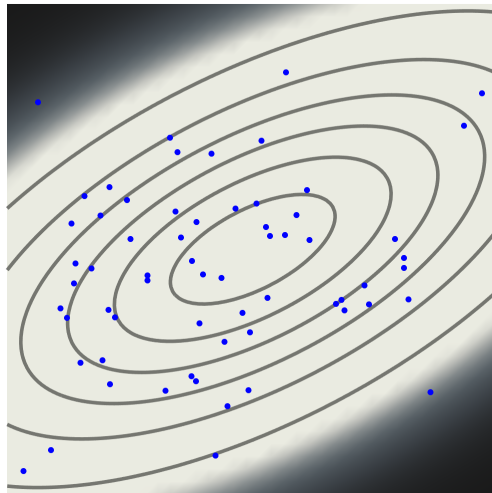


MacMCMC (v1.6) User Guide

Michael P. McLaughlin

November, 2022



***MacMCMC* (v1.6) User Guide**

Copyright © 2022 by Michael P. McLaughlin. All rights reserved.

DISCLAIMER:

MacMCMC is intended solely as a tool for the convenience of users. It makes no guarantee or warranty of any kind that its use and/or its output are appropriate for any purpose. All such decisions are at the discretion of the user.

Any brand names and product names included in this work are trademarks, registered trademarks, or trade names of their respective holders.

This document created using \LaTeX and TeXShop.

Figures created using *MacMCMC* or [Mathematica](#)TM (with [SetAxes](#))

Contents

A Word to the Wise (and others)	v
1 Quickstart	1
1.1 Things To Try	1
1.2 System Note	2
2 Basic Procedure	3
2.1 Installation	3
2.2 Menus	3
2.3 Input	5
2.4 Setup Options	7
2.5 Run Status	9
2.6 Output	10
3 Modeling Language	13
3.1 Models	13
3.2 Mixtures	18
3.3 What Might Go Wrong?	21
4 Goodness-of-fit	23
4.1 General Procedure	23
4.2 Credible Interval	24
4.3 Goodness-of-fit Examples	24
A Distributions	34
A.1 Continuous	34
A.2 Discrete	37
A.3 Mixtures	38
A.4 Generic	39
B Functions	40
B.1 Operators	40
B.2 Functions	41

B.3	Reserved Constants	42
C	Technical Details	43
C.1	Software	43
C.2	MCMC	43
C.3	Marginal Likelihood	45
C.4	Mixture Relabeling	45
C.5	Marginal-plot Smoothing	46
C.6	Goodness-of-fit Credible Intervals	46
D	Examples	47

List of Figures

1.1	Plot Showing Goodness-of-fit (mean estimates)	2
2.1	Initial Menu	4
2.2	Analyze Menu	5
2.3	Enzyme Model	6
2.4	Data Window	7
2.5	Setup Dialog	8
2.6	Status Window (Sampling phase)	9
2.7	Status Window (Done)	10
2.8	Report Window	11
2.9	Marginal for Parameter A	12
3.1	Enzyme Model2	14
3.2	Marginal for Parameter sig	17
3.3	Model vs. Data: With and Without Model Error	18
3.4	Salaries: Mixture Model	19
3.5	Salaries: Data and Model (mean estimates)	21
4.1	Daytime Model	25
4.2	Goodness Dialog for Daytime Example	26
4.3	Goodness-of-fit for Daytime Example	26
4.4	Hale-Bopp Model	27
4.5	Goodness-of-fit for Hale-Bopp Example	28
4.6	Body-temperature Model	29
4.7	Goodness Dialog for Body-temperature Example	29
4.8	Goodness-of-fit for Body-temperature Example	30
4.9	Hyphens Model	31
4.10	Goodness-of-fit for Hyphens Example	32
4.11	Marginals for Zero-count and Two-count	33

List of Tables

- 3.1 Results for model1 17
- 3.2 Results for model2 17
- 3.3 Results for Salaries 20

- D.1 Examples 48
- D.2 Functionality vs. Examples 48

A Word to the Wise (and others)

The DISCLAIMER on the copyright page says it all but perhaps you missed it so here it is again with further elaboration.

First, *MacMCMC* is a (MacOS-only) application for performing Bayesian inference by carrying out MCMC analyses as described in detail in *Data, Analysis and Inference*, a free ebook which constitutes the second half of this published offering. Users not already very familiar with this approach to data analysis are **strongly** urged to read this book. This *User Guide* is only software documentation; it says almost nothing about how to carry out Bayesian inference. The cited ebook does, with many examples, including most of the examples in this Guide. [6]

Second, *MacMCMC* takes user-defined models as input and the opportunities to make a mistake in creating such input can often prove embarrassing. It is even possible to construct a model that will cause *MacMCMC* to crash! The usual reason is that array indices, or their priors, are defined in such a way as to overrun the actual array bounds (see pg. 21). An MCMC model is like a little computer program in many respects. Be warned!

MacMCMC will probably catch most syntactic errors (typos, misspellings, mismatched parentheses, etc.) but it will not catch semantic errors. These arise because the model does not, in fact, describe what the user thinks it does. Bayesian inference is extremely powerful when done correctly but it requires careful thought and a fair amount of effort.

If you are looking for software to provide quick-and-dirty answers, look elsewhere.

MICHAEL P. McLAUGHLIN
MCLEAN, VA
NOVEMBER, 2022
MPMCL 'AT' CAUSASCIENTIA.ORG

Chapter 1

Quickstart

THIS chapter is for those who cannot wait to see the program working. Assuming that *MacMCMC* is installed in (copied to) the Applications folder, follow the six steps below to run the enzyme example. This is a weighted, nonlinear regression analogous to traditional [least-squares](#). The input (*enzyme.mcmc*, *enzyme.dat*) and non-optional output (*report.txt*, *trace.txt*) will be described in detail in the following chapters.

It is assumed, here and elsewhere, that you are familiar with the MacOS user interface and standard Finder operations. To see *MacMCMC* in action, do the following:

1. Go to the Quickstart folder.
2. Double-click *enzyme.mcmc* (or Open this model from the application).
3. Input the data (File/Data... or Shift-Command-D) and select *enzyme.dat*.
4. Compile the model (Analyze/Compile or Command-K).
5. Do Setup (Analyze/Setup... or Command-M).
6. Click the Run button (in Setup dialog).

The program will execute in about ten seconds¹ producing two output files in the local (data-file) folder. The screen will contain two new windows: one for the (saved) Report and one (unsaved) for the first of three marginal plots corresponding to the three Monitored unknowns in the model (two parameters plus one “Extra”).

1.1 Things To Try

If you Restart (File/Restart or Command-R), the model must be recompiled. In that case, *report.txt* and *trace.txt* (saved automatically) will be overwritten. Restart and increase Setup option # **Samples per walker** to get smoother marginals (and a longer trace).

¹on a standard 2017 iMac

Try the Smoothing tool in the marginal window for parameter A and note that the axis labels may be edited (but not the tick marks). All plots may be saved as PDF or PNG.

Select parameter B (or RSq) from the drop-down menu in the Status window then click Reset to get the corresponding marginal plot.

MacMCMC can usually make its own goodness-of-fit plots if the model so specifies (see Chapter 4) but you can also make them from the posterior output in the trace utilizing your favorite graphing software. An example is shown in Figure 1.1.²

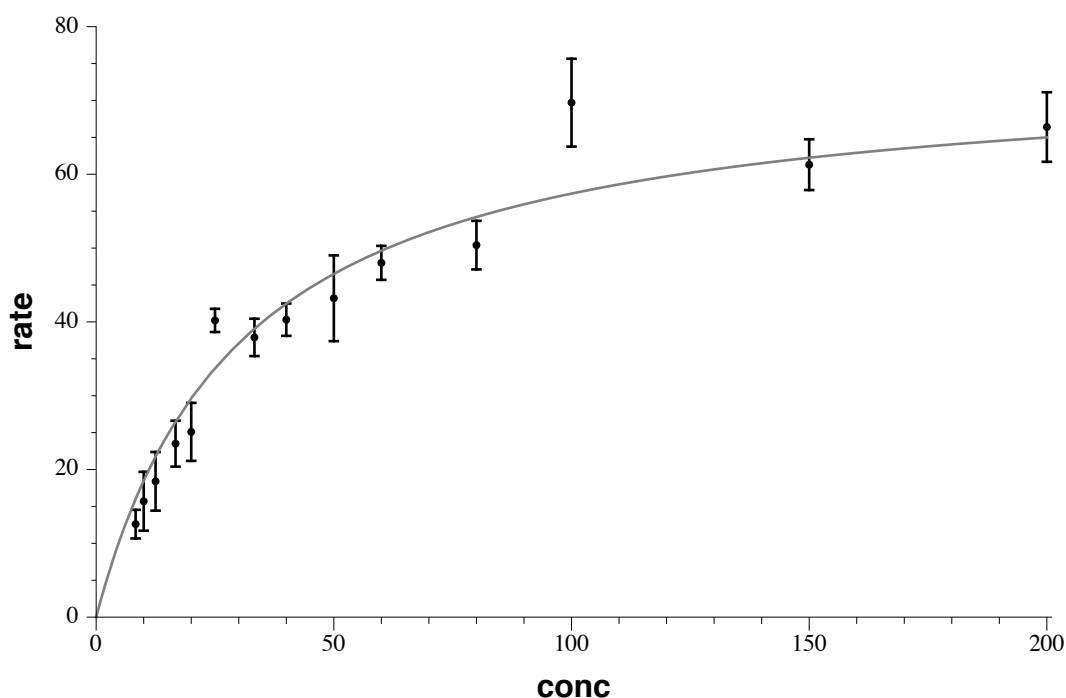


Figure 1.1: Plot Showing Goodness-of-fit (mean estimates)

If you understand how MCMC works, then you know that you can use the trace to answer *any question* about the analysis provided that the model is valid and the MCMC run successful.

1.2 System Note

This was a very short MCMC run. In some cases, with a very large amount of data and a large model, runs may take a long time to execute. The system Energy Saver preferences should be set to “sleep Never” or equivalent; otherwise, a really long run might be halted before it is finished.

²created in Mathematica (with SetAxes)

Chapter 2

Basic Procedure

IN this Guide, all of the operational features of *MacMCMC* are described in roughly the order in which they are encountered when running the program. This includes installation, menus, dialogs and input/output. Details on the requirements and structure of model pseudocode are described in the following chapter.

2.1 Installation

There are no special requirements regarding the installation of *MacMCMC*. Just follow the instructions in the file *INSTALLATION.rtf*.¹ Place *MacMCMC* in the Applications folder and other files in this package wherever it is convenient. Adding *MacMCMC* to the Dock is optional.

Note: While *MacMCMC* is running, there will always be one or more hidden sub-processes running as well. They will all be named *MacMCMC_M2C2*. These processes are “headless” and cannot be seen except in the Activity Monitor. These sub-processes of *MacMCMC* will be terminated whenever the main application Quits in the normal fashion.

If you have not already done so, this would be a good time to carry out the steps listed in the Quickstart chapter so that the material in the following Sections will be familiar.

2.2 Menus

Here, we discuss only those menu items specifically relevant to *MacMCMC*. The rest are common to all MacOS applications.

All menu screenshots depict the appearance of menus for the usual startup with many items disabled (dimmed) when not appropriate.²

¹updated for MacOS Catalina and later

²Theses screenshots were taken with MacOS Ventura.™

2.2.1 *MacMCMC* and File Menus

These two menus are typical of almost any application.

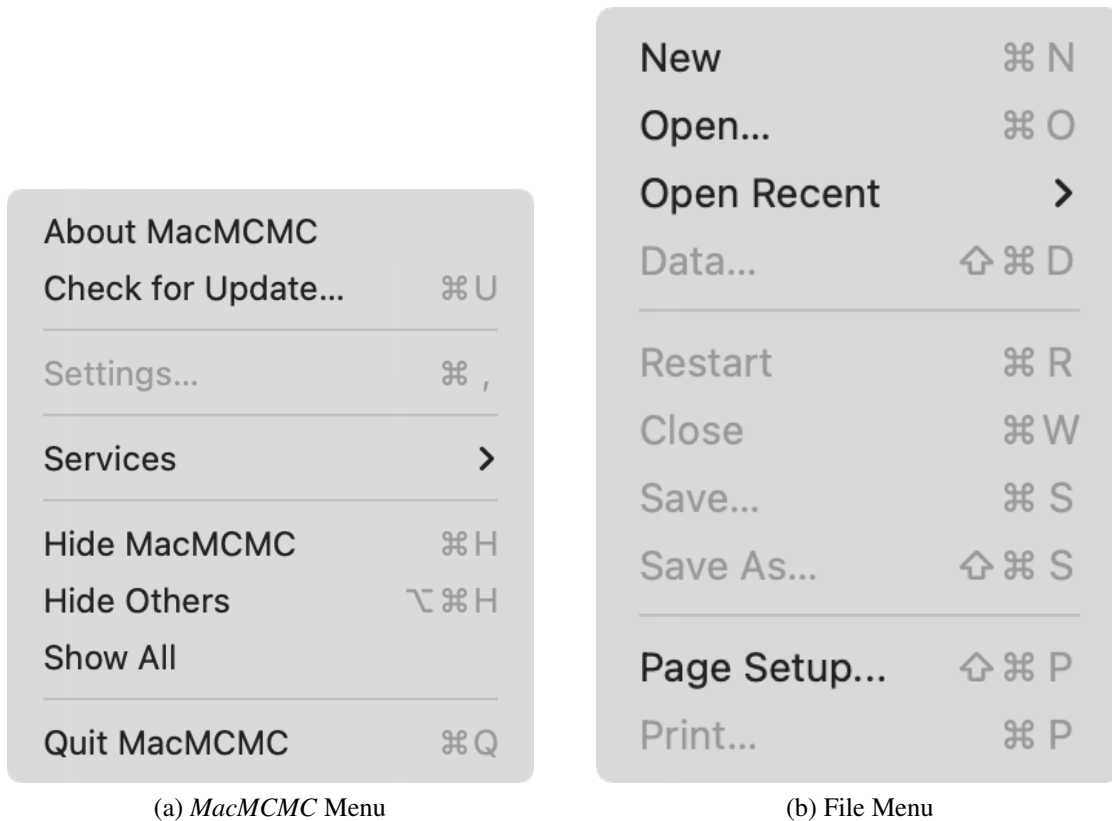


Figure 2.1: Initial Menus

The primary point of interest in the *MacMCMC* menu is the Check for Update... item. This will *not* update *MacMCMC* in place; it merely points the user to the website when there is a later version since other package elements might also be new.

The File menu as shown above is waiting for a model (*.mcm*) file. If *MacMCMC* is opened by double-clicking or drag-and-dropping a model file, the File menu will appear with the Open items dimmed and the Data item enabled, waiting for a data file. The next step, compilation, cannot take place without both a model and some data. The model must be input first.

Only one analysis can be done at a time.

2.2.2 Analyze Menu

Analysis consists in applying a model to the observed data. This model is a user input and there is, of course, **no guarantee whatever** that it is correct (or even sensible).³ Before the model can be applied, it must first be compiled then Setup options chosen. These actions are accomplished through the Analyze menu.

Given a model and data, this Analyze menu will appear as follows:

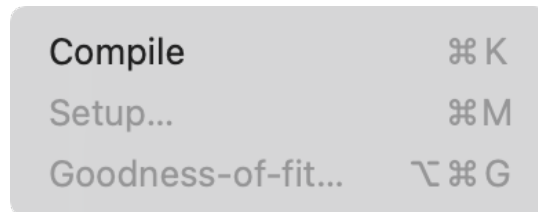


Figure 2.2: Analyze Menu

Successful compilation will enable the Setup... menu. Editing the model, which can be done in place, will require recompilation. The goodness-of-fit procedure, if any, cannot be selected until everything else has finished (see chap. 4).

2.3 Input

Model and data files must be plain ASCII (UTF-8) textfiles with **no** styling, accents, etc.

2.3.1 Model

All model files must have extension *mcmc*. Models are editable. Blank lines are ignored. C++ and C comments are permissible. Figure 2.3 displays the model window (vertically extended to show everything) for the Quickstart example.

The seven sections, {Constants: ... Monitored:} are mandatory. Extras, however, are optional so this section may be empty. There *must* be at least one uncertain parameter (from Bayes' Rule). At least one uncertain quantity *must* be Monitored.

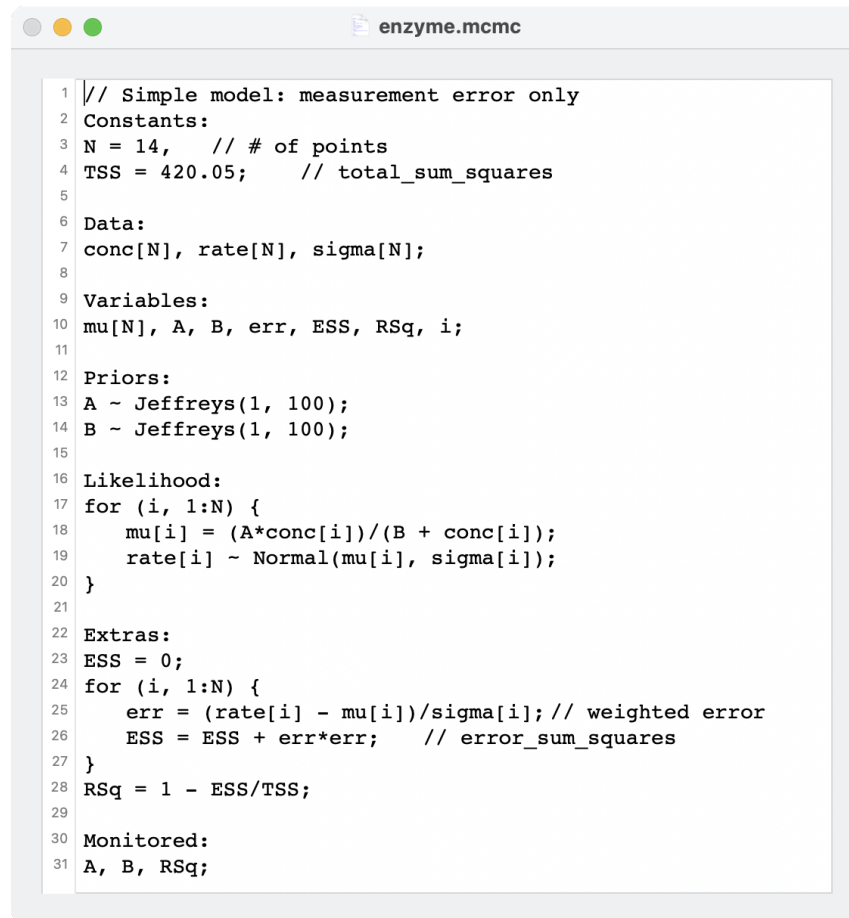
Choosing New from the File menu creates an empty (unsaved) model.

2.3.2 Data

All data files must have extension *dat*. Data files are *not* editable. Figure 2.4 shows a screenshot of the Data window for the Quickstart example.

MacMCMC does *not* support creating data files; these must be created and saved as **tab-delimited text** from some other application, e.g., a spreadsheet. Requirements and options include the following:

³See pg. v.



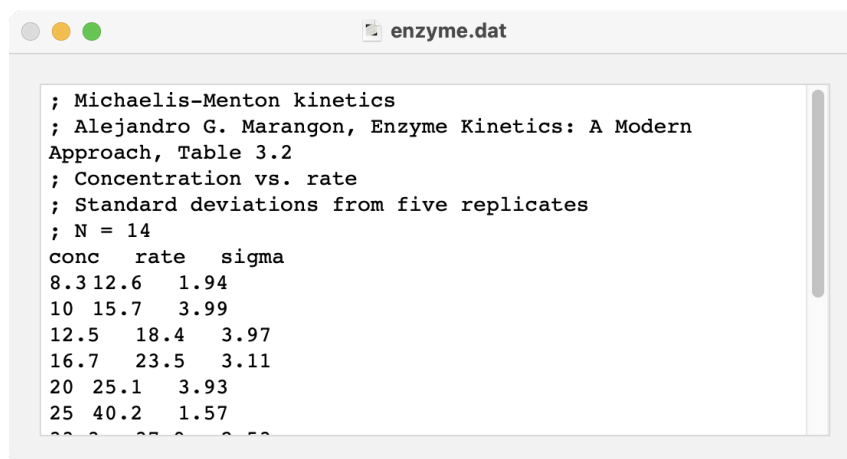
```

1 // Simple model: measurement error only
2 Constants:
3 N = 14,    // # of points
4 TSS = 420.05;    // total_sum_squares
5
6 Data:
7 conc[N], rate[N], sigma[N];
8
9 Variables:
10 mu[N], A, B, err, ESS, RSq, i;
11
12 Priors:
13 A ~ Jeffreys(1, 100);
14 B ~ Jeffreys(1, 100);
15
16 Likelihood:
17 for (i, 1:N) {
18     mu[i] = (A*conc[i])/(B + conc[i]);
19     rate[i] ~ Normal(mu[i], sigma[i]);
20 }
21
22 Extras:
23 ESS = 0;
24 for (i, 1:N) {
25     err = (rate[i] - mu[i])/sigma[i]; // weighted error
26     ESS = ESS + err*err;    // error_sum_squares
27 }
28 RSq = 1 - ESS/TSS;
29
30 Monitored:
31 A, B, RSq;

```

Figure 2.3: Enzyme Model

- The tab-delimited data array must be rectangular with one variable per column, as in the example, and *no blank lines or entries*. For multi-dimensional data, see below.
- A semicolon in the leftmost position indicates a full-line comment.
- A comment, beginning with a semicolon, may be appended to a data record.
- The first non-comment record must contain the data variable names. **There may be no comment on this line.**
- Data columns of unequal length must be filled (only at the bottom!) with asterisks to complete the rectangular array. See Example SAT.
- The data file must be completely consistent with the model, variable names and the number of datapoints in particular.



```

; Michaelis-Menton kinetics
; Alejandro G. Marangon, Enzyme Kinetics: A Modern
Approach, Table 3.2
; Concentration vs. rate
; Standard deviations from five replicates
; N = 14
conc  rate  sigma
8.3 12.6  1.94
10  15.7  3.99
12.5 18.4  3.97
16.7 23.5  3.11
20  25.1  3.93
25  40.2  1.57

```

Figure 2.4: Data Window

Multi-dimensional data

A data array may have up to three dimensions. Simple vectors are input as described above. If the observable has 2 or 3 dimensions, it must be input as a single, **row-major** column. Thus, the column for $x[10][3][2]$ (length = $10*3*2$) would be entered in the order given by the output of the following pseudocode:

```

for (r = 1;r <= 10;r++)
  for (c = 1;c <= 3;c++)
    for (d = 1;d <= 2;d++)
      Print(x[r][c][d]);

```

If there are 1-dimensional observables in addition to $x[10][3][2]$, they would have to be input as column vectors of length = 10 = (x rows) followed enough asterisks to make all data columns of equal length. See Example Body where the data are input as $WtHt[247][2]$.

Note: With multi-dimensional data, the reserved datum NA (\equiv not-available) is valid. What effect this will have on results will depend on the model.⁴

2.4 Setup Options

There are two levels of Options (see Fig. 2.5). In most cases, the defaults will suffice so nothing need be changed. The Run button will start MCMC. Cancel triggers a Restart.

⁴ NA is stored internally as $NaN = 0/0$.

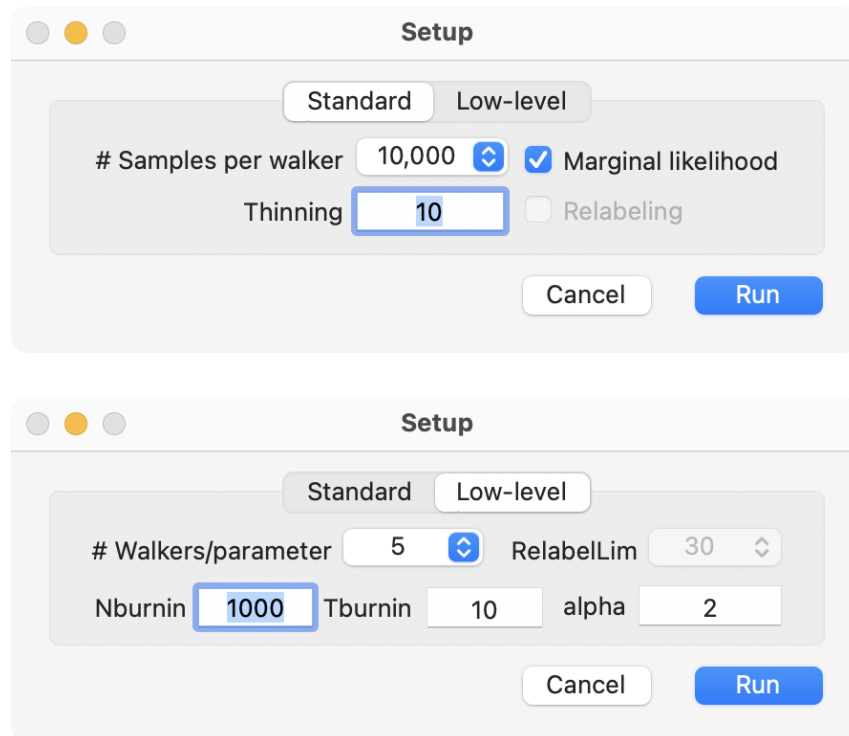


Figure 2.5: Setup Dialog

2.4.1 Standard Options

These Options are those most likely to be changed.

Samples per walker determines the size of the final sample. Since there will be several walkers (see Appendix C), the final sample will likely be a bit larger than expected so that sub-processes will all generate a sub-sample of the same size in order that the Gelman-Rubin statistic (see below) will be unbiased.

Thinning determines how many states are *skipped* for each trace entry (default = 9).

Marginal likelihood ON by default. Useful for model comparison (see Examples). Can be turned off to save time.

Relabeling Enabled only when the model is one of the three built-in mixtures (see below).

2.4.2 Low-level Options

These options adjust the internal workings of MCMC as it is implemented in *MacMCMC* (see Appendix C).

Walkers/parameter There must be at least two. The default (5) is a reasonable choice.

Nburnin and **Tburnin** control the burn-in phase.

RelabelLim Enabled only when the model is one of the three built-in mixtures.

Alpha Controls the step size for proposals and must be >1 . This Option is included only for the sake of completeness; there is no reason to change it.

2.5 Run Status

The Status dialog is displayed during and after a run. Figure 2.6 shows this dialog for the Quickstart example. The progress bar reflects all sub-processes (usually 3 or 7). The phase of the run (Burn-in, Sampling, Refining MAP parameters, Marginal-likelihood integration, ...) is shown below the progress bar. All sub-processes are in the same phase at all times (see Appendix C). A Cancel at any time aborts the run completely and executes a Restart.

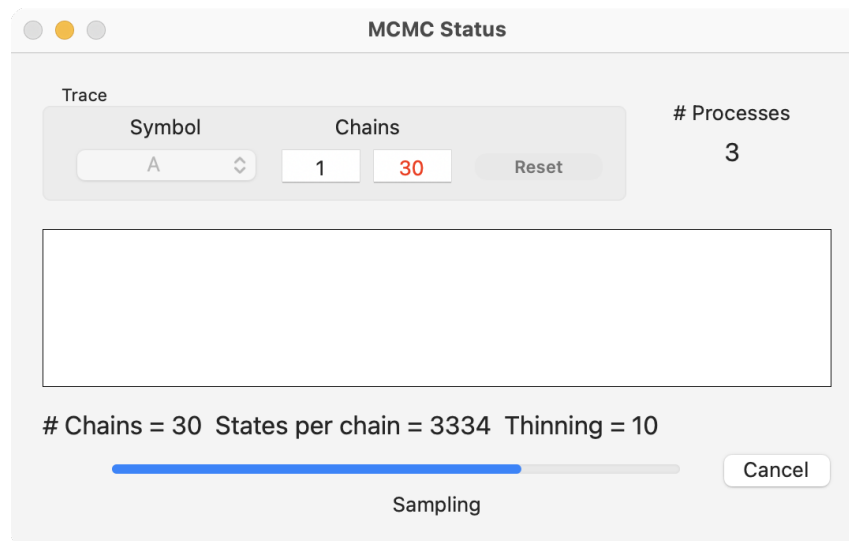


Figure 2.6: Status Window (Sampling phase)

When the run is finished, the Status dialog will show a pair of trace plots (black and red) for two selected chains (walkers) for the first of the monitored variables in the model, indicated by Symbol (see Fig. 2.7). This is a typical MCMC trace plot and serves to indicate whether these two chains were well mixed (as they should be). In *MacMCMC*, there are many more chains than usual so there are many chances for a pair of chains to be poorly mixed. If the model is working well, this should not happen. If mixing is poor, then this can usually be fixed by lengthening the burn-in phase. However, if the model is somehow inappropriate, poor mixing may be unavoidable. If, for any reason, the trace is bad (e.g., a column is constant), this will abort the run with an error message followed by a Restart.

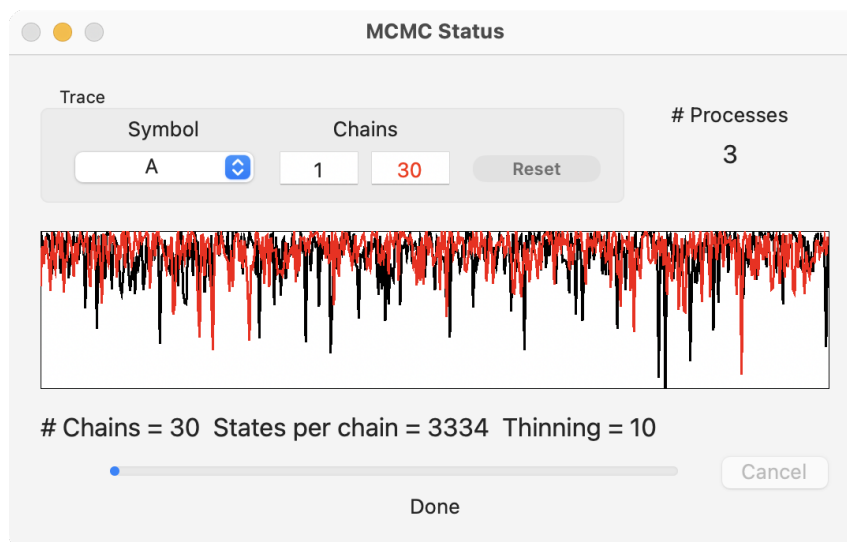


Figure 2.7: Status Window (Done)

The displayed symbol, and/or the pair of chains, can be changed but *Reset must* be clicked for the change to take effect.

2.6 Output

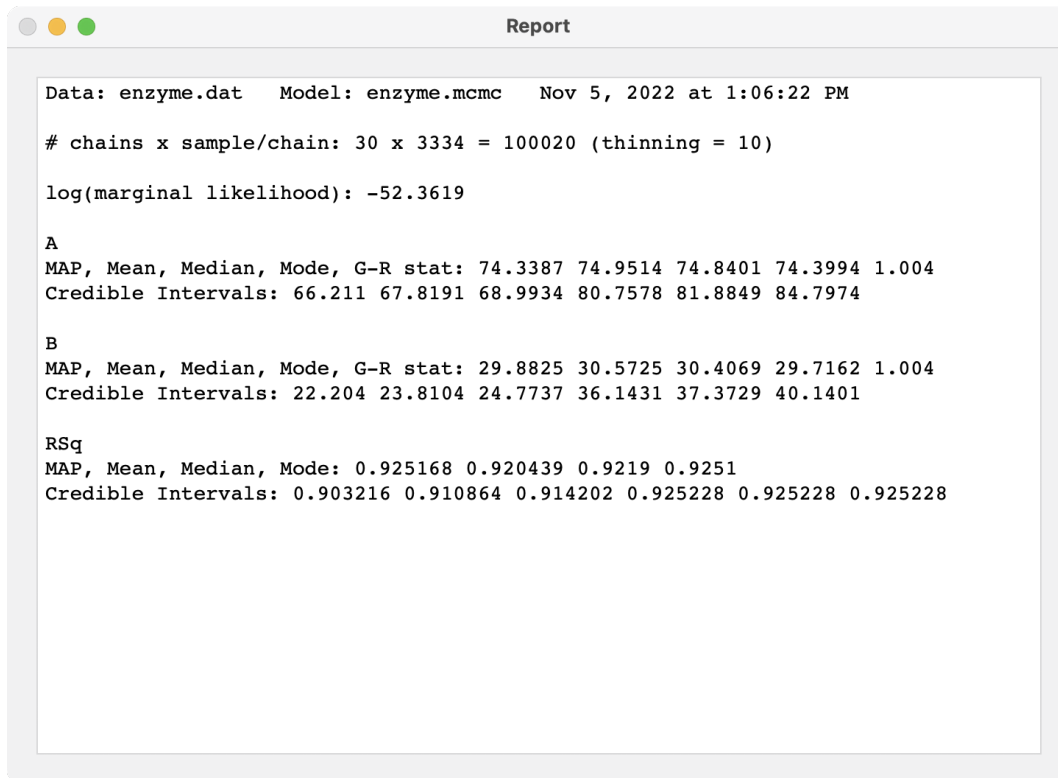
When the run is finished, files *report.txt* and *trace.txt* will be saved to the folder containing the data. The report will also be shown in a window. Figure 2.8 shows the Report window for the Quickstart example. After the header, information is summarized for some common quantities plus everything that was monitored during the run.

2.6.1 Trace

In this run, there were two parameters, *A* and *B*, and five walkers per parameter for a total of ten walkers. The target sample size (given 10,000 per walker) was thus 100,000. However, there were three sub-processes⁵ and 100,000 is not a multiple of three so the final sample size was rounded up to $30 \times 3334 = 100,020$. Since thinning was set to 10, there were, in fact, 1,000,200 post-burn-in iterations altogether (for 3 sub-processes with 10 walkers each). The second line of the report shows how many of the visited states wound up in the trace. The saved trace is sorted by walker so here it contains 30 blocks of 3334 states in the order visited by the respective walkers.

The dimensions of the saved trace array are given in the first line in the trace file. The second line in the file lists the column symbols. The rest of the file gives the trace—all

⁵See Status dialogs above. This number is computer-specific.



```

Report

Data: enzyme.dat   Model: enzyme.mcmc   Nov 5, 2022 at 1:06:22 PM

# chains x sample/chain: 30 x 3334 = 100020 (thinning = 10)

log(marginal likelihood): -52.3619

A
MAP, Mean, Median, Mode, G-R stat: 74.3387 74.9514 74.8401 74.3994 1.004
Credible Intervals: 66.211 67.8191 68.9934 80.7578 81.8849 84.7974

B
MAP, Mean, Median, Mode, G-R stat: 29.8825 30.5725 30.4069 29.7162 1.004
Credible Intervals: 22.204 23.8104 24.7737 36.1431 37.3729 40.1401

RSq
MAP, Mean, Median, Mode: 0.925168 0.920439 0.9219 0.9251
Credible Intervals: 0.903216 0.910864 0.914202 0.925228 0.925228 0.925228

```

Figure 2.8: Report Window

tab-delimited text. In addition to the monitored variables, the trace array also contains a zeroth (leftmost) column giving the value of the log(posterior) for that state.⁶

2.6.2 Marginal Likelihood

Unless it was deselected in the Setup dialog, the log(marginal likelihood) is shown in the third line of the report.

2.6.3 Monitored Variables

There must be at least one monitored variable. In the example shown in Figure 2.8, there are three: two parameters, A and B, plus RSq = the weighted R-squared metric for goodness-of-fit (where RSq = 1 is a perfect fit), computed as an Extra.

For each monitored variable, the first line reports the MAP, mean, median and mode values as found from the corresponding marginal. Credible-interval limits are in order: lower-99, lower-95, lower-90, . . . , upper-99. With parameters (not Extras), the Gelman-Rubin statistic (G-R stat) for chain mixing is also reported (to three decimal places, see pg. 45). Once again, a value of one is perfect. A good value should be less than 1.01.

⁶All logs, here and elsewhere, are natural logs.

2.6.4 Marginals

A marginal plot is shown for the monitored variable appearing in the Status dialog (Reset to change). Figure 2.9 shows the marginal for parameter A in the Quickstart example.

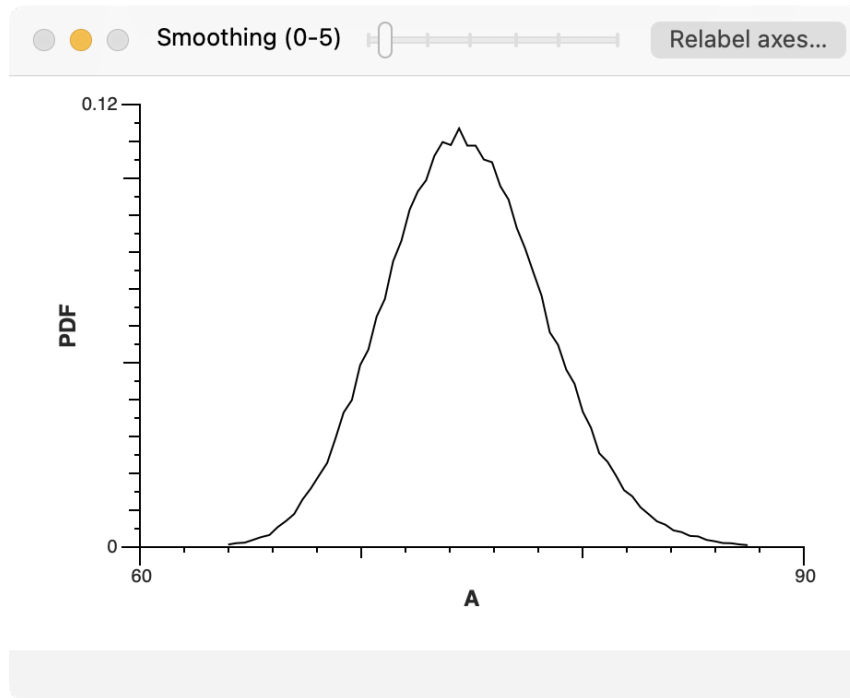


Figure 2.9: Marginal for Parameter A

This plot may be smoothed, unless the variable is discrete, and the axis labels may be edited.⁷ However, axis range and/or tick marks may *not* be changed.

In rare cases, the plotted marginal may be offset to avoid too many digits in a tick label (see Example MP). Of course, this can be avoided by offsetting the data. Proper data encoding, to maximize precision, is usually a good idea.

All *MacMCMC* plots may be saved in PDF format (the default) or PNG format.⁸

⁷Smoothing does not change the trace in any way.

⁸PNG, better on webpages, is of lower (bitmap) quality.

Chapter 3

Modeling Language

EVERY MCMC application/package inputs a model, described in pseudocode, for analyzing the data. This model is interpreted by a *parser* to execute computer code. Parsers vary so any MCMC application is therefore parser-specific to some extent. This chapter describes the *MacMCMC* modeling language compatible with (required by) its parser. Please pay attention to the details since software tends to be rather fussy about doing things in exactly the way that it expects and violating this precept is apt to give unexpected results—something that may or may not be evident upon casual inspection.

We shall use the Quickstart (enzyme) example once again to illustrate the various parts of a typical model. Also, some hints are provided suggesting how things might go wrong.

In addition to common distributions and functions, *MacMCMC* implements three built-in mixture models. These have some special requirements and will be discussed separately.

3.1 Models

The Quickstart (enzyme) model discussed earlier was about as simple as it could be and did not take advantage of all of the modeling expertise that one might expect. In particular, it utilized the individual *measurement errors* provided in the data but it did not acknowledge that there might be additional error due to the possibility that the formula for the rate equation itself might be incorrect.¹ Here, we shall adopt an alternate model in which this deficiency is corrected. To this end, we shall define a new, unknown parameter, *sig*, to describe this additional error. This will enable us to i) see whether we get an “improved” goodness-of-fit (weighted R-squared closer to one) and ii) perform a *model comparison* to assess whether the new model is really better (more credible) overall.

Our new model, model2, is shown in Figure 3.1. We shall explain the parts of a typical model by going through this model section by section focusing on required syntax and associated features. As noted earlier, all seven sections (Constants:, etc.) are required but Extras are optional and so this section could have been empty.

¹Engineers usually call this *system error*.

```

1 // Measurement error plus (hypothetical) model error
2 Constants:
3 N = 14, // # of points
4 TSS = 420.05; // total_sum_squares
5
6 Data:
7 conc[N], rate[N], sigma[N];
8
9 Variables:
10 mu[N], A, B, sig, err, ESS, RSq, i;
11
12 Priors:
13 A ~ Jeffreys(1, 100);
14 B ~ Jeffreys(1, 100);
15 sig ~ HalfNormal(5);
16
17 Likelihood:
18 for (i, 1:N) { // with error
19     mu[i] = (A*conc[i])/(B + conc[i]);
20     rate[i] ~ Normal(mu[i], sqrt(sigma[i]^2 + sig^2));
21 }
22
23 Extras:
24 ESS = 0;
25 for (i, 1:N) { // RMS-weighted error
26     err = (rate[i] - mu[i])/sqrt(sigma[i]^2 + sig^2);
27     ESS = ESS + err*err; // error_sum_squares
28 }
29 RSq = 1 - ESS/TSS;
30
31 Monitored:
32 A, B, sig, RSq;

```

Figure 3.1: Enzyme Model2

3.1.1 General

The parser considers each model section to be a vector of statements. Evaluation consists of interpreting these statements **in the order that they appear** in the model. This **must** be the same top-down order seen in the corresponding directed, acyclic graph (DAG).

Every statement must be terminated by a semicolon. For-loops must be delimited by braces. Blank lines are ignored wherever they occur. Comments are always acceptable and may be C++-style (as shown) or C-style.

3.1.2 Constants

This section must contain one statement with multiple definitions separated by commas.

This is the place to declare any quantity that need not be recomputed at each iteration. At a minimum, it should define a symbol for the number of datapoints. The right-hand-side (RHS) of a definition here may also be an expression, e.g., $\sqrt{2 \cdot \pi}$.

3.1.3 Data

Data variables must always be declared as an array even if there is only one value. Their declarations must be C-style with their corresponding dimensions, e.g. $x[N][2]$. In `model2`, data variables are simple vectors of size N . Indexing always begins with 1 (*unlike C-style*) so that *MacMCMC* pseudocode is consistent with the literature.² These data declarations must match the data-file columns exactly—symbol and dimension(s). However, the order of declaration is arbitrary.

3.1.4 Variables

Every variable used in the model, including loop counters, must be declared here. These variables are effectively global. Since statements are parsed sequentially, a variable may be reused, e.g., as a temporary quantity. As in any programming language, no variable may be reused while its value is still needed. Obviously, parameter variables should not be reused since they are never temporary. In `model2`, variable *err* is reused with each datapoint and ESS updated (lines 24–28).

(Hyper-)parameters must be initialized before any iteration may be computed. This is done randomly by selecting from their priors but the latter will also have parameters unless they are (fixed) founder nodes. Here, as always, it is *essential* that statements in the model be listed in a “top-down” fashion so that all variables on the RHS of every statement will have been given a value before being referenced. Otherwise, they will contain “garbage” and the initial trial run executed by *MacMCMC* could fail. If that happens, *MacMCMC* will immediately abort with an “Unknown runtime error” message.

There are two special options that may relate to a variable (or data) symbol:

- Any symbol with an underscore as the last character³ is treated as an integer. The internal, double-precision value is *rounded* to the nearest integer when necessary and in all output except mean values (see Examples Hyphens and Lizards).
- Symbols beginning with lower-case “zz” are treated as indicators and will not be monitored regardless of what it says in the model. Also, computation of the marginal likelihood will be disabled if there is an indicator since that would be problematical.

3.1.5 Priors

Every parameter and hyper-parameter must have exactly one prior. In `model2`, there are three prior statements (lines 13–15). These are stochastic relationships, described with a tilde instead of an equals sign.⁴ The Priors section may also include any deterministic relationships (equations) required or, indeed, anything that need not be recomputed with each datapoint.

²See, however, Appendix C.

³deleted from plots

⁴The <- symbol may be substituted for an equals sign (common in the literature).

3.1.6 Likelihood

Unless there is just one datapoint, the likelihood will consist of a for-loop with syntax as shown in lines 18–21. In some cases, the data may be sorted in some fashion and the likelihood loop split into pieces accordingly (see Example FishCount).

Occasionally, an entire vector must be referenced. The *MacMCMC* syntax for the entire $\mu[N]$ vector is $\mu[]$ (see Example Body). Note, however, that this syntax is valid only for the rightmost dimension. That is, $x[][1]$ is not valid.

3.1.7 Extras

This section may be empty and anything computed here could alternatively be computed in post-processing provided that all needed variables have been monitored and, therefore, have been saved in the trace.

Extras are computed at the end of an iteration after all needed quantities are known. In this example, we compute the value of (weighted) R-squared, RSq , at each state in order to assess goodness-of-fit.

3.1.8 Monitored

There must be a least one monitored variable (parameter or Extra) and it must not be constant in the trace. Any constant column in the trace will abort the run.⁵

In *model2*, four unknowns are monitored. They will appear in the trace, to the right of *logPosterior*, and in the Status sub-menu in alphabetical order. If an entire vector of k unknowns is monitored, e.g., $p[]$, the trace will list $p[1], p[2], \dots, p[k]$.⁶

3.1.9 Results for the Enzyme Example

Numerical results for monitored parameters for the original Quickstart example, *model1*, and the improved *model2* (both with default options) are given in Table 3.1 and Table 3.2, respectively. The reported values were those for the runs done for this documentation. If you rerun these examples, the report will contain slightly different values due to (expected) MCMC noise which can be reduced by generating a larger sample.

Figure 3.2 shows the marginal for *model2* parameter *sig*. This plot is very different from the (HalfNormal) prior for *sig* indicating that there was enough information in the data to say something useful about this parameter.

The R-squared metric compares the model to the data using the total distance-squared between the model curve and the datapoints. In this case, *model2* is better. Nevertheless, such simplistic approaches to model comparison are sub-optimal. The mathematically preferred way to compare two models is *via* their relative marginal likelihood. [6, pg. 111]

⁵The trace will have been saved by this time, however.

⁶Trace files can get very large.

Table 3.1: Results for model1

Unknown	Estimate		95% Credible Interval	
	MAP	Mean	Lower Limit	Upper Limit
A	74.3387	74.9514	68.9934	81.8849
B	29.8825	30.5725	23.8104	37.3729
RSq	0.925168	0.920439	0.910864	0.925228

Table 3.2: Results for model2

Unknown	Estimate		95% Credible Interval	
	MAP	Mean	Lower Limit	Upper Limit
A	76.4444	78.0646	66.9975	89.9513
B	33.3419	35.5137	23.3249	49.0680
sig	2.70553	3.41427	1.40108	5.77610
RSq	0.970641	0.970092	0.949538	0.988485

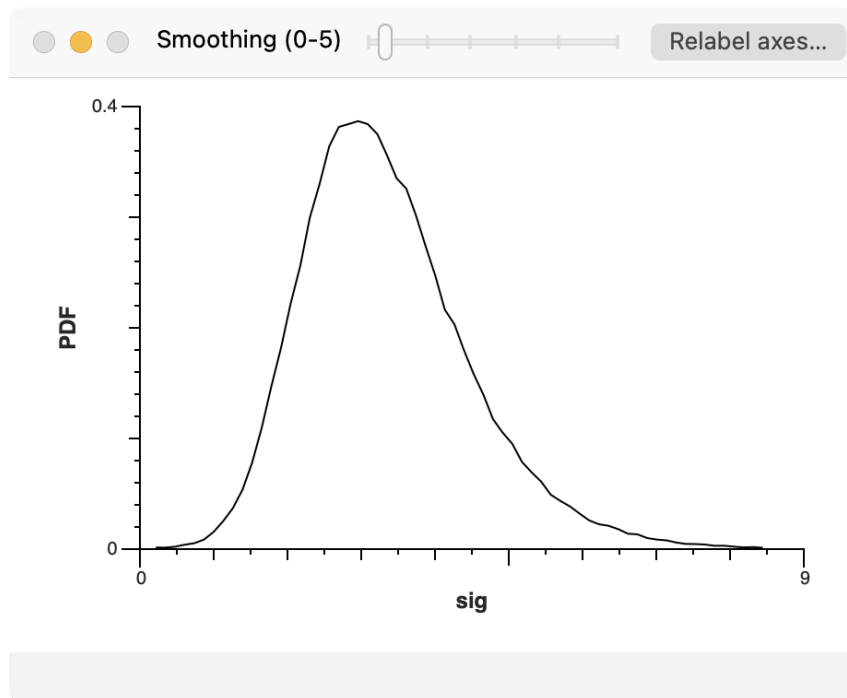


Figure 3.2: Marginal for Parameter sig

This permits computation of the probability that one model is better (more credible) than another based on the full posterior whatever the model forms or the number of parameters.

Here, the $\log(\text{marginal likelihood})$ values for model1 and model2 are -52.3619 and -47.2738 , resp. Since the value for model2 is the larger, it is truly the better model. The

complete computation is given below, probability being computed from the odds.

$$\text{odds model2 better} \equiv \text{odds}M2 = \exp(-47.2738 - (-52.3619)) = 162.082 \quad (3.1)$$

$$\text{Prob(model2 better)} = \frac{\text{odds}M2}{1 + \text{odds}M2} = 0.99387 \quad (3.2)$$

Therefore, by this state-of-the-art Bayesian technique, the probability that model2 is better than model1 is more than 99 percent.⁷

Note that there is *always* a penalty for adding a parameter to a model (i.e., enlarging the parameter space). Other things being equal, $\log(\text{marginal likelihood})$ will *decrease* with each added parameter. Here, the benefit of incorporating model error outweighs the penalty of adding parameter *sig*.

Finally, the plot in Figure 3.3 shows how the two models compare to the data. This plot uses mean parameter estimates from each report.

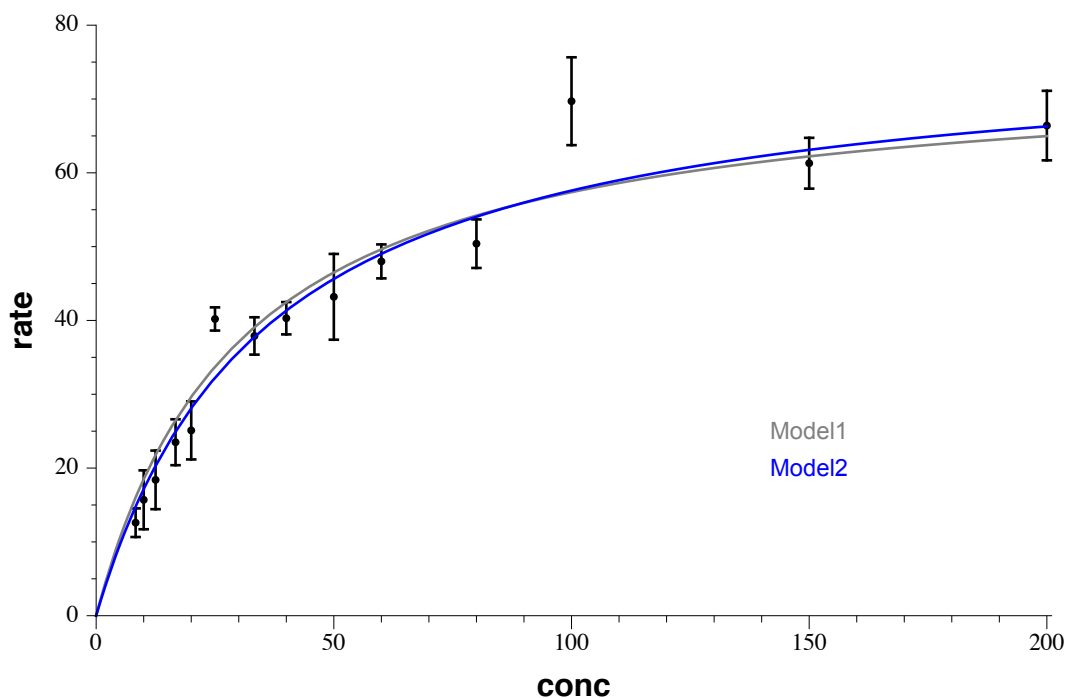


Figure 3.3: Model vs. Data: With and Without Model Error

3.2 Mixtures

Discrete mixtures consist of a weighted combination of distributions, the weights being positive and summing to one. One can always add a mixture distribution to a model “by

⁷This computation should vary very little from run to run.

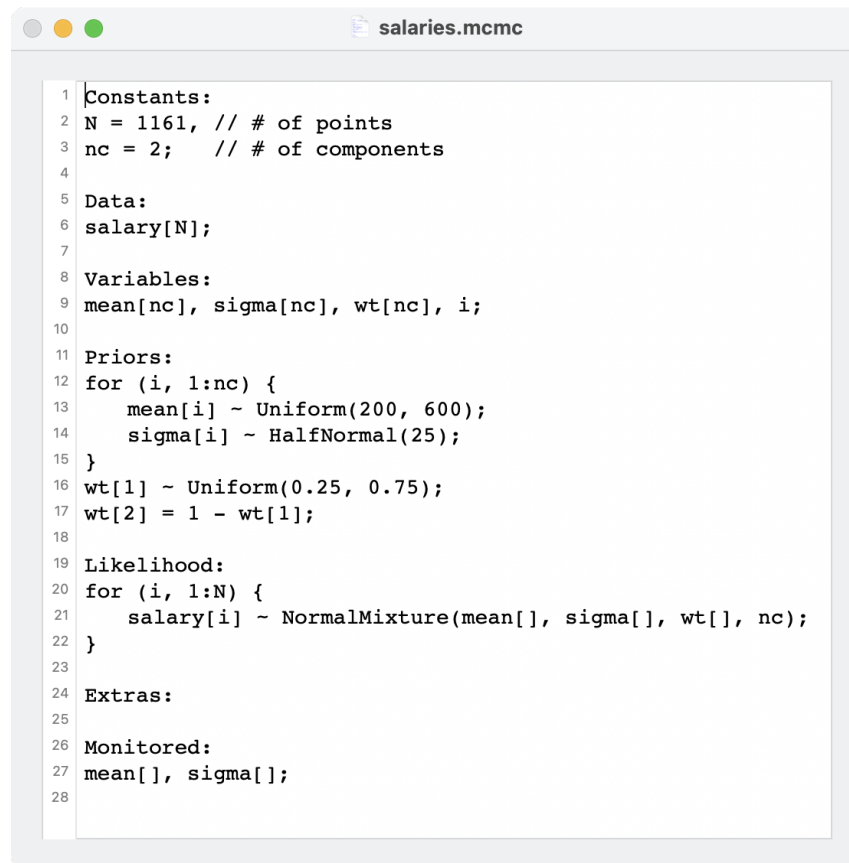
hand” but *MacMCMC* implements three common mixtures as part of its built-in repertoire. These are mixtures of Normal, BivariateNormal and Poisson (see Appendix A.3).

Note: In *MacMCMC*, one of these three may be employed as a likelihood only, never as a prior. Also, there can be at most one built-in mixture in the model.

Built-in mixtures have special requirements in the modeling language, as follows:

- The arguments for the mixture components must be entered as full vectors except for the final argument.
- The final argument must be the (constant) number of components (greater than one).
- The Monitored section must contain all argument vectors *except* the weight vector. No other monitored variables are allowed.

All of these are illustrated in Example Salaries as shown in Figure 3.4. Note that good results for a homogeneous (same form for all components) mixture such as this requires a *lot* of data especially when there is significant overlap between components.



```
1 Constants:
2 N = 1161, // # of points
3 nc = 2; // # of components
4
5 Data:
6 salary[N];
7
8 Variables:
9 mean[nc], sigma[nc], wt[nc], i;
10
11 Priors:
12 for (i, 1:nc) {
13     mean[i] ~ Uniform(200, 600);
14     sigma[i] ~ HalfNormal(25);
15 }
16 wt[1] ~ Uniform(0.25, 0.75);
17 wt[2] = 1 - wt[1];
18
19 Likelihood:
20 for (i, 1:N) {
21     salary[i] ~ NormalMixture(mean[], sigma[], wt[], nc);
22 }
23
24 Extras:
25
26 Monitored:
27 mean[], sigma[];
28
```

Figure 3.4: Salaries: Mixture Model

3.2.1 Relabeling

Homogeneous mixtures are susceptible to *label switching*. If the component labels (here, 1 and 2) are permuted, the posterior value is unchanged so component labels are inherently ambiguous. This problem is unavoidable and affects both the run options⁸ and the output.

The default is to try to correct this problem by relabeling the trace columns (see Setup dialog and Appendix C). This is *much* easier said than done! The number of attempts (default = 30) is specified with the low-level option, `RelabelLim`. With overlapping data, relabeling is problematical since some points are ill-defined by definition. It is always difficult in any case.

When relabeling is in effect, there will be two additional output files as well as a change to the report. One new output file is *traceRL.txt* which is the relabeled trace. The report is based on the contents of this file, not the original *trace.txt*. Another new output file is *probAssign.txt*. This file contains one row for each datum and one column for each component. The values listed are the posterior probabilities that datum[i] comes from component[k]. Finally, the report will not contain a value for log(marginal likelihood) since this cannot be computed in the absence of valid labels.⁹ The last line in the report contains posterior estimates for the weight of each component.

3.2.2 Results for the Salaries Example

Figure 3.5 shows the relabeled model compared to the data.¹⁰ The (weighted) individual components are shown as dashed curves; the mixture PDF is the solid curve.

Numerical results for a typical run are presented in Table 3.3. The values listed are based on the relabeled trace which, as noted, is unlikely to be 100-percent correct.

Table 3.3: Results for Salaries

Unknown	Estimate		95% Credible Interval	
	MAP	Mean	Lower Limit	Upper Limit
mean[1]	481.942	483.159	460.562	507.353
sigma[1]	91.7212	92.0972	83.2677	100.836
wt[1]	—	0.475716	—	—
mean[2]	369.992	370.767	361.478	379.995
sigma[2]	53.3872	53.9432	47.852	60.081
wt[2]	—	0.524284	—	—

⁸disabled when not appropriate

⁹This option is disabled for built-in mixtures.

¹⁰another Mathematica+SetAxes plot

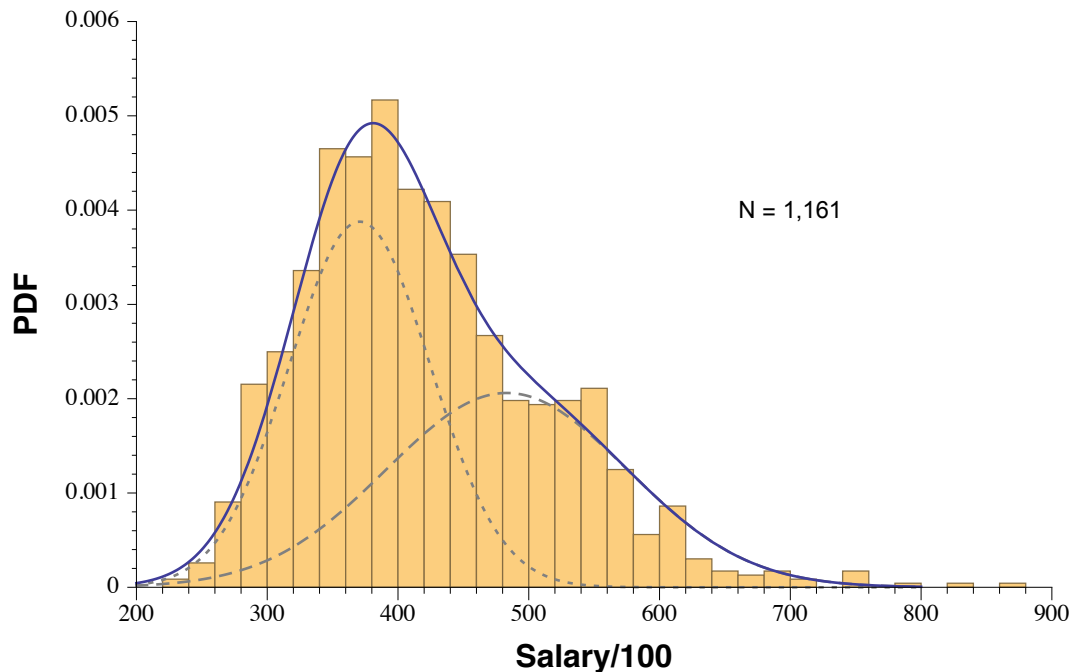


Figure 3.5: Salaries: Data and Model (mean estimates)

3.3 What Might Go Wrong?

There are many things that can go wrong in an MCMC run. Some of them will be caught by the parser and some will be evident in the marginals and/or the trace plots. In general, there are several things that must be kept in mind whenever a model is being formulated. The list below is not comprehensive but it does offer some useful hints.

- Check distributions and functions to make sure that they agree with their *MacMCMC* definitions (Appendix A). Such definitions often differ in the literature. Also, check the order in which their arguments are supposed to appear.
- Do not model discrete quantities as continuous or *vice versa* (see sect. 3.1.4).
- Check that every variable is declared in the Variables block, that its dimensions are correct and that subsequent indices are compatible *everywhere*. *MacMCMC* does not check array bounds. This is not a syntax error and will not be caught!

Violating array bounds is a common mistake and will usually crash any program. If *MacMCMC* crashes (does not Quit normally), the *MacMCMC_M2C2* processes will not terminate. If this happens, the easiest fix is to run *MacMCMC* and then Quit immediately. The absence of *MacMCMC_M2C2* processes can be verified using /Applications/Utilities/Activity Monitor.

- Check that every variable, apart from for-loop counters, appears on the LHS of a

statement *before* it ever appears on the RHS of a statement. Variables must be given a valid value somehow and *MacMCMC* processes model statements in the order in which they appear in the model.

- Check that any reused variable is “out of scope” before it is used again.
- Make sure that every (hyper-)parameter has exactly one prior.
- Check that variables that should be normalized are normalized and those that should be positive cannot be otherwise at any iteration.
- Make sure that priors for parameters of bounded likelihoods **never** contradict the data. Therefore, a bounded likelihood requires prior knowledge of the theoretical bounds. Such likelihoods are sometimes problematical with respect to convergence and/or integration.
- Data variables may be used *only* in a likelihood statement. Do *not* use some data to help define an informative prior then re-use the same data in the run. To do so invalidates the MCMC algorithm, and Bayesian inference in general, even if the program appears to execute without visible error.

If necessary, draw a *random* sub-sample of the dataset and use *that* in exploratory analysis but do not re-use it thereafter.

- Make sure that the child-parent relationship between all parameters and priors is maintained throughout the entire model. Every valid MCMC model must always have a **directed, acyclic graph (DAG)**. It cannot have any “loops” in it, however indirect. Otherwise, a “child” becomes its own “ancestor” and MCMC will fail.
- Note that posteriors may have multiple (even false) local optima. Also, *MacMCMC* utilizes many independent walkers (see Appendix C). If a prior is “too vague”, then a walker might get “caught” in a false local optimum giving very poor results.
- Relabeling can fail (be incomplete). If that happens, it is worth trying again, perhaps with a longer burn-in.
- Finally, if you are new to MCMC analyses, set *MacMCMC* aside until you have read the ebook cited on page v. Study the models in that book very carefully. Also, try out the Examples supplied with this package and summarized on page 48.

Chapter 4

Goodness-of-fit

NO data analysis is really complete unless/until it is demonstrated that the model used describes the data. By definition, a good model should be able to substitute for the data—to answer questions that could have been answered by the input data or by future data. If a model does not describe the input dataset, then it is not a model for that dataset. Therefore, after an MCMC run has finished, one should perform some sort of goodness-of-fit test to compare the inferred model to the data.

Comparing model to data can be done in many ways since the data are given and the model is implicit in the MCMC trace. How one makes this comparison is limited only by the expertise of the analyst. *MacMCMC* provides some very basic functionality to aid in this process. This is not the primary purpose of *MacMCMC* and there are a few limitations. Nevertheless, this functionality should assist in making a judgment regarding goodness-of-fit in most cases. The general procedure is described below followed by three examples.

4.1 General Procedure

Posterior-predictive techniques provide what is arguably the best approach to goodness-of-fit testing. [2, sect. 6.3] This capability was added in *MacMCMC* (version 1.3) but had to be compatible with everything already present. For this reason, the addition was achieved with an optional *Goodness:* section for the *MacMCMC* model plus three dummy variables (symbols): *gdnsX*, *gdnsX_* and *gdnsY*. These variables are pre-defined and intended solely for goodness-of-fit testing; they should not be used for anything else.

The *Goodness:* section, if present, must contain a single statement summarizing the relationship between the model used and the data. This target relationship must be a simple equation or distribution; more complicated models (e.g., [logistic regression](#)) will require more elaborate testing. For subsequent 2-D plotting, the data variables must be referenced in the model using the dummy symbols above and all non-constant symbols on the RHS of the target relationship **must** appear in the trace which means that they must be an uncertain parameter or be monitored (or both). Currently, *MacMCMC* has little checking capability

in this regard. The examples below should make the possibilities clear.

There are three limitations, some due to the requirements of the *MacMCMC* software and some simply graphical. The target relationship must

- Describe an equation (assumed continuous) or a univariate distribution
- Not be one of the built-in mixtures since the component weights are never monitored (see sect. 3.2)
- Not be a `Generic(.)` distribution (see sect. A.4) since a user-defined $\log(\text{PDF})$ formula might be incorrect or unnormalized. Also, the CDF will be unavailable to *MacMCMC*.

The goodness-of-fit output will be one of three plots. For equations, this will be a simple X–Y plot. For a distribution, the plot will be a [quantile-quantile \(q–q\) plot](#), with the data on the abscissa, if the data are continuous or a PDF histogram if they are discrete. This plot will generate a new document which may be saved as usual.

When appropriate, goodness-of-fit is enabled (see Fig. 2.2) after the MCMC run has finished, bringing up a dialog from which the user must specify the quantities associated with the dummy symbols given above. The *gdnsX* symbol, or its discrete equivalent (with an underscore), will always refer to the variable plotted on the abscissa. For an equation, that means the “independent” variable. In this case, the LHS of the target relationship must be *gdnsY* and this symbol must be identified accordingly. With distributions, there is only one quantity under consideration. It must be symbolized *gdnsX* (or *gdnsX_* when discrete) and appear on the LHS of the target relationship.

4.2 Credible Interval

In *MacMCMC* (version 1.5+), the goodness-of-fit plot has a *CredInterval95* option. This option functions independently of the Mean/MAP choices (see below). When checked, the 95-percent credible interval band for predicted *y*-values is shown in light gray. This option is disabled for equations when there are fewer than four points and, with distributions, when there are fewer than two points. The computation of this credible interval band is straightforward but requires a lot of work. For details, see Appendix C.

4.3 Goodness-of-fit Examples

Here, we present four examples described as follows:

Daytime an equation with a very small amount of measurement error

Hale-Bopp an equation with significant measurement error (plus credible interval)

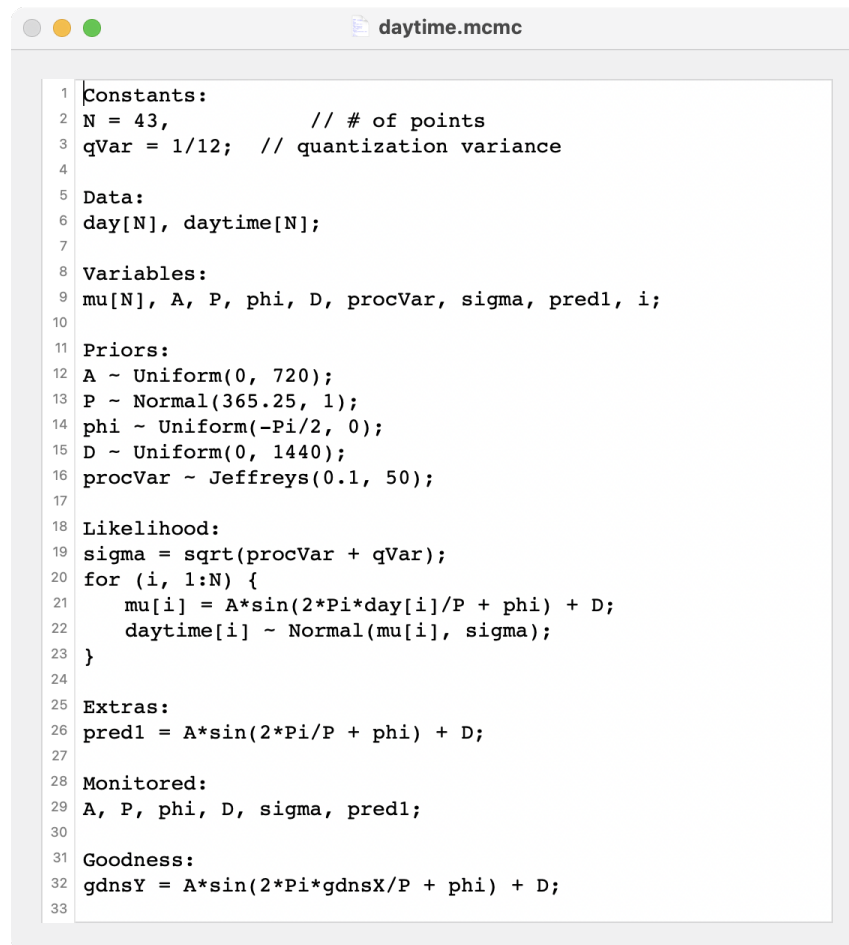
Body Temperature a continuous distribution

Hyphens a discrete distribution

All of these are in the Examples folder (see Appendix D).

4.3.1 Daytime

The Daytime dataset consists of 43 points each being the time in between sunrise and sunset (rounded off to the nearest minute) in Boston, Massachusetts, USA over three years. The model is a sine wave as shown in Figure 4.1—a nonlinear, unweighted regression.



```

1 Constants:
2 N = 43,           // # of points
3 qVar = 1/12;     // quantization variance
4
5 Data:
6 day[N], daytime[N];
7
8 Variables:
9 mu[N], A, P, phi, D, procVar, sigma, pred1, i;
10
11 Priors:
12 A ~ Uniform(0, 720);
13 P ~ Normal(365.25, 1);
14 phi ~ Uniform(-Pi/2, 0);
15 D ~ Uniform(0, 1440);
16 procVar ~ Jeffreys(0.1, 50);
17
18 Likelihood:
19 sigma = sqrt(procVar + qVar);
20 for (i, 1:N) {
21     mu[i] = A*sin(2*Pi*day[i]/P + phi) + D;
22     daytime[i] ~ Normal(mu[i], sigma);
23 }
24
25 Extras:
26 pred1 = A*sin(2*Pi/P + phi) + D;
27
28 Monitored:
29 A, P, phi, D, sigma, pred1;
30
31 Goodness:
32 gdnsY = A*sin(2*Pi*gdnsX/P + phi) + D;
33

```

Figure 4.1: Daytime Model

In addition to priors for the sine-wave parameters, there is an additional prior for the modeling error variance, *procVar*, to be combined with the [quantization error](#), *qVar*, since the times have been rounded off. Total error, *sigma*, is taken to be the [RMS](#) value of these two errors as shown in line 19. The target relationship described above is in line 32.

When goodness-of-fit is selected, a dialog appears so that the user can specify the identity of the dummy variables. The result of this specification is shown in Figure 4.2. The independent variable is *day* and the dependent variable is *daytime* just as given in the model and the data file.

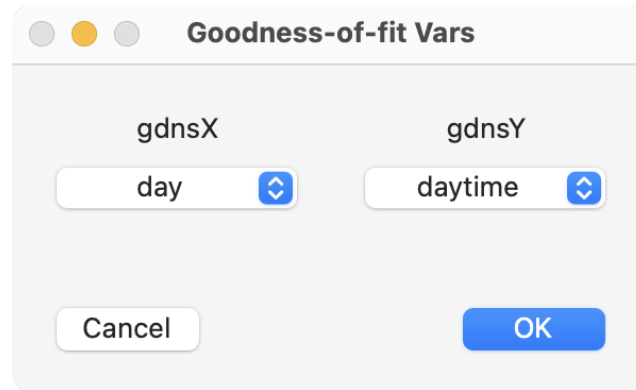


Figure 4.2: Goodness Dialog for Daytime Example

Since the target relationship is an equation, goodness-of-fit will be displayed as an X–Y plot. Here, the plot appears as shown in Figure 4.3 (model shown in gray).

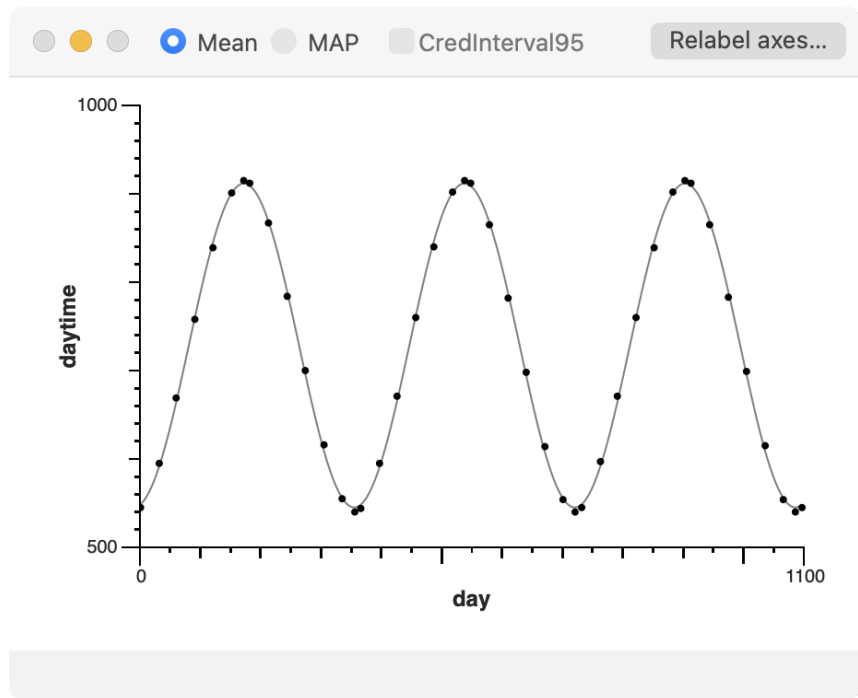


Figure 4.3: Goodness-of-fit for Daytime Example

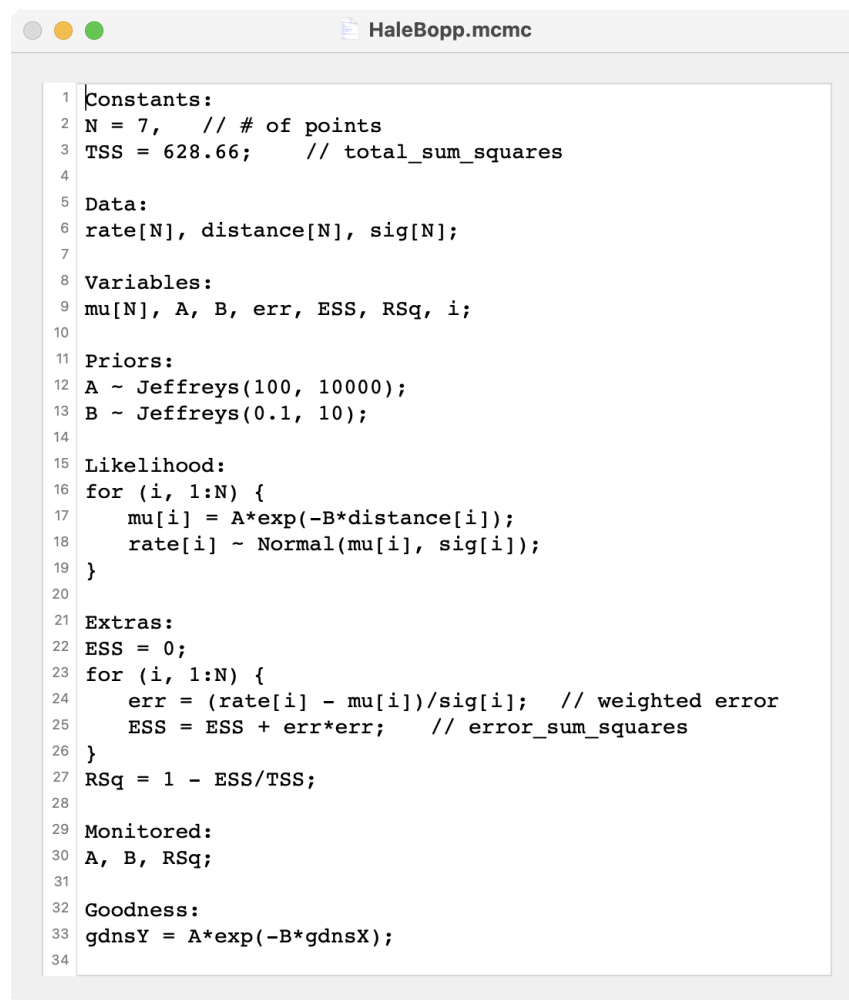
Mean parameters are used for this plot by default but MAP parameters may be used

instead as the radio buttons suggest. As with marginal plots (e.g., Fig. 3.2), the axes may be relabeled and the plot saved to a file. The tick marks cannot be changed.

It should be apparent that this model is a very good fit to the data even though it is only approximately correct. The report gave a mean value for σ of just 4.5 minutes.

4.3.2 Hale-Bopp

The data for this analysis are measurements of the rate of release of cyanide radical (CN) from comet Hale-Bopp in units proportional to molecules/second as a function of distance from the Sun in astronomical units (AU). [8] This analysis is a weighted regression with a lot of measurement error. The model is shown in Figure 4.4. The R-squared metric was computed as an Extra.



```

1 Constants:
2 N = 7, // # of points
3 TSS = 628.66; // total_sum_squares
4
5 Data:
6 rate[N], distance[N], sig[N];
7
8 Variables:
9 mu[N], A, B, err, ESS, RSq, i;
10
11 Priors:
12 A ~ Jeffreys(100, 10000);
13 B ~ Jeffreys(0.1, 10);
14
15 Likelihood:
16 for (i, 1:N) {
17     mu[i] = A*exp(-B*distance[i]);
18     rate[i] ~ Normal(mu[i], sig[i]);
19 }
20
21 Extras:
22 ESS = 0;
23 for (i, 1:N) {
24     err = (rate[i] - mu[i])/sig[i]; // weighted error
25     ESS = ESS + err*err; // error_sum_squares
26 }
27 RSq = 1 - ESS/TSS;
28
29 Monitored:
30 A, B, RSq;
31
32 Goodness:
33 gdnsY = A*exp(-B*gdnsX);
34

```

Figure 4.4: Hale-Bopp Model

The goodness-of-fit plot (plus credible interval) is shown in Figure 4.5.

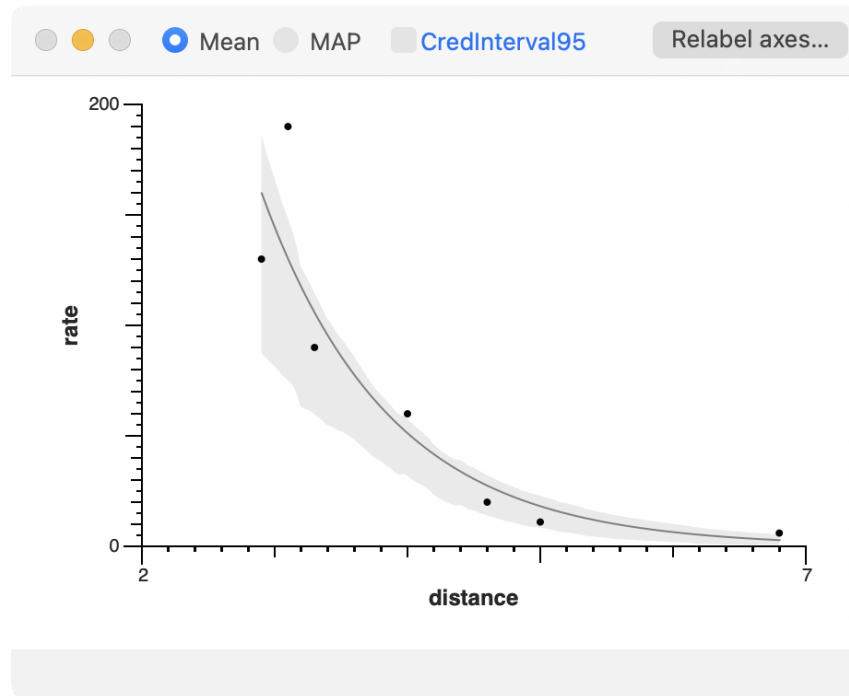


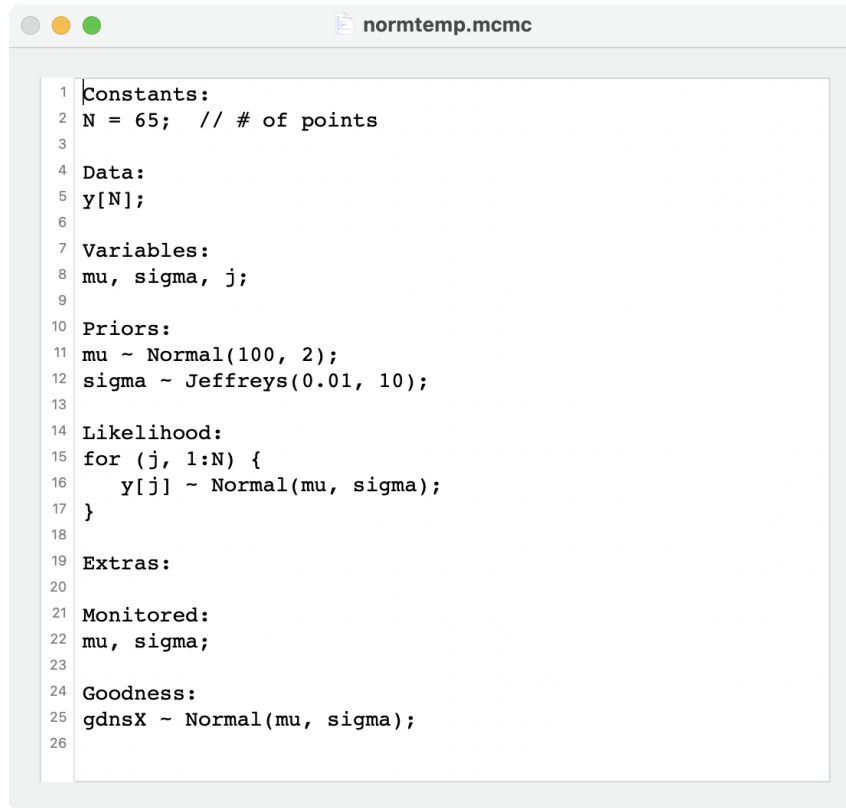
Figure 4.5: Goodness-of-fit for Hale-Bopp Example

Here, the credible interval is quite large, as expected given the experimental error. In the Daytime example, it was so small that it was almost invisible.

4.3.3 Body Temperature

This example illustrates the goodness-of-fit procedure when the data are described by a continuous distribution. The data consist of 65 measurements of normal body temperature for adult males. [9] We shall assume that they are normally distributed as described by the model in Figure 4.6.

The dialog box is slightly different for a distribution since there is only one observed quantity in the target relationship (see Fig. 4.7). The output in this case is the q–q plot shown in Figure 4.8. A perfect result (shown in gray) would be the straight line $f(x) = x$. The result here is good but not great (a judgment learned through experience).



```
1 Constants:
2 N = 65; // # of points
3
4 Data:
5 y[N];
6
7 Variables:
8 mu, sigma, j;
9
10 Priors:
11 mu ~ Normal(100, 2);
12 sigma ~ Jeffreys(0.01, 10);
13
14 Likelihood:
15 for (j, 1:N) {
16   y[j] ~ Normal(mu, sigma);
17 }
18
19 Extras:
20
21 Monitored:
22 mu, sigma;
23
24 Goodness:
25 gdnsX ~ Normal(mu, sigma);
26
```

Figure 4.6: Body-temperature Model

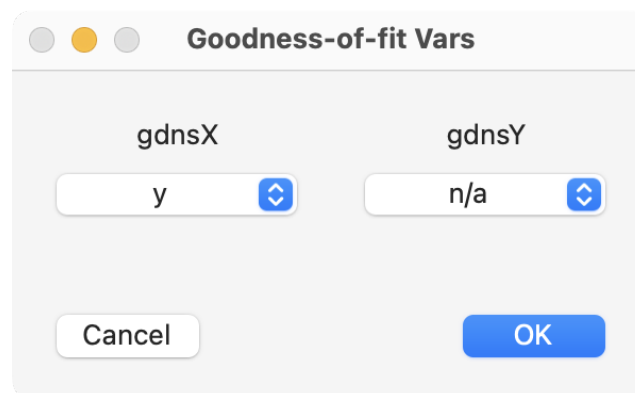


Figure 4.7: Goodness Dialog for Body-temperature Example

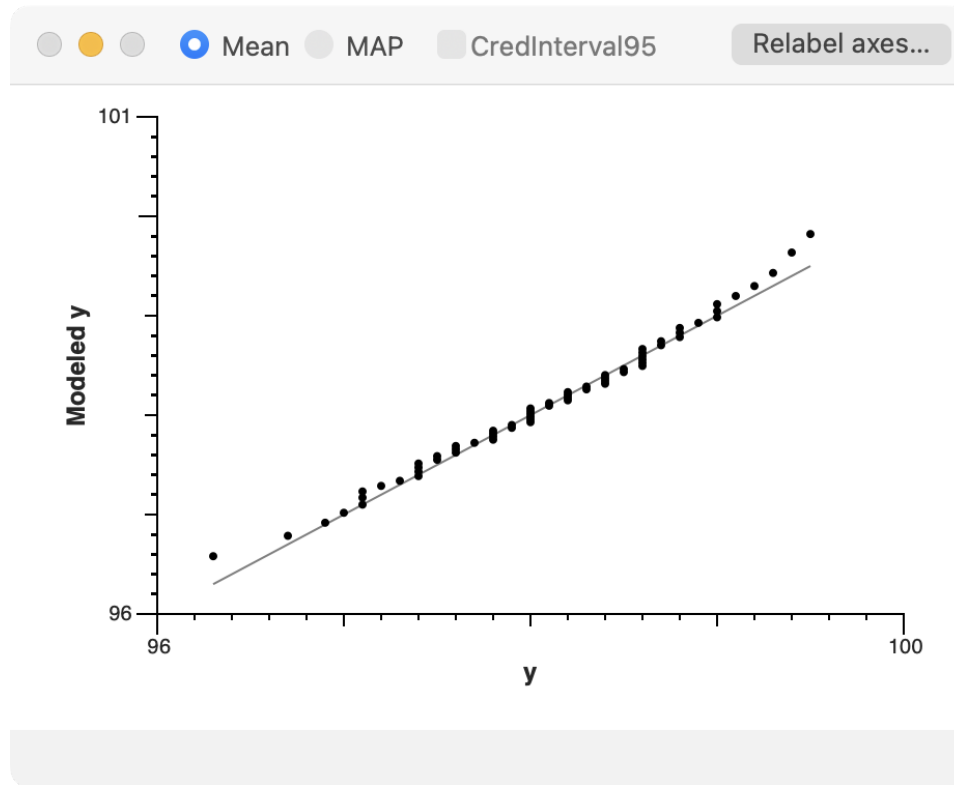


Figure 4.8: Goodness-of-fit for Body-temperature Example

4.3.4 Hyphens

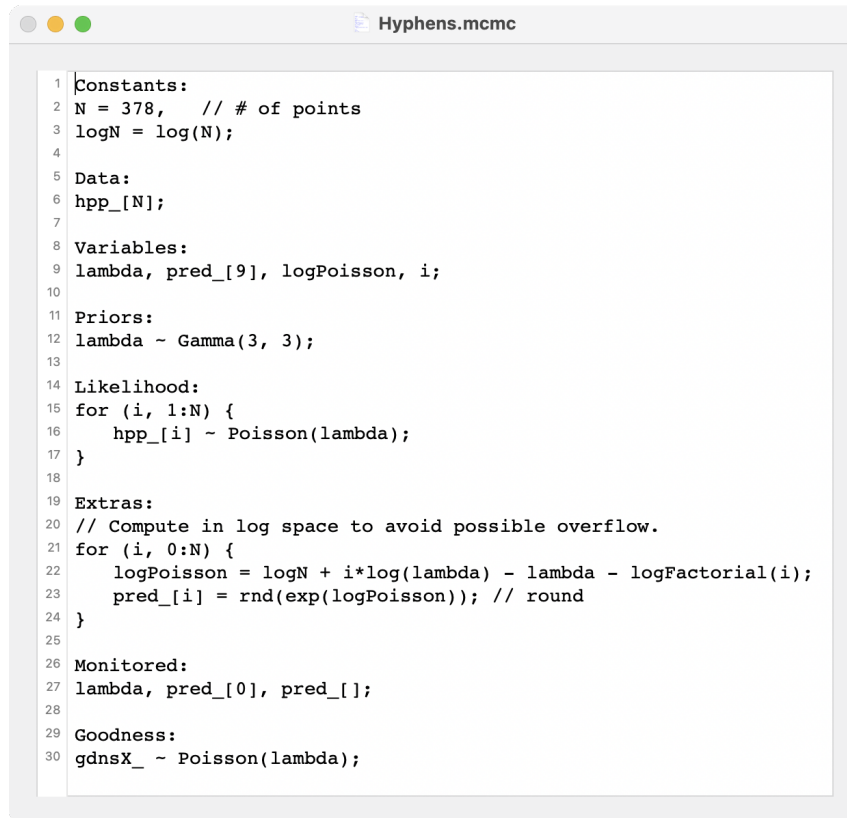
Our final example is a distribution with discrete data. Here, the data are 378 points each of which is the count of the number of lines on each page of a book I was reading that ended with a hyphen. The counts ranged from zero to eight and I thought this might make a good example of a Poisson distribution so that is the model used here (see Fig. 4.9).

The dialog is the same as in the previous example except for the name of the discrete variable, *hpp*. The Poisson parameter, *lambda*, is also the mean of that distribution. The report gave a mean value for *lambda* of 2.7 (out of an average of 18 lines per page).¹

There is no theoretical reason why these data must be Poisson but the goodness-of-fit plot (Fig. 4.10, with model shown in gray) is not bad for a one-parameter model. With discrete data, a q–q plot is not appropriate so a PDF histogram is output instead.

In this model, predictions are made for the number of terminal hyphens on a page. These predictions are declared here as a discrete vector, *pred*[_9], computed as an Extra in lines 21–24 and, of course, monitored. Note the syntax for monitoring these predictions. *pred*[_0] is defined, contrary to MCMC convention, but the *symp*[] syntax always assumes 1-based vectors.

¹The data average is also 2.7.



```

1 Constants:
2 N = 378, // # of points
3 logN = log(N);
4
5 Data:
6 hpp_[N];
7
8 Variables:
9 lambda, pred_[9], logPoisson, i;
10
11 Priors:
12 lambda ~ Gamma(3, 3);
13
14 Likelihood:
15 for (i, 1:N) {
16   hpp_[i] ~ Poisson(lambda);
17 }
18
19 Extras:
20 // Compute in log space to avoid possible overflow.
21 for (i, 0:N) {
22   logPoisson = logN + i*log(lambda) - lambda - logFactorial(i);
23   pred_[i] = rnd(exp(logPoisson)); // round
24 }
25
26 Monitored:
27 lambda, pred_[0], pred_[1];
28
29 Goodness:
30 gdnsX_ ~ Poisson(lambda);

```

Figure 4.9: Hyphens Model

With $\lambda = 2.7$, this model predicts $\text{pred}_[0] = N \exp(-2.7) = 25.4$ or 25 (after rounding). The report gave a mean prediction of 25.3 with a 95-percent credible interval of 21–29.² Figure 4.10 indicates that the observed zero count (20) was smaller than this prediction. Figure 4.11 shows the full marginals for $\text{pred}_[0]$ and the mode, $\text{pred}_[2]$.

²Note that a *MacMCMC* report does not round off mean values.

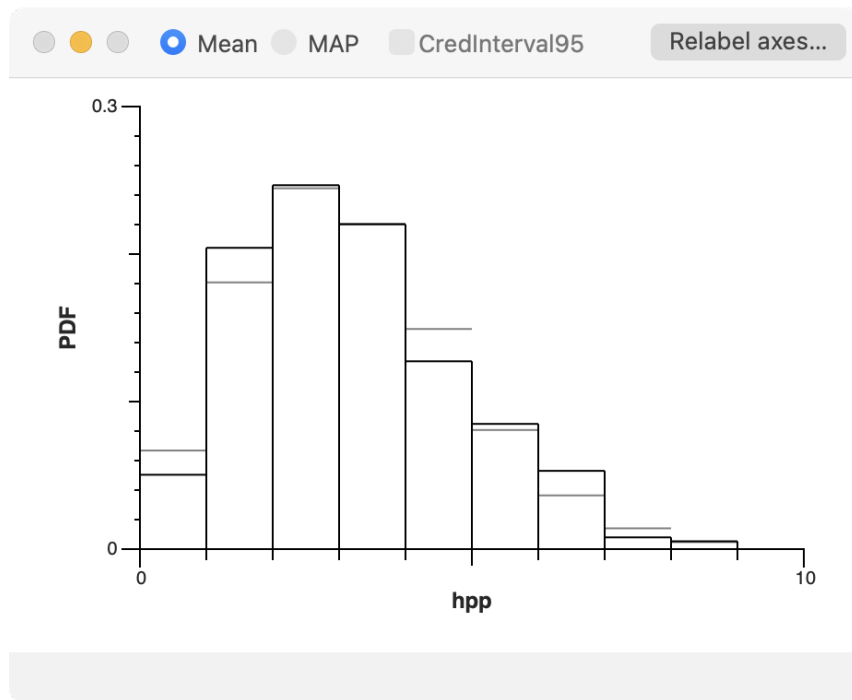


Figure 4.10: Goodness-of-fit for Hyphens Example

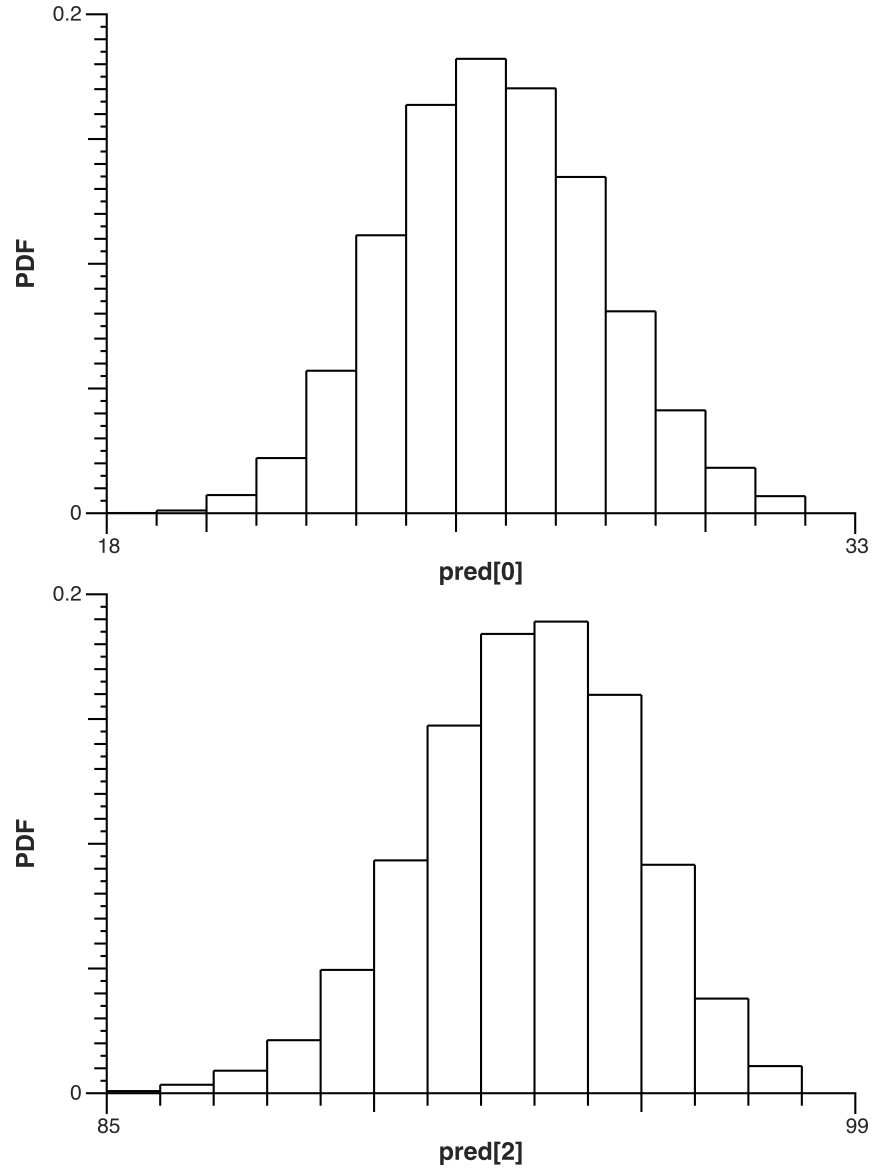


Figure 4.11: Marginals for Zero-count and Two-count

Appendix A

Distributions

This Appendix lists the definitions of all distributions defined in *MacMCMC*. Users are cautioned to examine the name and parameter list carefully since alternate definitions exist for many of them. Parameters must be input in the order shown regardless of the (user-defined) symbol. See Examples folder for sample usage where noted.

A.1 Continuous

Location parameters may be bounded. Scale parameters must be > 0 and shape > 0 unless otherwise noted. To avoid overflow, do not allow shape parameters to exceed 100. All univariate PDFs have units that are the reciprocal of the unit (if any) of the variate.

Beta

$$\text{Beta}(\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1} \quad ; 0 < x < 1$$

where $B(\alpha, \beta)$ is the [beta function](#) and α, β are shape parameters.

Exponential

$$\text{Exponential}(\lambda) = \frac{1}{\lambda} \exp\left(-\frac{x}{\lambda}\right) \quad ; x \geq 0$$

where $\lambda = \text{scale}$.

Gamma

$$\text{Gamma}(\alpha, \beta) = \frac{x^{\alpha-1}}{\Gamma(\alpha) \beta^\alpha} \exp\left(-\frac{x}{\beta}\right) \quad ; x > 0$$

where $\alpha = \text{shape}$, $\beta = \text{scale}$ and $\Gamma(\cdot)$ is the (complete) [gamma function](#).

HalfNormal

$$\text{HalfNormal}(s) = \frac{1}{s} \sqrt{\frac{2}{\pi}} \exp\left[-\frac{x^2}{2s^2}\right] \quad ; 0 \leq x < \infty$$

where s = scale (but *not* the standard deviation).

Jeffreys

Jeffreys is Uniform in log space.

$$\text{Jeffreys}(a, b) = \frac{1}{x \log(b/a)} \quad ; 0 < a \leq x \leq b$$

Laplace

$$\text{Laplace}(\mu, s) = \frac{1}{2s} \exp\left(-\frac{|x - \mu|}{s}\right) \quad ; -\infty < x < \infty$$

where μ = mean and s = scale.

Logistic

$$\text{Logistic}(\mu, s) = \frac{1}{s} \exp\left(-\frac{x - \mu}{s}\right) \left[1 + \exp\left(-\frac{x - \mu}{s}\right)\right]^{-2} \quad ; -\infty < x < \infty$$

where μ = mean and s = scale.

LogNormal

$$\text{LogNormal}(\mu, \sigma) = \frac{1}{x \sigma \sqrt{2\pi}} \exp\left[-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right] \quad ; 0 < x < \infty$$

where μ = mean (location) and σ = standard deviation (scale) in log space.

ModifiedJeffreys

This is the same as Jeffreys but with a Uniform distribution when $x \leq a$.

$$\text{ModifiedJeffreys}(a, b) = \frac{1}{(x + a) \log((a + b)/a)} \quad ; 0 \leq x, 0 < a < b$$

Normal

$$\text{Normal}(\mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right] \quad ; -\infty < x < \infty$$

where μ = mean (location) and σ = standard deviation (scale).

Pareto

$$Pareto(s, \alpha) = \frac{\alpha s^\alpha}{x^{\alpha+1}} \quad ; 0 < s \leq x$$

where s = scale (also lower bound) and α = shape.

SkewNormal

$$SkewNormal(\xi, s, \alpha) = \frac{2}{s\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) \Phi(\alpha z) \quad ; -\infty < x < \infty$$

where $z = (x - \xi)/s$, α = shape and $\Phi(\cdot)$ is the standard Normal CDF. Right-skewed when shape > 0 ; left-skewed when shape < 0 .

StudentT

$$StudentT(\mu, s, \nu) = \frac{\Gamma((\nu + 1)/2)}{\Gamma(\nu/2) s \sqrt{\pi \nu}} \left(1 + \frac{z^2}{\nu}\right)^{-\frac{\nu+1}{2}} \quad ; -\infty < x < \infty$$

where $z = (x - \mu)/s$, ν = shape > 0 (not necessarily an integer). Note: If $\nu = 1$, this reduces to the **Cauchy distribution** which has no finite moments!

Triangular

Triangular is bounded between a and b , with mode = c (which may be a bound).

$$Triangular(a, b, c) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & ; a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)} & ; c < x \leq b \end{cases}$$

Uniform

$$Uniform(a, b) = \frac{1}{b-a} \quad ; a \leq x \leq b$$

UniformHW

$$UniformHW(\mu, \delta) = Uniform(\mu - \delta, \mu + \delta) \quad ; \delta \equiv \text{half-width} > 0$$

Weibull

$$Weibull(k, \lambda) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} \exp\left[-\left(\frac{x}{\lambda}\right)^k\right] \quad ; 0 < x < \infty$$

for k (shape) and λ (scale).

BivariateNormal

This distribution is implemented for *likelihoods only*; it may *not* be used as a prior. It returns a vector. See Example Body.

$$\text{BivariateNormal}(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)[] = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{z_x^2 + z_y^2 - 2\rho z_x z_y}{2(1-\rho^2)}\right] ; -\infty < z_x, z_y < \infty$$

where $z_x = (x - \mu_x)/\sigma_x$ (and similarly for z_y); $-1 \leq \rho \leq 1$ is the correlation coefficient.

A.2 Discrete

Distributions that are special cases of Multinomial have p or $p[]$ as the *first* argument unlike many literature definitions. Multinomial and Categorical distributions may have $p[]$ indexed from zero. However, $p[0]$, if Monitored, must be specified separately.

Bernoulli

$$\text{Bernoulli}(p) = \begin{cases} p & ; \text{if } x = 1 \text{ (True)} \\ 1 - p & ; \text{if } x = 0 \text{ (False)} \end{cases}$$

where $x \in \{0, 1\}$ and $p = \text{Prob}(\text{True})$.

Binomial

For MCMC, p and n cannot both be completely unknown.

$$\text{Binomial}(p, n) = \binom{n}{x} p^x (1-p)^{n-x} ; x = 0, 1, 2, \dots$$

where $\binom{n}{x} = \frac{n!}{x!(n-x)!}$ is a **binomial coefficient** = the number of ways to choose x out of n .

Categorical

$$\text{Categorical}(p[]) = \begin{cases} p_i & ; x = i \\ 0 & ; x = \text{invalid index} \end{cases}$$

DiscreteUniform

$$\text{DiscreteUniform}(a, b) = \frac{1}{b - a + 1} ; x \in \{a, a + 1, \dots, b - 1, b\}$$

Geometric

$$Geometric(p) = p(1-p)^x \quad ; x = 0, 1, 2, \dots$$

Multinomial

This distribution is implemented for *likelihoods only*; it may *not* be used as a prior. For MCMC, values of n must not be completely unknown.

$$Multinomial(p[], n) = \frac{n!}{n_0!n_1! \dots n_k!} p_0^{x_0} p_1^{x_1} \dots p_k^{x_k} \quad ; x = 0, 1, 2, \dots$$

NegativeBinomial

For MCMC, p and n cannot both be completely unknown.

$$NegativeBinomial(p, n) = \binom{n+x-1}{n-1} p^n (1-p)^x \quad ; x = 0, 1, 2, \dots$$

In this form, x is the number of “failures” prior to observing the n^{th} “success” and where $\text{Prob}(\text{success}) = p$. This form permits x to have all integral values starting with zero.

Poisson

$$Poisson(\lambda) = \frac{\lambda^x}{x!} \exp(-\lambda) \quad ; x = 0, 1, 2, \dots$$

where $\lambda > 0$ is the mean (and the variance).

A.3 Mixtures

Built-in (homogeneous) mixtures require very special handling. See Section 3.2. In the cases below, $w[]$ is a vector of weights and nc is the number of components. Weights must be > 0 and total one. All other parameters are as defined above.

These mixtures are implemented for *likelihoods only*; they may *not* be used as priors.

NormalMixture

$$NormalMixture(\mu[], \sigma[], w[], nc) = \sum_{k=1}^{nc} w[k] Normal(\mu, \sigma)[k]$$

PoissonMixture

$$PoissonMixture(\lambda[], w[], nc) = \sum_{k=1}^{nc} w[k] Poisson(\lambda)[k]$$

BivariateNormalMixture

This distribution returns a vector. See Example Iris.

$$BivariateNormalMixture(\mu_x[], \mu_y[], \sigma_x[], \sigma_y[], \rho[], w[], nc)[] = \sum_{k=1}^{nc} w[k] BivariateNormal(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)[k]$$

A.4 Generic

This distribution is meant to handle distributions not otherwise implemented.

$$Generic(expr, lb, ub) = \text{user-defined}$$

The first argument, *expr*, is the expression for the (natural) log of the PDF (*assumed correct and normalized*).

lb, ub are meant to be reasonable bounds for initialization purposes since *MacMCMC* will not know how to draw a sample from a Generic distribution. Instead, it will sample a *Uniform(lb, ub)* distribution.

See Example Gumbel and Section C.2.3.

Appendix B

Functions

This Appendix lists the definitions of all operators and functions known to the *MacMCMC* parser. Most of them should be familiar.

Study the examples provided in this package and in the associated [ebook](#) for proper usage.

B.1 Operators

B.1.1 Arithmetic

+ plus

- minus

* times

/ divide

^ exponentiation

= equals

<- equals

~ is distributed as

B.1.2 Logical

All of these return 1 (true) or 0 (false) which can be used as numbers.

< less than

<= less than or equals

> greater than

>= greater than or equals

== is equal to

&& and

|| or

B.2 Functions

exp exponential

log natural logarithm

sin sine of angle (in radians)

cos cosine of angle (in radians)

tan tangent of angle (in radians)

asin arcsine of angle (in radians $\rightarrow [-\pi/2, \pi/2]$)

acos arccosine of angle (in radians $\rightarrow [0, \pi]$)

atan arctangent of angle (in radians $\rightarrow [-\pi/2, \pi/2]$)

atan2(y, x) arctangent of angle (y/x $\rightarrow [-\pi, \pi]$)

abs absolute value

mod(y, x) y mod x

min of two expressions

max of two expressions

sqrt square root

rnd round to nearest integer

logFactorial natural logarithm of N! where N is an integer ≥ 0

invLogit inverse **logit**, returning probability p

phi standard Normal CDF

sum of 1-D vector, from 1 to length

mean of 1-D vector, from 1 to length

B.3 Reserved Constants

Pi 3.14159...

Inf infinity

NA datum not available (in a multi-dimensional array)

Appendix C

Technical Details

The material in this Appendix is provided in order that knowledgeable users can assess the credibility of some internal details of *MacMCMC*. Techniques are all based on existing art and references are cited where appropriate.

C.1 Software

MacMCMC and its sub-process, *MacMCMC_M2C2*, were developed in Xcode (v14.1). Languages used include C, C++, Objective-C, Objective-C++, Flex and Bison. The top-level, controlling process is *MacMCMC*; CPU-intensive computations are passed to one of an array of *MacMCMC_M2C2* processes. *MacMCMC* is an NSDocument application; *MacMCMC_M2C2* is a headless console tool.

C.2 MCMC

C.2.1 Variables

Internally, all data and non-scalar variables are constructed as arrays of size $sz+1$ so that indices $[1..sz]$ will be valid. Therefore, a zeroth element always exists even though indices are assumed to start at 1. This zeroth element *can* be used in a model (e.g., with Categorical distributions) provided all else is consistent but this is must be done with care. It is safest to assume that there is no zeroth element.

All array elements are initialized to $-\infty$. If a referenced value is not overwritten by the model, this (garbage) value will cause the initial trial run to fail.

C.2.2 Algorithm

MacMCMC utilizes ensemble MCMC exclusively—essentially a parallelized version of the algorithm of Goodman and Weare. [4] The number of walkers in an ensemble depends

on the number of parameters with the default being five walkers per parameter.

Parallelization is achieved by partitioning the sample, *not* the walkers. Each parallel `MacMCMC_M2C2` process creates a full complement of walkers to produce a subset of the total sample with each walker (chain) randomly initialized by selecting variates from the relevant priors. The final sample (trace) is the union of states visited by all walkers. Thinned states are not saved in the trace. Thinning default = 10.

This proliferation of chains can result in excessive memory requirements when the number of parameters becomes large. Available memory is checked in advance of a run.

In each `MacMCMC_M2C2` process, at each iteration, proposals are made for all model parameters for every walker. Updating of all walkers is thereafter executed synchronously.

C.2.3 Initialization

Variables are initialized by drawing random variates from their modeled distribution with parameters already determined from ancestor nodes. The latter *must* have already been initialized (in a previous model statement). A single draw is used for discrete variables and the average of five draws for continuous variables.

C.2.4 Burn-in

Burn-in must be complete before any sampling can be done. Burn-in parameters, `Nburnin` and `Tburnin`, may be chosen from the Low-level Setup dialog.

Burn-in continues for `Nburnin` iterations or until 90 percent of the walkers in a sub-process have moved at least `Tburnin` times, whichever is fewer.

C.2.5 Credible Intervals

Credible intervals are highest-posterior-density (HPD) intervals. Each is computed by sorting its trace column (low to high) then examining the values bounding every window of states containing `p` percent of the trace. The shortest range of values found is reported as the `p`-percent credible interval.

This procedure is appropriate mainly for unimodal marginals.

An extended set of credible intervals is used in the marginal -likelihood integration (see below).

C.2.6 Mode

The mode of a continuous variable is estimated by splitting the sorted trace column into 1,000 bins and reporting the midpoint of the tallest bin as the mode. Discrete variables have `binwidth = 1` and the tallest is reported as the mode.

C.2.7 MAP parameters

In most cases, the maximum *a posteriori* (MAP) parameters reported are determined by refining the best-found parameter vector using the Nelder-Mead simplex algorithm. [7] However, if the number of unknown parameters (those with a prior) is ≥ 20 and/or there are indicator variables, MAP parameters are those found that have the largest log(posterior). Checking may be done by increasing the MCMC sample size (trace length).

C.2.8 Chain Quality

The only statistic reported (for parameters, not Extras) is the Gelman-Rubin statistic for chain mixing. [3] This uses the saved trace which is sorted by chain (walker).

C.3 Marginal Likelihood

The numerical integration of the posterior, yielding the marginal (global) likelihood, is carried out using Nested Restricted Monte Carlo Integration. [5] This procedure involves partitioning the posterior hypervolume, restricted to that exhibited by the trace, into nested “shells” defined, initially, by the {30, 50, 70, 90, 95, 99, 99.9, 99.99}-percent credible intervals. Thus split, the integration of these shells is carried out in parallel and totaled to give the marginal likelihood.

The convergence criterion is a standard error less than 0.001. If this is not achieved with the starting shells, the tail of the posterior is split uniformly into five additional shells and these integrated sequentially. Checking the final integral may be done by increasing the MCMC sample size (trace length).

Numerical integration is performed using [Quasi-Monte Carlo](#) integration implemented via [Sobol sequences](#). Note that integration cannot be done in log space so log(posterior) values are first offset by subtracting the MAP value of the log(posterior). This is added back at the end and log(marginal likelihood) reported to three decimal places which is more than sufficient for model-comparison purposes.

Computation of the marginal likelihood is enabled by default but can be disabled. It is always disabled for built-in mixtures since these are subject to label switching and correct weights are unavailable.

C.4 Mixture Relabeling

If the model likelihood is one of the built-in homogeneous mixtures (pg. 38), *MacMCMC* will (by default) attempt to undo any label switching and restore a correct set of marginals. The algorithm employed is NORMLH which utilizes the trace alone and is otherwise ignorant of the model. [10] Like all such relabeling algorithms, it is not perfect but appears

to give acceptable results in most cases. As always, components that are well-separated will give the best results; overlapping components are inherently ambiguous.

Relabeling failures are somewhat random. If relabeling results in a warning that it was incomplete, it is usually worth restarting the run, perhaps with a larger relabeling limit (low-level option).

C.5 Marginal-plot Smoothing

Marginal plots for continuous monitored parameters and Extras are shown in raw form but these plots may be smoothed. Smoothing is carried out in frequency space using an algorithm adapted from that of Aubanel and Oldham. [1] The user interface for this (optional) operation is rather generous, allowing oversmoothing in nearly all cases.

C.6 Goodness-of-fit Credible Intervals

These (optional) credible intervals are for *predictions*. The general procedure involves determining prediction-interval limits using 1,000 random rows from the trace instead of the entire trace. Abscissa values are determined in different ways depending on the nature of the analysis.

- For equations, the X-range of the data is divided into 120 equal segments and the resulting 121 values used with each of the 1,000 parameter sets to compute predicted Y-values. The 1,000 Y-values are then used to get 121 95-percent credible intervals. The plot is generated using a cubic-spline fit to the top and bottom of the credible-interval band.
- With a q-q plot, the X-values are the original datapoints. This plot does not utilize a cubic spline; it merely connects the points.
- With a histogram plot, the X-values are the bin boundaries. A credible-interval band is constructed for each bin.

To minimize runtime, the abscissa values are distributed to available sub-processes (all using the same trace rows).

Appendix D

Examples

Table D.1: Examples

1	Body	Height (cm) vs. weight (kg) of adult men
2	Carbon-14	^{14}C specific activity (counts/min-g) vs. age (years)
3	Daytime	Daytime in Boston, MA, USA (min) vs. day number
4	Enzyme	Concentration vs. rate
5	FishCount	Count of fish caught
6	Gumbel	US major-league baseball: best batting averages (1876-2019)
7	Hale-Bopp	Cyanide radial release: rate vs. distance
8	Hyphens	Count per page of line-ending hyphens
9	Iris	Fischer's iris data
10	Lizards	Capture-recapture data
11	MP	Historical melting point data for n -octadecane
12	NormalTemp	Normal body temperature for adult males
13	Salaries	College faculty (hundreds of dollars)
14	SAT	Northern Virginia, USA, schools (2014)
15	TriplePoint	Data for triple point of n -nonadecane

Table D.2: Functionality vs. Examples

Functionality (or example type)	Example #														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Continuous Distribution	•					•			•		•	•	•	•	•
Discrete Distribution					•			•		•					
Equation		•	•	•			•								
BivariateNormal Dist.	•								•						
Generic Distribution					•	•				•					•
Truncated Distribution															•
Mixture Distribution					•				•				•		
Relabeled Mixture									•				•		
Goodness-of-fit			•				•	•				•			•
Multidimensional Input	•								•						
Extras			•	•			•	•						•	

Bibliography

- [1] AUBANEL, E. E., AND OLDHAM, K. B. Fourier smoothing. *Byte* (February 1985), 207–218.
- [2] GELMAN, A., ET AL. *Bayesian Data Analysis*, 3rd ed. Chapman & Hall/CRC, 2014.
- [3] GELMAN, A., AND RUBIN, D. B. Inference from iterative simulation using multiple sequences. *Statistical Science* 7, 4 (1992), 457–472.
- [4] GOODMAN, J., AND WEARE, J. Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science* 5, 1 (2010), 65–80.
- [5] GREGORY, P. C., AND FISCHER, D. A. A Bayesian periodogram finds evidence for three planets in 47 ursae majoris. *MNRAS* 403 (2010), 731.
- [6] MCCLAUGHLIN, M. P. *Data, Uncertainty and Inference*. 2019.
- [7] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes, 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [8] RAUER, H., ET AL. Optical observations of comet Hale-Bopp (C1995 O1) at large heliocentric distances before perihelion. *Science* 275 (1997), 1909.
- [9] RUBENSTIEN, R. Y. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., 1981.
- [10] YAO, W., AND LINDSAY, B. G. Bayesian mixture labeling by highest posterior density. *Journal of the American Statistical Association* 104 (2009), 758–767.