is identical to what was described in Section 4.3. However, instead of displaying the projections of the parameter posterior, the final screen will contain an estimate for the logarithm of the marginal likelihood.

We estimated the logarithms of the marginal likelihoods for the problems described above as in Table 5.1. And corresponding Bayes factors demonstrate extremely strong evidential support for the models used originally for simulating the data.

	`goodwin3.data`	`goodwin5.data`
`goodwin3.xml`	361.1978 ± 0.3120	-81.6697 ± 0.5016
`goodwin5.xml`	96.5167 ± 0.5998	353.2483 ± 2.9273

Table 5.1: The logarithms of the marginal likelihoods for nonlinear oscillator models.

6 Detailed Description of the User Interface Elements

6.1 Main Menu

Main menu of BioBayes consists of three categories: *Projects*, *Window* and *Help*.
The items available in the *Projects* category are:

New... This item is used for creating new files and projects. A standard Eclipse wizard for creating new files is started when this item is selected.

Import... This item can be used for importing external models, datasets or task descriptions into user's project.

Save This item is used for saving file which is currently open in the editor (right) area. This item is disabled if the file has not been modified since last saving.

Save As This item is used for saving file which is currently open in the editor (right) area with a different name.

Preferences... This item opens a preferences pane for the programme. User can change the number of ODE solvers run in parallel using this pane. Other standard options accessible from the preferences pane are keyboard shortcuts for common operations and general appearance of the programme interface.

Exit This menu item terminated the execution of the programme.

Only one operation is accessible through the *Window* category, it is *Window* → *Open View* → **Other...** This item allows user to open one of the standard views of the programme. This item can be used to reopen the Project Browser if it was closed by mistake.

Help category provides the following options:

About BioBayes This item displays a standard information window where additional information on BioBayes configuration may be found. Such additional information includes details on the versions of separate plugins included with the programme.

Update... This item initiates the programme update process. The programme will try to contact the update web sites (this includes but is not limited to Glasgow University web site), compare available versions of the programme modules [plugins] to the list of modules currently installed on user's computer, download and install new versions of the modules if they are available.

Add New Features... This item allows users to install new modules [plugins] which are published at the updates web site and are not extensions or updates of any of the installed modules.

Manage Extensions... This item allows users to see and manage groups of modules which are installed.

6.2 Project Browser

Project Browser is one of the standard views which is typically located in the left side of the user interface. The project browser displays a tree structure of user workspace. Projects form the top level of the workspace structure. Each project is split into three categories: *models*, where all the models are stored; *data*, where all the datasets are stored; and *tasks* where problem descriptions are collected.

If you have closed this view by mistake, you can reopen it through *Window* → *Open View* → *Other...* menu.

6.3 Model Editor

Model editor is a window which opens in the editor area when user selects a model in the Project Browser. There are two tabs In the bottom part of the editor window: *Summary* and *XML*.

The *Summary* tab displays general information about the model. The very first textual line in this tab displays the name of currently open model. Number of equations (species) and model parameters are listed in the *Model Structure* box next. And the final box, called *Simulation* demonstrates a typical behaviour of the model. Users can change the simulation time and the number of intermediate simulation steps using the controls provided.

The second tab, called *XML* allows users to edit XML representation of the models manually.

BioBayes does not provide any graphical tools for editing SBML models, however external programmes may be used for such editing.

6.4 Dataset Editor

Dataset editor opens in the editor area when user selects a dataset in the Project Browser. This editor has two tabs accessible form the bottom part of the editor. The first one is

called *Data* and the second one is *XML*. The *XML* tab can be used for manual editing of the XML representation of the dataset. *Data* tab is the main visual editor of the datasets.

The first line in the *Data* tab allows users to define a short descriptive name for the dataset. The second line enables the selection of the noise model used with data defined in current dataset. The options are *Normal* and *Log-Normal* which define standard normal and log-normal error models correspondingly.

The next group of controls in the dataset editor is actual table of data. *Edit Columns* button opens a window depicted in Figure 6.1.

Rows can be removed from the data table by pressing crossed buttons to the very left column of data table representation. Adding new rows is possible by pressing a blue "plus" button below the main table. The actual data values can be edited by clicking them in the table. We provide a Matlab function for converting Matlab matrices into BioBayes datasets. The function is available from BioBayes official web site, in the downloads area.

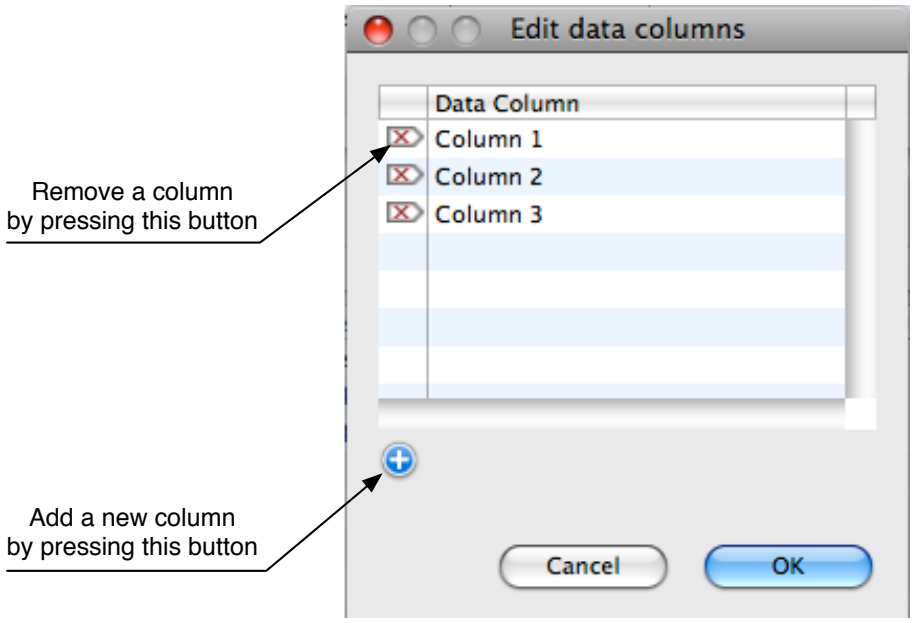

Figure 6.1: A window which allows adding and removing columns from datasets.

The group of controls named *Data Interpretation* (depicted in Figure 6.2) is used to define how each value in the data table will be treated while simulating the model. This table matches one of the available interpretation types to each of the data columns. There are four interpretation types available:

Time corresponds to columns which index time during the model simulation. The readouts in the corresponding data row will then be matched to model simulation at this particular simulation time. This column type does not require any additional parameters.

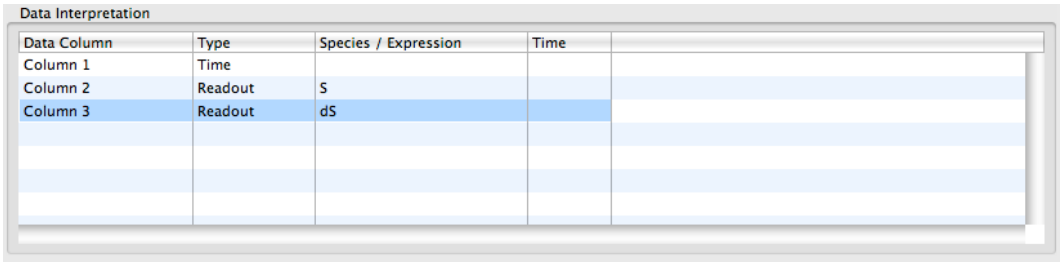

Figure 6.2: A group of controls which define how each value in the data table will be treated while simulating the model.

Initial corresponds to different values that have to be used as initial values for model variables during separate simulations. For example, if the experiment was conducted using titration, different concentrations of the titrant can be defined using this column type. This column type requires *Species / Expression* parameter to be additionally defined. The value of this parameter corresponds to the symbolic name of the model variable which will be set to corresponding initial values during the simulation.

Change corresponds to columns which define sudden changes of the model variables during the simulation. This can be used, for example, if some stimulus is added in the later stage of the experiment. This column type requires two parameters: *Species / Expression* which defines the symbolic name of the variable which will be updated during the simulation, and *Time* which defines a specific simulation time at which such update has to be made.

Readout corresponds to specific measurement made in the experiment. This column type requires *Species / Expression* parameter to be defined. The variable name defined as the *Species / Expression* parameter will then be matched to data defined in this dataset.

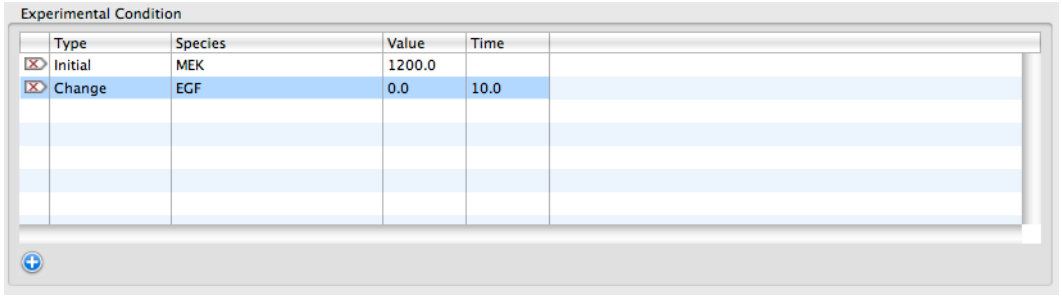

Figure 6.3: A group of controls which defines initial values and sudden value changes during the simulation which are global for all of the conditions defined in the current data set.

The next group of controls called *Experimental Condition* (depicted in Figure 6.3)) defines initial values and sudden value changes during the simulation which are global for all of the conditions defined in the current data set. There are only two types of such conditions: *Initial* and *Change*. The interpretation of these types match similar column interpretation types defined above.

6.5 Task Editor

Task editor opens in the editor area when user selects a task definition in the Project Browser. This editor has four tabs accessible form the bottom part of the editor:

Task Task parameters, models and datasets used in the task, and *"Run Task"* button are placed in this tab.

Priors Prior distributions for model parameters are defined in this tab.

XML This tab can be used for manual editing of the XML representation of the task definition.

Results This tab is used to monitor the execution of the task, and for reading the results when the task is finished.

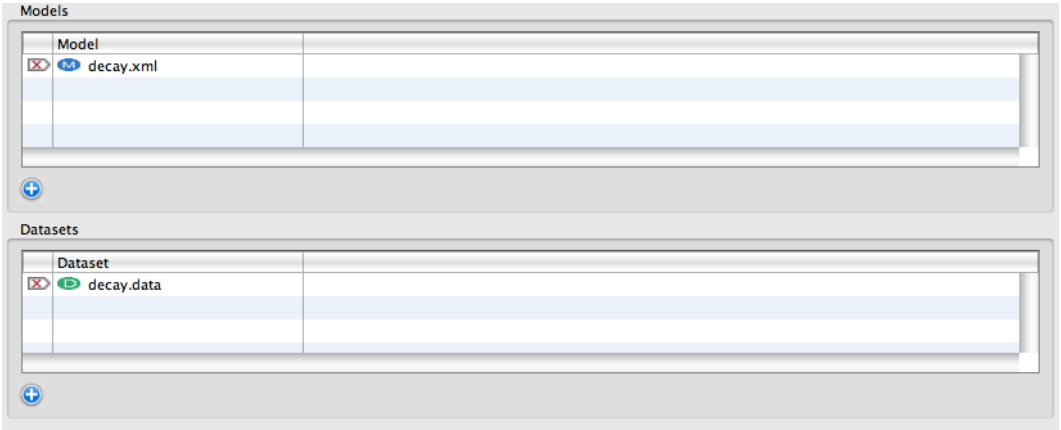

Figure 6.4: *Models* and *Datasets* tables define which SBML models and datasets have to be used in this task.

The *Task* tab contains the *"Run Task"* button which runs the task when pressed. The *Models* and *Datasets* tables depicted in Figure 6.4 define which SBML models and datasets have to be used in this task. Adding new elements and removing existing elements from these tables can be performed by pressing blue "plus" and red crossed "remove" buttons.

The *Task parameters* group of controls depicted in Figure 6.5 defines general properties of the task. The first line of this group defines a descriptive name of the task, and

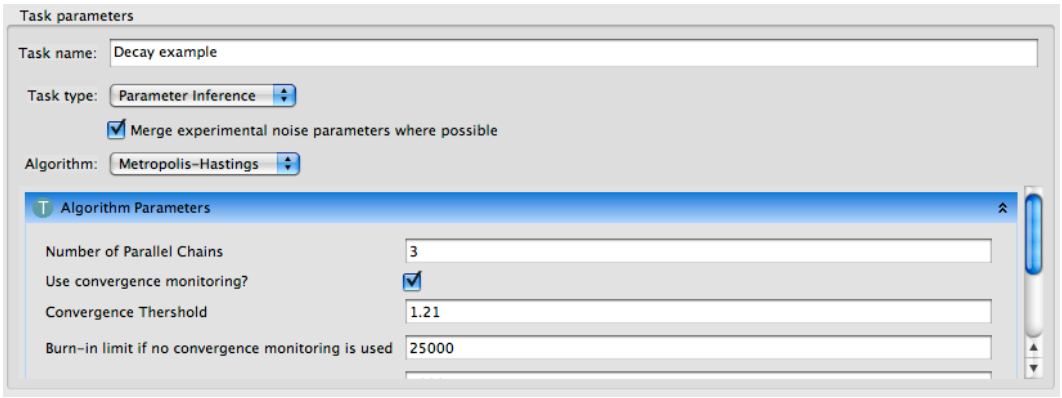

Figure 6.5: The *Task Parameters* group is used to define general properties of the task.

the second line allows users to select which type of task has to be used. The default installation of BioBayes defined two types of tasks: *Parameter Inference* and *Model Comparison*, however, additional task types may be defined by third party plugins.

Merge experimental noise parameters where possible option defines whether one common noise random variable has to be used for the statistical model employed in the task, or a separate variable has to be created for each of the datasets employed.

Algorithm selection box allows users to choose which algorithm to use for the task. The default version of BioBayes supports three algorithms: *Metropolis-Hastings* and *Population MCMC* for parameter inference and *Annealing-Melting Integration* for model comparison. The list of available algorithms may be extended by third party plugins.

Algorithm Parameters group is specific for each of the available algorithms, and defines the parameters used while executing the algorithm. The default algorithms supplied with BioBayes use the following parameters, *Metropolis-Hastings* algorithm parameters:

Number of Parallel Chains defines the number of Markov chains to be run simultaneously and independently. Samples produced with these independent chains are used to judge the convergence of the Markov chains to the posterior distribution according to Gelman-Rubin $\hat{R}$ statistic.

Use convergence monitoring? This option defines whether Gelman-Rubin $\hat{R}$ statistic must be used for judging the convergence of the Markov chains to the true posterior distribution, or a hard limit of the proposal steps during the "burn-in" phase of the algorithm has to be used instead.

Convergence Threshold is the critical value for the $\hat{R}$ statistic used for convergence monitoring. Once the statistic for all of the model variables reaches a value less or equal to the selected critical value, the chains are judged as converged to the true posterior, and the main sample will be produced starting from the current state of the chains.

Burn-in limit if no convergence monitoring is used is a hard limit on the number of proposals during the "burn-in" phase of the algorithm. This option is only used if *Use convergence monitoring?* option is not selected.

Posterior Sample Size defines the size of the posterior sample produced after the Markov chains are judged to be sampling from the true posterior distribution. This sample can be exported to be used with external tools after the task is finished.

Sample thinning. Take every Nth allows users to perform sample thinning and take only some of the Markov chain states into the actual posterior sample. The values different from 1 are useful to reduce autocorrelation of the posterior sample, however such values require longer time for task execution.

Plots refresh rate defines how often the plots in the *Results* tab have to be updating when executing the task.

Population MCMC algorithm parameters:

Number of Parallel Chains defines the number of Markov chains to be used in Parallel Tempering schedule of the Population MCMC algorithm.

Number of Simultaneous Samplers defines the number of Population MCMC schedules to be run simultaneously and independently. Samples produced with these independent samplers are used to judge the convergence of the Markov chains to the posterior distribution according to Gelman-Rubin $\hat{R}$ statistic.

Use convergence monitoring? This option defines whether Gelman-Rubin $\hat{R}$ statistic must be used for judging the convergence of the samplers to the true posterior distribution, or a hard limit of the proposal steps during the "burn-in" phase of the algorithm has to be used instead.

Convergence Threshold is the critical value for the $\hat{R}$ statistic used for convergence monitoring. Once the statistic for all of the model variables reaches a value less or equal to the selected critical value, the samplers are judged as converged to the true posterior, and the main sample will be produced starting from the current state of the chains.

Burn-in limit if no convergence monitoring is used is a hard limit on the number of proposals during the "burn-in" phase of the algorithm. This option is only used if *Use convergence monitoring?* option is not selected.

Posterior Sample Size defines the size of the posterior sample produced after the samplers are judged to be sampling from the true posterior distribution. This sample can be exported to be used with external tools after the task is finished.

Sample thinning. Take every Nth allows users to perform sample thinning and take only some of the Markov chain states into the actual posterior sample. The values different from 1 are useful to reduce autocorrelation of the posterior sample, however such values require longer time for task execution.

Plots refresh rate defines how often the plots in the *Results* tab have to be updating when executing the task.

Annealing-Melting Integration algorithm utilises the *Population MCMC* sampler, and has precisely the same algorithm parameters.

Figure 6.6: The *Priors* tab of the task editor is used to define the prior distributions for the model parameters.

The *Priors* tab in the task editor (depicted in Figure 6.6 is used to define the prior distributions for the model parameters. The value of the model parameter can either be fixed to a certain value, or assigned one of the five prior distributions: *Uniform*, *Gaussian*, *Gamma*, *Log-Normal* or *Inverse Gamma*.

The *Results* tab is used to monitor the execution of the task, and for reading the results when the task is finished. The contents of this tab depend on the algorithm used. Common types of the monitoring plots are:

Convergence Monitor depicts current values for the Gelman-Rubin $\hat{R}$ statistic if it is used for the convergence monitoring.

Parameter Monitor depicts current samples used in the Markov chains (as dots). This plot is parametrised by the model parameter name, and tempering temperature if used in Population MCMC setting.

Parameter Distributioin depicts a kernel smoothed version of the current posterior sample. This plot is parametrised by the model parameter name, and tempering temperature if used in Population MCMC setting.

Prior vs. Posterior compares a kernel smoother plot of the parameter prior to the kernel smoothed plot of the current approximation of the parameter posterior. This plot is useful for assessing information gain from the experimental data.

Acceptance Rate plots the proportion of the proposed parameter values which have been accepted into the sample. The best sampling performance is typically achieved when these values are between 25% and 40%.

When the task is completed, the *Results* tab will usually display marginal distributions of model parameter posterior for *Parameter Inference* tasks, or the estimated value of the marginal likelihood for *Model Comparison* tasks. *Parameter Inference* tasks additionally allow exporting the posterior sample as a text file, which then can be used with external tools for additional analysis, these tasks also report an estimate of the marginal likelihood computed using the *Posterior Harmonic Mean* estimator.

7 Programme Interface

7.1 Schemata

Schemata for the XML descriptions of data and task definitions used in BioBayes are available as a download from the corresponding area of the official web site.

7.2 Converting Matlab Data into Datasets

A Matlab function for converting matrices into BioBayes datasets has been developed by request of BioBayes users. This function is available for download from the *downloads* area of the official web site.

Parameter posterior samples can be imported to Matlab after manually removing the first line of the exported file. To load the edited file into Matlab as a matrix use

```
P = load('filename.txt');
```

command.

Acknowledgements

This research is funded by Microsoft Research within the "Modelling and Predicting in Biology and Earth Sciences 2006" programme.

Mark Girolami is an EPSRC Advanced Research Fellow, EP/E052029/1.

Bibliography

Burbeck, S. and Jordan, K. E. (2006).
An assessment of the role of computing in systems biology.
IBM J. RES & DEV., **90**(6), 529–543.

de Jong, H. (2003).
Modeling and simulation of genetic regulatory systems: a literature review.
Lecture Notes in Computing Science, **2602**, 149–162.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995).
Bayesian Data Analysis.
Chapman & Hall.

Jasra, A., Stephens, D. A., and Holmes, C. C. (2007).
On population-based simulation for static inference.
Stat Comput, **17**, 263–279.

Lawrence, N. D., Sanguinetti, G., and Rattray, M. (2007).
Probabilistic inference of transcription factor concentrations and gene-specific regulatory activities.
In B. Schölkopf, J. C. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA.

M. Hucka *et al.* (2003).
The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models.
Bioinformatics, **19**(4), 524–531.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2002).
Numerical Recipes in C++: The Art of Scientific Computing.
Cambridge University Press.

Rogers, S., Khanin, R., and Girolami, M. (2006).
Bayesian model-based inference of transcription factor activity.
BMC Bioinformatics, **8**(2).

Voit, E. O. (2000).
Computational Analysis of Biochemical Systems.
Cambridge University Press.

Vyshemirsky, V. and Girolami, M. A. (2008).
Bayesian ranking of biochemical system models.
Bioinformatics, **doi:10.1093/bioinformatics/btm607**.