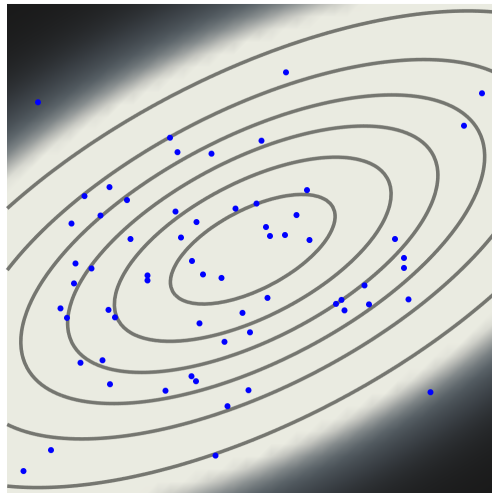


MacMCMC (v2.1) User Guide

Michael P. McLaughlin

November, 2025



***MacMCMC* (v2.1) User Guide**

Copyright © 2025 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 was produced using [L^AT_EX](#) and [TeXShop](#).

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

Contents

A Word to the Wise	v
1 Quickstart	1
1.1 Things To Try	2
1.2 System Note	3
2 Basic Procedure	4
2.1 Installation	4
2.2 Menus	5
2.3 Input	6
2.4 Setup Options	9
2.5 Run Status	10
2.6 Output	11
3 Modeling Language	15
3.1 Models	15
3.2 Mixture Distributions	21
3.3 What Might Go Wrong?	26
4 Goodness-of-fit	28
4.1 Goodness-of-fit Plot	29
4.2 Goodness-of-fit Examples	29
A Distributions	39
A.1 Continuous	39
A.2 Discrete	42
A.3 Generic	44
A.4 Mixtures	44
B Functions	46
B.1 Operators	46
B.2 Functions	47
B.3 Reserved Constants	48

C	Technical Details	49
C.1	Software	49
C.2	MCMC	49
C.3	Marginal Likelihood	51
C.4	Mixture Relabeling	52
C.5	Marginal-plot Smoothing	52
C.6	Goodness-of-fit Credible Intervals	52
D	Examples	53

List of Figures

1.1	Plot Showing Goodness-of-fit (mean estimates)	2
1.2	Goodness-of-fit Plot Plus Credible-interval Band	3
2.1	Initial Menus	5
2.2	Analyze Menu	6
2.3	Enzyme Model	7
2.4	Data Window	8
2.5	Setup Dialog	9
2.6	Status Window (Sampling phase)	11
2.7	Status Window (Done)	11
2.8	Report Window	12
2.9	Marginal for Parameter A	14
2.10	Marginal for trueMP	14
3.1	Enzyme Model2	16
3.2	Marginal for Parameter sig	20
3.3	Model vs. Data: With and Without Model Error	21
3.4	Salaries: Mixture Model	23
3.5	Salaries: Data and Model (mean estimates)	25
3.6	Salaries: Goodness-of-fit	25
3.7	A Bad Trace	26
4.1	Daytime Model	30
4.2	Goodness-of-fit for Daytime Example	31
4.3	Hale-Bopp Model	32
4.4	Goodness-of-fit for Hale-Bopp Example	33
4.5	Body-temperature Model	34
4.6	Goodness-of-fit for Body-temperature Example	35
4.7	Hyphens Model	36
4.8	Goodness-of-fit for Hyphens Example	37
4.9	Marginals for Zero-count and Two-count	38

List of Tables

- 3.1 Results for Model1 19
- 3.2 Results for Model2 20
- 3.3 Results for Salaries 24

- D.1 Examples 54
- D.2 Functionality vs. Examples 54

A Word to the Wise

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. [6] Users not already familiar with this approach to data analysis are **strongly** urged to read that book. This *User Guide* is software documentation only; 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.

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 fail! An MCMC model is like a little computer program in many respects. A lot can go wrong (see pg. 26). Be warned!

One common reason for failure is that array indices, or their priors, are defined in such a way as to overrun the actual array bounds or that some statement will *become* invalid, later in the run, in some unforeseen manner.

The parser 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.

MacMCMC is intended for the serious user—someone who needs the correct solution. If you are looking for software to provide quick-and-dirty answers, look elsewhere.

MICHAEL P. MCLAUGHLIN
MCLEAN, VA
NOVEMBER, 2025
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) → no need to change anything.
6. Click the Run button in the Setup dialog.

The program will execute in about three 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”).

All *MacMCMC* plots can be saved as either PDF (the default) or PNG.²

¹on a standard 2023 iMac

²PNG, sometimes better on webpages, is of lower (bitmap) quality.

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).

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).

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

Goodness-of-fit is important. *MacMCMC* can (in most cases) make its own goodness-of-fit plot if the model includes a goodness statement (see Chapter 4). Try Goodness-of-fit from the Analyze menu. In this example, you will get a window like that in Figure 1.1.³ This plot utilizes the posterior Mean parameters (the default).

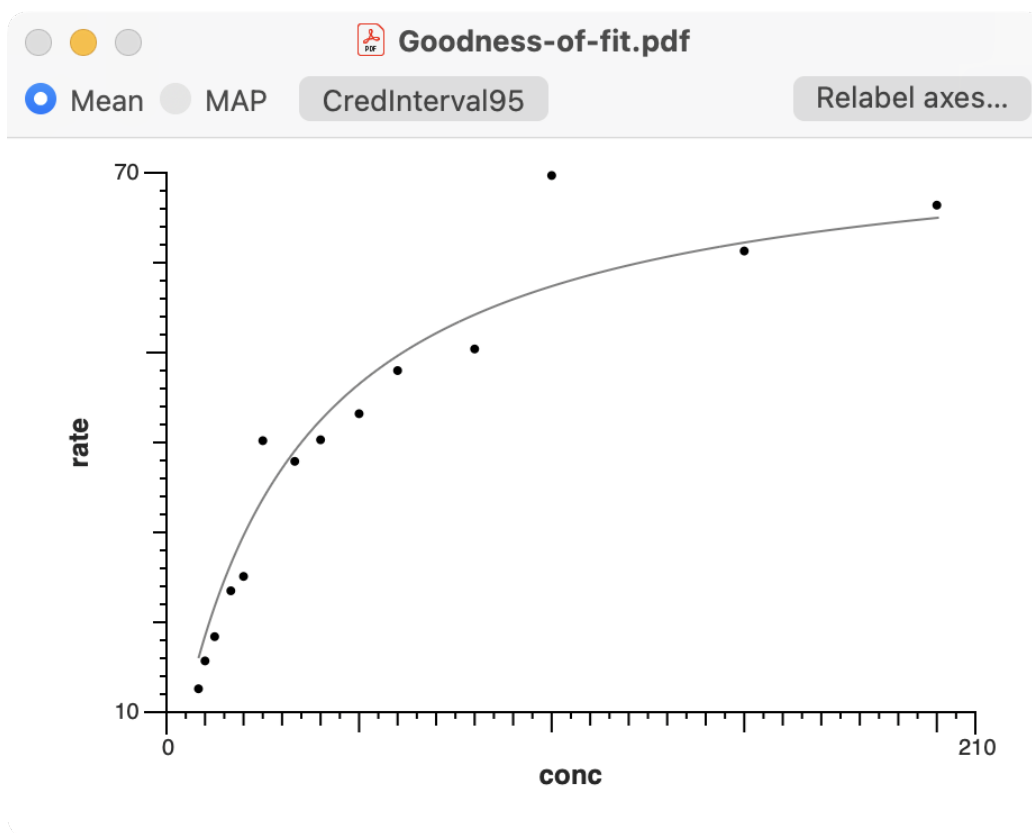


Figure 1.1: Plot Showing Goodness-of-fit (mean estimates)

This graph compares the data to the targeted [Michaelis-Menten](#) model (gray curve). If you toggle the CredInterval95 button, you will get Figure 1.2 showing the 95-percent credible-interval band around the model curve.

³Screenshots shown in this Guide reflect MacOS Sequoia™ design.

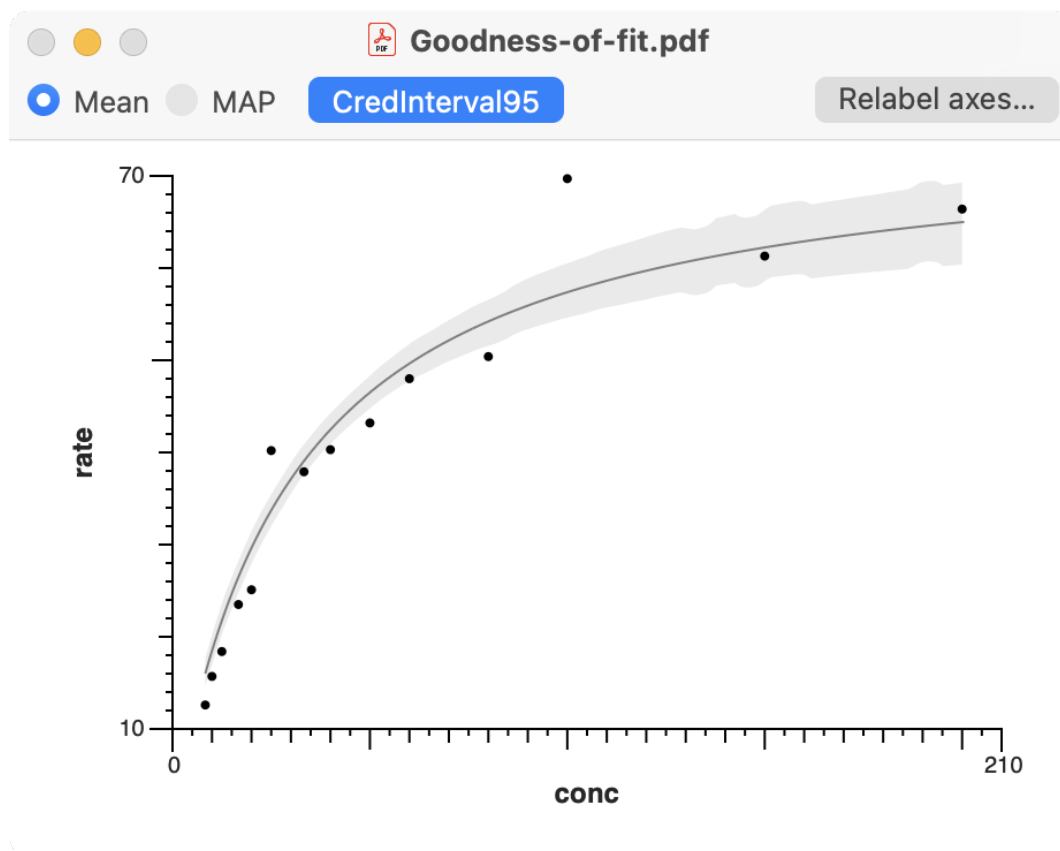


Figure 1.2: Goodness-of-fit Plot Plus Credible-interval Band

1.2 System Note

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

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, structure and syntax 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*. 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.

Do **NOT** attempt to run two copies of *MacMCMC* simultaneously; they will interfere with each other.

Abnormal Termination

If *MacMCMC* crashes (*extremely* unlikely), its sub-processes will persist as “zombies”. These *must* be eliminated or else they will interfere with subsequent runs. The simplest way to get rid of them is to open *MacMCMC* then Quit immediately. Of course, they can also be killed in the MacOS Activity Monitor in the usual way.

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.

2.2.1 *MacMCMC* and File Menus

These two menus are typical of almost any application.

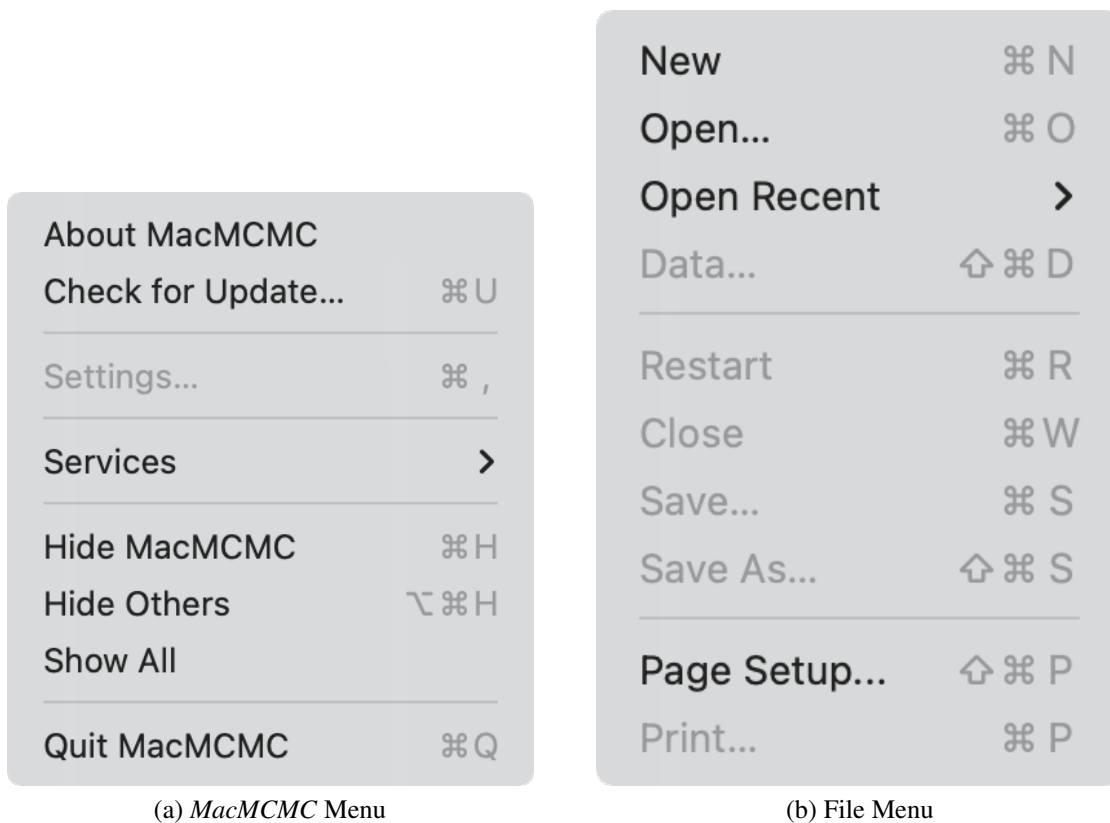


Figure 2.1: Initial Menus

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 may be a later version since other package elements might also be new.

The File menu as shown above is waiting for a model (*.mcmc*) file. If *MacMCMC* is opened by double-clicking or drag-and-dropping a model file, the File menu will appear with the New and 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, enabling Close. Restart to begin again with the (possibly edited) model and data.

2.2.2 Analyze Menu

Analysis consists in utilizing the model to “explain” the observed data. This model is a user input and there is, of course, **no guarantee whatever** that it is correct (or even sensible).¹ 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, the Analyze menu will appear as follows:

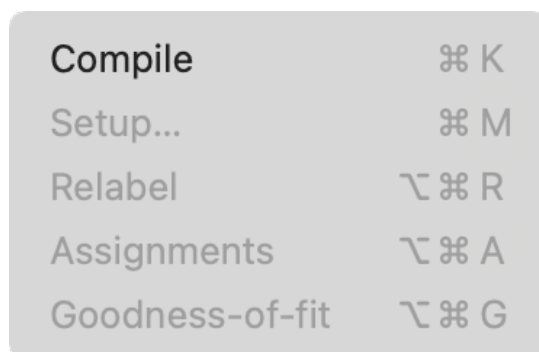


Figure 2.2: Analyze Menu

Successful compilation will enable the Setup... menu. Editing the model, which can be done in place, will require recompilation. The remaining menu items may become enabled if/when appropriate.

2.3 Input

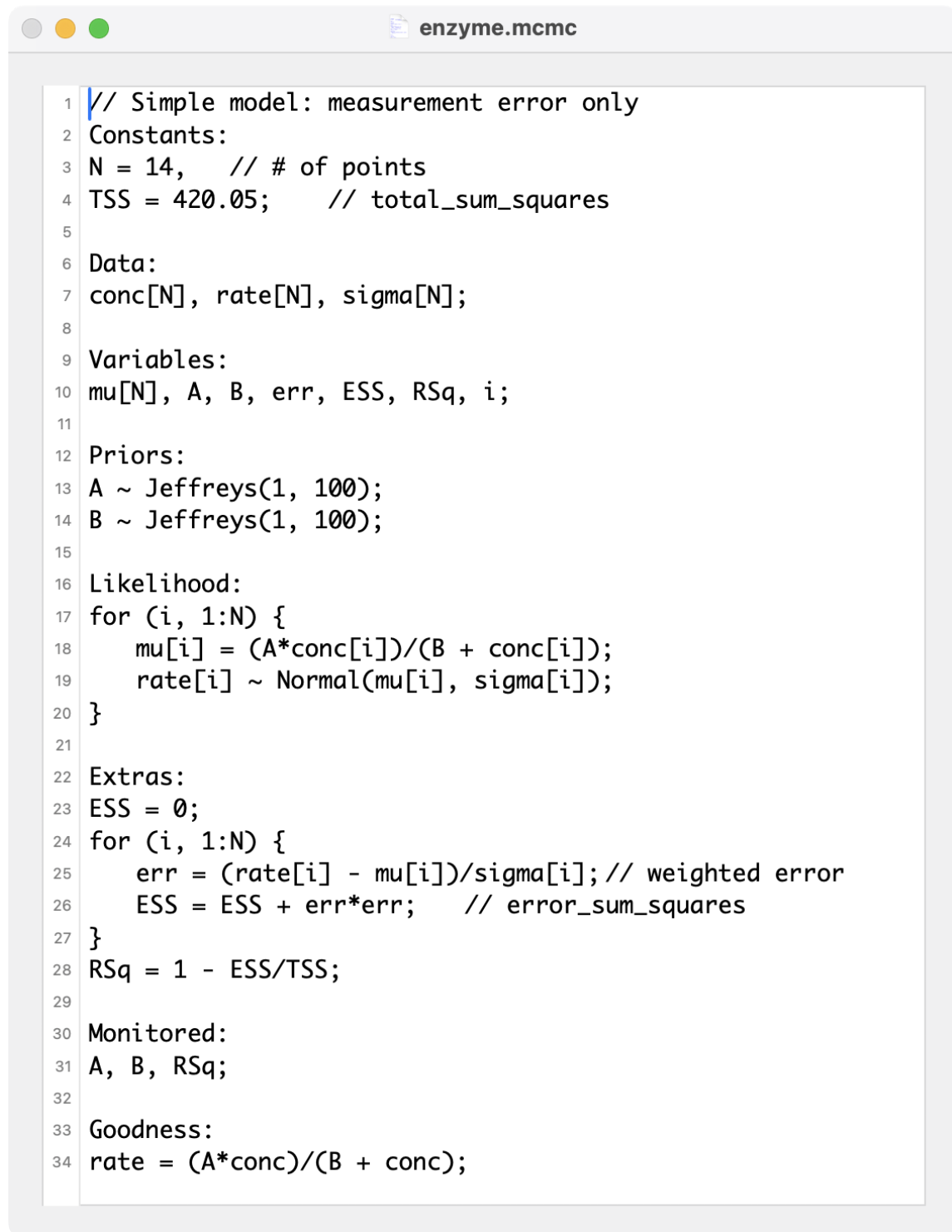
Model and data files must be plain ASCII (UTF-8) textfiles with **no** styling, accents, etc.

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 blocks, {Constants ... Monitored} are mandatory. Extras, however, are optional so this block may be empty. The final, Goodness block is completely optional but, if present, must contain just one statement (see Chapter 4). There *must* be at least one uncertain parameter (i.e., with a prior for Bayes’ Rule). At least one uncertain quantity *must* be Monitored. Constants must never be monitored.

¹See pg. v.



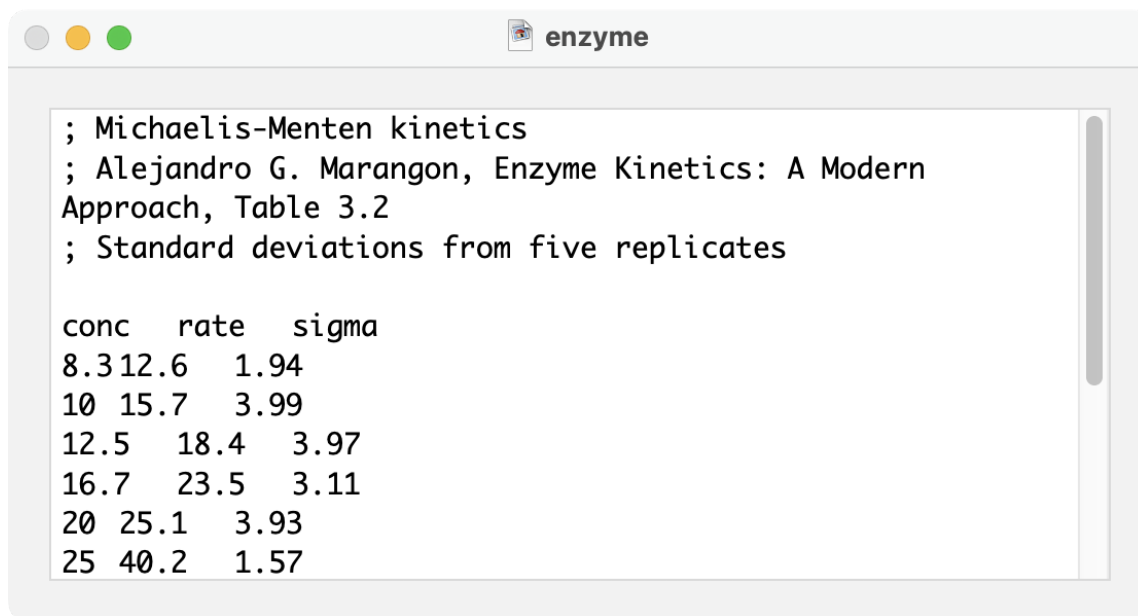
```
1 // Simple model: measurement error only
2 Constants:
3 N = 14, // # of points
4 TSS = 420.05; // total_sum_squares
5
6 Data:
7 conc[N], rate[N], sigma[N];
8
9 Variables:
10 mu[N], A, B, err, ESS, RSq, i;
11
12 Priors:
13 A ~ Jeffreys(1, 100);
14 B ~ Jeffreys(1, 100);
15
16 Likelihood:
17 for (i, 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;
32
33 Goodness:
34 rate = (A*conc)/(B + conc);
```

Figure 2.3: Enzyme Model

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.



```
; Michaelis-Menten kinetics
; Alejandro G. Marangon, Enzyme Kinetics: A Modern
Approach, Table 3.2
; Standard deviations from five replicates

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

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:

- The tab-delimited data array must be rectangular with one variable per column, as in the example, with *no blank lines or entries*. For multivariate 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 must 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.

Multivariate data

Any N datapoints must be scalars or 2-D vectors encoded as $x[N]$ or $x[N][2]$, respectively. In the data file, the values for every variable must be in a single column. Vectors must be in row-major order. See Example Body where the data are input as `WtHt[247][2]`.

In *MacMCMC* v2.1, built-in multivariate relationships include the BivariateNormal and TrivariateNormal distributions. See Appendix A.1.

2.4 Setup Options

The MCMC procedure allows many options specifying how the run is to be carried out. Several of these will be listed in the output Report (see Sect. 2.6).

There are two levels of Options (see Fig. 2.5), described below. In most cases, the defaults will suffice so nothing need be changed. The Run button will start the MCMC traversal. Cancel triggers a Restart after which the model, possibly edited (without saving), must be recompiled. (For more details, see Appendix C).

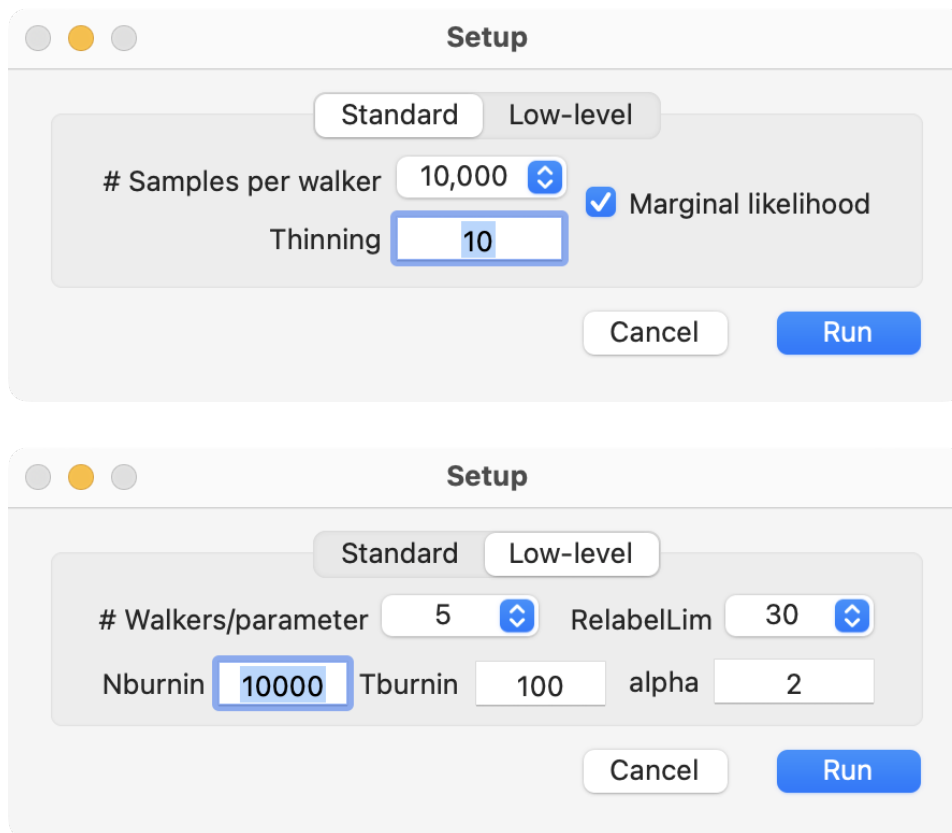


Figure 2.5: Setup Dialog

2.4.1 Standard Options

These Options are those most likely to be changed.

Samples per walker determines the size of the final (thinned) sample \rightarrow *trace*. 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 iterations are *skipped over* during a traversal (default = 9). Thinning inhibits autocorrelation.

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

2.4.2 Low-level Options

These options adjust the internal workings of MCMC as implemented in *MacMCMC*.

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

Nburnin and Tburnin control the burn-in phase.

RelabelLim Limit on relabeling attempts. Ignored when Relabeling is disabled.

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 during the sampling phase. 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.

When the run is finished, the Status dialog will show a pair of trace plots (black and red) for two selected chains (walkers) for a monitored variable 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 the trace itself is bad (e.g., a column is constant), this will abort the run with an error message followed by a Restart.

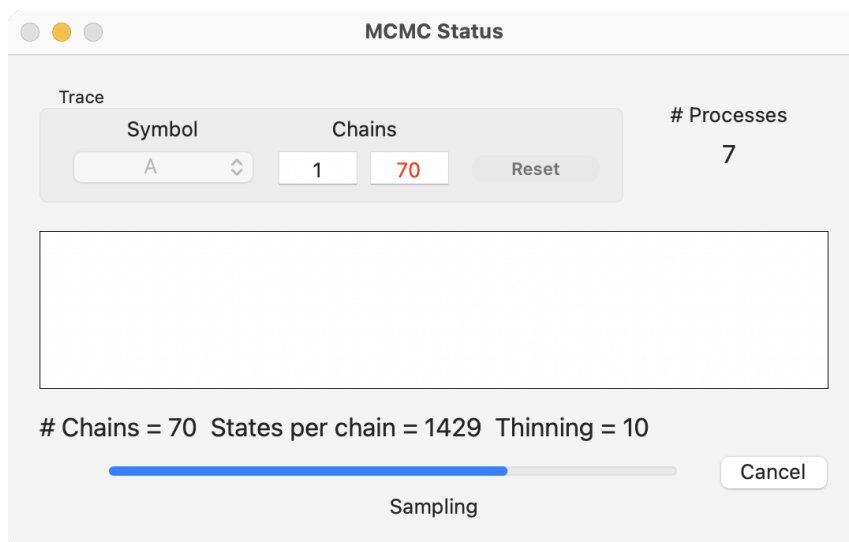


Figure 2.6: Status Window (Sampling phase)

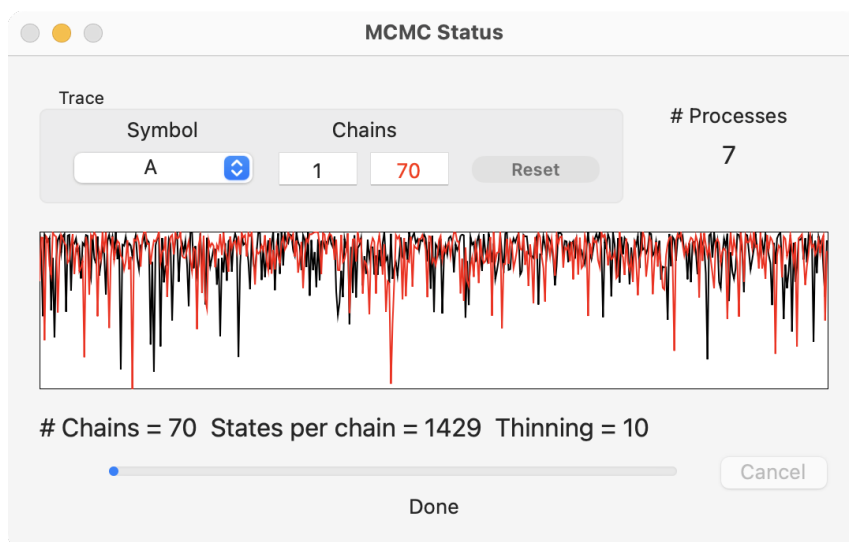


Figure 2.7: Status Window (Done)

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 a Report window for the Quickstart example. After the header, information about the run is summarized

followed by the marginal likelihood (unless deselected) then the posterior results for the monitored quantities (see below).

```

Report

Data: enzyme  Model: enzyme.mcmc  Sep 25, 2024 at 4:12:27 PM

Datapoints: 14
Parameters: 2
Walkers (chains) per parameter: 5
Sample size per walker: 10000 (after thinning = 10)
Total sample size: 100030 (7 processes)
Nburnin, Tburnin: 10000, 100
Runtime: 3 sec

log(marginal likelihood): -52.3626

A
MAP, Mean, Median, Mode, G-R stat: 74.3387 74.8899 74.7813 74.8449 1
Credible Intervals: 66.0059 67.7664 69.0028 80.8088 81.8812 84.548

B
MAP, Mean, Median, Mode, G-R stat: 29.8825 30.5239 30.3556 30.8254 1.002
Credible Intervals: 22.2062 23.947 24.7059 36.0636 37.5006 40.0349

RSq
MAP, Mean, Median, Mode: 0.925168 0.920456 0.921916 0.925096
Credible Intervals: 0.903187 0.910955 0.914238 0.925228 0.925228 0.925228

```

Figure 2.8: Report Window

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 seven sub-processes² and 100,000 is not a multiple of seven so the final sample size was rounded up to $70 \times 1429 = 100,030$. Since thinning was set to 10, there were, in fact, 1,000,300 post-burn-in iterations altogether (for 7 sub-processes with 10 walkers each). The saved trace is sorted by walker so here it contains 70 blocks of 1429 states in the order visited by the respective walkers.

All posterior information generated by the run is contained in the trace. All outputs—point estimates, credible intervals, marginals, goodness-of-fit—use the trace as their raw material. **Given a successful MCMC model, whatever is knowable about the unknowns can be found from this posterior trace.**

²See Status dialogs above. This number is computer-specific.

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 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.³

2.6.2 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 goodness-of-fit metric (where RSq = 1 is a perfect fit), computed as an Extra (see Fig 2.3).

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 (percent), lower-95, lower-90, . . . , upper-99. With parameters (not Extras), the Gelman-Rubin statistic (G-R stat) for chain mixing is then reported (see pg. 51). Once again, a value of one is perfect. A good value should be less than 1.01.

2.6.3 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.

This plot may be smoothed, unless the variable is discrete, and the axis labels may be edited.⁴ However, an 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, for instance, Figure 2.10⁵ from Example MP in which all the datapoints were in the range [300–302]. Of course, this can be avoided by offsetting the data. Proper data encoding, to maximize precision, is usually a good idea.

³All logs, here and elsewhere, are natural logs.

⁴Smoothing does not change the trace in any way.

⁵figure from file saved as PDF

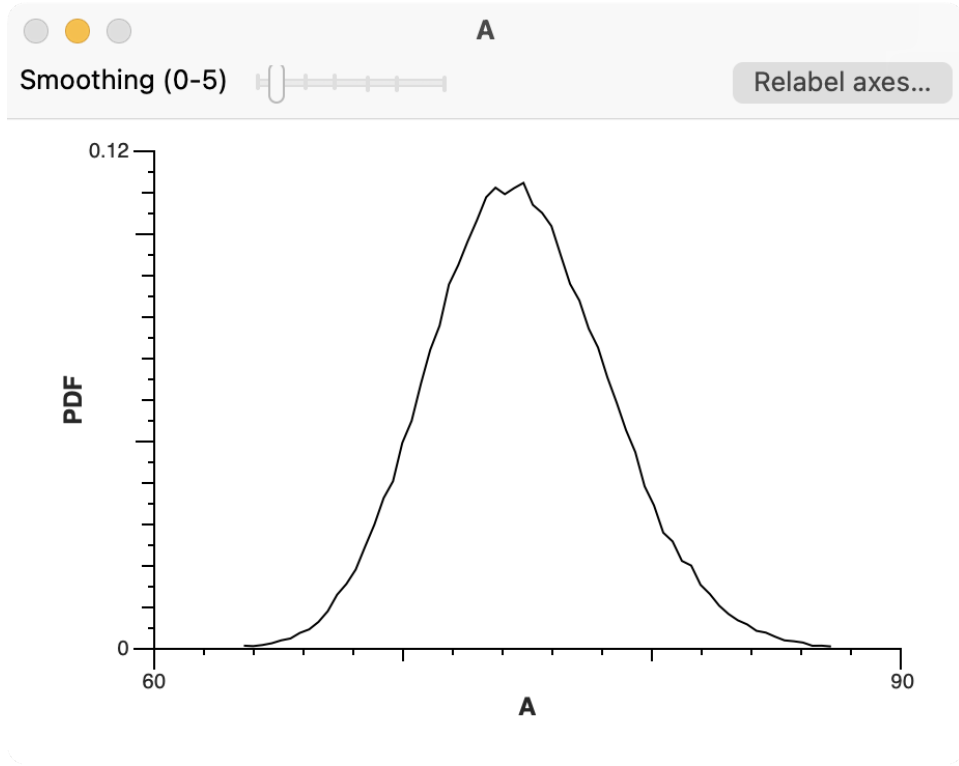


Figure 2.9: Marginal for Parameter A

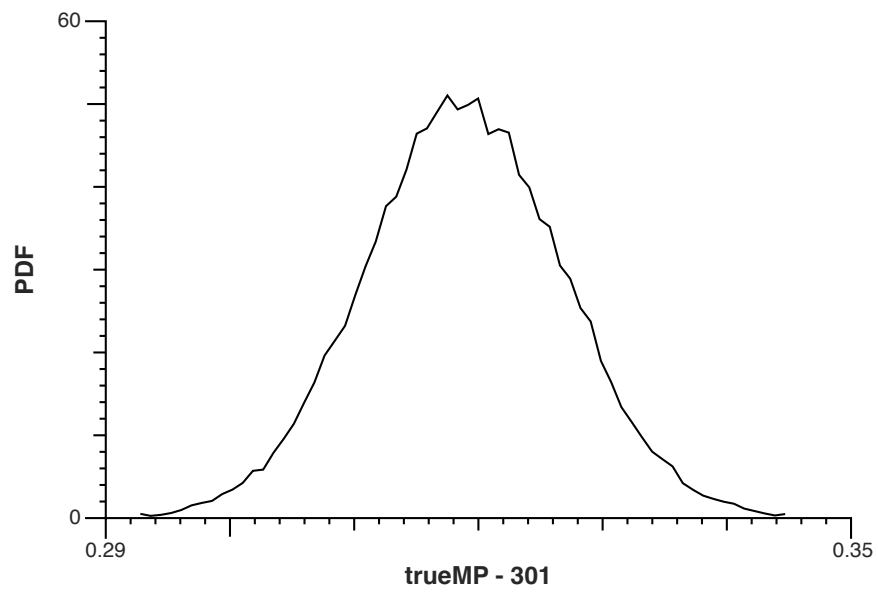


Figure 2.10: Marginal for trueMP

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.

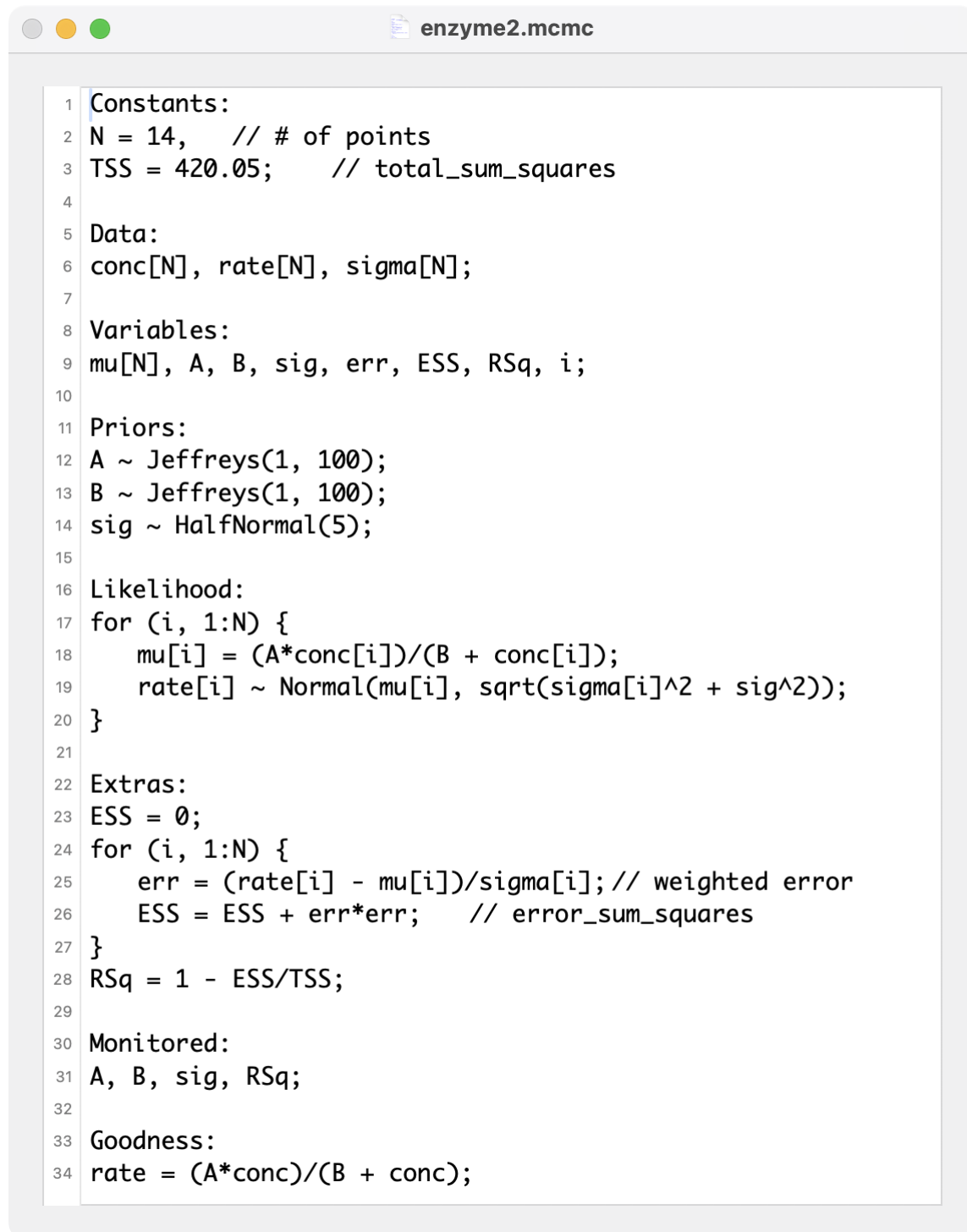
In addition to common functions and distributions, *MacMCMC* can handle generic distributions as well as discrete-mixture models. The latter will be discussed separately.

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 listing block by block focusing on required syntax and associated features.

¹Engineers often call this *system error*.



```
1 Constants:
2 N = 14,    // # of points
3 TSS = 420.05;    // total_sum_squares
4
5 Data:
6 conc[N], rate[N], sigma[N];
7
8 Variables:
9 mu[N], A, B, sig, err, ESS, RSq, i;
10
11 Priors:
12 A ~ Jeffreys(1, 100);
13 B ~ Jeffreys(1, 100);
14 sig ~ HalfNormal(5);
15
16 Likelihood:
17 for (i, 1:N) {
18     mu[i] = (A*conc[i])/(B + conc[i]);
19     rate[i] ~ Normal(mu[i], sqrt(sigma[i]^2 + sig^2));
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, sig, RSq;
32
33 Goodness:
34 rate = (A*conc)/(B + conc);
```

Figure 3.1: Enzyme Model2

3.1.1 General

The parser considers each model block 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.

Indexing always begins with 1 (*unlike C-style*) so that *MacMCMC* pseudocode will be consistent with the literature.²

3.1.2 Constants

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

Declare in this block any quantity (*must* be a scalar) 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., $\text{sqrt}(2*\text{Pi})$.

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 .

Data declarations must match the datafile columns exactly—symbol and dimension(s). However, the order of the declarations in the datafile and pseudocode 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 executed 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 25–26).

(Hyper-)parameters must be initialized before any iteration may be computed. This is done by selecting randomly 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. That is, no variable may appear on the RHS of any statement before it has appeared on the LHS of a previous statement. Otherwise, it 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.

²See, however, Appendix C.

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

- A symbol with an underscore as the last character³ defines an integer, required when, for instance, it will be described by a discrete distribution. 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 whenever there is an indicator variable 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 12–14). These are stochastic relationships, described with a tilde instead of an equals sign (or <- symbol). The Priors block may also include any deterministic relationships (equations) required or, indeed, equations for any variable that need not be recomputed with each datapoint.

In order for Bayesian inference to output a correct posterior,

a prior must NEVER be based on any data to be analyzed

except, perhaps, for data columns meant only to describe (label) the points in some way.

Note: *MacMCMC* can propose scalars only so multivariate distributions cannot be used as priors. The same is true for Multinomial and Categorical distributions; their components must be proposed separately.

3.1.6 Likelihood

Unless there is just one datapoint, the likelihood will consist of a for-loop with syntax as exemplified in lines 17–20. In some analyses, the data may be sorted in some fashion and the likelihood loop split into pieces accordingly (see Example SAT).

There **must** be a stochastic relationship to “explain” the data (here, line 19) with a data variable on the LHS of the statement. There may also be additional statements defining temporary variables, e.g., line 18. If the latter are not to be monitored or referenced in a Goodness statement, then they may be reused as needed.

Occasionally, an entire vector must be referenced. The *MacMCMC* syntax for an entire $x[N]$ vector is $x[]$ (see Example Body). Note, however, that this syntax is valid only for the rightmost dimension. That is, $x[][1]$ is not valid. Also, if $x[0]$ is defined, it is not included in $x[]$ (see Example Hyphens).

³deleted from plots

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, in order to assess the traditional (frequentist) goodness-of-fit.

3.1.8 Monitored

There must be at 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.⁴

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., $p[1]$, the trace will list $p[1]$, $p[2]$, \dots , $p[k]$.⁵

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.

Table 3.1: Results for Model1

Unknown	Estimate		95% Credible Interval	
	MAP	Mean	Lower Limit	Upper Limit
A	74.3387	74.8899	67.7664	81.8812
B	29.8825	30.5239	23.9470	37.5006
RSq	0.92517	0.92046	0.91096	0.92523

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 as described. By this criterion, Model1 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, Sect. 12.5] This method outputs the probability that one model is

⁴The trace will have been saved by this time, however.

⁵Trace files can get very large.

Table 3.2: Results for Model2

Unknown	Estimate		95% Credible Interval	
	MAP	Mean	Lower Limit	Upper Limit
A	76.4444	78.0696	67.1640	90.0171
B	33.3419	35.5252	23.3780	48.9063
sig	2.7055	3.4063	1.3524	5.7121
RSq	0.92214	0.90660	0.86629	0.92523

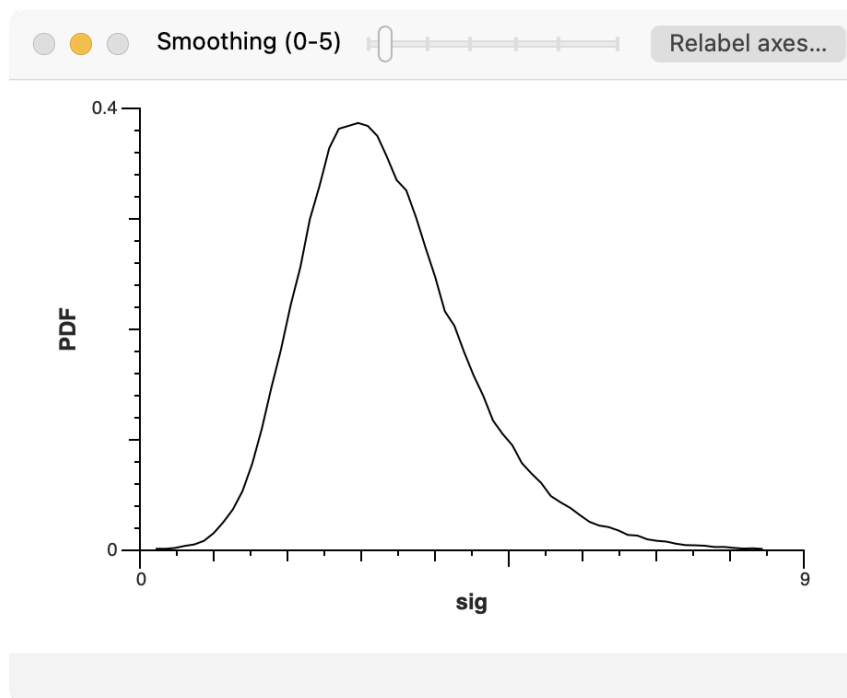


Figure 3.2: Marginal for Parameter sig

better (more credible) than another based on the full posterior whatever the model form or the number of parameters.

Here, reported $\log(\text{marginal likelihood})$ values for Model1 and Model2 are -52.3626 and -47.2677 , 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.

$$\text{odds Model2 better} \equiv \text{odds}M2 = \exp(-47.2677 - (-52.3626)) = 163.188 \quad (3.1)$$

i.e., 163::1 in favor of Model2.

$$\text{Prob}(\text{Model2 better}) = \frac{\text{odds}M2}{1 + \text{odds}M2} = 0.99391 \quad (3.2)$$

Therefore, by this state-of-the-art Bayesian technique, the probability that Model2 is better

than Model1 is more than 99 percent.⁶

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.⁷ This plot uses the mean parameter estimates from each report.

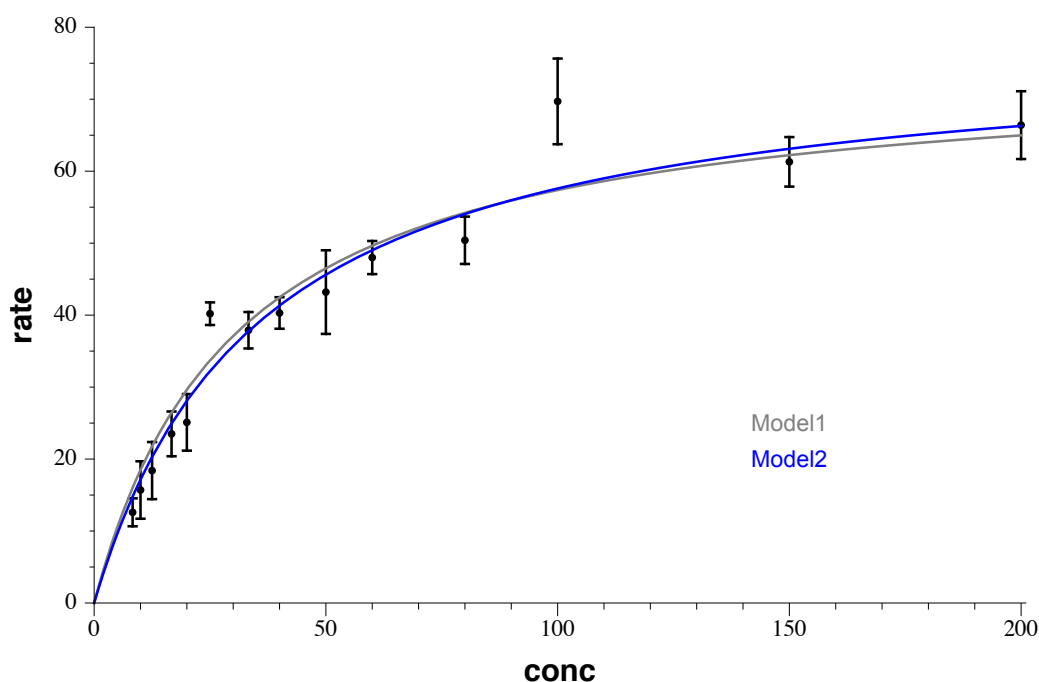


Figure 3.3: Model vs. Data: With and Without Model Error

3.2 Mixture Distributions

A (discrete) mixture distribution is a weighted combination of distributions (components), with all weights greater than zero and summing to one. *MacMCMC* defines the distribution *Mixture*(\cdot) (see Appendix A.4). There are, however, some limitations, as follows:

- The components must be all continuous or all discrete, matching the data.
- All components must have the same support (domain).
- No component may be a *Mixture*(\cdot), *Inflated*(\cdot), *Generic*(\cdot), *DiscreteGeneric*(\cdot) or doubly-bounded distribution. (See complete list on pg. 44)

⁶This computation should vary very little from run to run.

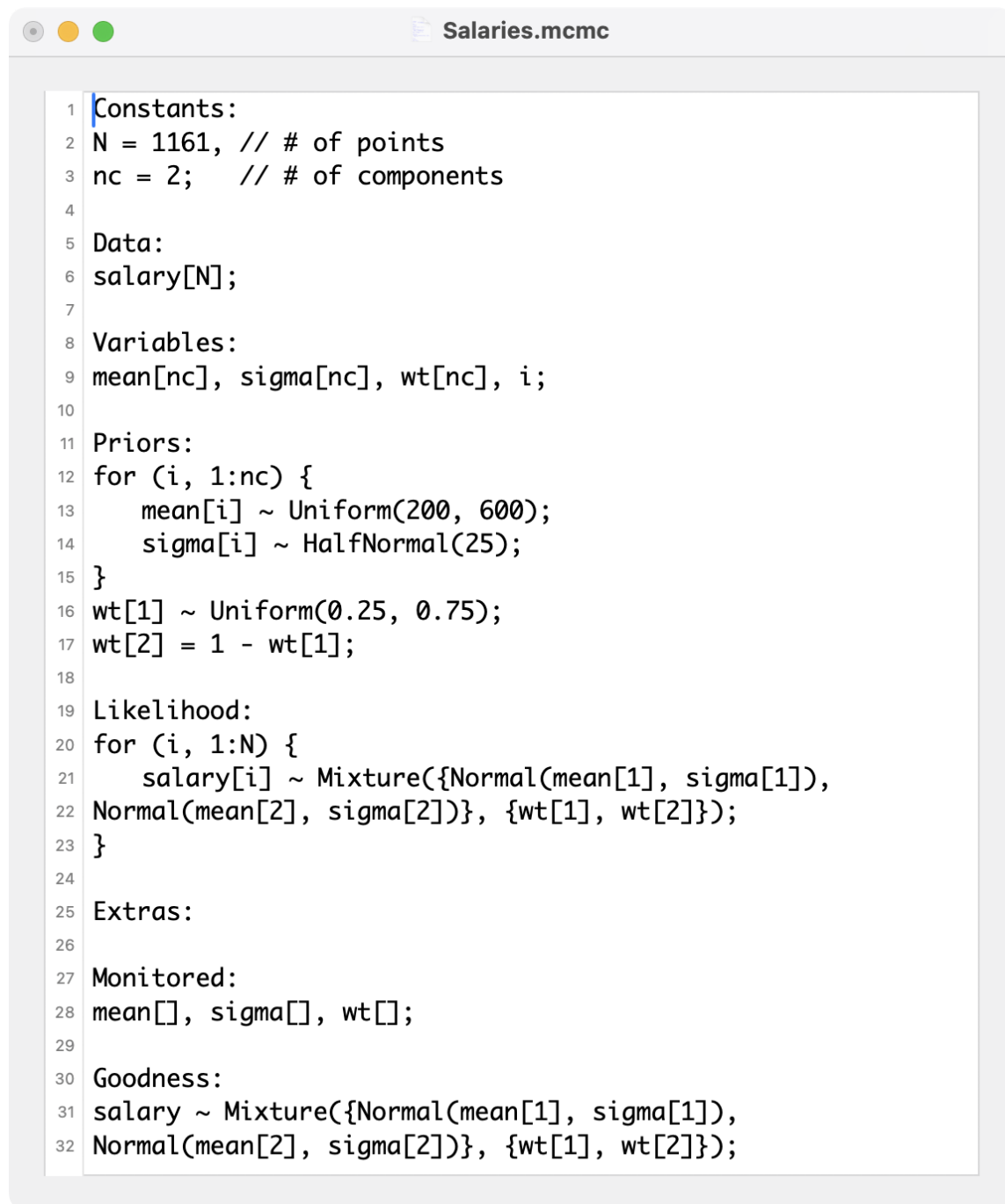
⁷Plot created offline in Mathematica (with *SetAxes*).

- Component arguments must be simple symbols, not formulas.
- The Mixture likelihood statement may not be split.

If all components are of the same form, then the mixture is termed *homogeneous* and *label switching* becomes a serious problem (see below). If a homogeneous Mixture is in the likelihood block, then there may be only one distribution statement in that block, i.e., no split for-loop. Otherwise, relabeling will be disabled.

The Salaries example will illustrate (see Fig. 3.4). The model likelihood is a Gaussian mixture with two components. As the initial (bimodal!) marginals will show all too well, relabeling will be necessary—an option executed from the Analyze menu.

It is worth noting that mixtures typically require a lot of data, a large trace and, perhaps, a really long burn-in phase.



```
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] ~ Mixture({Normal(mean[1], sigma[1]),
22 Normal(mean[2], sigma[2])}, {wt[1], wt[2]});
23 }
24
25 Extras:
26
27 Monitored:
28 mean[], sigma[], wt[];
29
30 Goodness:
31 salary ~ Mixture({Normal(mean[1], sigma[1]),
32 Normal(mean[2], sigma[2])}, {wt[1], wt[2]});
```

Figure 3.4: Salaries: Mixture Model

3.2.1 Label Switching

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.⁸ In favorable cases it can be fixed by relabeling the trace columns (see Analyze menu and Appendix C). However, 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.

Important: Relabeling, if selected, will apply only to a mixture in the likelihood block. Therefore, using a homogeneous mixture as a prior is not recommended unless its hyperpriors are sufficiently informative that label switching is minimized.

When relabeling is in effect, there will be two additional output files: *traceRL.txt* which is the relabeled trace and *ReportRL.txt*, a revised Report.

Another optional file is *probAssign.txt* (Analyze/Assignments menu), available once relabeling is finished. 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]. The last line in this file contains estimates for the weight of each component after relabeling. These are usually a bit different from the weights in *ReportRL*.

3.2.2 Results for the Salaries Example

Figure 3.5 shows the model (reabeled) compared to the data.⁹ The (weighted) individual components are shown as dashed curves; the mixture PDF is the solid curve. Figure 3.6 shows a goodness-of-fit plot, plus credible interval, output by *MacMCMC*.

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	—	—

⁸It also affects maximum-likelihood computations.

⁹another Mathematica+SetAxes plot

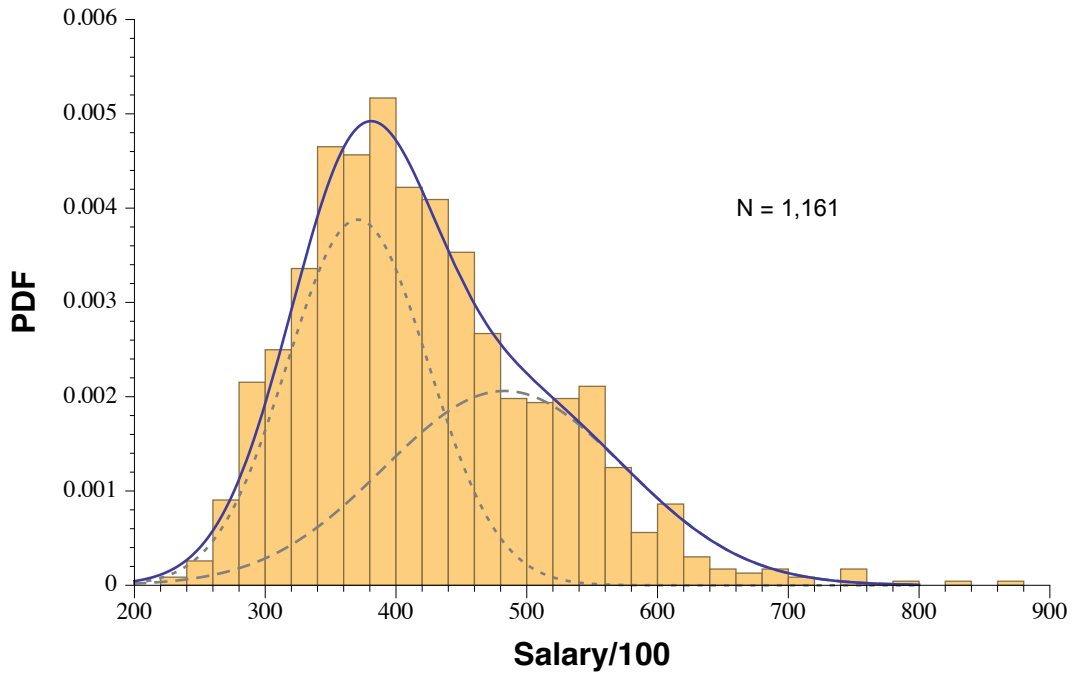


Figure 3.5: Salaries: Data and Model (mean estimates)

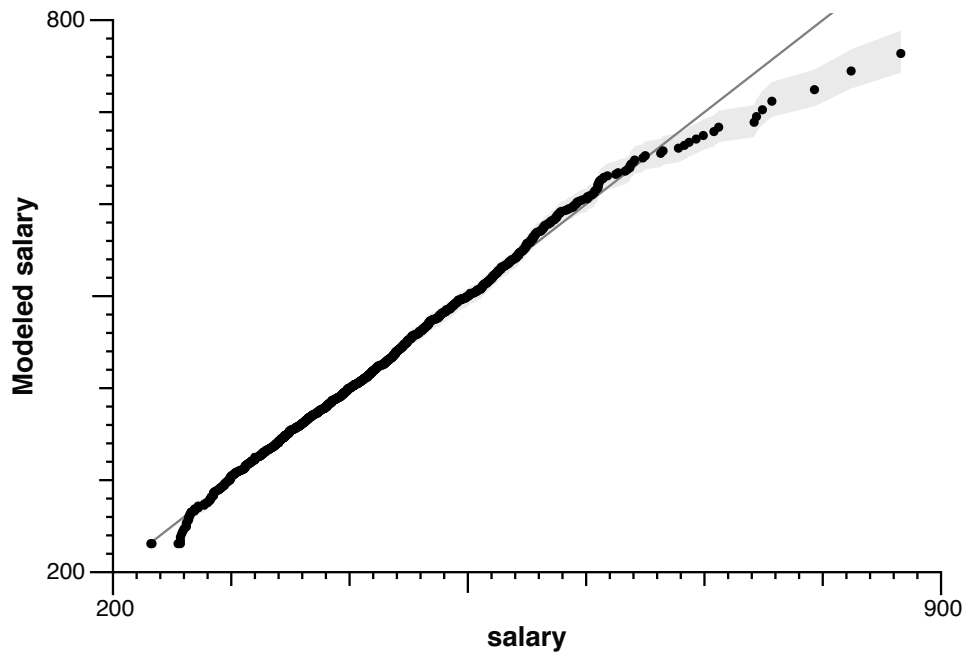


Figure 3.6: Salaries: Goodness-of-fit

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. A bad trace shows poor mixing of the black and red lines— something like this:

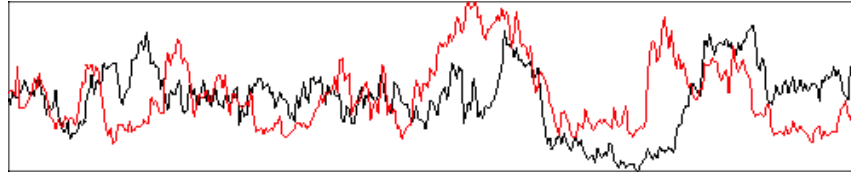


Figure 3.7: A Bad Trace

Regarding the model, there are several things that must be kept in mind. The list below is not comprehensive but it does offer some useful hints.

- Check distributions to make sure that they agree with their *MacMCMC* definitions (Appendix A). Such definitions sometimes differ in the literature. Also, check the order in which their arguments are supposed to appear. Check function syntax as well (Appendix B).
- 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! It is also a common mistake and will usually crash any program. (see pg. 4)
- A prior with LHS of the form `var[symbol][[]]` is invalid but, again, it is not a syntax error and will probably not be caught. Multivariate priors are not allowed.
- 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 variables that should be normalized *are* normalized and those that should be positive cannot be otherwise at any iteration.
- Check that any reused variable is “out of scope” before it is used again.
- Make sure that every (hyper-)parameter has exactly one prior.
- Make sure that priors for bounds of bounded likelihoods **never** contradict the data. If this happens, MCMC will fail. A bounded likelihood requires prior knowledge of the theoretical bound(s). Unless this is simply zero, the run will almost certainly give an error.

- 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. At the very least, credible intervals will be too narrow.

If necessary, draw a *random* sub-sample from the dataset and use *that* in exploratory analysis, to construct more informative priors, 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 DAG which does not 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 one or more walkers might get “caught” in a false local optimum giving very poor results, usually evident from a bad trace (see Fig. 3.7).
- Relabeling can fail (be incomplete). If that happens, it is worth trying again. You can always Restart, of course.
- 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 54.

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 sophisticated functionality to aid in this process. It can generate goodness-of-fit plots automatically with no additional input from the user. That said, *MacMCMC* was not designed as a plotting application and there is a price to pay for the convenience it offers. Specifically, the Goodness block, if present,

- Must contain a single statement describing the data
- Must have one data symbol on the LHS and just one data symbol on the RHS
- Must not use any indexed symbols
- Must not depend on data arrangement (e.g., sorted data)
- Must not depend on data grouping (e.g., *via* some input covariate)
- Must not describe a multivariate or Generic relationship

The reason for these limitations is that plotting often requires abscissa points not in the data or from an unavailable closed-form CDF. Its advanced credible-band functionality (see below) also makes some demands of its own.

In spite of this, the functionality available is very powerful and not generally found in MCMC software.

4.1 Goodness-of-fit Plot

MacMCMC creates one of three kinds of plot depending on the nature of the relationship:

X–Y Plot for deterministic relationships (equations)

Quantile–quantile (q–q) Plot for continuous stochastic relationships

Histogram for discrete stochastic relationships

All will display the data as black dots (or lines) and the model curve (or lines) in gray. Plotting utilizes *posterior-predictive* techniques, arguably the best approach to goodness-of-fit testing. [2, Sect. 6.3] The model shown will use either posterior Mean (default) or MAP parameters. The `CredInterval95` button will toggle the corresponding credible-interval band ON or OFF (default). The `Relabel axes...` button will enable that capability for the axes only, not the tick marks. Figure 1.1 showed all of this earlier.

4.1.1 Credible Interval

In *MacMCMC* (version 2.0+), the goodness-of-fit plot has a `CredInterval95` toggle. This option functions independently of the Mean/MAP choices. When clicked, 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.

4.2 Goodness-of-fit Examples

Here, we present four examples, with credible intervals, as follows:

Daytime an equation with a very small amount of measurement error

Hale-Bopp an equation with significant measurement error

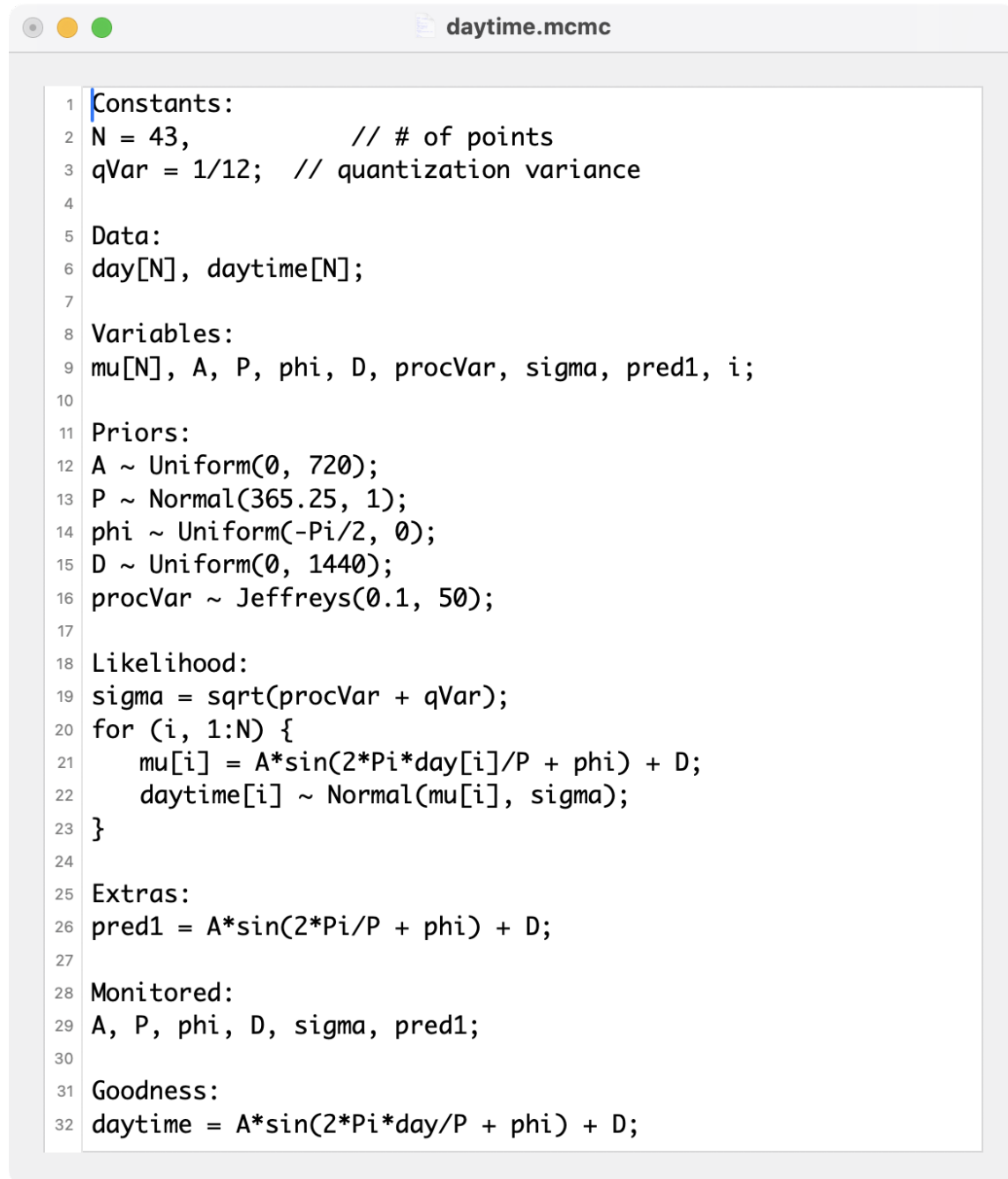
Body Temperature a continuous distribution

Hyphens a discrete distribution

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

4.2.1 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 daytime = A*sin(2*Pi*day/P + phi) + D;
```

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.

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 4.2](#).

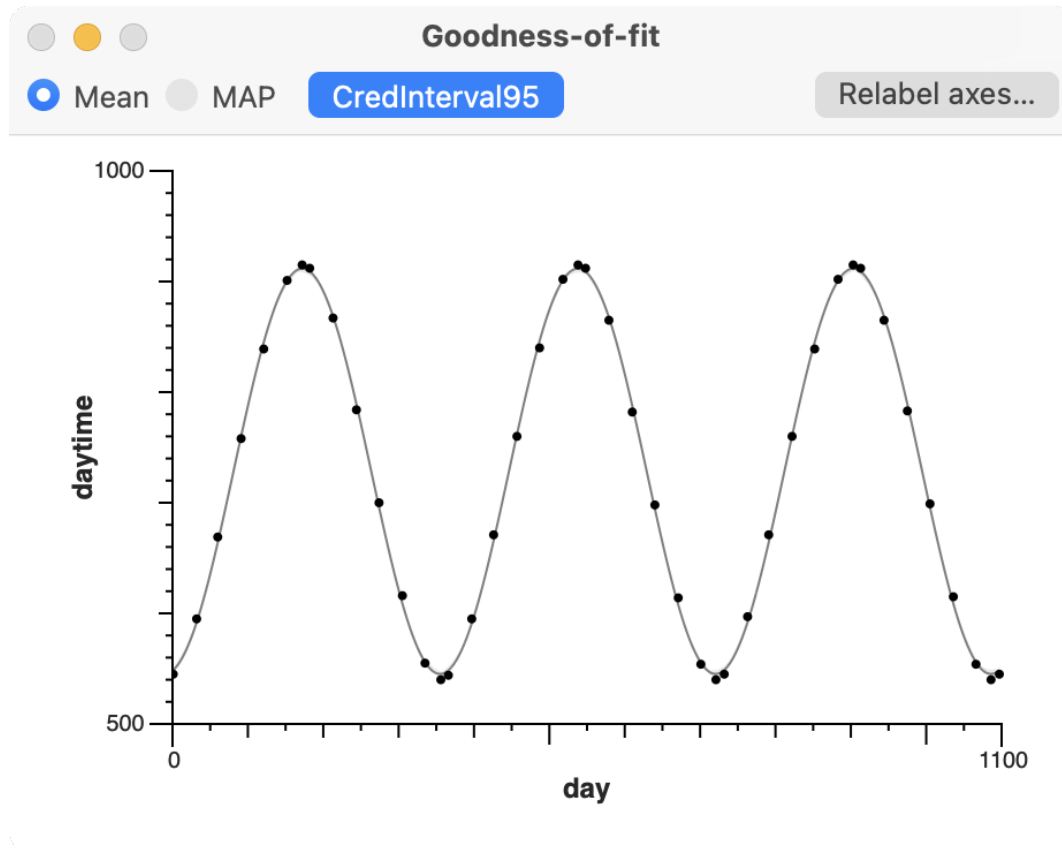
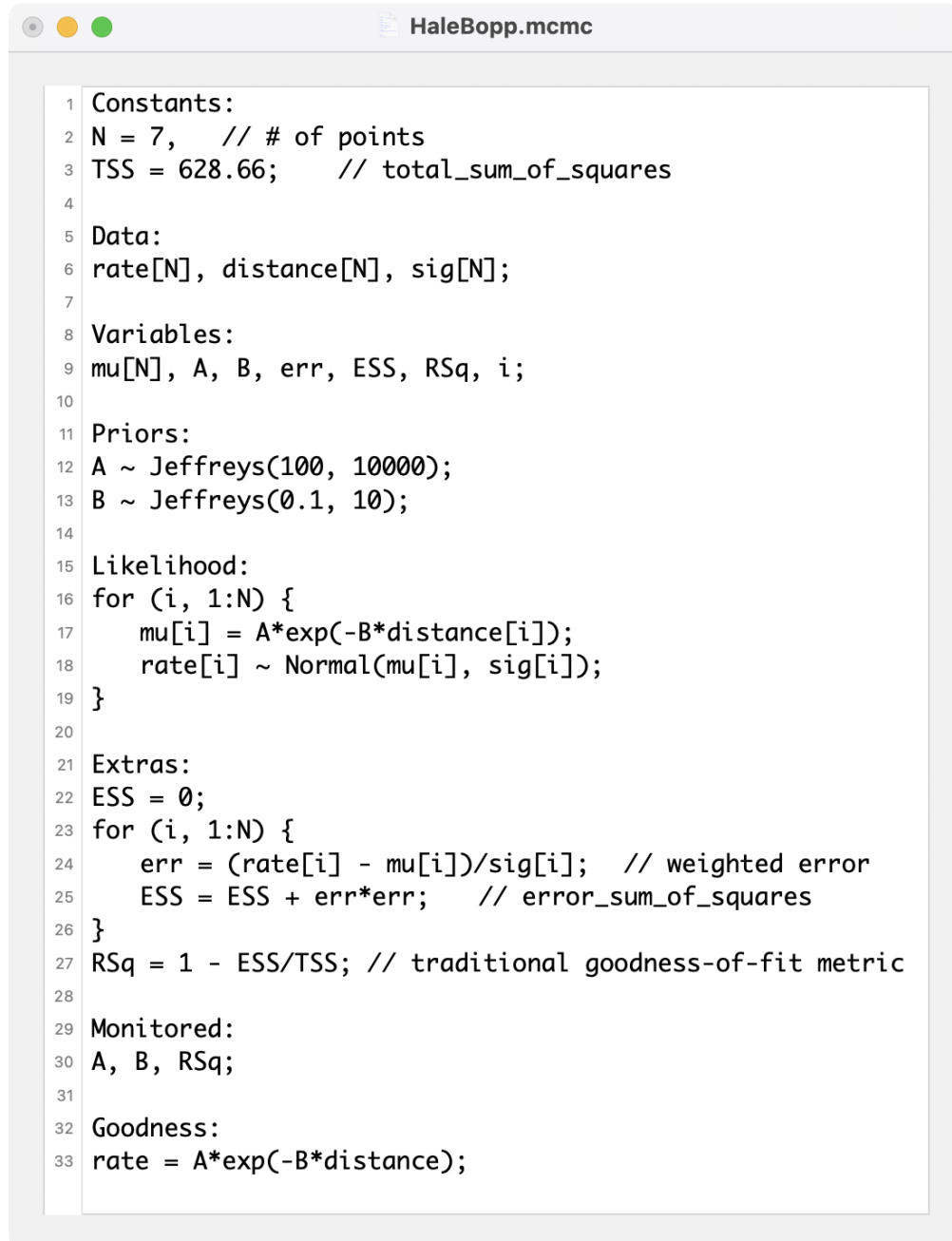


Figure 4.2: Goodness-of-fit for Daytime Example

It should be apparent that this model is a very good fit to the data even though it is only approximately correct. In fact, the credible-interval band is almost too small to see. The report gave a mean value for *sigma* of just 4.5 minutes.

4.2.2 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] The model is shown in Figure 4.3. The R-squared metric was computed as an Extra.



```

1 Constants:
2 N = 7, // # of points
3 TSS = 628.66; // total_sum_of_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_of_squares
26 }
27 RSq = 1 - ESS/TSS; // traditional goodness-of-fit metric
28
29 Monitored:
30 A, B, RSq;
31
32 Goodness:
33 rate = A*exp(-B*distance);

```

Figure 4.3: Hale-Bopp Model

The goodness-of-fit plot is shown in Figure 4.4. There is a lot of measurement error, so the credible-interval band is quite large

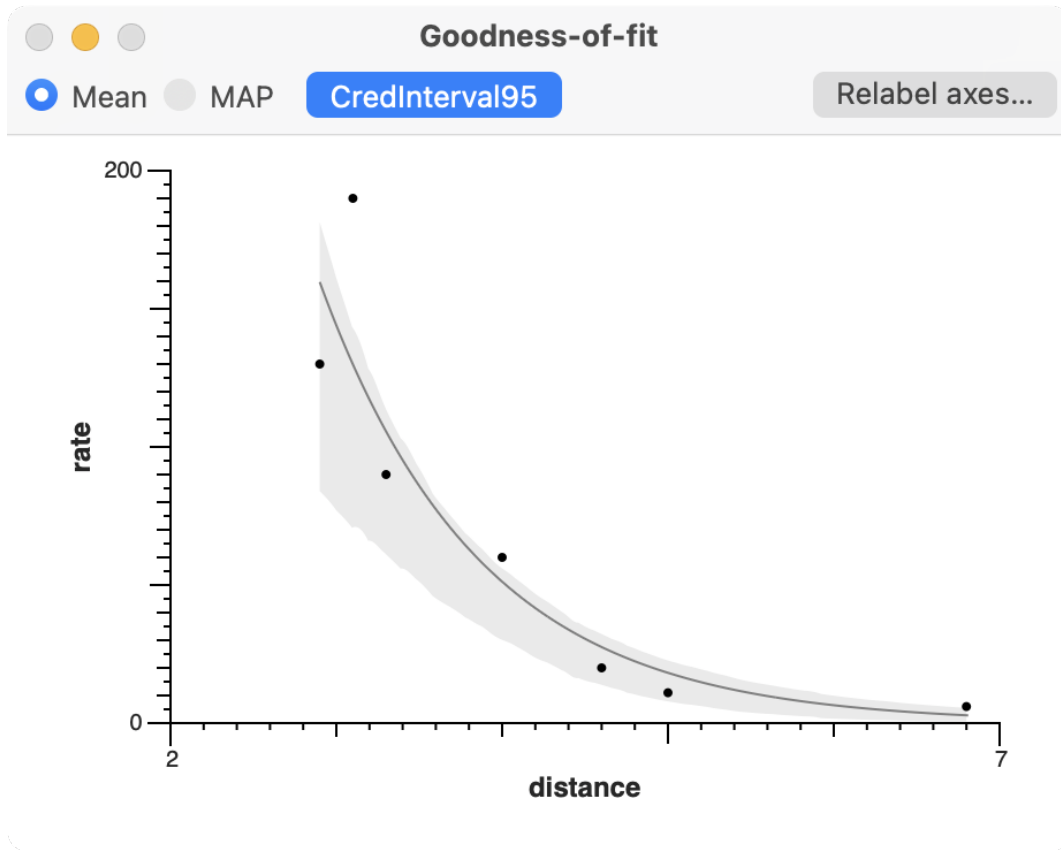
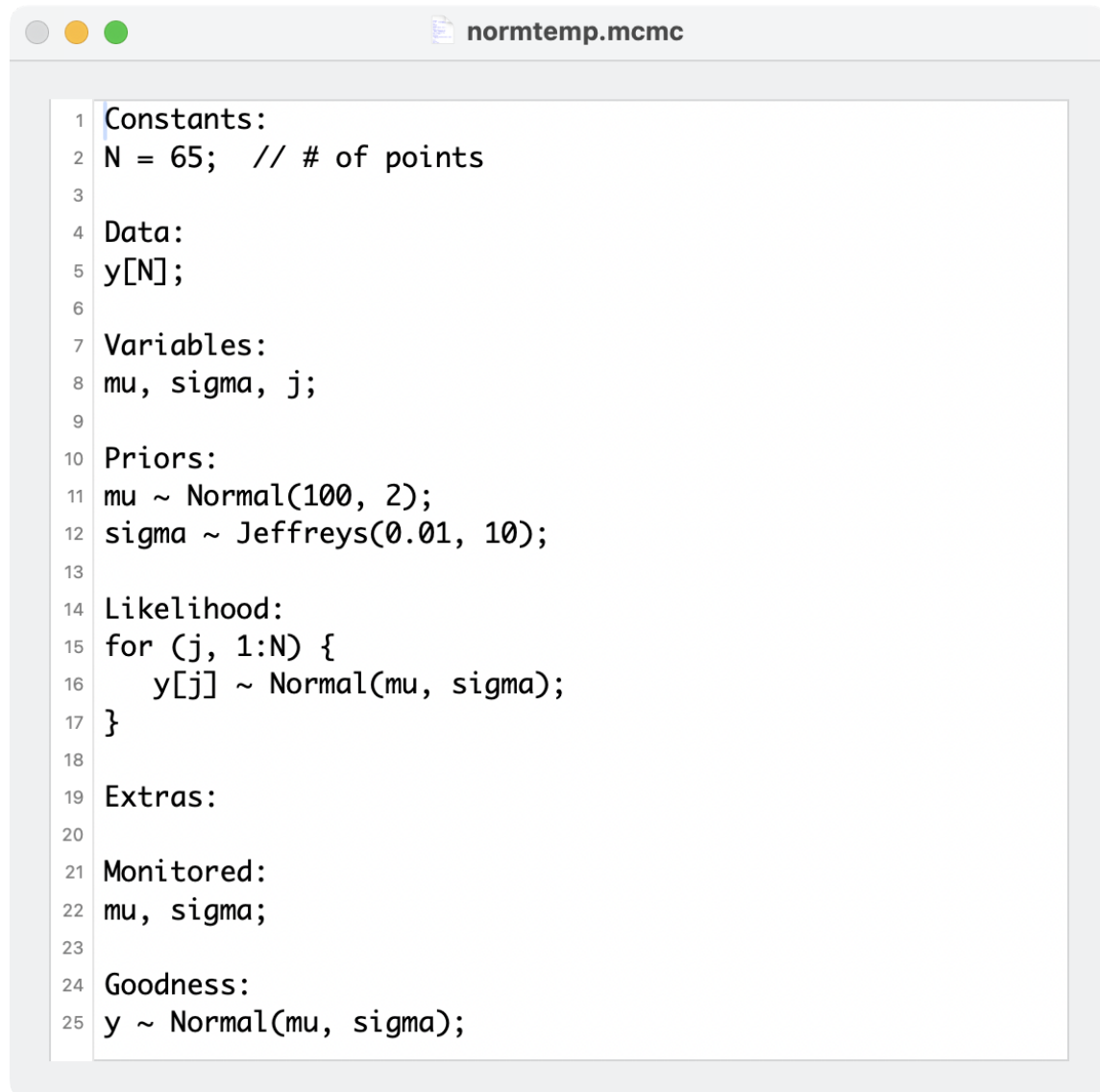


Figure 4.4: Goodness-of-fit for Hale-Bopp Example

4.2.3 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 $\sim \text{Normal}(\mu, \sigma)$ as described by the model in Figure 4.5.



```
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 y ~ Normal(mu, sigma);
```

Figure 4.5: Body-temperature Model

The goodness-of-fit (q–q) plot is shown below.

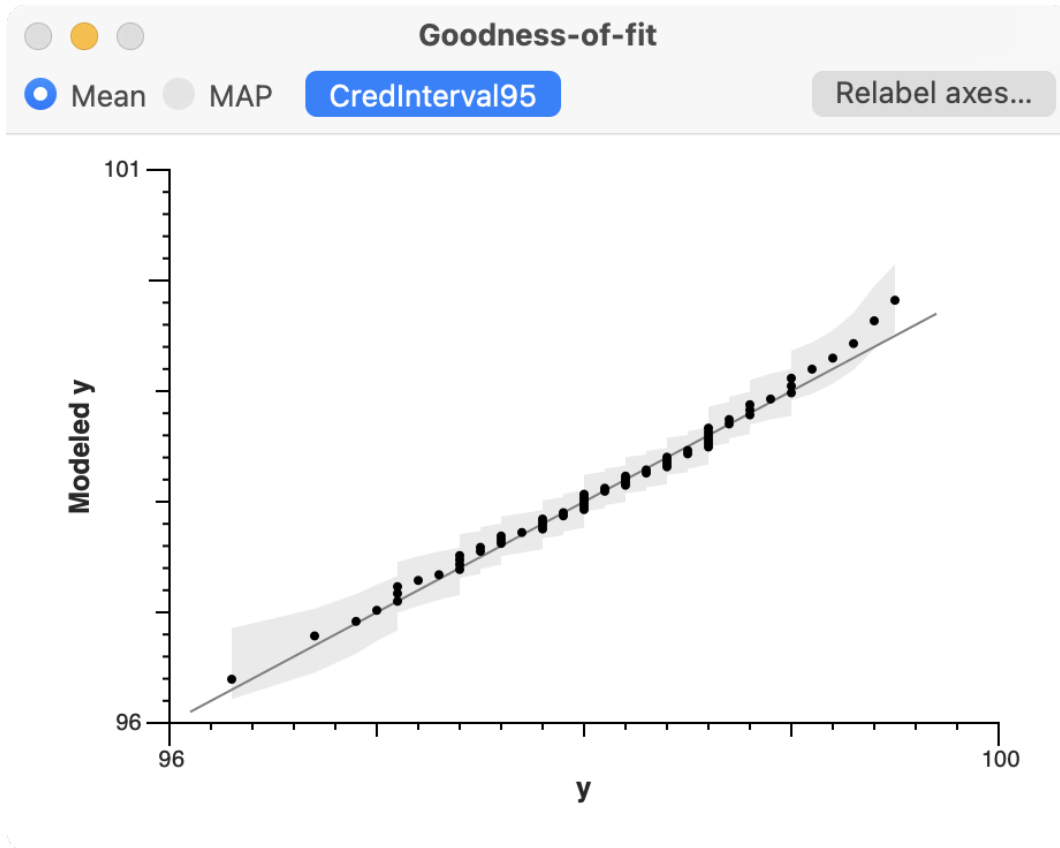
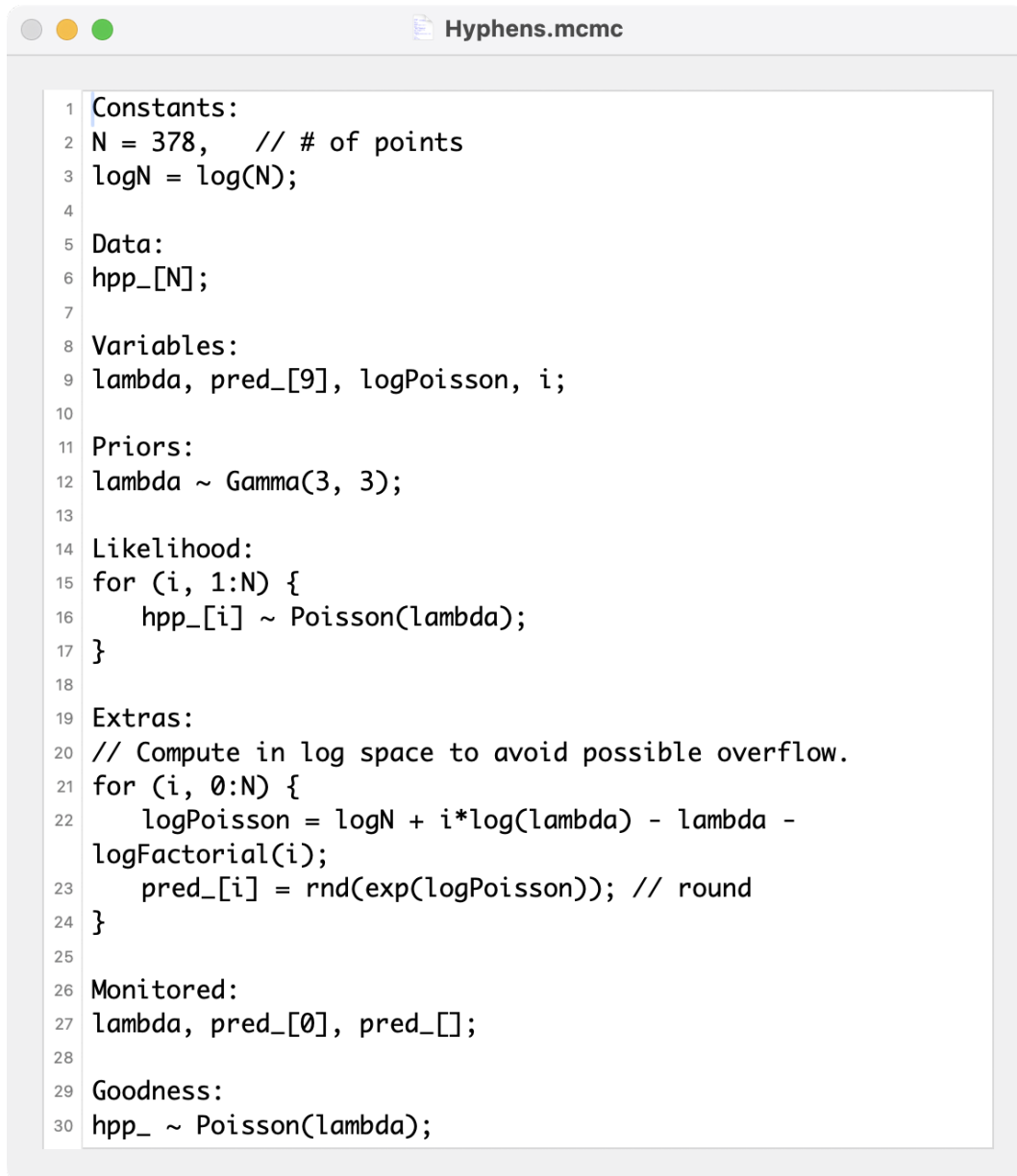


Figure 4.6: Goodness-of-fit for Body-temperature Example

4.2.4 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. 4.7).



```
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 -
23   logFactorial(i);
24   pred_[i] = rnd(exp(logPoisson)); // round
25 }
26 Monitored:
27 lambda, pred_[0], pred_[];
28
29 Goodness:
30 hpp_ ~ Poisson(lambda);
```

Figure 4.7: Hyphens Model

The Poisson parameter, λ , is also the mean of that distribution. The report gave a mean value for λ 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 below is not bad for a one-parameter model. With discrete data, a q–q plot is not appropriate so a PMF histogram is output instead.²

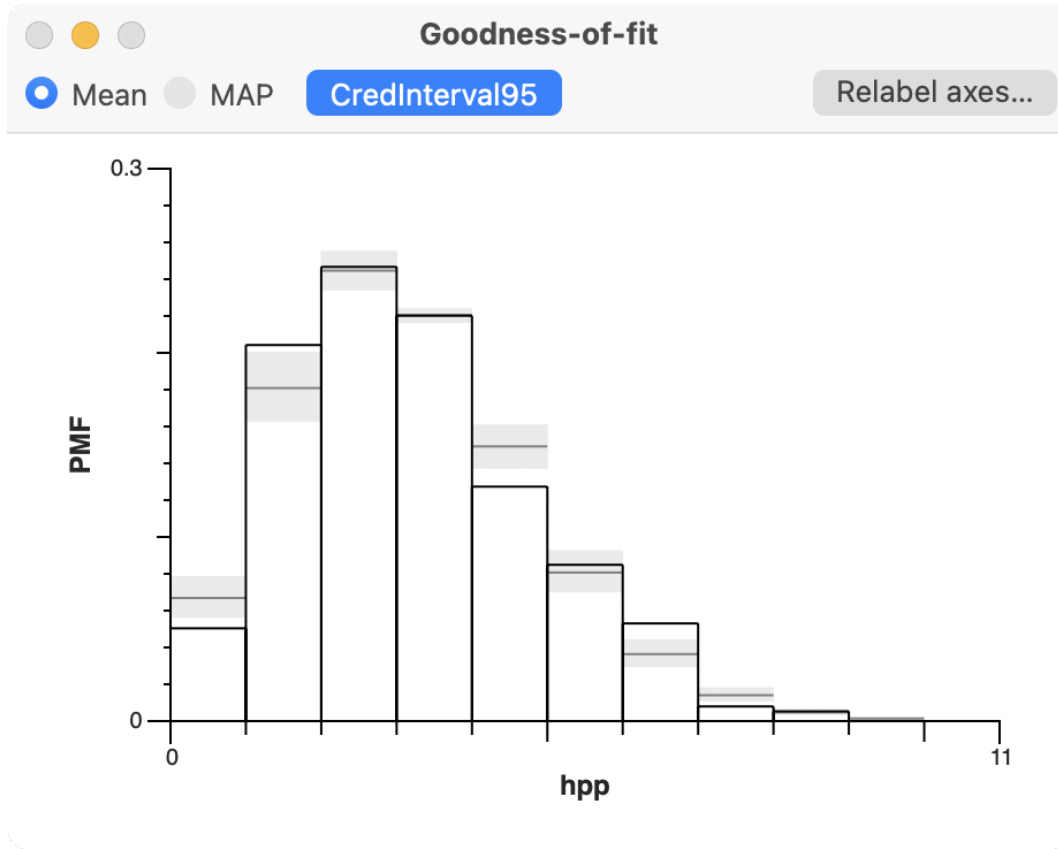


Figure 4.8: Goodness-of-fit for Hyphens Example

In this model, predictions are made for the number of terminal hyphens on a page. These predictions are declared here as a discrete vector, $pred_[]$, computed as an Extra in lines 21–23 and, of course, monitored. Note the monitoring syntax for $pred_ [0]$ which is here defined, contrary to MCMC convention, so it must be monitored separately.

With $\lambda = 2.7$, this model predicts $pred_ [0] = N \exp(-2.7) = 25.4$ or 25 (after rounding). The report gave a mean prediction of 25.3 with a 95-percent credible interval of 21–29.³ Figure 4.8 indicates that the observed zero count (20) was smaller than this prediction. Figure 4.9 shows the full marginals for $pred_ [0]$ and the data mode, $pred_ [2]$.

¹The data average is also 2.7.

²hpp_ shown as hpp in plots.

³Note that mean values are not discrete.

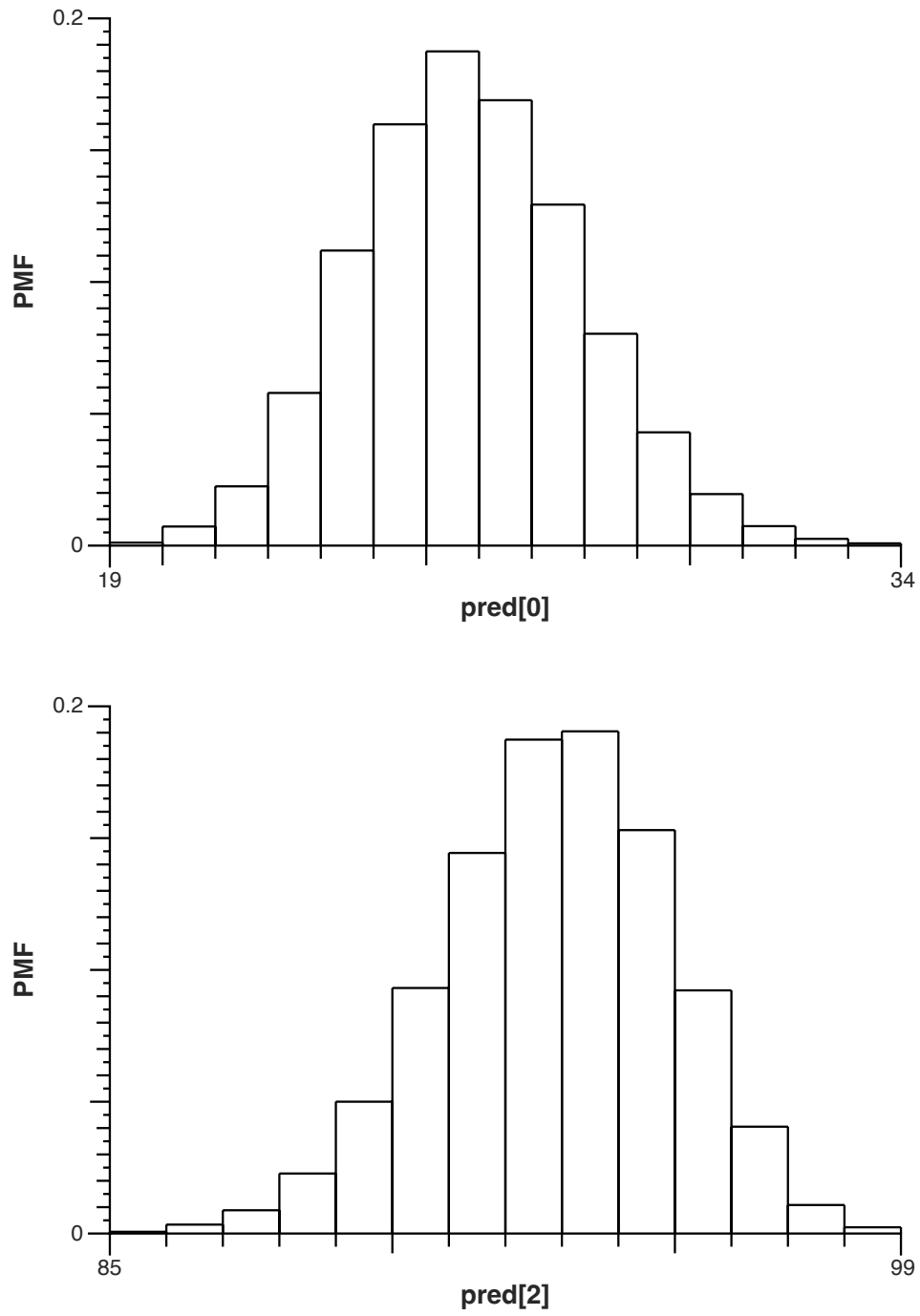


Figure 4.9: Marginals for Zero-count and Two-count

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

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

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

Exponential

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

where $\lambda =$ scale.

Gamma

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

where $\alpha =$ shape, $\beta =$ 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

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

UniformMH

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

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 describes a vector of two variables. See Example Body.

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

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

TrivariateNormal

This distribution is implemented for *likelihoods only*; it may *not* be used as a prior. It describes a vector of three variables.

$$\text{TrivariateNormal}(\mu_1, \mu_2, \mu_3, \sigma_1, \sigma_2, \sigma_3, \rho_{1,2}, \rho_{1,3}, \rho_{2,3}) \llbracket = (2\pi)^{-3/2} \det(\Sigma)^{-1/2} \exp\left(-1/2 (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right) \quad ; -\infty < \mathbf{x} < \infty$$

where \mathbf{x} , μ are vectors, $\rho_{i,j}$ the correlation coefficients and Σ the covariance matrix.

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 unknown. Typically, p is unknown.

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

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

Categorical

This distribution is implemented for *likelihoods only*; it may *not* be used as a prior. Validity of p_0 checked internally.

$$\text{Categorical}(p[]) = p_i \quad ; x = i = 0, 1, 2, \dots$$

DiscreteUniform

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

Geometric

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

Multinomial

This distribution is implemented for *likelihoods only*; it may *not* be used as a prior. Also, the value for the total, n , *must* be known and equal to $\sum x_i$. Validity of p_0 checked internally.

$$\text{Multinomial}(p[], n) = \frac{n!}{x_1!x_2! \dots x_k!} p_1^{x_1} p_2^{x_2} \dots p_k^{x_k} \quad ; x = 0, 1, 2, \dots$$

NegativeBinomial

For MCMC, p and n cannot both be unknown.

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

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 integral values starting with zero.

Poisson

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

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

A.3 Generic

This distribution is meant to handle distributions not otherwise implemented.

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

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.

DiscreteGeneric

This is the discrete analog of the Generic distribution.

A.4 Mixtures

Mixtures of any number of distributions are defined but implementation requires several limitations (see Sect. 3.2). In addition to mixtures in general, a special case, Inflated, is defined for discrete distributions.

Mixture

$$\text{Mixture}(\{\text{dist1}(\cdot), \text{dist2}(\cdot), \dots\}, \{\text{wt1}, \text{wt2}, \dots\})$$

where $\text{dist1}(\cdot)$, etc. is the syntax as described in Section A.1 and wt1 , etc. are the respective weights which *must* sum to one.

See Example Salaries in Section 3.2.

Invalid Components

None of the following may be a Mixture component:

- Mixture, Inflated
- Generic, DiscreteGeneric
- Uniform, UniformMH, DiscreteUniform, Triangular, Beta
- Jeffreys, ModifiedJeffreys

Inflated

An inflated distribution is a discrete distribution in which one bin, almost always for the value zero, contains excess probability mass modeled as Uniform within that bin.

$$\text{Inflated}(\{dist(\cdot)\}, \{b, wt\})$$

where $dist(\cdot)$ is a discrete distribution, b the bin to be inflated and wt the *excess* weight to be assigned to that bin. Thus, the weight for $dist$ will be $(1 - wt)$.

See Example Tornadoes.

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

These return 1 (true) or 0 (false) and can be used as numbers (See Example TriplePoint).

< 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

erf standard error function

phi standard Normal CDF

sum of 1-D vector, from 1 to length

mean of 1-D vector, from 1 to length

circularize(x, y) restrict x to [-y, y], e.g., for an angle

B.3 Reserved Constants

Pi 3.14159... (to double-precision)

Inf infinity

NA datum not available (in a multivariate 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 (v26.1.1). 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. There are 7,300 LOC.

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 must be done with care. It is safest to assume that there is no zeroth element. Arrays are defined for one or two dimensions only.

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.

Note: The reserved datum *NA* (\equiv not-available) is defined. What effect this will have on results will depend on the model so any such datum should probably be deleted.¹

¹or, at least, commented out. *NA* is stored internally as $\text{NaN} = 0/0$.

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. There must be at least two per parameter. To facilitate mixing, the number of walkers is limited to a maximum of 25.

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

Unknown variables are initialized by drawing random variates from their priors. All of the parameters of 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 all 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. Note that HPD intervals are, in general, *not unique*. Still, intervals from MCMC are typically reliable and reproducible (apart from simulation noise).

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 $\text{binwidth} = 1$ 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 there are indicator variables, MAP parameters are those found that have the largest $\log(\text{posterior})$. 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 $\log(\text{posterior})$ values are first offset by subtracting the MAP value of the $\log(\text{posterior})$. This is added back at the end and $\log(\text{marginal likelihood})$ reported to three decimal places which is more than sufficient for model-comparison purposes.

This computational scheme is not foolproof. With thousands of datapoints and many unknowns, some integrations might still exhibit underflow, reporting a value of minus infinity.

Computation of the marginal likelihood is enabled by default but can be disabled.

C.4 Mixture Relabeling

If the model likelihood is a homogeneous mixture, *MacMCMC* can try 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. Components that are well separated will give the best results. Needless to say, overlapping components are inherently ambiguous.

Relabeling failures are somewhat random. If relabeling results in a warning that it was incomplete, it is usually worth repeating this step or even 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

Examples supplied as part of this package are listed on the following page.

Table D.1: Examples

1	Body	Height (cm) vs. weight (kg) of adult men
2	Carbon-14	^{14}C specific activity (counts/min per gram) vs. age (years)
3	Daytime	Daytime in Boston, MA, USA (min) vs. day number
4	Enzyme	Concentration vs. rate (Michaelis-Menten model)
5	Tornadoes	Count of tornado fatalities, Oklahoma, USA (1950–2023)
6	Gumbel	US major-league baseball: best batting averages (1876-2019)
7	Hale-Bopp	Cyanide radical 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 salaries (in hundreds of dollars)
14	SAT	Northern Virginia, USA, schools (2014)
15	TriplePoint	Triple-point temperature 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		•	•	•			•								
Bivariate Normal Dist.	•								•						
Generic Distribution						•				•					•
<i>Truncated Distribution^a</i>															•
Mixture Distribution					•				•				•		
Relabeled Mixture									•				•		
Inflated Distribution					•										
Goodness-of-fit			•	•	•		•	•				•	•		•
Multivariate Data	•								•						
Extras			•	•			•	•						•	

^aimplemented as Generic

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*, 2 ed. 2024.
- [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.