



FlexBayes User Manual

November 2010

TIBCO Software Inc.
Seattle, Washington

IMPORTANT INFORMATION

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN *TIBCO SPOTFIRE S+ LICENSES*). USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO Software Inc., TIBCO, Spotfire, TIBCO Spotfire S+, Insightful, the Insightful logo, the tagline "the Knowledge to Act," Insightful Miner, S+, S-PLUS, TIBCO Spotfire Axum, S+ArrayAnalyzer, S+EnvironmentalStats, S+FinMetrics, S+NuOpt, S+SeqTrial, S+SpatialStats, S+Wavelets, S-PLUS Graphlets, Graphlet, Spotfire S+ FlexBayes, Spotfire S+ Resample, TIBCO Spotfire Miner, TIBCO Spotfire Statistics Services, TIBCO Spotfire S+ Server, and TIBCO Spotfire Clinical Graphics are either registered trademarks or trademarks of TIBCO Software Inc. and/or subsidiaries of TIBCO Software Inc. in the United States and/or other countries. All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. This

software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

Copyright © 1996-2010 TIBCO Software Inc. ALL RIGHTS RESERVED. THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

TIBCO Software Inc. Confidential Information

Reference

The correct bibliographic reference for this document is as follows:

FlexBayes User Manual, TIBCO Software Inc.

Technical Support

For technical support, please visit <http://spotfire.tibco.com/support> and register for a support account.

ACKNOWLEDGMENTS

TIBCO Spotfire S+ would not exist without the pioneering research of the Bell Labs S team at AT&T (now Lucent Technologies): John Chambers, Richard A. Becker (now at AT&T Laboratories), Allan R. Wilks (now at AT&T Laboratories), Duncan Temple Lang, and their colleagues in the statistics research departments at Lucent: William S. Cleveland, Trevor Hastie (now at Stanford University), Linda Clark, Anne Freeny, Eric Grosse, David James, José Pinheiro, Daryl Pregibon, and Ming Shyu.

The Spotfire FlexBayes for S+ library has been supported by NIH SBIR grants 5 R44 CA085108-01, 02, 03.

Contributors include: Xiaobin Dong, Dave Henderson, Kjell Konis, R. Douglas Martin, Alejandro Murua, James Schimert, Dawn Woodard, and Yihui Zhan.

We also appreciate advice from Sid Chib and Robert Kass.

TIBCO SPOTFIRE S+ BOOKS

Note about Naming

Throughout the documentation, we have attempted to distinguish between the language (S-PLUS) and the product (Spotfire S+).

- “S-PLUS” refers to the engine, the language, and its constituents (that is objects, functions, expressions, and so forth).
- “Spotfire S+” refers to all and any parts of the product beyond the language, including the product user interfaces, libraries, and documentation, as well as general product and language behavior.

The TIBCO Spotfire S+[®] documentation includes books to address your focus and knowledge level. Review the following table to help you choose the Spotfire S+ book that meets your needs. These books are available in PDF format in the following locations:

- In your Spotfire S+ installation directory (**\$HOME\help** on Windows, **\$HOME/doc** on UNIX/Linux).
- In the Spotfire S+ Workbench, from the **Help ► Spotfire S+ Manuals** menu item.
- In Microsoft[®] Windows[®], in the Spotfire S+ GUI, from the **Help ► Online Manuals** menu item.

Spotfire S+ documentation.

| Information you need if you... | See the... |
|---|---|
| Must install or configure your current installation of Spotfire S+; review system requirements. | <i>Installation and fs Administration Guide</i> |
| Want to review the third-party products included in Spotfire S+, along with their legal notices and licenses. | <i>Licenses</i> |

Spotfire S+ documentation. (Continued)

| Information you need if you... | See the... |
|---|---|
| Are new to the S language and the Spotfire S+ GUI, and you want an introduction to importing data, producing simple graphs, applying statistical models, and viewing data in Microsoft Excel [®] . | <i>Getting Started Guide</i> |
| Are a new Spotfire S+ user and need how to use Spotfire S+, primarily through the GUI. | <i>User's Guide</i> |
| Are familiar with the S language and Spotfire S+, and you want to use the Spotfire S+ plug-in, or customization, of the Eclipse Integrated Development Environment (IDE). | <i>Spotfire S+ Workbench User's Guide</i> |
| Have used the S language and Spotfire S+, and you want to know how to write, debug, and program functions from the Commands window. | <i>Programmer's Guide</i> |
| Are familiar with the S language and Spotfire S+, and you want to extend its functionality in your own application or within Spotfire S+. | <i>Application Developer's Guide</i> |
| Are familiar with the S language and Spotfire S+, and you are looking for information about creating or editing graphics, either from a Commands window or the Windows GUI, or using Spotfire S+ supported graphics devices. | <i>Guide to Graphics</i> |
| Are familiar with the S language and Spotfire S+, and you want to use the Big Data library to import and manipulate very large data sets. | <i>Big Data User's Guide</i> |
| Want to download or create Spotfire S+ packages for submission to the Comprehensive S-PLUS Archive Network (CSAN) site, and need to know the steps. | <i>Guide to Packages</i> |

Spotfire S+ documentation. (Continued)

| Information you need if you... | See the... |
|--|------------------------------------|
| Are looking for categorized information about individual S-PLUS functions. | <i>Function Guide</i> |
| If you are familiar with the S language and Spotfire S+, and you need a reference for the range of statistical modelling and analysis techniques in Spotfire S+. Volume 1 includes information on specifying models in Spotfire S+, on probability, on estimation and inference, on regression and smoothing, and on analysis of variance. | <i>Guide to Statistics, Vol. 1</i> |
| If you are familiar with the S language and Spotfire S+, and you need a reference for the range of statistical modelling and analysis techniques in Spotfire S+. Volume 2 includes information on multivariate techniques, time series analysis, survival analysis, resampling techniques, and mathematical computing in Spotfire S+. | <i>Guide to Statistics, Vol. 2</i> |

CONTENTS

| | | |
|------------------|---|-----------|
| Chapter 1 | Introduction | 1 |
| | Overview | 2 |
| | FlexBayes Features | 4 |
| | Introductory Example using FlexBayes | 7 |
| | Using FlexBayes | 14 |
| | Featured Examples | 16 |
| | Validation | 18 |
| | Using this Manual | 19 |
| Chapter 2 | Background | 21 |
| | Overview | 22 |
| | Comparison with Frequentist Theory and Practice | 23 |
| | Summarizing Posterior Inference | 25 |
| | Computation: Markov Chain Monte Carlo | 26 |
| | Hierarchical Models | 28 |
| Chapter 3 | Assessing Convergence of Monte Carlo Markov Chain Algorithms | 29 |
| | Overview | 30 |
| | MCMC Control Parameters | 31 |
| | Practical Considerations | 32 |

| | |
|--|-----------|
| Chapter 4 The Hierarchical Linear Mixed Effects Model | 35 |
| Model Description | 36 |
| Example: Orthodont Data | 42 |
| Chapter 5 The Hierarchical Poisson Mixed Model | 51 |
| Model Description | 52 |
| Example 1: Pump Data | 57 |
| Example 2: Epilepsy Data | 66 |
| Algorithms for the Hierarchical Poisson Model | 76 |
| Chapter 6 Hierarchical Binomial Mixed Model | 81 |
| Model Description | 82 |
| Example: Toxoplasmosis Data | 87 |
| Algorithms for the Hierarchical Binomial Model | 94 |
| Index | 95 |

INTRODUCTION

1

| | |
|--|-----------|
| Overview | 2 |
| FlexBayes Features | 4 |
| Workflow | 4 |
| Prior Specification | 5 |
| Introductory Example using FlexBayes | 7 |
| Prior Specification | 7 |
| MCMC Control Specification | 8 |
| Fit Model | 8 |
| Diagnostics | 8 |
| Print / Summary | 8 |
| Plot | 9 |
| Comparison with Least Squares Estimates | 11 |
| Using FlexBayes | 14 |
| Installing, Starting, and Quitting FlexBayes | 14 |
| Installing Dependencies | 14 |
| Getting Help | 15 |
| Featured Examples | 16 |
| Clinical Examples | 16 |
| Validation | 18 |
| Using this Manual | 19 |
| Intended Audience | 19 |
| Organization | 19 |
| Typographic Conventions | 20 |

OVERVIEW

FlexBayes is a software library for modeling data using the *Bayesian* paradigm for statistical inference (see for example Gelman *et. al.* 2004).

Bayesian modeling offers the following advantages over classical or frequentist modeling:

- Uses more realistic models with many parameters.
- Provides a natural way to integrate out “nuisance” parameters, or missing data.
- Quantifies uncertainty about model parameters in terms of a posterior probability distribution, which combines information from (1) prior understanding and (2) observed data.
- Provides a computational advantage of using priors to stabilize computation in ill-defined problems (for example, too many variables relative to the number of cases). Among other advantages, it helps with collinearity.

Until recently, Bayesian inference has been hampered by computational problems. These problems have been overcome partly by the wide availability of fast and inexpensive computer resources, and by the use of Monte Carlo Markov Chain (MCMC) methods to accomplish inference by simulation.

FlexBayes implements MCMC algorithms for a variety of models as displayed in Table 1.1. The interface for FlexBayes is the Spotfire S+ command line.

Table 1.1: *Models in the FlexBayes library*

| FlexBayes Models | Function Name |
|---------------------------------|---------------|
| Hierarchical linear mixed model | bh1m |

Table 1.1: *Models in the FlexBayes library*

| FlexBayes Models | Function Name |
|--|----------------------|
| Hierarchical Poisson mixed model | bhpm |
| Hierarchical binomial (logistic) mixed model | bhbm |
| Flexible model-fitting | posteriorSamples |

The function `bhlm` fits a hierarchical linear mixed effects model, while the functions `bhpm` and `bhbm` fit hierarchical Poisson and binomial mixed effects models with optional overdispersion. In this manual we focus on documenting the functions `bhlm`, `bhpm`, and `bhbm`.

The FlexBayes package also allows for more flexible Bayesian model-fitting using the `posteriorSamples` function, which provides an interface to the BUGS (Bayes Using Gibbs Sampling) open-source engine. Many examples and ample documentation for the `posteriorSamples` function are provided in the FlexBayes Help files. (To access this documentation, call `help(posteriorSamples)` after loading FlexBayes.) In addition, help for the BUGS model specification language and the BUGS engine are available in the OpenBUGS manual (<http://mathstat.helsinki.fi/openbugs/>).

FLEXBAYES FEATURES

Workflow

Table 1.2 organizes functions in FlexBayes according to activities in the workflow. Also see Table 1.1 to find the models which correspond to the model-fitting functions.

Table 1.2: *Analysis activities and corresponding function names.*

| Activity | Functions |
|---------------------------------|--|
| Specify priors | bhlm.prior, bhpm.prior, bbbm.prior |
| Specify MCMC control parameters | bhlm.sampler, bhpm.sampler, bbbm.sampler |
| Fit models | bhlm, bhpm, bbbm, posteriorSamples |
| Diagnose convergence of MCMC | traceplot, autocorr, autocorr.plot, crosscorr, crosscorr.plot, lagged.crosscorr, effectiveSize, gelman.diag, gelman.plot, geweke.diag, geweke.plot, heidel.diag, raftery.diag, cumuplot |
| View parameter inferences | summary, densplot |
| Manipulate posterior objects | varnames.posterior, nvar.posterior, niter.posterior, nchain.posterior, thin.posterior, start.posterior, end.posterior, [.posterior, [[.posterior, window.posterior, as.mcmc.list.posterior, getSamples |
| Validate model output | validate.package(package = "FlexBayes") cgrFlexBayes |

Prior Specification

All generalized linear models in FlexBayes involve specifying priors for (1) the regression coefficients, and / or (2) variance or covariance parameters. Specify each of these priors by calling a function that creates a `bayes.distribution` object, which consists of the following two components:

- `name` is the name of the prior distribution.
- `parameters` gives values for the distribution parameters.

For example, to fit a linear model:

$$Y = \gamma_0 + \gamma_1 X + \varepsilon$$

Create an object specifying a two-dimensional Gaussian prior to be used for $\gamma = (\gamma_0, \gamma_1)$, with zero mean and diagonal covariance matrix, by calling

```
> bayes.normal(mean.vector= c(0,0), covmat= diag(2))
normal with:

mean vector :
[1] 0 0

covariance matrix :
      [,1] [,2]
[1,] 1.00 0.00
[2,] 0.00 1.00

k0 :
[1] 1
```

If you assume that the errors ε are distributed normally with a common variance, you can specify an inverse chisquare prior for the variance, with 3 degrees of freedom and unit scale, by calling:

```
> bayes.invChisq(df= 3, sigma0.sq= 1)
invChisq with:

df :
[1] 3

sigma0.sq :
```

[1] 1

Table 1.3 summarizes the available prior distribution functions implemented in FlexBayes. For each particular model, see the corresponding chapter for details on acceptable priors for that model.

Table 1.3: *Functions to specify priors.*

| Regression Coefficients | Description |
|-------------------------|------------------------|
| bayes.normal | normal |
| bayes.t | student t |
| bayes.nonInformative | “non-informative” flat |

| Variance/Covariance Parameters | Description |
|--------------------------------|----------------------------|
| bayes.invChisq | inverse chisquare |
| bayes.invWishart | inverse Wishart |
| bayes.massPoint | Dirac delta |
| bayes.nonInfoPower | “noninformative” power law |
| bayes.duMouchel | DuMouchel |
| bayes.uniformShrinkage | uniform shrinkage |

INTRODUCTORY EXAMPLE USING FLEXBAYES

The data frame `stack.dat` combines the `stack.loss` and `stack.x` data sets included with Spotfire S+. These data are from the operation of a plant for the oxidation of ammonia to nitric acid, measured on 21 consecutive days.

The goal is to model `Loss` (percent of ammonia lost times 10) as a linear function of `Air.Flow` (air flow to the plant), `Water.Temp` (cooling water inlet temperature), and `Acid.Conc` (acid concentration as a percentage).

Prior Specification

On the command line, proceed as follows.

The first step is to specify the priors for the linear model coefficients and the variance.

Specify a four-dimensional Gaussian prior for the fixed effects

$\gamma = (\gamma_0, \gamma_1, \gamma_2, \gamma_3)$, with zero mean and diagonal covariance (zero covariances and variance = 1),

```
> coefPrior = bayes.normal(mean.vector= zero, covmat=
  identity)
```

`zero` and `identity` are function names that produce a zero vector and identity covariance of the correct dimension. This call is equivalent to the following:

```
> coefPrior = bayes.normal(mean.vector= rep(0,4), covmat=
  diag(4))
```

Specify an inverse chisquare prior for the variance of the outcome, with 3 degrees of freedom and unit scale, by:

```
> varPrior = bayes.invChisq(df= 3, sigma0.sq= 1)
```

Combine the separate priors into a linear model prior:

```
> modelPrior = bhlm.prior( fixed.coef= coefPrior,
  error.var = varPrior )
```

MCMC Control Specification

The function `bhlm.sampler` specifies control parameters for the Gibbs sampler. In particular, to run three chains, generating 1000 samples from the posterior distribution (beyond the *burn-in* samples), call

```
> stackSampler <- bhlm.sampler( nChains = 3,
                                nSamples=1000,
                                init.point = "prior" )
```

Here multiple chains are run, each starting at randomly chosen initial values, to allow Gelman-Rubin diagnosis of convergence of the Gibbs sampler.

Fit Model

Produce samples from the posterior distribution by calling

```
> stack.bhlm <- bhlm( fixed.formula = Loss ~ .,
                      data = stack.dat,
                      prior = modelPrior,
                      sampler = stackSampler)
```

Diagnostics

As discussed in Chapter 3, Assessing Convergence of Monte Carlo Markov Chain Algorithms, FlexBayes allows the user to run several MCMC convergence diagnostics.

To perform diagnostics on the parameter values, the diagnostics functions operate directly on the fit posterior object. Run these functions as follows. (They call functions in the open-source **coda** package for Spotfire S+, and so require the installation of the **coda** package. It can be obtained as described in the section Installing Dependencies on page 14.)

```
> traceplot(stack.bhlm)
> autocorr.plot(stack.bhlm)
```

For information on the interpretation of these diagnostics, see the section Practical Considerations on page 32. Here the diagnostics do not give evidence of substantial lack of convergence.

Print / Summary

To print the fitted model, just type the model name, `stack.bhlm`, which is equivalent to `print(stack.bhlm)` and `summary(stack.bhlm)`. The result includes output for each of the chains: the means, standard deviations, and percentiles 2.5%, 25%, 50%, 75% and 97.5% of the marginal posterior distributions for each coefficient:

```
> stack.bhlm
*** Posterior Distribution from the Bayesian Model ***
Call:
bhlm(fixed.formula = Loss ~ ., data = stack.dat, prior =
modelPrior, sampler = stackSampler)

# of Chains: 3
Starting Iteration: 1001
Ending Iteration: 2000
Thinning: 1
# of Samples: 1000

1. Summary statistics:

              Mean      S.D.
(Intercept) -0.2084 0.98790
  Air.Flow   0.8179 0.15250
 Water.Temp  0.9738 0.39400
 Acid.Conc. -0.6036 0.08265
      SIGMA  3.9120 0.63490

2. Quantiles:

              2.5 %    25 %    50 %    75 %    97.5 %
(Intercept) -2.1370 -0.8533 -0.2254  0.4620  1.6980
  Air.Flow   0.5192  0.7179  0.8163  0.9176  1.1190
 Water.Temp  0.1787  0.7255  0.9719  1.2370  1.7490
 Acid.Conc. -0.7619 -0.6576 -0.6048 -0.5506 -0.4401
      SIGMA  2.9080  3.4790  3.8390  4.2790  5.3910
```

Here three chains were run with no thinning, and 1000 samples were obtained from each chain.

Plot

Use the `densplot` function to display estimated marginal posterior distributions of the parameters in the model.

```
> densplot(stack.bhlm)
```

This produces posterior density plots, shown in Figure 1.1, that are estimated from the sampled values of the parameters.

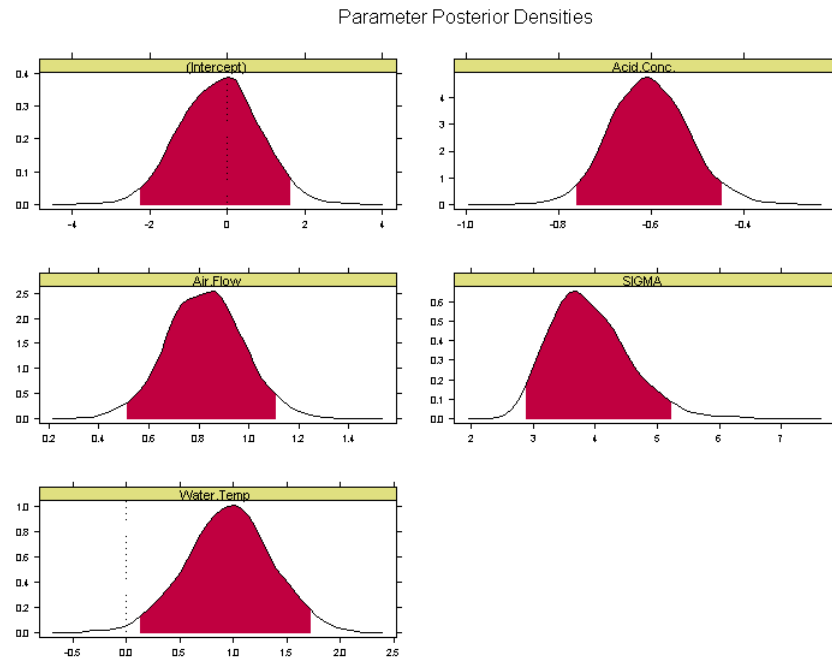


Figure 1.1: *Estimated posterior densities for the stack example.*

It is also possible to view bivariate density estimates for the parameters, shown in Figure 1.2, by calling `bivarDensplot(stack.bh1m)`. The only two parameters that show substantial correlation are the coefficients for `Air.Flow` and `Water.Temp.`

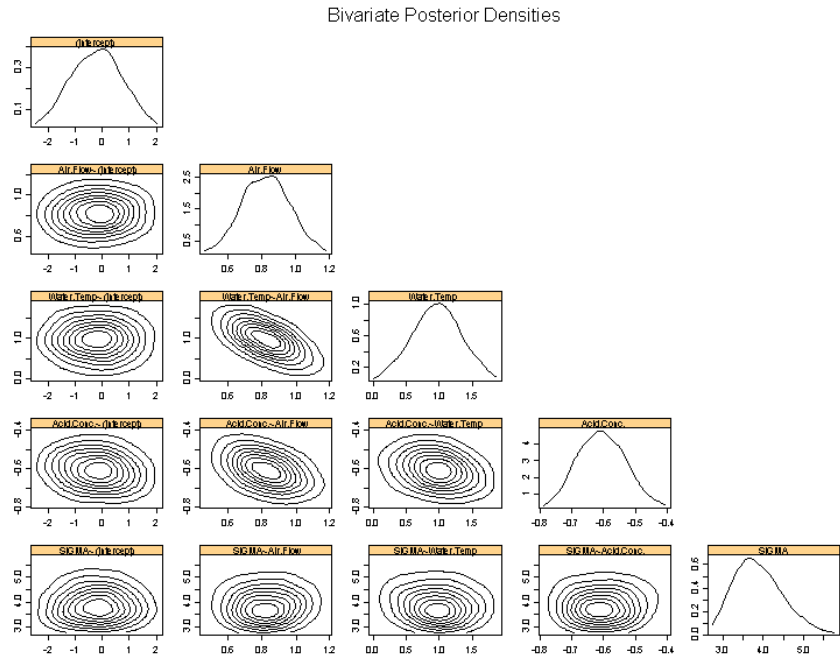


Figure 1.2: *Bivariate density plots for the stack example.*

Comparison with Least Squares Estimates

You can compare your results easily with those obtained by least squares linear regression as follows. The data frame `stack.dat` is built into FlexBayes, so you can apply least squares regression through the Spotfire S+ GUI. Make sure you have set your Database Filter to view the FlexBayes library objects. (See the section Installing, Starting, and Quitting FlexBayes on page 14.)

1. Select the response and predictor variables from the right-hand **Object Explorer** pane, and then choose **Statistics -> Regression -> Linear Regression** from the **Spotfire S+** toolbar.
2. On the **Printed Results** region of the **Results** page of the dialog that appears, clear the **Long Output** box and select the **Short Output** box.

3. Click **OK** to see the **Report Window** results shown in Figure 1.1.

Notice that the least squares coefficient values in Figure 1.1 are similar to the means of the Bayes posterior distribution, except for the intercept and `Acid.Conc.` This is because the prior variance for the intercept is small in the Bayesian model. This, together with the corresponding prior mean of 0, shrinks the intercept estimate toward 0.

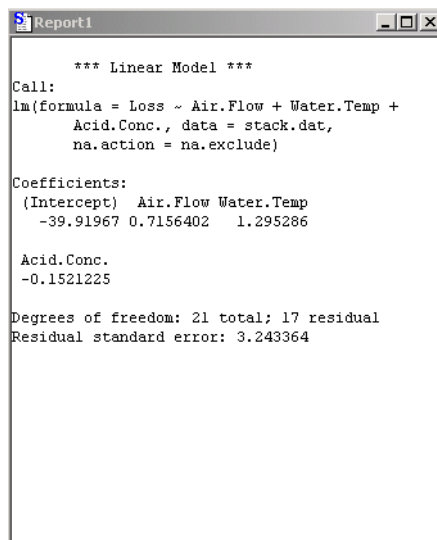


Figure 1.3: *Coefficient Values for Least Squares Fit of Linear Model to `stack.dat`.*

Instead, if we run the Bayesian analysis with “non-informative” priors, then the least squares and Bayesian analyses agree very closely:

```
> modelPrior = bhlm.prior( fixed.coef= "non-informative",
                           error.var= bayes.nonInfoPower(-1) )
> stack.bhlm <- bhlm( fixed.formula = Loss ~ .,
                      data = stack.dat,
                      prior = modelPrior,
                      sampler = stackSampler )

> stack.bhlm
*** Posterior Distribution from the Bayesian Model ***
Call:
```

```
bhlm(fixed.formula = Loss ~ ., data = stack.dat, prior =  
modelPrior, sampler = stackSampler)
```

```
# of Chains: 3  
Starting Iteration: 1001  
Ending Iteration: 2000  
Thinning: 1  
# of Samples: 1000
```

1. Summary statistics:

| | Mean | S.D. |
|-------------|----------|---------|
| (Intercept) | -40.1500 | 13.2100 |
| Air.Flow | 0.7138 | 0.1468 |
| Water.Temp | 1.3050 | 0.4025 |
| Acid.Conc. | -0.1505 | 0.1708 |
| SIGMA | 3.4180 | 0.6197 |

2. Quantiles:

| | 2.5 % | 25 % | 50 % | 75 % | 97.5 % |
|-------------|----------|----------|----------|-----------|----------|
| (Intercept) | -64.4600 | -48.2500 | -39.9800 | -31.75000 | -14.8800 |
| Air.Flow | 0.4336 | 0.6176 | 0.7155 | 0.80750 | 0.9993 |
| Water.Temp | 0.5611 | 1.0640 | 1.3050 | 1.55800 | 2.0660 |
| Acid.Conc. | -0.4814 | -0.2631 | -0.1537 | -0.04911 | 0.1662 |
| SIGMA | 2.4290 | 2.9720 | 3.3210 | 3.71300 | 4.8300 |

USING FLEXBAYES

FlexBayes is currently available only for Windows. If you have used Spotfire S+ before, getting started with FlexBayes is easy. If you have not used Spotfire S+ before, consult the *Spotfire S+ User's Manual* to learn more about Spotfire S+ before proceeding.

Installing, Starting, and Quitting FlexBayes

FlexBayes requires the installation of Spotfire S+ version 8.0 or later. To install FlexBayes, just copy the Windows binary version of the package into one of the user's Spotfire S+ library directories. Then:

1. Start Spotfire S+. (See your *Spotfire S+ User's Guide* for more detailed instructions on starting Spotfire S+.)
2. Load the FlexBayes library in your Spotfire S+ session:

```
> library(FlexBayes)
```

You can then view the FlexBayes data sets by clicking +, located to the left of FlexBayes in the Spotfire S+ object explorer.

Installing Dependencies

Certain model-fitting and convergence diagnosis functions in FlexBayes require the installation of the open-source Spotfire S+ packages BRugs, R2WinBUGS, and/or coda and the open-source standalone software WinBUGS. If you call a function that requires one of these dependencies, and the dependency is not installed, you will see an error message informing you of the required dependencies.

The Spotfire S+ packages BRugs, R2WinBUGS, and coda can be found on TIBCO's CSAN website:

<http://spotfire.tibco.com/csan>

Copy the Windows binary versions of the packages into one of the Spotfire S+ library directories.

The open-source software WinBUGS can be found at

<http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml>

Version 1.4.2 or later of WinBUGS is required to use the `posteriorSamples` function with the `engine = "WinBUGS"` option. Free registration of the WinBUGS software is necessary; instructions are available on the WinBUGS website.

Getting Help

Initial help for FlexBayes is available by calling

```
> help(FlexBayes)
```

FlexBayes provides additional help files for virtually all FlexBayes functions. (Some functions intended for internal use have no help files.) For example, from the command line, you can obtain help and examples for the function `bh1m` by typing

```
> help(bh1m)
```

or

```
> ?bh1m
```

In particular, detailed documentation and examples are available in the help files for the main FlexBayes model-fitting functions, namely `bh1m`, `bhpm`, `bhbm`, and `posteriorSamples`.

FEATURED EXAMPLES

Additional examples showcasing the use of the FlexBayes functions are provided in the help files for a number of data sets, including `epilepsy`, `crossoverTrial`, `drugAdverseEvents`, `effToxDose`, `schools`, `seeds`, `ratsWeight`, `pumps`, `salamanders`, and `stackBayes`. The first four in this list are clinical examples. The others are well-known data sets from a variety of areas for which standard analyses have been published. The latter are useful for comparative purposes and illustrate the use of the FlexBayes generalized linear model functions `bhlm`, `bhpm`, `bhbm`.

Clinical Examples

An analysis of the efficacy of a drug in reducing epileptic seizure counts can be found in the help file for the `epilepsy` data set:

```
> help(epilepsy)
```

This analysis fits a Bayesian hierarchical adaptation of a Poisson-outcome model that was fit to the same data set by Breslow and Clayton (1993). It does so using the FlexBayes `bhpm` function.

The `crossoverTrial` help file gives an analysis of a two-period crossover trial for comparing two treatments, as described in Gelfand et al. (2004). The code in the help file fits a Bayesian hierarchical linear model by calling the FlexBayes `posteriorSamples` function, which allows the flexible specification needed for this model.

A late-phase drug trial safety analysis is provided in the help file for the `drugAdverseEvents` data set. The number of occurrences of each of several hundred adverse events in the treatment and control groups is modeled and analyzed in order to detect potentially elevated rates of some adverse events. A Bayesian hierarchical model is used that borrows strength across adverse events, addressing the issues of multiplicity and rare events. This model is fit by calling the FlexBayes `posteriorSamples` function.

The `effToxDose` help file shows a combined Phase I-Phase II dose-finding study. Data on the effect of a drug on mortality rates for acute eschemic stroke is gathered adaptively in cohorts. After each cohort, estimated efficacy and toxicity at each dose level are combined in order to choose the estimated best dose, and that dose is assigned to the next cohort. The “best” dose is one that does not have

strong evidence of toxicity and that has the highest probability of efficacy. The Bayesian models and dose choice criterion are as described by Thall and Russell (1998) and Thall and Cook (2004). One of the available models is fit by calling the FlexBayes `bhbm` function; an alternative model is fit using the FlexBayes `posteriorSamples` function.

VALIDATION

FlexBayes includes a set of scripts for validating the output of the model-fitting functions. Each of these scripts compares the posterior distribution obtained by a call to one of the FlexBayes proprietary model-fitting functions (bh1m, bhpm, bhbm) with the posterior distribution obtained from the open-source software BUGS via a call to the `posteriorSamples` function. Various models are tested, and this set of validation scripts will continue to expand with each release of FlexBayes.

For the beta release of FlexBayes, the validation scripts can be run by calling:

```
> validate( list.files( system.file("validate",  
package="FlexBayes") ), system.file("validate",  
package="FlexBayes") )
```

This call does take a while to run. One can also run each of the validation scripts separately by changing the first argument in the call to `validate` to the name of the test script; these separate calls are quite fast. If you have Spotfire S+ Version 8.1 or later, it is possible to run all the validation scripts using a simpler command:

```
> validate.package( "FlexBayes" )
```

USING THIS MANUAL

This manual describes how to use the FlexBayes library, and includes detailed descriptions of the principal FlexBayes functions and objects.

Intended Audience

Like the FlexBayes library, this book is intended for statisticians, researchers, and other analysts involved in analyzing data. This book is not meant to be a text book in Bayesian methods; we refer you to the cited works for recommended reading in this area.

For users familiar with Spotfire S+, this manual contains all the information most users need to begin making productive use of FlexBayes. Users who are *not* familiar with Spotfire S+ should read their *Spotfire S+ User's Manual*, which provides complete procedures for basic Spotfire S+ operations, including graphics manipulation, customization, and data input and output.

Other useful information can be found in the *Spotfire S+ Guide to Statistics*. This manual describes how to analyze data using a variety of statistical and mathematical techniques, including classical statistical inference, time series analysis, linear regression, ANOVA models, generalized linear and generalized additive models, loess models, and nonlinear regression.

Organization

The main body of this book is divided into 6 chapters which take you step-by-step through the FlexBayes library.

- Chapter 1 (this chapter) introduces you to FlexBayes, lists its features, and tells you how to use this manual.
- Chapter 2 briefly gives background information, which may be skimmed at first, and revisited as needed.
- Chapter 3 discusses convergence of a Markov Chain Monte Carlo algorithm and convergence diagnostics.
- Chapter 4 discusses the hierarchical linear model.
- Chapter 5 discusses the hierarchical Poisson model.
- Chapter 6 discusses the hierarchical binomial model.

Typographic Conventions

This book uses the following typographic conventions:

- The *italic font* is used for emphasis, and also for user-supplied variables within UNIX, DOS, and Spotfire S+ commands.
- The **bold font** is used for UNIX and DOS commands and filenames, as well as for chapter and section headings. For example,

setenv S_PRINT_ORIENTATION portrait

In this font, both “ and ” represent the double-quote key on your keyboard (").

- The typewriter font is used for Spotfire S+ functions and examples of Spotfire S+ sessions. For example,

```
> ?bhlm
```

Displayed Spotfire S+ commands are shown with the Spotfire S+ prompt >.

| | |
|--|-----------|
| Overview | 22 |
| Comparison with Frequentist Theory and Practice | 23 |
| Prior, Likelihood, Posterior | 23 |
| Predictive Distribution | 24 |
| Complex Models | 24 |
| Summarizing Posterior Inference | 25 |
| Computation: Markov Chain Monte Carlo | 26 |
| Gibbs Sampling | 26 |
| Metropolis Algorithm | 26 |
| Hierarchical Models | 28 |

OVERVIEW

This chapter discusses background information regarding Bayesian methods (Gelman *et. al.*, 2004). You might want to skim this chapter, and then return to it as needed as you read the rest of the manual.

First, to put Bayesian methodology into context, we briefly compare with the *frequentist* school of inference, which has dominated statistical theory and practice until recently.

Bayesian inference is performed using parameter posterior summaries such as quantiles, moments, and modes. Posterior inference is discussed in the section Summarizing Posterior Inference on page 25.

Second, we discuss Markov Chain Monte Carlo (MCMC), which is a computational technique that overcomes computational barriers in the Bayesian approach. MCMC has increased interest in the Bayesian methods as much as any inherent philosophical or logical advantage of Bayesian thinking (Gelman *et. al.*, 2004). However, assessing convergence of MCMC is an important practical problem. To address this problem, we discuss simple diagnostic procedures in Chapter 3, Assessing Convergence of Monte Carlo Markov Chain Algorithms.

Third, we discuss hierarchical regression models, which are fitted by the main functions in FlexBayes and which are essential for any data in which the covariates are defined at different levels.

COMPARISON WITH FREQUENTIST THEORY AND PRACTICE

Bayesian methods explicitly use probability to quantify uncertainty. After fitting a probability model to data, Bayesian inference summarizes the result by a probability distribution on unobserved quantities such as the parameters of the model, or predictions for new observations. These probability statements condition on the observed value of the data.

This leads to a common sense interpretation of statistical conclusions. For example, you might interpret a Bayesian interval for a parameter as having high probability of containing the parameter. In contrast, the frequentist interpretation of confidence intervals is based on hypothetical repetitions of similar inferences. In general, the frequentist approach evaluates the procedure over the distribution of possible data values, conditional on the true unknown value of the unobserved quantity.

In many simple analyses, the two approaches lead to similar conclusions. However, Bayesian methods extend easily to more complex problems.

Prior, Likelihood, Posterior

The Bayesian approach to inference specifies two distributions:

- a prior distribution $P(\theta)$ for the parameters that reflects knowledge about θ before seeing the data, and
- a distribution $P(y|\theta)$ for the data given parameters.

The posterior distribution combines these distributions, and reflects knowledge about θ updated after seeing the data:

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{\int P(y|\theta)P(\theta)d\theta}$$

If data are added sequentially to the data set, Bayesian inference is natural because one can perform a sequential analysis in which the new prior distribution is the posterior distribution from the previous analysis.

Predictive Distribution

A future observation $\tilde{\mathbf{y}}^*$ may be predicted using the predictive distribution $P(\mathbf{y}^*|\mathbf{y})$, which is calculated from the posterior distribution as follows:

$$P(\mathbf{y}^*|\mathbf{y}) = \int P(\mathbf{y}^*|\boldsymbol{\theta})P(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

Importantly, this approach accounts for the uncertainty in estimating $\boldsymbol{\theta}$ during estimation of \mathbf{y}^* . By contrast, the maximum likelihood approach performs predictive inference using $P(\mathbf{y}^*|\hat{\boldsymbol{\theta}})$, the data density evaluated at the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$.

Complex Models

Directly quantifying uncertainty encourages fitting models with many parameters and multilayer probability specifications, such as hierarchical models. The Bayesian paradigm gives freedom to establish complex models by supplying a conceptually simple method for coping with multiple parameters.

Although a realistic model might require many parameters, interest usually focuses on a smaller number of parameters. The other parameters are *nuisance* parameters. The state of knowledge about the parameters of interest can be obtained through their *marginal* posterior distribution, obtained by integrating over the nuisance parameters in the *joint* posterior distribution of all parameters. Equivalently, using simulation, you sample from the joint posterior distribution, but only look at the parameters of interest.

SUMMARIZING POSTERIOR INFERENCE

Bayesian inference often consists of summaries of the posterior distribution. For example, one might estimate the “plausible” range of a parameter using a 99% posterior interval. Alternatively, one might be interested in the plausible range for a particular function of the parameters, or for a random variable that depends on the parameters. Posterior and predictive intervals can be used as estimates of the plausible ranges of these quantities. Posterior density plots or contour plots of posterior density functions are also useful summaries of the highest-probability values of the quantities of interest and the relationship between the quantities of interest.

Estimated posterior or predictive intervals and density plots can be easily obtained using a set of samples from the posterior distribution. One can obtain posterior samples using MCMC methods, thereby making sophisticated inferences possible even in complex models. For more information about posterior inference using a sample, see Simon and Bruce (1991), Smith and Gelfand (1992), or Albert (1993).

COMPUTATION: MARKOV CHAIN MONTE CARLO

Monte Carlo Markov Chain (MCMC) methods are a type of Monte Carlo method, which estimates features of an unknown distribution $\pi(x)$ by either sampling from that distribution or suitably reweighting samples drawn from some other appropriately chosen distribution. MCMC methods obtain samples by constructing a Markov chain with equilibrium distribution equal to $\pi(x)$. If the chain is run for a long time, simulated values from the chain can be used to summarize features of $\pi(x)$, often through familiar exploratory data analysis tools like the histogram or box plot.

A number of algorithms are used for constructing Markov chains with specified equilibrium distributions. These algorithms include Metropolis-Hastings (Metropolis et al. (1953), Hastings (1970)), the Gibbs sampler (Geman and Geman (1984), Ripley (1977), Ripley (1979), Gelfand and Smith (1990), Zeger and Karim (1991)), data augmentation (Tanner and Wong (1987)), and sequential imputation (Kong and Wong (1991)). For more information on the available methods and their properties, see Gelman and Rubin (1992), Geyer (1992), Smith and Roberts (1992), and Tierney (1991).

Gibbs Sampling The Gibbs sampler is useful when it is difficult to sample from the joint posterior distribution, but simple to simulate from the *full conditional distributions* of each parameter given the data and the other parameters: $P(\theta_j | y, \{\theta_i, i \neq j\})$ for $j = 1, \dots, K$. Gibbs sampling is useful because it reduces the problem to a simpler sequence of problems, each of which deals with one parameter at a time. Gibbs sampling alternately simulates from each of these distributions. After sufficient iterations, the procedure stabilizes, converging to the stationary distribution (the desired posterior distribution).

Metropolis Algorithm

The Metropolis-Hastings (MH) algorithm is a stochastic procedure which generates steps in parameter space from a *proposal* distribution. The algorithm generates posterior samples by generating from the proposal distribution and then accepting or rejecting the step with a particular probability. The probability of acceptance depends on both

the proposal distribution and the underlying distribution, and is given in Metropolis *et. al.* (1953) and Hastings (1970). A high rejection rate slows the exploration of the space by the sampler.

MH is used as a component of many Markov chain Monte Carlo algorithms used for statistical inference (Gilks *et. al.* 1996, Liu 2001). MH is used by both frequentists and Bayesians to perform inference by calculating expectations for high dimensional distributions.

Suppose X is a vector of K random variables with distribution π . In Bayesian applications, X might consist of model parameters and missing data, and π is the posterior distribution. In frequentist applications, X might consist of data or random effects, and π is the likelihood.

HIERARCHICAL MODELS

The Bayesian approach shows great practical advantage in hierarchical models.

Assume that data are collected from many groups of units that are somehow similar, such as groups of subjects, animals, or cities. Rather than making inferences separately for each group, it is often desirable to combine the information from the various groups in order to better understand the phenomenon under study.

A natural way to approach the problem is to build a multiple-level “hierarchical model”, which includes a level specifying the distributions of the individual observations as a function of some group-specific parameters, as well as a level specifying the common distribution of the group parameters. In this way, hierarchical models allow modeling of individual heterogeneity as well as population-level correlation structure. FlexBayes implements several hierarchical regression models formed by distributions from the exponential family.

ASSESSING CONVERGENCE OF MONTE CARLO MARKOV CHAIN ALGORITHMS

3

| | |
|---------------------------------|-----------|
| Overview | 30 |
| MCMC Control Parameters | 31 |
| Practical Considerations | 32 |
| Time Series Plots | 32 |
| Autocorrelation Plots | 32 |
| Geweke's Diagnostic | 32 |
| Gelman and Rubin's Diagnostic | 33 |

OVERVIEW

The goal of Monte Carlo Markov Chain (MCMC) methods is to use simulated values from a Markov chain which has been run long enough to be very close to its limiting distribution. In practice, it is extremely difficult to know with absolute certainty that the chain is close enough; instead, users typically apply diagnostics to try to detect any problematic lack of convergence of the algorithm. Such diagnostics are crucial, because lack of convergence of the chain can lead to incorrect posterior inferences.

This chapter discusses the convergence diagnostics implemented in FlexBayes, as well as other practical considerations during the use of MCMC, including the choice of starting values.

MCMC CONTROL PARAMETERS

In MCMC, the goal is to accurately estimate characteristics of the posterior distribution $P(\theta|Y_{obs})$, such as its moments and quantiles. Convergence is given by the law of large numbers and occurs when the sample summaries are sufficiently close to the posterior quantities they estimate.

To estimate a quantity $g = g(\theta)$ of interest such as a point estimate, standard error, interval estimate, or posterior probability (such as a “Bayes p -value”), collect iterates

$$g(\theta^{k+1}), g(\theta^{k+2}), \dots, g(\theta^{k+n}) .$$

Here, k is the *burn-in period* and n is the Monte Carlo sample size. If k is large enough to ensure stationarity and n/k is large enough for the law of large numbers to apply, then the sample can be used to estimate posterior quantities (for example, estimate the posterior mean $E(g|Y_{obs})$ by the mean of $g(\theta^{k+i})$ over i). The burn-in period k theoretically should be chosen to be large enough to make $g(\theta^k)$ practically independent of $g(\theta^0)$.

The goal of convergence diagnostics is to determine a good value for k , by plotting or analyzing $\{g(\theta^i)\}$.

PRACTICAL CONSIDERATIONS

The FlexBayes library contains several functions for convergence diagnostics, including:

1. Trace (time series) plots for each parameter, i.e. a plot of parameter values versus iteration number.
2. Autocorrelation plots for each parameter.
3. Geweke's diagnostic plots.
4. Gelman and Rubin's diagnostic plots.

Most of these diagnostic functions call functions in the Spotfire S+ open-source package **coda**, and require this package to operate. For instructions on obtaining and installing **coda**, see the section Installing Dependencies on page 14.

For more information on these and other convergence diagnostics, see Cowles and Carlin (1996).

Time Series Plots

After convergence, time series (trace) plots should not show a trend, nor should iterates k steps apart have more than negligible correlation. Trace plots are available using the FlexBayes function `traceplot`.

Autocorrelation Plots

After convergence, autocorrelation function (ACF) plots should die out. The sample ACFs, excluding burn-in, should fall within approximate 0.05-level critical values for testing that the ACFs are zero. Autocorrelation plots are available using the FlexBayes function `autocorr.plot`.

Geweke's Diagnostic

Geweke (1992) proposes a diagnostic which is especially appropriate for diagnosing the convergence of the mean of some scalar function of the sampled variables. Typically the function is of one particular parameter, but the function could also be based on several parameters. In particular, -2 times the log of the posterior density can be used to assess convergence of the joint posterior.

Based a single chain, Geweke's method tests whether the means from different parts of a chain are statistically equal. If not, there is evidence that the chain has not converged. The variances needed in

performing the test are motivated by time series spectral analysis, which are appropriate because the posterior samples are usually dependent.

In particular, the chain is divided into two “windows” containing the first n_1 and last n_2 observations respectively. Geweke suggests taking the first tenth, and the last half, of the samples. These are the defaults in the `FlexBayes` function `geweke.diag`. If the whole chain is stationary, the means early and late in the sequence should be similar. Geweke’s diagnostic G is the difference between the two means divided by the estimated standard error of their difference. Asymptotically (in the limit of the number of iterations of the chain), G is normally distributed with mean 0 and variance 1. Therefore extreme values of G (e.g. above 3 or below -3) suggest that the chain has not converged during the first window.

FlexBayes Implementation

The function `geweke.diag` calculates Geweke’s diagnostic by calling the function of the same name in the open-source Spotfire S+/R library `coda`. The arguments are described in the help files for `geweke.diag` in `FlexBayes` and the **`coda`** library. As discussed above, extreme values give evidence that the chain has not yet converged. It might be reasonable to base further inference only on samples that occur after the Geweke diagnostic values become moderately sized.

Gelman and Rubin’s Diagnostic

One possible cause of poor convergence is that if the posterior distribution is multimodal, the Markov chain simulation may appear to have converged but really be trapped in one or several modes of the posterior distribution. In this situation, multiple replications of the chain might become trapped in distinct modes, revealing the problem. Gelman and Rubin (1992) devised a diagnosis procedure based on this idea, which uses multiple chains run from “overdispersed” starting values.

The Gelman-Rubin diagnostic compares the within- and between-chain variances for each variable in order to estimate the factor by which the standard deviation of the marginal posterior distribution of each parameter might be reduced if the chain were run to infinity.

FlexBayes Implementation

The FlexBayes function `gelman.diag` obtains Gelman-Rubin's diagnostic by calling the function of the same name in the open-source Spotfire S+/R library **coda**. The arguments are described in the help files for `gelman.diag` in the FlexBayes and coda libraries.

The rate of convergence of a Markov chain to stationarity partly depends on the starting values or starting distribution. As a general guideline, Schafer (1997) recommends using starting values that are near the center of the posterior. For example, use a maximum likelihood estimate or posterior mode obtained from running an EM algorithm, if one is available.

For multiple chains, Gelman and Rubin (1992) recommend starting values that are *overdispersed* (exhibit greater variability) relative to $P(\theta|Y_{obs})$. This increases the chance of discovering multiple modes of the distribution if they exist.

In order to obtain such overdispersed values, Schafer (1997) recommends using the bootstrap method as follows. Repeat the following M times:

1. Draw with replacement n^* rows from Y_{obs} to obtain a bootstrap sample Y_{obs}^b .
2. Calculate the maximum likelihood estimate of the parameters, $\hat{\theta}_b = \hat{\theta}(Y_{obs}^b)$.

If we take n^* to be smaller than n , say $n^* = \frac{n}{2}$, then $\hat{\theta}_b$ tends to be overdispersed relative to $P(\theta|Y_{obs})$. Take care, because a reduced data set size might lead to problems such as collinearity.

THE HIERARCHICAL LINEAR MIXED EFFECTS MODEL

4

| | |
|--------------------------------|-----------|
| Model Description | 36 |
| The First-Level Model | 36 |
| The Link Function | 37 |
| The Second-Level Model | 37 |
| Prior Specification | 38 |
| Example: Orthodont Data | 42 |
| Fitting the Hierarchical Model | 44 |
| Convergence Diagnosis | 46 |
| Posterior Inference | 46 |
| Bayesian Mixed Effects Model | 47 |

MODEL DESCRIPTION

The function `bhlm` fits a two-stage *hierarchical linear mixed effects* model. For background on hierarchical models, see the section Hierarchical Models on page 28. A mixed effects model contains one or both of fixed and random effects.

Hierarchical models are often applied when there are multiple groups of correlated outcomes. The situation of independent and identically distributed data is included as a special case where the number of groups is one.

For an example of grouped data, consider the data set `Orthodont` provided with `Spotfire S+`. It consists of measurements of the distance from the pituitary gland to the pterygo-maxillary fissure. These measurements were taken at two-year intervals from age eight to fourteen on 16 male and 11 female children; the measurements are shown in Figure 4.1. More details on the data can be found in Potthoff and Roy (1964).

The multiple measurements from the same subject are correlated; therefore the groups in the hierarchical model correspond to the subjects. More generally, the function `bhlm` can be applied to any continuous-valued outcomes y_{ij} where $i = 1, 2, \dots, n_j$ indexes the measurements in each of several groups $j = 1, 2, \dots, J$.

The First-Level Model

Let y_{ij} denote the i -th measurement in group j . The first-level model specifies the distribution of the individual responses y_{ij} . In the simplest case, these are assumed to be normal with individual mean θ_{ij} and a possibly group-specific variance σ_j^2 :

$$y_{ij} \sim N(\theta_{ij}, \sigma_j^2)$$

Often a common outcome variance is assumed, so that $\sigma_j^2 = \sigma^2$.

The variance parameters σ_j^2 or σ^2 are usually unknown, but `FlexBayes` also allows for the case that they are known.

The Link Function

The means θ_{ij} are assumed to have the standard linear mixed effects structure:

$$\theta_{ij} = m_{ij}\gamma + x_{ij}\beta_j$$

where m_{ij} and x_{ij} denote the q -dimensional fixed-effect predictors (covariates) and p -dimensional random-effect predictors, respectively, and γ, β_j are the q -dimensional fixed-effect coefficients and p -dimensional random effects, respectively. The fixed-effect coefficients take a common value across all groups (treatments), while the random effects are group-specific.

The Second-Level Model

The random effects β_j are assumed to have the linear model structure

$$\beta_j = z_j\alpha + u_j$$

where z_j is an r -dimensional vector of predictors associated with the j -th group or treatment, α is the corresponding $r \times p$ -dimensional matrix of coefficients, and u_j is an r -dimensional random vector of errors. The errors u_j are assumed to follow a multivariate normal distribution with mean equal to the zero vector and variance equal to $\tau^2 V$, where either V is the identity matrix and τ^2 is assigned a univariate prior, or $\tau^2 V$ is given an inverse Wishart prior.

The second-level coefficients α can be thought of as population parameters. Several types of prior distributions are available for α ; for instance, the multivariate normal prior distribution with known mean vector α_0 and known variance-covariance matrix V_o .

If you wish to fit a non-hierarchical mixed-effects model, you can omit the second-level model specification, so that $\beta_j \sim N(0, \tau^2 V)$.

Prior Specification

FlexBayes allows you to choose from a number of prior distributions for the coefficients (α, γ) and the variances σ_j^2 (alternatively, σ^2) and τ^2 (alternatively, $\tau^2 V$).

Table 4.1: Prior distributions available for a hierarchical mixed effects model.

| Parameter | Available prior distributions |
|----------------------------|--|
| α, γ | normal, t, non-informative flat |
| σ_j^2 OR σ^2 | inverse chi-square, uniform shrinkage, non-informative power, non-informative DuMouchel, known |
| τ^2 OR $\tau^2 V$ | inverse chi-square, uniform shrinkage, non-informative power, non-informative DuMouchel. inverse Wishart. |

Fixed Effect Coefficients

Often the fixed effect coefficients γ are given a flat non-informative prior. However, you also can specify a normal or t prior for γ .

Second-level coefficients

If enough information is present in the data, you can afford to have a flat non-informative prior on α . However if you have knowledge about α , then you should use it and put it into a non-flat prior. For example, say that experts in the field inform you that the most reasonable value of α is the vector $(0, 1)$ (here we assume that α is two-dimensional), but they do not know how far away α could plausibly be from $(0, 1)$. In this case you may specify a t prior for alpha with a “large” scale matrix, and several degrees of freedom:

```
> priorA <- bayes.t( c(0,1), 100*diag(2), df = 3 )
```


Variance Components

In a hierarchical model, the choice of prior for the variance parameters is important and difficult. The following priors can be specified for both variance components σ^2 and τ^2 , except for the case when the variance is known, which can only be specified for the outcome variance σ^2 , and the case where $\tau^2 V$ is assigned an inverse Wishart prior. You can also specify different priors for σ_j^2 associated to different groups $j = 1, \dots, J$.

In most applications, little or no prior information is available. You can specify any of the following three vague priors to reflect this uncertainty.

Vague priors

- **“non-informative” power:** This prior is of the form

$$p(\sigma^2) \propto \sigma^{2\kappa} \quad \kappa < 0$$

A proper posterior for the outcome variance σ^2 is guaranteed if $\kappa > (-n/2)$, where n is the number of observations. When using group-specific variances σ_j^2 , this prior is guaranteed to yield proper posteriors for specific values of κ depending on the number of observations in each group; FlexBayes always verifies that the power κ supplied in the prior specification gives a proper posterior. For the random effect variance parameter τ^2 the analogous restriction is $\kappa > -(Jp)/2$, where p is the dimension of the random effect coefficient vector β_j and J is the number of groups.

- **uniform shrinkage:** This prior is of the form (Daniels 1999)

$$p(\sigma^2) = (\sigma_0^2 / (\sigma^2 + \sigma_0^2)^2) \quad \sigma_0 > 0$$

One can show easily that σ_0^2 is the median of this distribution. This prior is diffuse in the sense that both its expectation and the expectation of the inverse $1/\sigma^2$ are infinite.

- **DuMouchel:** This prior has the form

$$p(\sigma^2) = \sigma_0 / (2\sigma(\sigma_0 + \sigma)^2) \quad \sigma_0 > 0$$

Similarly to the uniform shrinkage prior, this is a proper but diffuse distribution.

You might specify the following two informative priors for the variance components.

Informative Priors

- **Scaled inverse chi-square:** This prior has the form

$$p(\sigma^2) \propto \sigma^{-2(\nu_0/2 + 1)} \exp\{-(\nu_0 \sigma_0^2) / (2\sigma^2)\}$$

It is specified by two parameters, the degrees of freedom $\nu_o > 0$ and the scale σ_o^2 . The degrees of freedom give weight to your knowledge of the variance, which is assumed to lie about σ_o^2 . The larger ν_o , the more influence the prior has on the posterior of σ^2 .

- **Inverse Wishart:** This prior is the generalization of the above inverse chi-square prior to multivariate variance-covariance matrices. You may use it for $\tau^2 V$ when you do not want to restrict the random effects coefficients β_j to have equal variances, and/or you want to account for possible correlation among these coefficients. You can create this prior by calling, for example:

```
> priorTau2 = bayes.invWishart( df= 2,
                                scale= 100*diag(2) )
```

for a two-dimensional β_j .

Known Variances

In the rare case when the variances σ_j^2 (alternatively, σ^2) are known, you can specify a degenerate Dirac delta prior. For example,

```
> priorSigma2 = bayes.massPoint( c( 0.25, 0.50 ) )
```

creates a prior giving probability one to the event that $\sigma_1^2 = 0.25$ and $\sigma_2^2 = 0.50$.

EXAMPLE: ORTHODONT DATA

The Orthodont data set provided with Spotfire S+ consists of measurements of the distance from the pituitary gland to the pterygo-maxillary fissure. These measurements were taken at two-year intervals from age eight to fourteen on 16 male and 11 female children. More details can be found in Potthoff and Roy (1964). An analysis of these data carried out by Pinheiro and Bates (2001) using the mixed-effects model library nlme of Spotfire S+ found clear differences in the growth pattern between female and male children. Differences in the growth curves within each child are also visible from the curve plots in Figure 4.1. Pinheiro and Bates suggest fitting a mixed effects linear growth model to the data.

$$distance_{ij} = \gamma_o + \gamma_1 age_{ij} + \gamma_2 sex_j \times age_{ij} + \beta_j age_{ij} + \varepsilon_{ij}$$

with $\gamma = (\gamma_o, \gamma_1, \gamma_2)$ common among all children (fixed effects), and β_j subject-dependent (random effects). For this data set, $i = 1, \dots, 4$ and $j = 1, \dots, 27$.

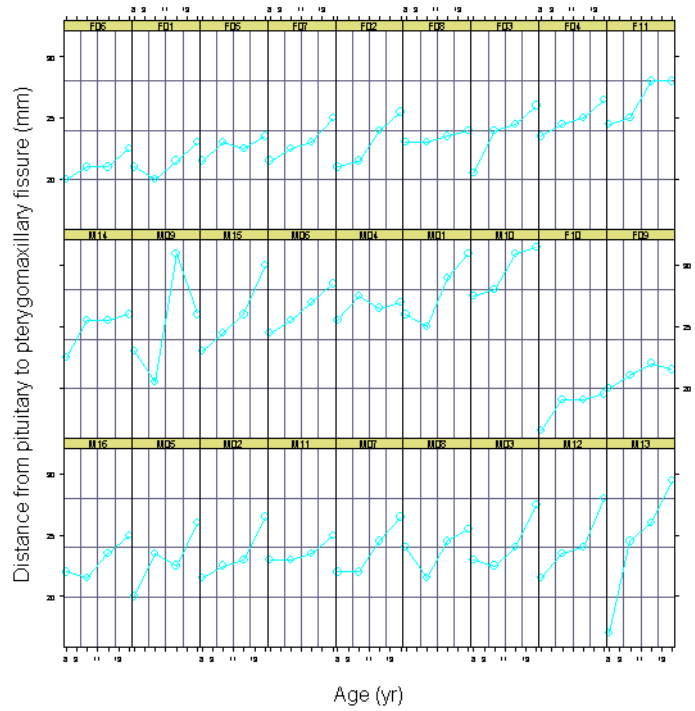


Figure 4.1: Growth curve plots of all 27 children in the Orthodont data set. M01 to M16 correspond to the 16 male subjects, and F01 to F11, to the female ones.

Alternatively, our hierarchical linear model setup allows us to model the variation in the growth rate due to sex in the second-level model

$$\beta_{kj} = \alpha_{ko} + \alpha_{k1}sex_j + u_{kj}$$

$k = 0, 1$ (corresponding to the subject intercept and slope, respectively), which seems to be a more “natural” way to model the growth rate as sex-dependent. The first-level model is then

$$distance_{ij} = \beta_{0j} + \beta_{1j}age_{ij} + \varepsilon_{ij}$$

Note that by including a second-level model there is no need to include fixed effects into the first level. The fixed effects are taken over by the population model for (β_{0j}, β_{1j}) .

Fitting the Hierarchical Model

First we show how to fit the hierarchical model (the second model described above). It is recommended that you work with the centered age variable (age-11). To fit the `Orthodont` data from the command line start by setting the priors for the model parameters

```
> alpha.prior = bayes.nonInformative()  
> error.prior = bayes.nonInfoPower( -1 )  
> betaCov.prior = bayes.invWishart(df= 2, scale= diag(2)/2)
```

You need to put all these priors in a single prior specification object

```
> orthodont.prior = bhlmm.prior( error.var= error.prior,  
  level2.coef= alpha.prior, random.var= betaCov.prior,  
  common.error.var= 2 )
```

This creates a list with the prior information needed to fit a hierarchical model to the `Orthodont` data. The specification `error.var=bayes.nonInfoPower(-1)` means that the prior density for σ^2 is proportional to $1/\sigma^2$ (which is a power law prior with power -1). The specification `level2.coef= bayes.nonInformative()` means that the second-level coefficients α have a flat improper prior. The specification `random.var= bayes.invWishart(df=2,scale=diag(2)/2)` means that the random effects covariance matrix $\tau^2 V$ has an inverse Wishart prior with two degrees of freedom and a diagonal scale matrix with entries $1/2$. The last term in the list, `common.error.var = 2`, specifies that all

the groups share the same variance, i.e. $\sigma_j^2 = \sigma^2$, for $j = 1, \dots, 27$. To specify the likelihood (error) structure of the model, set

```
> orthodont.likelihood = bhlml.likelihood(type= "normal")
```

The errors are assumed to be normally distributed; t-distributed errors are the alternative (given by `type= "t"`). Specify the number of chains desired in the simulation, as well as the control specifications for the chains and the way the parameters are to be initialized in each chain:

```
> orthodont.sampler = bhlml.sampler(nBurnin= 2000, nSamples=
1000, nChains= 3, init.point="prior")
```

Setting `init.point` to “prior” tells the program to initialize all parameters (within each chain) by random draws from t distributions that are centered at the prior mean/median (because α, γ have flat priors, the prior median does not exist and is nominally taken to be zero). See the help file for `bhlml.sampler` for more details on these default draws. Setting `init.point` to “user’s choice” would allow you to explicitly specify the initial values for all the model parameters in all chains.

Launch the fitting routine by calling

```
> orthodont.post = bhlml(
  random.formula= distance ~ I(age-11),
  level2.formula= ~Sex,
  group.formula= ~Subject,
  data= Orthodont,
  prior= orthodont.prior,
  likelihood= orthodont.likelihood,
  sampler = orthodont.sampler
)
```

The specification `random.formula= distance ~ I(age-11)` indicates that there is both a random “intercept” parameter (β_{0j}) and a random slope (β_{1j}) corresponding to the centered age variable. The specification `level2.formula= ~Sex` indicates that each of these random effects (both intercept and slope) have a mean structure given by an intercept plus a sex fixed effect. Setting `group.formula= ~Subject` indicates that the groups j correspond to the subjects.

Convergence Diagnosis

The resulting object `orthodont.post` is of class `posterior`. To check convergence of the three chains to the same parameter distributions take a look at the Gelman-Rubin plots. Obtain these using the `gelman.plot` function, which by default shows only the first six parameters:

```
> gelman.plot( orthodont.post )
```

To view the autocorrelation plots, type

```
> autocorr.plot( orthodont.post )
```

Posterior Inference

You can look at the model fit results by typing

```
> orthodont.post
```

By default, this prints only the first 30 parameters. Examine the complete results by typing:

```
> summary(orthodont.post, maxVars=80)
```

which yields:

```
*** Posterior Distribution from the Bayesian Model ***
Call:
bhlmm(random.formula = distance ~ I(age - 11),
level2.formula = ~ Sex,
group.formula = ~ Subject, data = Orthodont, prior =
orthodont.prior,
likelihood = orthodont.likelihood, sampler =
orthodont.sampler)

# of Chains: 3
Starting Iteration: 2001
Ending Iteration: 3000
Thinning: 1
# of Samples: 1000

1. Summary statistics:

              Mean      S.D.
MEASUREMENT ERROR [ SIGMA ]  1.30200 0.12090
      (Intercept):(Intercept) 24.98000 0.49910
              Sex:(Intercept) -2.32300 0.79990
      (Intercept):I(age - 11)  0.78180 0.13210
```



```

Sex:I(age - 11) -0.30950 0.20980
RANDOM:TAU:1.1 3.50800 1.23500
RANDOM:TAU:1.2 0.06855 0.21670
RANDOM:TAU:2.1 0.06855 0.21670
RANDOM:TAU:2.2 0.20490 0.07414
(Intercept):distance:M16 23.23000 0.62480
I(age - 11):distance:M16 0.60720 0.24850
(Intercept):distance:M05 23.24000 0.62080
I(age - 11):distance:M05 0.81110 0.24570
(Intercept):distance:M02 23.57000 0.63100
...

```

- The lines labelled (Intercept):(Intercept), Sex:(Intercept), (Intercept):I(age - 11), and Sex:I(age - 11) refer to the α parameters corresponding to the random intercept (the first two) and random slope of I(age - 11) (the last two), respectively.
- The lines labelled RANDOM:TAU:## correspond to the elements of the covariance matrix $\tau^2 V$ of the random effects.
- The lines labelled (Intercept):distance:<subj_ind> refer to the random intercept parameter for each subject and those labelled I(age - 11):distance:<subj_ind> correspond to the random slope parameter for each subject.

To visualize the parameter posterior densities, use the following (plots not shown here):

```
> densplot( orthodont.post )
```

Bayesian Mixed Effects Model

You can also fit a Bayesian version of the mixed effects model used by Pinheiro and Bates (2001) for the Orthodont data. The model is based on the following equation:

$$distance_{ij} = \gamma_o + \gamma_1 age_{ij} + \gamma_2 sex_j \times age_{ij} + \beta_j age_{ij} + \epsilon_{ij}$$

Here, the model is no longer hierarchical (there is no second level). Instead, all effects are specified in the first level. The subject variations are modeled through a random slope for the age predictor.

To fit this model, type the following at the command prompt:

```
> orthodont.prior = bhlm.prior( error.var=
  bayes.nonInfoPower( -1 ),
  fixed.coef= bayes.nonInformative(),
  random.var= bayes.invChisq(df = 3, sigma0.sq = 10),
  common.error.var= 2 )
> orthodont.sampler = bhlm.sampler(nBurnin= 2000,
  nSamples= 1000, nChains= 3,
  nThin = 10, init.point="prior")
> orthodont.post = bhlm(
  fixed.formula= distance ~ I(age-11) + Sex * I(age-11),
  random.formula = ~I(age-11)-1,
  group.formula= ~Subject,
  data= Orthodont, prior= orthodont.prior,
  likelihood= orthodont.likelihood,
  sampler = orthodont.sampler)
```

After doing convergence diagnostics, summarize the model output by typing:

```
> orthodont.post
```

The following is displayed:

```
*** Posterior Distribution from the Bayesian Model ***
Call:
bhlm(random.formula = ~ I(age - 11) - 1, fixed.formula =
distance ~ I(age -
  11) + Sex * I(age - 11), group.formula = ~ Subject, data
= Orthodont,
  prior = orthodont.prior, likelihood =
orthodont.likelihood, sampler =
  orthodont.sampler)

# of Chains: 3
Starting Iteration: 2001
Ending Iteration: 11991
```

Thinning: 10
of Samples: 1000

1. Summary statistics:

| | Mean | S.D. |
|-----------------------------|-----------|--------|
| (Intercept) | 24.970000 | 0.3115 |
| I(age - 11) | 0.783900 | 0.3329 |
| Sex | -2.315000 | 0.4824 |
| Sex:I(age - 11) | -0.301900 | 0.5186 |
| MEASUREMENT ERROR [SIGMA] | 2.428000 | 0.1889 |
| RANDOM:TAU | 1.199000 | 0.1690 |
| I(age - 11):distance:M16 | -0.191600 | 0.5654 |
| I(age - 11):distance:M05 | 0.065820 | 0.5717 |
| I(age - 11):distance:M02 | -0.016950 | 0.5575 |
| I(age - 11):distance:M11 | -0.371000 | 0.5656 |
| I(age - 11):distance:M07 | 0.004793 | 0.5684 |
| I(age - 11):distance:M08 | -0.342700 | 0.5685 |
| I(age - 11):distance:M03 | -0.024180 | 0.5545 |
| I(age - 11):distance:M12 | 0.184500 | 0.5617 |
| I(age - 11):distance:M13 | 0.959200 | 0.5677 |
| I(age - 11):distance:M14 | -0.207400 | 0.5774 |
| I(age - 11):distance:M09 | 0.155400 | 0.5548 |
| I(age - 11):distance:M15 | 0.281100 | 0.5645 |
| ... | | |

The first four parameters listed here are the fixed effects γ . SIGMA is the square root of the outcome variance and TAU is the square root of the random effect variance. The parameters I(age-11):distance:M# are the random slope parameters β_j for some of the male subjects. The remaining parameters in the model are not shown here. The above results for the fixed effects are very similar to the ones obtained through the hierarchical model.

THE HIERARCHICAL POISSON MIXED MODEL

5

| | |
|--|-----------|
| Model Description | 52 |
| The Individual Model | 52 |
| The Structural Model | 53 |
| The Second-Level Model | 55 |
| Prior Specification | 55 |
| Example 1: Pump Data | 57 |
| Prior Specification | 58 |
| MCMC Control Specification | 58 |
| Fitting the Model Using Gamma Conjugate | |
| Overdispersion | 59 |
| Convergence Diagnosis | 59 |
| Posterior Inference | 61 |
| Fitting the Log-Normal Model | 64 |
| Example 2: Epilepsy Data | 66 |
| Regression Models for the Epilepsy Data | 66 |
| Fitting the Mixed Model | 67 |
| Fitting the Hierarchical Model | 73 |
| Algorithms for the Hierarchical Poisson Model | 76 |
| MCMC Sampler for the Gamma Conjugate | |
| Overdispersion Case | 76 |
| MCMC Sampler for the No-Overdispersion Case | 78 |
| MCMC Sampler for the Log-Normal Overdispersion Case | 78 |

MODEL DESCRIPTION

The hierarchical Poisson regression implemented in the `FlexBayes` function `bhpm` enjoys the same multi-level structure as the hierarchical linear regression in `bhlm`. The Poisson hierarchy is illustrated by the following example.

Suppose there are J epilepsy patients in a study, each of which is monitored over several weeks of treatment. The number of seizures in week i for patient j could be assumed to be Poisson distributed with some rate λ_{ij} . The rates λ_{ij} for $j = 1, 2, \dots, J$ and $i = 1, 2, \dots, n_j$ are called the “individual parameters”. The approximate distribution of these parameters for the patients in treatment and control must be estimated in order to make inferences about the effect of treatment.

The individual parameters λ_{ij} may be modeled through a link as a linear function of a week effect along with some patient-specific random effects β_j . This component of the regression is sometimes called the “structural model”.

In turn, the mean of these random effects β_j may be modeled as a linear function of some patient-specific covariates (including a treatment indicator), just as is the case for the linear regression of the previous chapter. We will call the model for the random effects the “second-level model”.

Next, we describe each of the components (individual, structural, and second-level) of the Poisson hierarchy in detail. The `FlexBayes` Poisson hierarchical regression applies generally to any set of grouped and count-valued observations, where $j = 1, 2, \dots, J$ indexes the groups and $i = 1, 2, \dots, n_j$ indexes the observations.

The Individual Model

The individual model specifies the distribution of the i th observation y_{ij} in group j , given the individual parameter λ_{ij} :

$$y_{ij} | \lambda_{ij} \sim \text{Poisson}(e_{ij} \lambda_{ij})$$

In epidemiology the parameter λ_{ij} is often called the rate, and e_{ij} is called the exposure.

The Structural Model

The structural model is the model for the individual parameters λ_{ij} . The most commonly used Poisson model uses a log link function to relate the individual parameter λ_{ij} directly to the covariates and the random effects:

$$\log \lambda_{ij} = m_{ij} \gamma + x_{ij} \beta_j$$

where, just as in the linear model in the previous chapter, m_{ij} and x_{ij} are the fixed-effect predictors and random-effect predictors respectively, and γ, β_j are the fixed-effect coefficients and random-effect coefficients, respectively. This model is available in FlexBayes. Additionally, FlexBayes can be used to fit several models that allow some variability of $\log \lambda_{ij}$ around $m_{ij} \gamma + x_{ij} \beta_j$. One popular structural model is the gamma model, conjugate to the Poisson. In this model, λ_{ij} follows a gamma distribution

$$\lambda_{ij} | \xi_j, \mu_{ij} \sim \text{Gamma}\left(\xi_j, \frac{\xi_j}{\mu_{ij}}\right)$$

where $\xi_j > 0$ is a group-specific precision parameter and $\mu_{ij} > 0$ is the mean of the gamma distribution. Often a common value is used for the precision parameters ξ_j , so that $\xi_j = \xi$. The mean μ_{ij} is usually modeled through the log link equation

$$\log \mu_{ij} = m_{ij}\gamma + x_{ij}\beta_j$$

The parameter ξ_j^{-1} is an overdispersion parameter, meaning that

$\xi_j^{-1} = 0$ corresponds to a standard Poisson regression model with

log link function, $\log \lambda_{ij} = m_{ij}\gamma + x_{ij}\beta_j$, and that as ξ_j^{-1}

increases so does the variance of y_{ij} relative to this standard Poisson regression model. We will refer to this particular type of overdispersion as “gamma-conjugate” overdispersion, since λ_{ij} has been assigned a gamma distribution, and that distribution is conjugate to the Poisson.

An alternative to gamma-conjugate overdispersion is log-normal overdispersion, modeled with:

$$\log \lambda_{ij} \sim \text{Normal}(\mu_{ij}, \sigma_j^2)$$

where $\sigma_j^2 > 0$ and $\mu_{ij} = m_{ij}\gamma + x_{ij}\beta_j$, i.e. μ_{ij} is modeled as a linear function of the covariates. In practice, this type of overdispersion is a popular choice because with an appropriate prior, an MCMC scheme is simple to implement; see the section MCMC Sampler for the Log-Normal Overdispersion Case on page 78 for more details. The overdispersion parameter for the log-normal case is

σ_j^2 , which plays the same role as ξ_j^{-1} in the gamma conjugate case. Also similarly to the gamma-conjugate case, a common overdispersion parameter is often assumed, so that $\sigma_j^2 = \sigma^2$.

The Second-Level Model

The second-level model is specified exactly as in the hierarchical linear regression in the previous chapter. In particular,

$$\beta_j = z_j \alpha + u_j$$

where z_j are the predictors associated with the j -th group, α is the corresponding matrix of coefficients, and the vector u_j is distributed according to a multivariate normal distribution with mean equal to the zero vector and variance equal to $\tau^2 V$, where either V is the identity matrix and τ^2 is assigned a univariate prior, or $\tau^2 V$ is given an inverse Wishart prior.

Prior Specification

FlexBayes allows you to choose from a number of prior distributions for the coefficients (α, γ) , the random effect variance τ^2 (alternatively, $\tau^2 V$), and the optional overdispersion parameters σ_j^2 or ξ_j .

Fixed Effect Prior Specification

The regression coefficient vector γ can be given the same prior distributions as for the linear hierarchical model in the previous chapter, namely a normal prior, a t prior, or a flat (“non-informative”) prior.

**Prior
Specification for
the Gamma
Conjugate
Overdispersion
Parameter**

In the gamma conjugate overdispersion case, the prior for the precision parameter ξ (or the group-specific parameters ξ_j) can be specified as the *uniform shrinkage* distribution:

$$\pi(\xi) = \frac{z_0}{(\xi + z_0)^2}$$

Under the uniform shrinkage prior, the prior shrinkage factor $\xi/(\xi + z_0)$ is uniformly distributed in the interval $[0, 1]$. This prior is very vague, giving both ξ and $1/\xi$ infinite expectations. The constant z_0 is the median of ξ . Christiansen and Morris (1997) remark that small values of z_0 are less informative and lead to less shrinkage in the posterior than large values of z_0 . They recommend choosing $z_0 < \hat{\xi}$, where $\hat{\xi}$ is the maximum likelihood estimate of ξ .

**Prior
Specification for
the Log-Normal
Overdispersion
Parameter**

The log-normal overdispersion parameter σ^2 (alternatively, σ_j^2) is commonly assigned a scaled inverse chi-square prior:

$$\sigma^2 \sim \text{InvChisq}(v_o, s_o^2)$$

However, as in the hierarchical linear model, FlexBayes allows you to choose priors from a number of distributions. Refer to Table 4.1 for the available prior distributions for σ^2 (alternatively, σ_j^2).

**Prior
Specification for
the Second-Level
Model**

The prior specification for the second-level parameters τ^2 (alternatively, $\tau^2 V$) and α is exactly as in the linear hierarchical model. Refer to Table 4.1 for the available prior distributions for these parameters.

EXAMPLE 1: PUMP DATA

Here we give an example of an overdispersed Poisson regression with fixed effects. This example does not have a group structure, which in the framework of the previous section means that $J = 1$. For an example of the use of `bhpm` for fitting grouped data ($J > 1$), see the epilepsy data example in the section Example 2: Epilepsy Data on page 66.

The FlexBayes data frame `pumps` (Christiansen and Morris, 1997) contains counts of pump failures at a pressurized water reactor nuclear power plant. This data frame contains four variables z , e , y , and x giving the failure counts, the exposure (the time of operation in units of 1,048 hours), the ratio $y = z/e$, and a covariate (see Figure 5.1). The first six pumps ran intermittently, indicated by $x_i = -1$, while the four pumps with the largest exposures were operated continuously, indicated by $x_i = 1.5$. The exposures e_i are in units of 1,048 hours of operation.

| pumps | | | | | |
|-------|-----|-----|------|-------|---|
| | 1 | 2 | 3 | 4 | 5 |
| | z | e | y | x | |
| 1 | 1 | 1 | 1.00 | -1.00 | |
| 2 | 1 | 1 | 1.00 | -1.00 | |
| 3 | 4 | 2 | 2.00 | -1.00 | |
| 4 | 3 | 5 | 0.60 | -1.00 | |
| 5 | 22 | 10 | 2.20 | -1.00 | |
| 6 | 1 | 15 | 0.07 | -1.00 | |
| 7 | 19 | 30 | 0.63 | 1.50 | |
| 8 | 5 | 60 | 0.08 | 1.50 | |
| 9 | 5 | 90 | 0.06 | 1.50 | |
| 10 | 14 | 120 | 0.12 | 1.50 | |
| 11 | | | | | |
| 12 | | | | | |

Figure 5.1: The pump failure counts data set.

The quantity y_i is the i th pump failure rate in failures per 1,048 hours. The average rate for the intermittently operated pumps is 1.14, while the average rate for the continuously operated pumps is 0.22. The weighted average is 0.225.

Following Christiansen and Morris (1997) we use a Poisson regression with gamma-conjugate overdispersion for this data. We use a uniform shrinkage prior for the overdispersion parameter ξ , and set the prior median to be the weighted average 0.225 of the pump failure rates. The fixed effects are given a flat prior distribution, as is done in Christiansen and Morris (1997).

Prior Specification

First specify the priors for the parameters:

```
> pump.prior = bhpm.prior ( xi =  
  bayes.uniformShrinkage (0.225),  
  fixed.coef = "non-informative", common.glm = 2 )
```

Here `common.glm = 2` specifies that overdispersion is desired and that the overdispersion parameter should be common to all groups (although there is only one group for this example). The setting `xi = bayes.uniformShrinkage (0.225)` specifies a uniform shrinkage prior with median 0.225 for ξ . The "non-informative" specification for `fixed.coef` indicates the flat prior for the fixed effects.

MCMC Control Specification

Next, specify the MCMC control parameters, including the number of chains desired in the simulation, the number of samples desired from each chain, the burn-in length, and the way the parameters are to be initialized in each chain:

```
> pump.sampler = bhpm.sampler( nSamples=1000,  
  nChains = 3, nBurnin = 1000,  
  init.point = "prior", update.cov = 1 )
```

Simulating multiple chains allows us to use the Gelman-Rubin convergence diagnostic. Setting `init.point` to "prior" tells the program to initialize each parameter (within each chain) by a random draw from a t-distribution that is centered at the prior mean/median (nominally zero for a flat improper prior such as that specified for γ). Setting `update.cov = 1` directs the Metropolis-Hastings sampler to update the covariance structure of the proposal

distributions at each iteration. `update.cov = 0` would indicate that the covariance structure should be fixed throughout the simulation at the value computed during the first iteration of the sampler. This latter option might be useful if numerical errors arise when covariances are updated at each iteration, or when the number of predictor variables is very large and faster results are desired.

Fitting the Model Using Gamma Conjugate Overdispersion

Next, fit the model:

```
> pump.exposure = ~ e
> pump.fixed = z ~ x
> pump.post = bhpm(fixed.formula = pump.fixed,
  exposure.formula = pump.exposure, data = pumps,
  prior = pump.prior, sampler = pump.sampler,
  overdispersion = "gamma-conj" )
```

Here we have specified gamma-conjugate overdispersion. The fixed effects in the model consist of an intercept parameter and a slope parameter for x .

Convergence Diagnosis

The output `pump.post` is an object of class `posterior`. To check the convergence of the simulation, call the `traceplot` function on `pump.post`:

```
> traceplot(pump.post)
```

It looks like there might be a trend in the values of some parameters over the length of the chain. Perhaps the chain has not yet converged fully. As another convergence diagnostic, create autocorrelation plots for the parameters:

```
> autocorr.plot(pump.post)
```

It looks like there might be substantial autocorrelation out to lag 15 for the regression coefficients and the overdispersion parameter. Rerun the simulation, using thinning this time to reduce the autocorrelation:

```
> pump.sampler = bhpm.sampler( nSamples=1000,
  nChains = 3, nBurnin = 1000, nThin = 20,
  init.point = "prior", update.cov = 1 )
> pump.post = bhpm(fixed.formula = pump.fixed,
  exposure.formula = pump.exposure, data = pumps,
```

```
prior = pump.prior, sampler = pump.sampler,  
overdispersion = "gamma-conj" )
```

Recreating the autocorrelation plots shows that the autocorrelation is now negligible. Create Gelman-Rubin-Brooks plots as another convergence diagnostic:

```
> gelman.plot(pump.post)
```

This plots the Gelman-Rubin shrink factor for each parameter as a function of the iteration of the chain, yielding the plot in Figure 5.2. By default, only the first six parameters are plotted, but more can be obtained by setting the `maxVars` argument.

As described in Brooks and Gelman (1998), the shrink factor shown in the plots should be close to 1 for the entire length of the simulation. The shrink factor is loosely interpretable as an estimate of the factor by which the posterior interval of the parameter might shrink if the chain were run for a much longer time. Values up to about 1.2 are therefore probably acceptable. The Gelman-Rubin plots here do not show substantial lack of convergence.

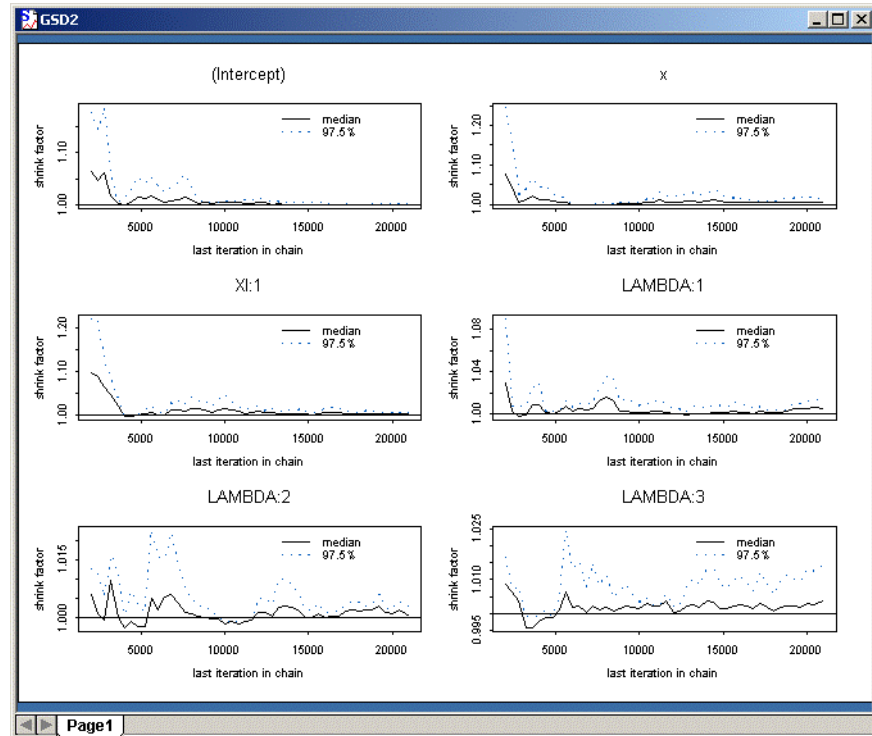


Figure 5.2: Gelman-Rubin-Brooks plots for the pumps MCMC.

Posterior Inference

Because the convergence diagnostics are acceptable, summarize the posterior distributions of the parameters:

```
> summary(pump.post)

*** Posterior Distribution from the Bayesian Model ***
Call:
bhpM(exposure.formula = pump.exposure, fixed.formula =
pump.fixed, data =
pumps, overdispersion = "gamma-conj", prior = pump.prior,
sampler = pump.sampler)
```

```
# of Chains: 3
Starting Iteration: 1001
```

Ending Iteration: 20981
Thinning: 20
of Samples: 1000

1. Summary statistics:

| | Mean | S.D. |
|-------------|----------|---------|
| (Intercept) | -0.41900 | 0.43360 |
| x | -0.64780 | 0.34310 |
| XI:1 | 1.00100 | 0.50810 |
| LAMBDA:1 | 1.09300 | 0.82540 |
| LAMBDA:2 | 1.08300 | 0.81820 |
| LAMBDA:3 | 1.74300 | 0.84920 |
| LAMBDA:4 | 0.67580 | 0.34880 |
| LAMBDA:5 | 2.11400 | 0.45080 |
| LAMBDA:6 | 0.12470 | 0.09182 |
| LAMBDA:7 | 0.57890 | 0.13860 |
| LAMBDA:8 | 0.09240 | 0.03797 |
| LAMBDA:9 | 0.06324 | 0.02610 |
| LAMBDA:10 | 0.12080 | 0.03105 |

2. Quantiles:

| | 2.5 % | 25 % | 50 % | 75 % | 97.5 % |
|-------------|----------|----------|----------|----------|---------|
| (Intercept) | -1.15800 | -0.70980 | -0.45400 | -0.17250 | 0.53450 |
| x | -1.28500 | -0.86960 | -0.65030 | -0.44300 | 0.05702 |
| XI:1 | 0.34500 | 0.64910 | 0.89320 | 1.22700 | 2.31200 |
| LAMBDA:1 | 0.10850 | 0.49510 | 0.89710 | 1.45600 | 3.22000 |
| LAMBDA:2 | 0.12160 | 0.49010 | 0.89470 | 1.43900 | 3.13100 |
| LAMBDA:3 | 0.53700 | 1.13000 | 1.59300 | 2.19400 | 3.79900 |
| LAMBDA:4 | 0.16620 | 0.42150 | 0.62540 | 0.86440 | 1.51800 |
| LAMBDA:5 | 1.32900 | 1.79700 | 2.08600 | 2.39400 | 3.10100 |
| LAMBDA:6 | 0.01126 | 0.05711 | 0.10410 | 0.16870 | 0.35550 |
| LAMBDA:7 | 0.33760 | 0.48050 | 0.56910 | 0.66620 | 0.86700 |
| LAMBDA:8 | 0.03473 | 0.06472 | 0.08722 | 0.11550 | 0.18080 |
| LAMBDA:9 | 0.02272 | 0.04451 | 0.05866 | 0.07866 | 0.12350 |
| LAMBDA:10 | 0.06827 | 0.09859 | 0.11840 | 0.13980 | 0.18880 |

The parameters that are summarized are the fixed effect coefficients corresponding to the intercept and to x , the overdispersion parameter ξ , and the rates λ . Here there is no group specification so $J = 1$ and $n_1 = 10$ (where J and n_j are as defined in the model description at the beginning of this chapter).

The posterior means for the regression coefficients and ξ are close to those reported by Christiansen and Morris (1997). The 95% credible (posterior) interval for the coefficient for x contains zero, so there is not strong evidence for a difference in failure rate between the two pump types. One can also look at the density plots for the parameters.

```
> densplot(pump.post)
```

yields the window in Figure 5.3. Three tabs are created, of which the first is visible. The other tabs show the density plots for the rest of the parameters.

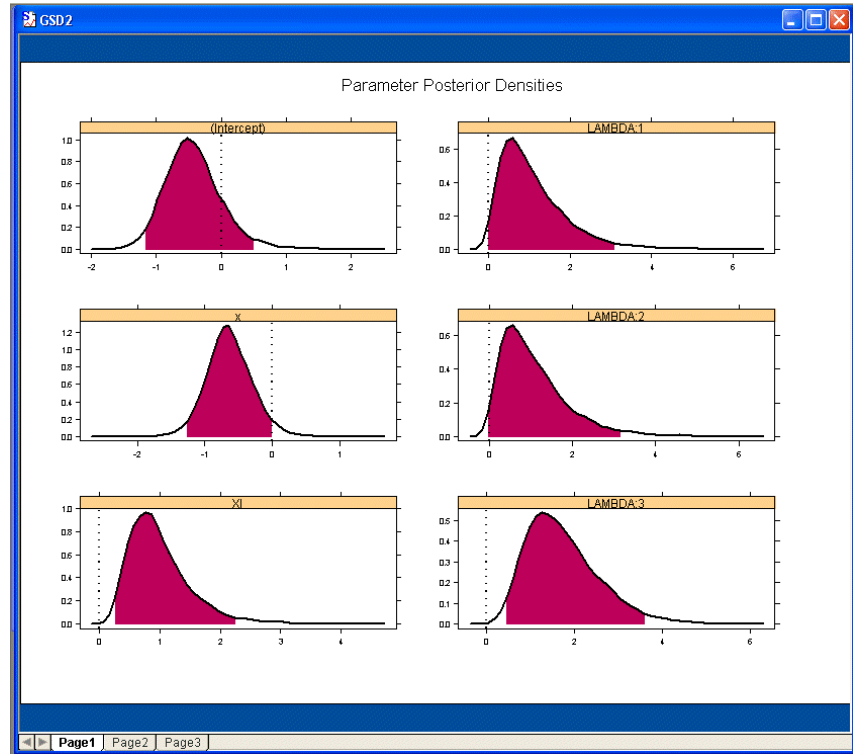


Figure 5.3: Posterior density plots for the first six parameters in the pumps example.

Note that the posterior distributions of the regression coefficients are nearly symmetric, while those of the overdispersion parameter ξ and some of the rates λ are right-skewed. Compare the posterior means for the rates to their empirical values, given by:

```
> pumps$y
[1] 1.00 1.00 2.00 0.60 2.20 0.07 0.63 0.08 0.06 0.12
```

Note that some shrinkage towards the common mean occurred in the Bayesian estimates relative to these raw rates.

Fitting the Log-Normal Model

You can also fit the Poisson log-normal model to the same data:

```
> pump.prior = bhpm.prior ( sigma2 =
  bayes.uniformShrinkage (0.5),
```

```
fixed.coef = "non-informative", common.glm = 2 )  
> pump.post = bhpm(fixed.formula = pump.fixed,  
  exposure.formula = pump.exposure, data = pumps,  
  prior = pump.prior, sampler = pump.sampler,  
  overdispersion = "log-normal" )
```

which gives similar results (by calling `summary(pump.post)`) to the gamma conjugate overdispersion case. The posterior interval for the x coefficient still contains zero, again indicating that there is not a significant effect of the pump type.

EXAMPLE 2: EPILEPSY DATA

FlexBayes includes the `epilepsy` data set, from Thall and Vail (1990), which gives seizure counts for 59 patients in the two weeks preceding each of four clinic visits. We use this example to showcase the ability of hierarchical regression models to fit grouped data. This example is also available by calling `help(epilepsy)` at the command prompt.

Out of the 59 patients, 31 received a particular medication while the others did not. The interest is in whether or not the medication is associated with decreased seizure counts. The seizure counts are grouped by patient, so $J = 59$ and $n_j = 4$ for each of $j = 1, \dots, J$.

Regression Models for the Epilepsy Data

Breslow and Clayton (1993) fit the following Poisson regression model (their model III) to the epilepsy data, where λ_{ij} is the underlying (unknown) seizure rate for patient j in the two weeks preceding visit i :

$$\log \lambda_{ij} \sim \text{Normal}(m_{ij}\gamma + \beta_j, \sigma^2)$$

Here m_{ij} is a vector of predictors consisting of an intercept, a treatment indicator, an indicator of week four, the logarithm of patient age, a variable capturing the baseline seizure count of the patient, and an interaction term between the “treatment” and “baseline” predictors. Week four is singled out because preliminary analysis showed that the counts in the fourth week were significantly lower than those in the other weeks.

Breslow and Clayton (1993) fit this model via estimating equations based on penalized quasilikelihood. We show how to perform Bayesian inference for the same model using the `bhpm` function.

Because several of the predictors are at the patient level, one can also consider the following hierarchical version of the model:

$$\log \lambda_{ij} \sim \text{Normal}(m_{ij}\gamma + \beta_j, \sigma^2)$$

$$\beta_j \sim \text{Normal}(z_j\alpha, \tau^2)$$

where m_{ij} now consists of just the indicator of week four and z_j is a vector containing the intercept, treatment, age and “baseline” predictors plus the interaction between “treatment” and “baseline”. We will also show how to fit this model using the `bhpm` function.

Fitting the Mixed Model

First, we demonstrate how to perform Bayesian inference for model III in Breslow and Clayton (1993) using `bhpm`. We start by loading and preparing the data. The `epilepsy` object is a list with one component `y` that gives the seizure counts, as well as components for the various predictors. `epilepsy$y` is a matrix with a row for each patient and a column for each visit, thus conforming to the j, i indexing scheme from the above description of the model.

`FlexBayes` requires that the data used to fit the model be in a data frame where one column gives the outcome counts, one column gives the group index j , and the other columns give the predictor values. Create such a data frame for the `epilepsy` data by calling:

```
> data(epilepsy)
> nSubj <- dim(epilepsy$y)[1]
> base <- log ( epilepsy$baseline / 4 )
> epilepsyBhpm <- data.frame(
  y = as.vector( epilepsy$y ),
  subj = rep( (1:nSubj), each = 4 ),
  visit4 = rep( epilepsy$visit4, nSubj ),
  base = rep( base, each = 4 ),
  treat = rep( epilepsy$treatment, each = 4 ),
  logAge = rep( log( epilepsy$age ), each = 4 ) )
```

Next, specify the prior distributions for the parameters:

```
> epilepsyPrior <- bhpm.prior( sigma2=
```

```
bayes.nonInfoPower(-1), common.glm = 2 )
```

Here we are specifying an improper power-law prior with power -1 for σ^2 (so that the prior density is proportional to $1/\sigma^2$). Because the priors for γ , α , and τ^2 are not specified, the default priors for these parameters are used. The defaults for γ and α are improper flat priors and the default for τ^2 is an improper power-law prior with power -1 (proportional to $1/\tau^2$).

Set the control parameters for the MCMC, including the thinning and the desired number of samples (after thinning):

```
> epilepsySampler <- bhpm.sampler( nSamples = 1000,
  nThin = 10, init.point = "user's choice",
  params.init = list( sigma2 = 1,
    fixed.coef = rep(0, 6), random.var = 1,
    random.coef = 0 ) )
```

Here we specify manually the initial values for the chain, because the default is to draw initial values from a distribution that is related to the prior; because the prior is very diffuse, the sampled values usually would be very far from the concentration of the posterior distribution. This leads to very slow convergence of the chain to stationarity. Choosing the initial values by hand leads to better convergence of the chain as long as the initial values are reasonable. The above specification indicates that the six fixed effects and the random effects for all of the groups are initialized at zero, and the random effect variance τ^2 and outcome variance σ^2 are initialized at 1.

Now we are ready to fit the model. Note that the following call to `bhpm` takes a few minutes to run:

```
> fixedEff <-
  y ~ visit4 + treat + base + logAge + treat * base
> randomEff <- ~ 1
> epilepsyPost <- bhpm( data = epilepsyBhpm,
  fixed.formula = fixedEff,
  random.formula = randomEff,
```

```
group.formula = ~ subj,
prior = epilepsyPrior,
sampler = epilepsySampler,
overdispersion = "log-normal" )
```

The specification `random.formula = ~ 1` indicates that there is a random intercept parameter (β_j) but no random slope parameters.

The argument `group.formula` sets the group structure for the model. Here the subjects form the groups (and thus the correlation structure of the seizure counts).

Convergence Diagnosis

Next, run several relevant convergence diagnostics on `epilepsyPost`. Autocorrelation plots (obtained by calling `autocorr.plot(epilepsyPost)`) show that slightly more thinning would be desirable (see Figure 5.4).

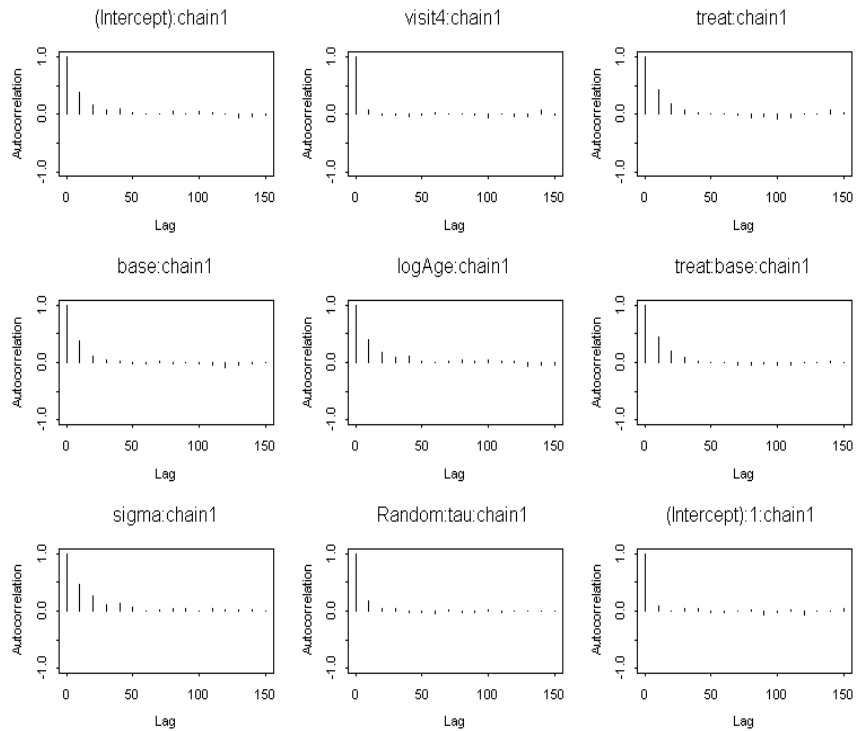


Figure 5.4: Autocorrelation plots for the epilepsy example.

Increasing the thinning to 40 leads to negligible autocorrelation (but increases the computing time by a factor of four). After increasing the thinning, generate trace (time series) plots with the following command.

```
> traceplot(epilepsyPost)
```

The first of these plots is shown in Figure 5.5. The plot is consistent with convergence to stationarity and good mixing. There is no trend in the mean or “stickiness” in the time series. Inspect each of the plots to verify that this holds for all of the parameters.

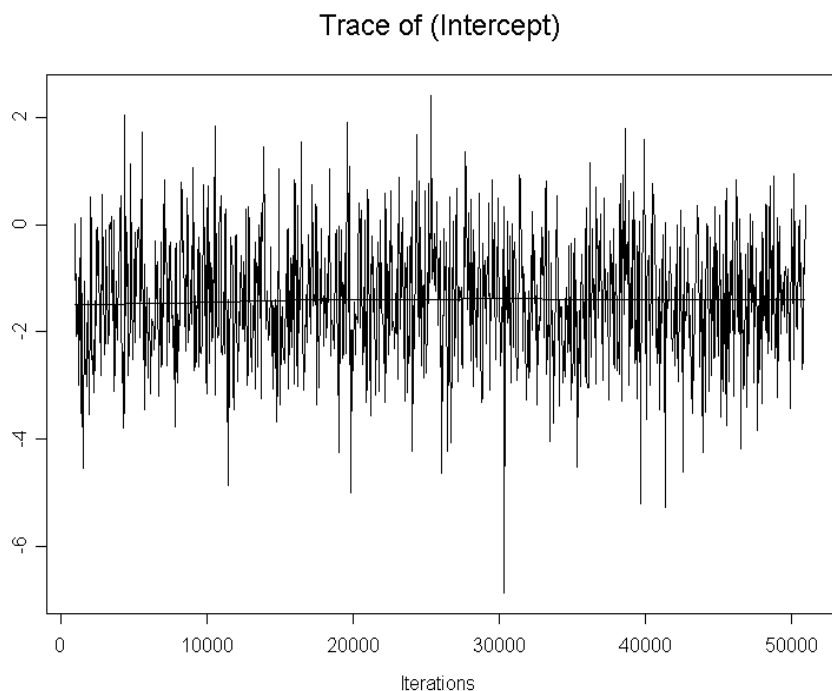


Figure 5.5: *The trace plot for the intercept in the epilepsy example.*

Posterior Inference

The convergence diagnostics are acceptable, so obtain the posterior summaries of the parameters by calling `summary(epilepsyPost)`. This yields the following (the summary statistic lists are truncated here to save space):

```
*** Posterior Distribution from the Bayesian Model ***
Call:
bhpm(random.formula = randomEff, fixed.formula = fixedEff,
group.formula = ~
```



```
subj, data = epilepsyBhpm, overdispersion = "log-normal",
prior =
epilepsyPrior, sampler = epilepsySampler)
```

```
# of Chains: 1
Starting Iteration: 1001
Ending Iteration: 40961
Thinning: 40
# of Samples: 1000
```

1. Summary statistics:

| | Mean | S.D. |
|---------------|----------|---------|
| (Intercept) | -1.43400 | 1.30200 |
| visit4 | -0.10100 | 0.09257 |
| treat | -0.94230 | 0.44100 |
| base | 0.88110 | 0.14080 |
| logAge | 0.49350 | 0.38410 |
| treat:base | 0.34570 | 0.22260 |
| sigma | 0.37050 | 0.04527 |
| Random:tau | 0.51660 | 0.07678 |
| (Intercept):1 | 0.04755 | 0.30860 |
| (Intercept):2 | 0.05692 | 0.29150 |

...

2. Quantiles:

| | 2.5 % | 25 % | 50 % | 75 % | 97.5 % |
|---------------|----------|----------|----------|----------|----------|
| (Intercept) | -3.95600 | -2.28900 | -1.40900 | -0.57020 | 1.05800 |
| visit4 | -0.28170 | -0.15820 | -0.10750 | -0.04004 | 0.08603 |
| treat | -1.80300 | -1.22200 | -0.94480 | -0.65500 | -0.08436 |
| base | 0.60280 | 0.78770 | 0.88410 | 0.96990 | 1.14900 |
| logAge | -0.26900 | 0.23490 | 0.49170 | 0.73960 | 1.25300 |
| treat:base | -0.08344 | 0.19410 | 0.34920 | 0.49620 | 0.78940 |
| sigma | 0.28840 | 0.33880 | 0.36980 | 0.39790 | 0.46390 |
| Random:tau | 0.38320 | 0.46300 | 0.51180 | 0.56720 | 0.67630 |
| (Intercept):1 | -0.55550 | -0.15070 | 0.04491 | 0.25410 | 0.65930 |
| (Intercept):2 | -0.54610 | -0.13600 | 0.06804 | 0.25210 | 0.58880 |

...

The first six parameters are the fixed effects. The others are called `sigma`, `Random:tau`, and `(Intercept):1`, `(Intercept):2`, and so on.

- `sigma` refers to the parameter σ .
- `Random:tau` refers to the parameter τ ; the prefix `Random:` is due to the fact that τ is the random effects standard deviation.
- `(Intercept):#` refers to the random effect β_j for $j = 1, \dots, 59$.

Not all of the parameter summaries are displayed by `FlexBayes`; by default, only the first thirty parameters are shown. One can increase the number of parameters that are displayed by setting the `maxVars` argument in the call to `summary` to be greater than thirty. However, this results in a fairly lengthy output. Alternatively, view the summaries for parameters 31-60 of the model by calling

```
> summary( epilepsyPost[, (31:60)] )
```

The parameters that are displayed are more of the subject effects β_j . Because `epilepsyPost` is of class `posterior`, the subsetting operation `epilepsyPost[, (31:60)]` is defined to return another object of class `posterior` containing the posterior samples for just parameters 31-60 of the `epilepsy` model. Therefore calling `summary` on this object generates the desired statistics.

To see the names of all of the parameters in the model, type `varnames(epilepsyBhpm)`. The parameters called `LAMBDA:1` through `LAMBDA:236` are the individual seizure rates λ_{ij} for each subject and visit; the numbering refers to the index of each data point in `epilepsyBhpm$y` rather than in the i, j indexing scheme.

The estimates of the fixed effects and the parameters σ and τ that we have obtained are close to those obtained by Breslow and Clayton (1993). This is not surprising, because the prior distributions used here are very vague.

The 95% credible interval for the treatment fixed effect is entirely below zero. Therefore there is evidence of a beneficial effect of the medication in reducing epileptic seizures. This is the same conclusion that one would draw from the analysis in Breslow and Clayton (1993).

Fitting the Hierarchical Model

Next we show how to fit the hierarchical version of the epilepsy model (the hierarchical version is described in the section Regression Models for the Epilepsy Data on page 66). We use the data frame `epilepsyBhpm` that was created by the code given in the previous subsection. Specify the MCMC control parameters with:

```
> epilepsySampler <- bhpm.sampler( nSamples = 1000,
  nThin = 40, init.point = "user's choice",
  params.init = list( sigma2 = 1,
    fixed.coef = 0, random.var = 1,
    random.coef = 0, level2.coef = rep(0, 5) ) )
```

Here we specified initial values for the fixed effect, the five second-level effects, the random effects (they are all initialized at zero), the random effect variance τ^2 and the outcome variance σ^2 .

Next, fit the model using the following code. Note that the call to `bhpm` takes quite a few minutes to run, in part because the thinning is large. For a quicker demo, reduce the thinning.

```
> epilepsyPrior <- bhpm.prior( sigma2=
  bayes.nonInfoPower(-1), common.glm = 2 )
> fixedEff <- y ~ visit4 - 1
> level2Eff <- ~ treat + base + logAge + treat * base
> epilepsyPost <- bhpm( data = epilepsyBhpm,
  fixed.formula = fixedEff,
  random.formula = ~ 1,
  group.formula = ~ subj,
  level2.formula = level2Eff,
  prior = epilepsyPrior,
```

```
sampler = epilepsySampler,
overdispersion = "log-normal" )
```

In this call, there is no intercept in the fixed effects because the intercept is included at the patient level of the hierarchy (`level2Eff` in the above code). Including intercepts at both levels would lead to nonidentifiability of the intercept parameters, exhibiting itself through poor mixing of the Markov chain (very high cross-correlation of the intercept parameters).

The specification `random.formula = ~ 1` indicates that β_j is a random intercept parameter. If we instead specified `random.formula = ~ visit4` then there would be both a random intercept term and a random coefficient for the “visit4” predictor.

Posterior Inference

Verify that the autocorrelation and trace plots are acceptable, and then summarize the posterior distribution using `summary(epilepsyPost)`, which gives the following result. (The summary statistic lists have been truncated to save space.)

```
*** Posterior Distribution from the Bayesian Model ***
Call:
bhpm(random.formula = randomEff, fixed.formula = fixedEff,
level2.formula =
level2Eff, group.formula = ~ subj, data = epilepsyBhpm,
overdispersion
= "log-normal", prior = epilepsyPrior, sampler =
epilepsySampler)

# of Chains: 1
Starting Iteration: 1001
Ending Iteration: 40961
Thinning: 40
# of Samples: 1000

1. Summary statistics:

              Mean      S.D.
visit4 -0.1006 0.08762
sigma  0.3697 0.04264
(Intercept):(Intercept) -1.3550 1.27800
treat:(Intercept) -0.9692 0.42760
base:(Intercept) 0.8741 0.14850
```

```
logAge:(Intercept)  0.4724 0.37820
treat:base:(Intercept) 0.3571 0.22170
      Random:tau    0.5137 0.07482
      (Intercept):1  1.1850 0.28190
      (Intercept):2  1.1910 0.28440

...

```

In this display, `visit4` refers to the fixed effect coefficient γ for visit four and `sigma` refers to the parameter σ . The lines labelled `(Intercept):(Intercept)` through `treat:base:(Intercept)` refer to the second-level coefficients α .

Once again, the fixed-effect coefficient estimates and conclusions match those from Breslow and Clayton (1993) closely.

ALGORITHMS FOR THE HIERARCHICAL POISSON MODEL

In this section, we explain the MCMC algorithms used to fit the hierarchical Poisson model without overdispersion, with gamma-conjugate overdispersion, and with log-normal overdispersion. Recall the model definitions from the section Model Description on page 52.

MCMC Sampler for the Gamma Conjugate Overdispersion Case

Because the gamma distribution is conjugate to the Poisson, the parameter λ_{ij} can be marginalized out for the gamma conjugate overdispersion case, so that the distribution of y_{ij} conditional on the rest of the parameters is the negative binomial distribution:

$$y_{ij} | \xi_j, B_{ij} \sim \frac{\Gamma(\xi_j + y_{ij})}{\Gamma(\xi_j) y_{ij}!} (1 - B_{ij})^{y_{ij}} B_{ij}^{\xi_j}$$

where B_{ij} is the shrinkage factor defined by $B_{ij} = \xi_j / (\xi_j + e_{ij} \mu_{ij})$.

The MCMC sampler in FlexBayes for the case of gamma-conjugate overdispersion is based on the Metropolis-Hastings algorithm, outlined as follows: initialize the parameter vector \mathbf{v} to some particular value $\mathbf{v}^{(0)}$ and repeat the following steps for $k = 0, 1, 2, \dots, N$:

- Generate \mathbf{v}^* from a proposal distribution $q(\mathbf{v}^{(k)}, \mathbf{v}^*)$ and u from a uniform distribution $U(0, 1)$;
- If $u < \rho(\mathbf{v}^{(k)}, \mathbf{v}^*)$ (where ρ is defined below) then set $\mathbf{v}^{(k+1)} = \mathbf{v}^*$ else set $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)}$.
- Return the values $\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}\}$.

The acceptance probability $\rho(\mathbf{v}, \omega)$ in the algorithm is given by the following, where $L(\mathbf{v}|y)$ is the likelihood function (the product over i, j of the density of $y_{ij}|\xi_j, B_{ij}$ as given above):

$$\rho(\mathbf{v}, \omega) = \frac{\pi(\omega)L(\omega|y)q(\omega, \mathbf{v})}{\pi(\mathbf{v})L(\mathbf{v}|y)q(\mathbf{v}, \omega)}$$

In practice, each element of the parameter vector is updated separately (the regression coefficients γ are updated as a block, as are the random effects β_j and the second-level coefficients α), conditional on the values of the rest of the parameters. The proposal distribution $q(\gamma^{(k)}, \gamma)$ for updating the regression coefficient vector γ is a multivariate normal distribution with mean $\gamma^{(k)}$ and covariance matrix $\Sigma_\gamma^{(k)}$, where $\Sigma_\gamma^{(k)}$ is the inverse of the negative Hessian of $L(\mathbf{v}|y)$, evaluated at $\gamma^{(k)}$. The proposal distribution for the random effects β_j is also a multivariate normal distribution, with the mean and covariance chosen in the analogous fashion. The proposal distribution for each overdispersion parameter ξ_j (alternatively, the common overdispersion ξ) is a univariate log-normal distribution with the mean and variance chosen analogously to that for γ .

The parameter α has a closed-form posterior distribution (normal) conditional on the values of the other parameters, so it is updated according to its conditional posterior distribution. The parameter τ^2 (alternatively, $\tau^2 V$) has a closed-form posterior distribution if its prior is inverse chi-squared (alternatively, inverse Wishart); in this case it is updated from its conditional posterior, and otherwise it is updated via a Metropolis-Hastings step.

**MCMC Sampler
for the No-
Overdispersion
Case**

A standard Poisson regression has no overdispersion and the distribution of y_{ij} conditional on the rest of the parameters is the Poisson distribution,

$$y_{ij}|\lambda_{ij} \sim \text{Poisson}(e_{ij}\lambda_{ij})$$

This case is computed similarly to the case with gamma-Poisson overdispersion. The only difference is that there is no parameter ξ_j and the likelihood function $L(v|y)$ is the product over i, j of the Poisson density of $y_{ij}|\lambda_{ij}$.

**MCMC Sampler
for the Log-
Normal
Overdispersion
Case**

When there is log-normal overdispersion, it is straightforward to construct a MCMC sampler to generate samples from the joint posterior distribution of all of the parameters $(\{\theta_{ij} = \log \lambda_{ij}\}, \gamma, \{\beta_j\}, \{\sigma_j^2\}, \tau^2 V, \alpha)$.

Conditional on the parameters $(\gamma, \{\beta_j\}, \{\sigma_j^2\}, \tau^2 V, \alpha)$, the transformed individual means $\{\theta_{ij}\}$ have independent posterior distributions with θ_{ij} distributed according to the density proportional to

$$f(y_{ij}|\theta_{ij}) \times \phi(\theta_{ij}|\beta_j, \gamma, \sigma_j^2)$$

where $f(y_{ij}|\theta_{ij})$ is the Poisson density with mean $\exp(\theta_{ij})$ and $\phi(\theta_{ij}|\beta_j, \gamma, \sigma_j^2)$ is the normal density with mean $m_{ij}\gamma + x_{ij}\beta_j$ and variance σ_j^2 . In FlexBayes, the parameters θ_{ij} are updated individually using a Metropolis-Hastings step.

By combining the structural model and the prior distributions, it is easily seen that the regression coefficients γ , the random effects β_j , and the second-level coefficients α have closed-form (normal) posterior distributions. The variance parameters τ^2 (alternatively, $\tau^2 V$) and σ_j^2 (alternatively, σ^2) have closed-form posterior distributions if their priors are inverse chi-squared or inverse Wishart; in this case they are updated from their conditional posteriors, and otherwise they are updated via a Metropolis-Hastings step.

HIERARCHICAL BINOMIAL MIXED MODEL

6

| | |
|---|-----------|
| Model Description | 82 |
| The Individual Model | 83 |
| The Structural Model | 83 |
| The Second-Level Model | 85 |
| Prior Specification | 85 |
| Example: Toxoplasmosis Data | 87 |
| Fitting the Model | 87 |
| Convergence Diagnosis | 88 |
| Posterior Inference | 92 |
| Algorithms for the Hierarchical Binomial Model | 94 |

MODEL DESCRIPTION

The hierarchical binomial model implemented in the FlexBayes function `bhbm` has the same multi-level structure as the hierarchical Poisson and linear models. This is illustrated by the following example.

Suppose that a chain of stores has locations in a number J of distinct regions, and that in each region j there are n_j stores.

Associated with each store i in region j is a binomial parameter θ_{ij} that represents the proportion of individuals entering the store that make a purchase. The data for a particular store might consist of the number of customers y_{ij} who made a purchase on a particular day, out of a total of n_{ij} customers entering the store that day. In order to make inferences about the effect of a store's layout, advertising, service level, or other factor on purchasing, one must take into account the fact that stores in the same region may be correlated.

In order to take this into account, the “individual parameters” θ_{ij} may be modeled as a function of store-level predictors m_{ij} as well as a region effect β_j . We call this part of the model the “structural model.” The mean structure of the region effects β_j may in turn be modeled as a function of region-level factors (e.g. average income and education levels) using regression coefficients α . We will call this component of the regression the “second-level model.”

This type of hierarchical analysis of binomial data is appropriate when groups, clusters, or correlated observations are present in the data, in which case we use $j = 1, 2, \dots, J$ to index the group,

$i = 1, 2, \dots, n_j$ to index the data, and y_{ij} to denote the number of events out of n_{ij} trials. Next we describe each of the levels (individual, structural, and second-level) in detail, as they are implemented in the FlexBayes function `bhbm`.

The Individual Model

The individual-level model specifies that the numbers of events y_{ij} , given the individual parameters θ_{ij} , are independently distributed as:

$$y_{ij}|\theta_{ij} \sim \text{Binomial}(n_{ij}, \theta_{ij})$$

The Structural Model

The most common structural model corresponds to standard logistic regression, wherein the individual parameters θ_{ij} are modeled as:

$$\log(\theta_{ij}/(1 - \theta_{ij})) = m_{ij}\gamma + x_{ij}\beta_j$$

where, just as in the linear and Poisson models, m_{ij} and x_{ij} are the fixed-effect predictors and random-effect predictors respectively, and γ, β_j are the fixed-effect coefficients and random effects, respectively. This model is available by calling `bhbm`.

Alternatively, the `bhbm` function also allows two types of structural models that incorporate variability of $\log(\theta_{ij}/(1 - \theta_{ij}))$ around

$m_{ij}\gamma + x_{ij}\beta_j$. One such structural model uses the beta distribution, conjugate to the binomial. The model is parametrized in terms of its mean $\mu_{ij} \in (0, 1)$ and a group-specific precision parameter $\xi_j > 0$ as follows:

$$\theta_{ij}|\mu_{ij}, \xi_j \sim \beta(\xi_j\mu_{ij}, \xi_j(1 - \mu_{ij}))$$

The precision parameter ξ_j characterizes the individual-level variability and is often assumed to be common to all of the groups, so that $\xi_j = \xi$. The mean μ_{ij} is modeled through the logit link equation

$$\log(\mu_{ij}/(1 - \mu_{ij})) = m_{ij}\gamma + x_{ij}\beta_j$$

The parameter ξ_j^{-1} is also called an overdispersion parameter, meaning that $\xi_j^{-1} = 0$ corresponds to standard logistic regression, and that as ξ_j^{-1} increases so does the variance of y_{ij} relative to this standard logistic regression. We will refer to this particular type of overdispersion as “beta conjugate” overdispersion, since θ_{ij} has been assigned a beta distribution, and since that distribution is conjugate to the binomial.

An alternative to beta conjugate overdispersion is logit-normal overdispersion, modeled with:

$$\log(\theta_{ij}/(1 - \theta_{ij})) \sim N(\mu_{ij}, \sigma_j^2)$$

where $\sigma_j > 0$ and $\mu_{ij} = m_{ij}\gamma + x_{ij}\beta_j$, i.e. μ_{ij} is modeled as a linear function of the covariates. In practice, this model is a popular choice because with an appropriate prior, the MCMC scheme is simple to implement; see the section Algorithms for the Hierarchical Binomial Model on page 94 for more details. The overdispersion parameter for the logit-normal case is σ_j^2 , which plays the same role as ξ_j^{-1} in the beta conjugate case. Also similarly to the beta conjugate case, a common overdispersion parameter is often assumed, so that $\sigma_j^2 = \sigma^2$.

The Second-Level Model

The second-level model is specified exactly as in the hierarchical linear and Poisson models. In particular,

$$\beta_j = z_j \alpha + u_j$$

where z_j are the predictors associated with the j -th group or treatment, α is the corresponding matrix of coefficients, and the vector u_j is distributed according to a multivariate normal distribution with mean equal to the zero vector and variance equal to $\tau^2 V$, where either V is the identity matrix and τ^2 is assigned a univariate prior, or $\tau^2 V$ is given an inverse Wishart prior.

Prior Specification

FlexBayes allows you to choose from a number of prior distributions for the coefficients (α, γ) , the random effect variance τ^2 (alternatively, $\tau^2 V$), and the optional overdispersion parameters σ_j^2 or ξ_j .

Fixed Effect Prior Specification

The regression coefficient vector γ can be given the same prior distributions as for the linear hierarchical model, namely a normal prior, a t prior, or a flat (“non-informative”) prior.

Prior Specification for the Beta Conjugate Overdispersion Parameter

In the beta conjugate overdispersion case, the prior for the precision parameter ξ (or the group-specific parameters ξ_j) must be specified as the uniform shrinkage distribution. This distribution is described in section Prior Specification for the Gamma Conjugate Overdispersion Parameter on page 56.

**Prior
Specification for
the Logit-Normal
Overdispersion
Parameter**

The logit-normal overdispersion parameter σ^2 (alternatively, σ_j^2) is commonly assigned a scaled inverse chi-square prior:

$$\sigma^2 \sim \text{InvChisq}(v_o, s_o^2)$$

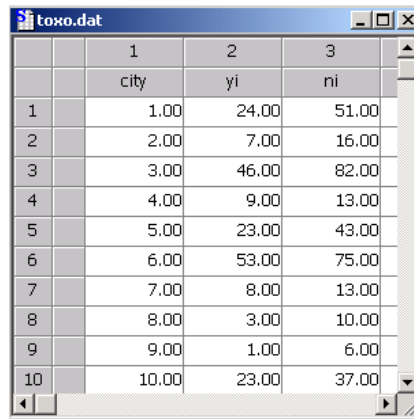
However, as in the hierarchical linear and Poisson models, FlexBayes allows you to choose priors from a number of distributions. Refer to Table 4.1 for the available prior distributions for σ^2 (alternatively, σ_j^2).

**Prior
Specification for
the Second-Level
Model**

The prior specification for the second-level parameters τ^2 (alternatively, $\tau^2 V$) and α is exactly as in the hierarchical linear and Poisson models. Refer to Table 4.1 for the available prior distributions for these parameters.

EXAMPLE: TOXOPLASMOSIS DATA

The data frame `toxо.dat` (Kass and Steffey 1989) contains observations on tests for toxoplasmosis, taken in 10 cities in El Salvador. y_i is the number of subjects that tested positive for toxoplasmosis in each city. n_i is the number of subjects tested in each city. A simple model for y_i is a binomial model with probability θ_i of testing positive for toxoplasmosis, given a total number n_i of trials. In terms of the framework given in the previous section, there is a single group, so that $J = 1$ and $n_1 = 10$.



| | 1 | 2 | 3 |
|----|-------|-------|-------|
| | city | yi | ni |
| 1 | 1.00 | 24.00 | 51.00 |
| 2 | 2.00 | 7.00 | 16.00 |
| 3 | 3.00 | 46.00 | 82.00 |
| 4 | 4.00 | 9.00 | 13.00 |
| 5 | 5.00 | 23.00 | 43.00 |
| 6 | 6.00 | 53.00 | 75.00 |
| 7 | 7.00 | 8.00 | 13.00 |
| 8 | 8.00 | 3.00 | 10.00 |
| 9 | 9.00 | 1.00 | 6.00 |
| 10 | 10.00 | 23.00 | 37.00 |

Figure 6.1: *The toxoplasmosis data set.*

Fitting the Model

In this section we fit the toxoplasmosis data from the command line, using a binomial regression model with beta conjugate overdispersion. First set the priors for the model parameters

```
> toxо.prior <- bһbm.prior( xi= bayes.uniformShrinkage(
median= 0.4 ), fixed.coef= "non-informative", common.glm= 2
)
```

By setting `common.glm= 2` we have specified that overdispersion should be included in the model, and that a common overdispersion parameter ξ should be used for all groups. In fact, there is only a single group for this example, so `common.glm= 2` is equivalent to

`common.glm= 1` and `common.glm= 0`. The above code also specifies a uniform shrinkage prior for ξ and a flat prior for the (fixed-effect) intercept γ . Next, create a list of MCMC control specifications:

```
> toxo.sampler <- bhbm.sampler(nBurnin= 2000, nSamples=
  1000, nThin= 25, nChains= 3, update.cov= 1, init.point=
  "prior")
```

Here we specify multiple chains in order to allow use of the Gelman-Rubin convergence diagnostic. Setting `init.point` to "prior" tells the program to initialize all parameters (within each chain) by random draws from distributions related to the priors for the parameters. For instance, the intercept γ is drawn from a t -distribution with 3 degrees of freedom centered at zero (the nominal "center" of the prior), in order to give a wide range of selection for the initial value of this parameter and to avoid very similar initial draws in the different chains. Setting `update.cov` to 1 directs the Metropolis-Hastings sampler to update the covariance structure of the proposal distribution at each iteration. If this argument is set to zero, then the covariance structure is fixed during the entire simulation at the value computed during the first iteration of the sampler. This latter option might be useful if numerical errors arise when updating covariances on each iteration, or when the number of covariates is very large and faster results are desired. Now you are ready to launch the fitting routine

```
> toxo.post <- bhbm(trials.formula= ~ ni, fixed.formula= yi
  ~ 1, data= toxo.dat, overdispersion= "beta-conj", prior=
  toxo.prior, sampler= toxo.sampler )
```

Note that `overdispersion="beta-conj"` specifies beta conjugate overdispersion as described in section Model Description on page 82. The specification `fixed.formula=~1` gives a single regression coefficient, namely an (fixed effect) intercept. The output `toxو.post` is an object of class `posterior`.

Convergence Diagnosis

You can look at autocorrelation plots for the parameters by typing

```
> autocorr.plot(toxo.post)
```

The first tab of the resulting window is given in Figure 6.2, showing the autocorrelation plots from the first chain for the intercept γ , the overdispersion parameter ξ , and the individual parameters θ_i . The autocorrelation plots for the rest of the individual parameters and the second and third chains are also generated but are not shown here. The autocorrelation plots show that slightly more thinning would be desirable. Increase the thinning in the above model fit by setting `nThin= 100` in the call to `bhbm.sampler`. Then produce the Gelman-Rubin-Brooks plots by calling `gelman.plot(toxo.post)`, which gives the plot shown in Figure 6.3. The only parameter that exhibits substantial lack of convergence by this measure is ξ .

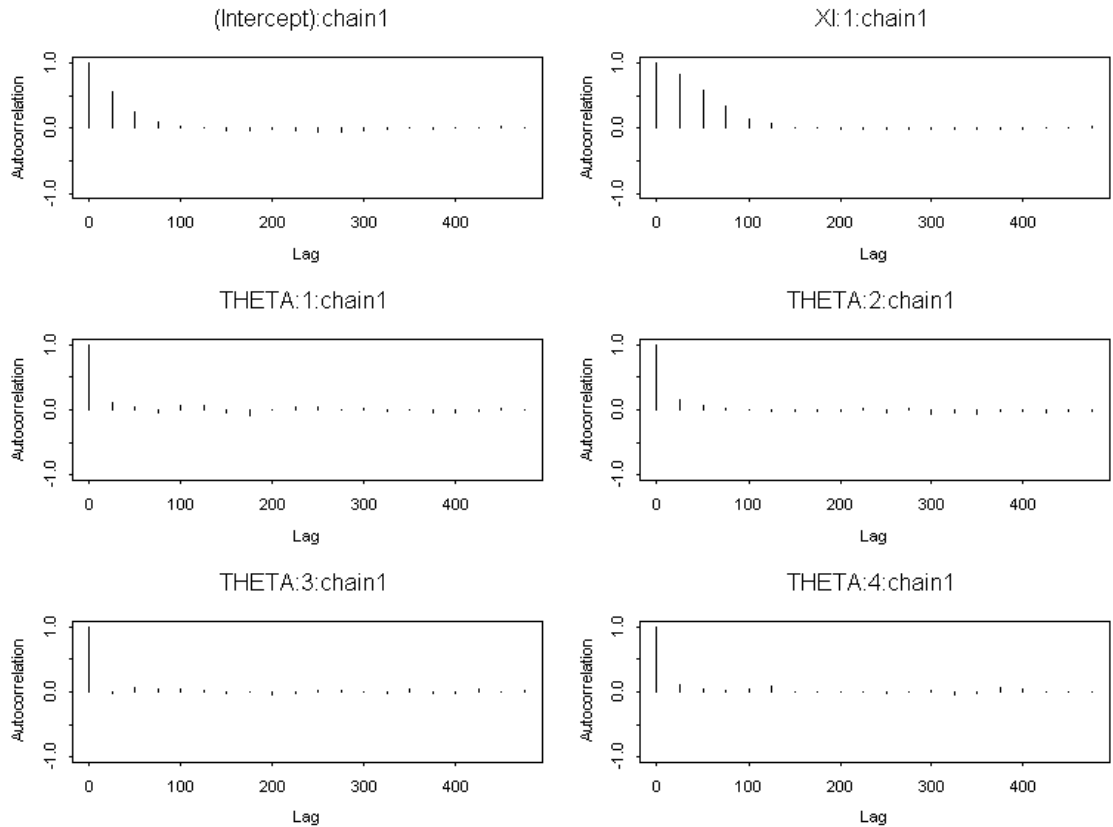


Figure 6.2: Autocorrelation plots for the toxoplasmosis example.

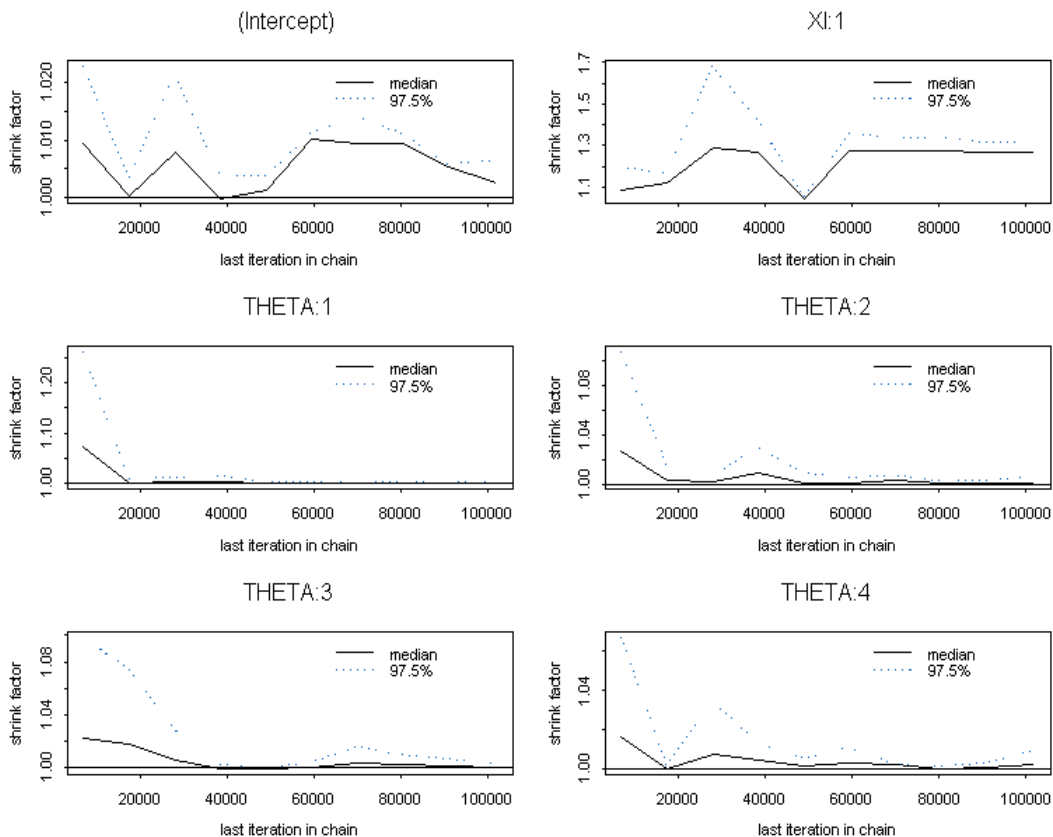


Figure 6.3: Gelman-Rubin-Brooks plots for the toxoplasmosis example.

Look at the estimated posterior density of ξ by calling `densplot(toxo.post)`, which yields the plot given in Figure 6.4.

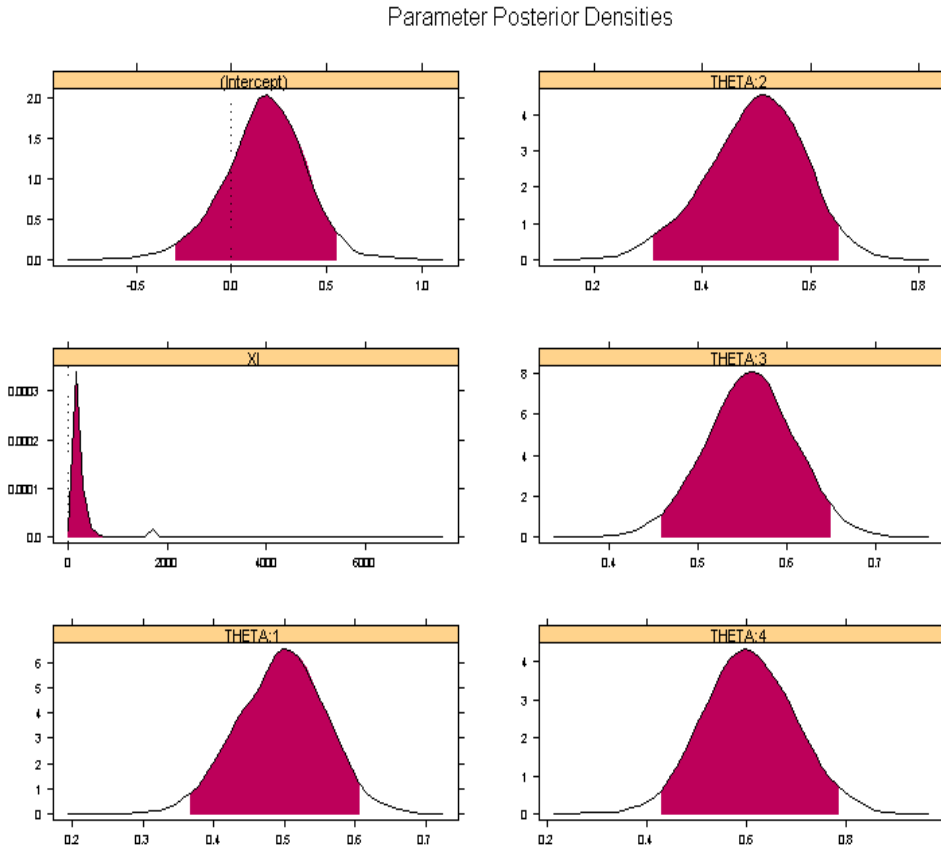


Figure 6.4: *Posterior density plots for the toxoplasmosis data.*

Notice that the estimated posterior density of ξ is very right-skewed.

This can be viewed more clearly by looking at the trace plot for ξ , generated by calling `traceplot(toxo.post)`. The Gelman-Rubin diagnostic assumes normality of the posterior density, so that in cases with very skewed posterior density it can yield invalid results.

Application of the Gelman-Rubin diagnostic to a log transformation of the posterior samples of ξ shows no substantial lack of convergence.

Posterior Inference

Create posterior summaries of `toxopost` by typing its name, which gives the following output. Be aware that in all of the chains we have only drawn 3,000 samples from the posterior distribution, so that our posterior inferences are subject to fairly high random variability; in order to obtain more accurate posterior inferences increase the value of `nSamples` in the above call to `bhbm.sampler`.

```
> toxopost
*** Posterior Distribution from the Bayesian Model ***
Call:
bhbm(trials.formula = ~ ni, fixed.formula = yi ~ 1, data =
toxodat,
      overdispersion = "beta-conj", prior = toxoprior, sampler =
      toxosampler )

# of Chains: 3
Starting Iteration: 2001
Ending Iteration: 101901
Thinning: 100
# of Samples: 1000

1. Summary statistics:
```

| | Mean | S.D. |
|-------------|---------|-----------|
| (Intercept) | 0.1747 | 0.21930 |
| XI | 66.8600 | 873.90000 |
| THETA:1 | 0.4959 | 0.06314 |
| THETA:2 | 0.4963 | 0.09041 |
| THETA:3 | 0.5573 | 0.04966 |
| THETA:4 | 0.6083 | 0.09236 |
| THETA:5 | 0.5404 | 0.06340 |
| THETA:6 | 0.6692 | 0.05416 |
| THETA:7 | 0.5764 | 0.08987 |
| THETA:8 | 0.4567 | 0.11030 |
| THETA:9 | 0.4414 | 0.12580 |

```
THETA:10  0.5958  0.06543
```

2. Quantiles:

| | 2.5 % | 25 % | 50 % | 75 % | 97.5 % |
|-------------|---------|---------|---------|---------|----------|
| (Intercept) | -0.2738 | 0.04328 | 0.1869 | 0.3145 | 0.5911 |
| XI | 3.5700 | 9.98300 | 17.8400 | 35.0900 | 215.9000 |
| THETA:1 | 0.3663 | 0.45510 | 0.4976 | 0.5399 | 0.6135 |
| THETA:2 | 0.2978 | 0.44000 | 0.5017 | 0.5572 | 0.6651 |

...

The estimated values of the individual parameters θ_i can be compared to their empirical values, given by

```
> toxo.dat$yi / toxo.dat$ni
```

This comparison shows that shrinkage of these parameters towards the common mean has occurred in the Bayesian model.

ALGORITHMS FOR THE HIERARCHICAL BINOMIAL MODEL

Computation in the hierarchical binomial model is analogous to the hierarchical Poisson model (see the section Algorithms for the Hierarchical Poisson Model on page 77 for details on the latter). The main difference is that in the binomial model, integrating out the individual parameters for the conjugate beta overdispersion case yields a beta-binomial distribution for the outcome counts, while in the Poisson model this integration yields a negative binomial distribution for the outcome counts. However, the consequences for computation are the same; for the beta conjugate overdispersion case it is possible to perform computation entirely ignoring the individual parameters. As for logit-normal overdispersion in the binomial model, the computation is analogous to that for log-normal overdispersion in the Poisson model. For these cases, the individual parameters must be explicitly sampled.

INDEX

Symbols

4

A

as.mcmc.list.posterior 4
autocorr 4
autocorr.plot 4

B

bayes.distribution 5
bayes.duMouchel 6
bayes.invChisq 6
bayes.invWishart 6
bayes.massPoint 6
bayes.nonInfoPower 6
bayes.normal 6
bayes.normal.mixture 6
bayes.t 6
bayes.t.mixture 6
bayes.uniformShrinkage 6
Bayesian 2
Bayes posterior distribution 12
Bayes Using Gibbs Sampling 3
bbm 3, 4
bbm.sampler 4
bhbm 3, 4
bhbm.sampler 4
bhlm 2, 4
bhlm.sampler 4
bhpm 3, 4
bhpm.prior 4

bhpm.sampler 4
blm 4
blm.prior 4
blm.sampler 4, 8
bpm 3, 4
bpm.prior 4
BRugs 14
BUGS 3
BUGS model 3

C

chisquare 7
coda 8
collinearity 2
covariance parameters 5
crosscorr 4
crosscorr.plot 4
cumuplot 4

D

densplot 4, 9

E

effectiveSize 4
end.posterior 4

G

gelman.diag 4
gelman.plot 4
Gelman-Rubin 8

getSamples 4
geweke.diag 4
geweke.plot 4
Gibbs sampler 8

H

heidel.diag 4

I

identity 7

L

lagged.crosscorr 4
least squares linear regression 11

M

Monte Carlo Markov Chain 2

N

name 5
nchain.posterior 4
niter.posterior 4
nvar.posterior 4

P

parameters 5
posterior 8
posteriorSamples 3, 4, 14

print 8

R

R2WinBUGS 14
raftery.diag 4
regression coefficients 5

S

stack.blm 8
start.posterior 4
summary 4, 8

T

thin.posterior 4
traceplot 4

V

validateFlexBayes 4
variance parameters 5
varnames.posterior 4

W

WinBUGS 14
window.posterior 4

Z

zero 7